Lyrics-based interpretation of songs

Aryan Madhanjith

University of Kwa-Zulu Natal aryan.madhanjith@gmail.com

ABSTRACT

A machine learning model capable of interpreting song lyrics has many uses, from education to assisting songwriters The goal of this project is to create such using a pre-trained BERT and BART models - on a dataset of annotated lyrics, where the annotations represent the interpretations of human experts. Due to the lack of computing resources, the models cannot be trained fully. However, the results are promising, as the models are improving as they train with incrementally larger subsets of the full dataset. This implies that they would meet expectations in a less resource-constrained environment.

1 INTRODUCTION AND BACKGROUND

1.1 Introduction

What do these lyrics mean?

"Strumming my pain with his fingers
Singing my life with his words
Killing me softly with his song
Killing me softly with his song
Telling my whole life with his words
Killing me softly with his song"

These lyrics are from the song "Killing Me Softly With His Song" originally performed by Roberta Flack and later covered by many other artists. The song is about a woman who is deeply moved by a performance. It suggests that the singer has a powerful ability to evoke emotions in the woman, as if he is strumming her pain with his fingers and singing her life with his words. The repetition of the line "Killing me softly with his song" emphasizes the intensity of the emotional impact. The singer's words are so powerful that they are able to tell the story of the woman's whole life. Overall, the lyrics convey a sense of vulnerability and raw emotion in the woman, who is being deeply affected by the singer's performance. The song is a testament to the power of music to touch the soul and evoke powerful emotions in the listener.

Another interpretation of the song "Killing Me Softly" by Roberta Flack (or the Fugees) could be that it describes the emotional impact of music on the listener. The singer describes being moved by a musician who is playing and singing with such passion and emotion that it feels like he is "strumming [her] pain" and "singing [her] life" with his words. The musician's performance is so powerful that it feels like he is "killing [her] softly" with his song. In this interpretation, the song is not about a romantic relationship, but rather about the transcendent power of music to connect with people on a deep emotional level. The listener is so moved by the musician's performance that she feels like she is being "killed" by the intensity of her emotions, yet at the same time she is grateful for the experience of being so deeply affected by the music. This interpretation is supported by the fact that the song has been covered by many different artists in various genres over the years, demonstrating its enduring appeal and ability to connect with people in different ways.

There are various websites and forums where people discuss and interpret song lyrics. Some of these include Genius, SongMeanings, and r/LyricInterpretations community from the social media platform Reddit. These platforms can offer different perspectives and interpretations of the same lyrics, which can be useful in gaining a broader understanding of a song's meaning..

The goal is to develop a model that is capable of providing a comprehensive and linguistically fluent understanding of the lyrics of a given song. It can be applied to various scenarios, such as:

1. 1. lyric-based song recommendations: If a user listens to songs having a positive and upbeat message, more such songs can be recommended by the model)
2. Parental controls: Parents, guardians can prevent songs with adult undertones and themes from reaching their children's ears.
3. Education: The model can be used in a learning environment by teachers and students alike to help understand poetry.
4. Song writing: The application does not have to be limited to just the listeners of a song, as artists can use interpretations to help in their creative processes or gain inspiration.

1.2 Background

The interpretation of song lyrics is one of the hardest problems in NLP. It goes beyond even text comprehension and question answering, wherein the answers can be found directly in the text. This is not the case for lyric interpretations, as many (if not most) songs feature figures of speech that explore deeper topics and themes without directly refereeing to them.

Additionally, songs and their analysis are often subjective, hence the same song can mean different things to different people. This makes it hard to give a "correct" interpretation for any one song.

Even more complex is when songs are not written to have a meaning: A Genius Lyrics [18] user attempts to explain the line "*What the hell am I tryin' to say?*" form the Nirvana song *On a Plain*, by referencing a quote from lead singer and song writer Kurt Cobain,

"People expect more of a thematic angle with our music. They always want to read into it. And before, I was just using pieces of poetry, and just garble—just garbage. Y'know, just stuff that would spew out of me at the time. And a lot of times when I write lyrics it's just at the last second, 'cause I'm really lazy. And then I find myself having to come up with explanations for it, you know?"

Even in this case, it is hard to know whether he was expressing his real creative process (as most of his songs do have a lot of meaning) or if it was his habit of evading interviewers.

Lastly, songs often require additional context to be fully appreciated, for example: Different time periods imposed a different set of beliefs, slang, and ideologies upon the artists.

2 LITERATURE REVIEW AND RELATED WORKS

2.1 Pre-trained models

The popularity and availably of educational resources for machine learning is growing. It has got to the point where ML courses are being implemented at a high-school level – with promising results [14]. Consider the exponential rate at which models are being added even to a single platform (Hugging Face), as shown in Figure 1:

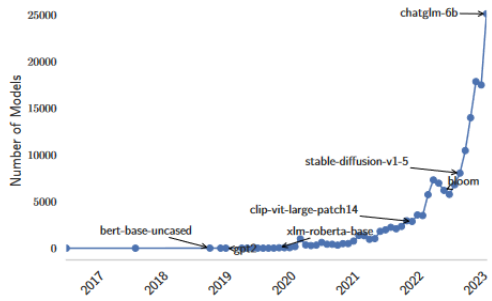

Figure 1 : Number of models on the Hugging Face platform over time [15]

However, this influx of people also means a directly proportional increase in the computing resources required to accommodate a larger community. These resources range from storage to GPUs, which are also being used by research teams to increase the capabilities of models (the ever-improving GPT models being a prominent example). The increase in resource usage is unfortunately directly proportional to CO_2 emissions and a larger carbon footprint [15]. These emissions are comparable to other sources (such as charging a phone to the amount of energy needed to power a home for a year) – as illustrated in Figure 2.

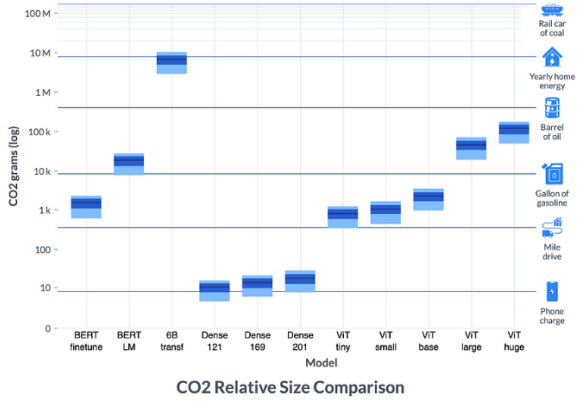

Figure 2 : comparing ML emissions to other sources. [16]

To be more mindful of our impact on the environment, we can use the concept of transfer learning [2]. It allows for machine learning models to utilize the captured knowledge of other, more sophisticated models for a specific task (such that the entire training process need not be repeated). This is analogous to people applying previous experiences and knowledge to solve new problems.

The Pre-Trained Model (PTM) is trained on a much larger and more general dataset, before being "fine-tuned" on a smaller, specific set of examples. This is useful in the domain of an NLP task, as these PTMs understand the syntax and lexicon of the English language.

Currently, the best-known approach to PTMs regarding NLP tasks is to use a transformer-based architecture as introduced in [3].

2.2 Transformers

This type of architecture revolves around the concept of "attention", which attempts to factor in the context of words. This is the often the main challenge of NLP tasks, for example: "Africa" can refer to the continent, a holiday destination, or a site of historical importance. There are two types of attention:

(1) Self-attention refers to calculating a vector representation of each word, in turn making use of 3 other vectors known as query, key and value. The attention score of each word is calculated as:

$$\mathrm{A(q,k,v)} = \sum_i \frac{exp(q * k < i >)}{\sum_j exp(q * k < j >)}$$

Figure 3 shows how a sample sentence can be broken down into such a vector representation.

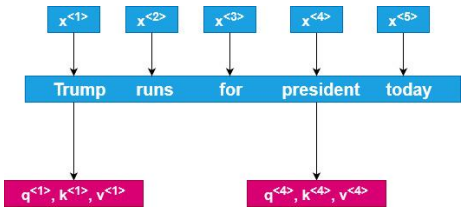

Figure 3: transformer vector representation

If $x^{<4>}$ is the word embedding, values are calculated as learned weight matrices:

- $q^{<4>} = W^Q * x^{<4>,}$
- $k^{<4>} = W^K * x^{<4>}$
- $v^{<4>} = W^V * x^{<4>.}$

Intuitively, q represents a question, e.g.: Who is running for president? Then, $q^{<3>} * k^{<1>}$ = "Trump" would be the answer. The highest value of $q^{<3>} * k^{<x>}$ represents the best answer. Thus, we obtain an attention score as shown in figure 3.

(2) Multi-head attention simply means that we calculate attention for the same sequence multiple times. Each time, the calculation remains the same, but we use different weights. These can correspond to asking different questions:

- W_1^Q, W_1^K, W_1^V = "What is happening?"
- W_2^Q, W_2^K, W_2^V = "When?"

Hence, we have attention (W1Q, W1K, W1V) = A<1>

(And similarly for $A^{<2>,} A^{<3>}$… $A^{<5>}$) for each set of weight matrices. Thus, the Multi-head attention is a concatenation of head 1, head 2 … head 5.

A benefit of multi-head attention is that the "heads" do not depend on each other, therefore, they can be computed in parallel.

Now we can understand the transformer architecture as shown in figure 5.

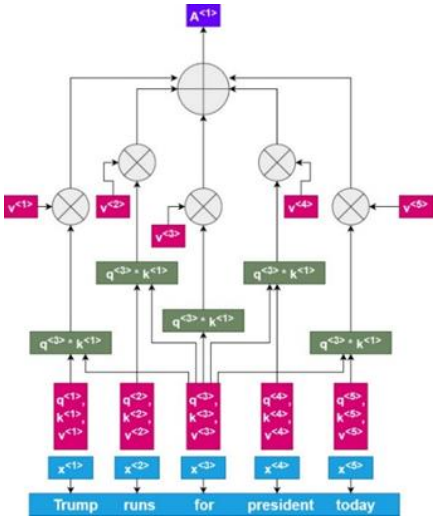

Figure 4 : calculating attention score

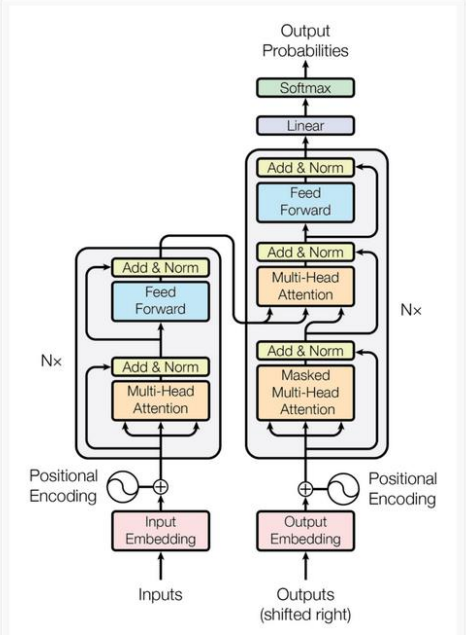

Figure 5: base transformer architecture [3]

The encoder and decoder share some common elements, namely: a fully connected neural network

consisting of a ReLU activation function, a positional encoding layer to capture the positions of words in a sequence, and a multi-head attention layer. Both the encoder and decoder have Nx layers, meaning that the output is used as input "x" times (using different wights and biases parameters at each iteration). Note that the encoder is meant to cater to all words in a sequence irrespective of their positioning. The decoder is only allowed to cater to words before the current one (as predictions only depend on previously seen words). This is done by adding a "mask" over the disallowed values so the decoder cannot use them.

Transformers are a good choice for the interpretation task as they can be trained using parallel computing and achieve performances superior to architectures employing convolutional or recurrent methods. We look further to BERT and BART as more desirable models that have improved upon the general transformer architecture.

2.3 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

A large shortcoming of the base transformer architecture is that the encoder is only capable of reading tokens from left to right. Language is complicated and dynamic, having many instances where even the whole sentence is required to understand the context of a single word. Therefore, being able to read tokens in only a single direction hampers model's ability to understand context.

The Bidirectional Encoder Representations from Transformers / BERT [4] is a model made for NLP applications using largely the same architecture as the base Transformer model.

The pre-training framework (refer to figure 6) involves gigantic text corpuses (namely English Wikipedia and the BooksCorpus - amounting to about 3.3 billion words) being applied to two tasks in equal measure: Masked Language Modelling and Next Sentence Prediction (NSP). The former concept is how BERT able to utilize bidirectional learning: It predicts a masked/hidden word (that the model sees as a "MASK" token) using the words to its left and right. The latter is how BERT learns the relationship between sentences: It must predict if a target sentence is semantically likely to follow a previous sentence. BERT has been further fine-tuned on a variety of NLP tasks with more specialized datasets.

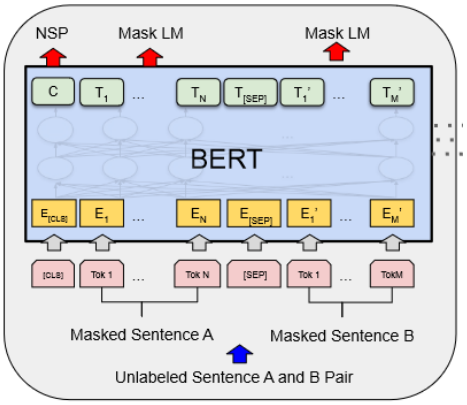

Figure 6: Bert pre-training process [4]

BERT uses a special classification "CLS" and separation "SEP" tokens, which denote the start of each of the first and second sentence in the pair respectively.

The evaluation methods consisted of using the

- Stanford Question Answering Dataset / SQuAD challenge (predict where the answer to a question lies in a passage from a Wikipedia page) measured by EM (exact match) and f1-scores.
- Situations With Adversarial Generations / SWAG (choose the most likely event given a sentence out of 4 possible options).
- General Language Understanding Evaluation / GLUE benchmark (evaluate language models on a range of tests) measured by metrics prevalent to each task.

In all cases, BERT outperformed the previous models and human performances (where applicable).

This is a powerful model for NLP applications and thus well suited be used for our lyric interpretation task.

2.4 BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension

BART [5] uses a base transformer architecture with the only change being that they use GeLU as the activation function. It is also similar to BERT in that it incorporates bidirectional learning but differs in 2 ways: (1) Cross attention is done at the last encoder hidden layer. (2) No additional feed-forward neural network before any word prediction.

Pre-training of the BART model was done using text from Wikipedia and select books and has 2 stages: The text is first transformed using various methods (see figure 7), with each transformation attempting to "teach" the model a different way of understanding context:

- Token masking: Similar to BERT, tokens are masked/hidden from the model. Purpose is to get the model to understand the context of words within a sentence.
- Token deletion: As opposed to masking, the word is removed completely without adding a MASK token. Purpose is to get the model to predict which words are missing from which positions.
- Sentence permutation: The sentences within a document are shifted around to appear in a random order. The model is allowed to learn the context sentences based on other sentences.
- Document rotation: A document is shifted such that it begins with a randomly selected token. Purpose is to get the model to predict what the document should start with.
- Text infilling: An amount of the text of a document is selected and replaced with a single MASK token. Purpose is for the model to learn how many tokens have been replaced.

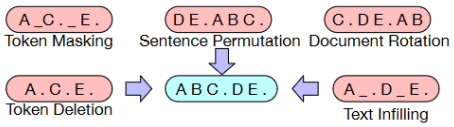

Figure 7: methods for adding noise [5]

The goal is for the model learns to undo these transformations in order to rebuild the original text. Hence why BART is referred to as a "de-noising" model. Note that, within BERT, the predictions of masked/missing tokens do not depend on each other. Therefore, it is harder to use BERT for text generation. Compared to BART, the predictions are made in an autoregressive fashion (i.e.: they can be used for generation more easily).

BART has been evaluated on several tasks:

- Outperforms BERT on the SQuAD challenge and several other GLUE tasks.
- On generative tasks such as summarization, dialogue (where the model generates responses based on a given personality) and Abstractive Question Answering, BART outperforms the previous best works.
- On translation tasks, BART was shown to be prone to overfitting, but still performed above the baseline.

BART is a PTM that is especially suited for NLP tasks because of the methods used in pre-training. It is well-adapted to comprehension tasks and text generation – which are critical elements of lyric interpretation. We can make use of the BART model fine-tuned on the CNN-Daily-Mail dataset.

2.5 PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization

The PEGASUS model [6] is also built on the base transformer architecture, but it aims to improve on the BART model in the realm of pre-training and masking. Its main usage is meant for abstractive text

summarization (where generated summaries are able to incorporate new words learned from the large corpus they are trained on).

Its pre-training is uses BERT's Masked Language Modeling (as seen) and a new method known as Gap Sentences Generation (GSP). Unlike BART, PEGASUS masks entire sentences as opposed to a masking a sequence of words within a sentence. The intuition behind this approach is that a summarizer model pre-trained as an extractive type (where a model does not generate new words in its summaries) would merely be able to copy sentences and not display a full understanding of the context.

The GSP process involves selecting the most important sentence of a document and replacing it with a single "MASK1" token. The other sentences also contain some random masking of one "MASK2" token per word as per a Masked LM approach (consider figure 8). These techniques are applied simultaneously – similarly to how BERT uses Masked LM and NSP in conjunction.

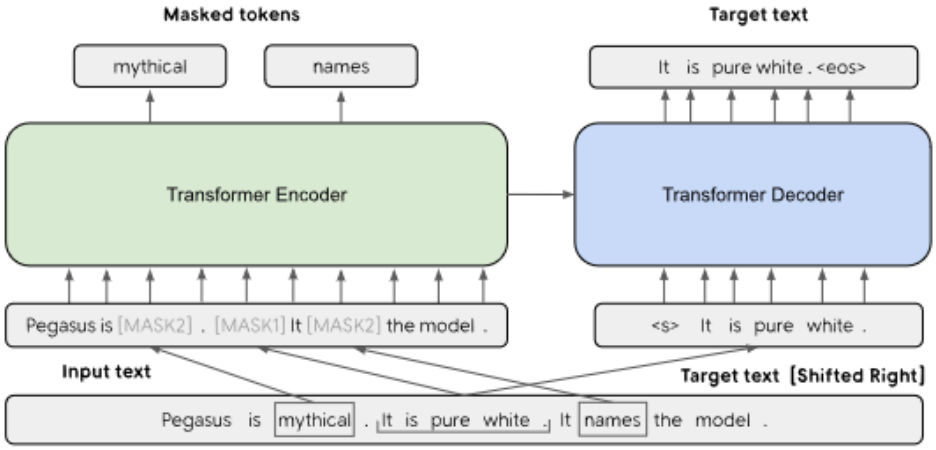

Figure 8: MLM and GSP in pre-training of PEGASUS

Pre-training of the PEGASUS base model utilizes the Colossal and Cleaned version of Common Crawl / C4 (a collection of web pages) and HugeNews (a dataset of news articles) that totals to over 4.5 TBs of data. The PEGASUS large model features more than double the amount of training parameters and further trains on several varied datasets (from the CNN-Daily-Mail to social media posts from the r/TIFU Reddit community). These extra pieces of training data added diversity in terms of length, style of writing, and general abstractive-ness.

The evaluations consisted of using ROUGE metrics and comparing them to previous state of the art / SOTA abstractive model results. Collectively, the large models outperformed the previous SOTA. "Collectively" as in 2 PEGASUS large models were used (each having been pre-trained on either the C4 or HugeNews dataset), the C4 version outperforms some SOTA and the HugeNews version outperforms others.

Interpreting lyrics can be seen as an abstractive summarization task as we are not trying to summarize the lyrics exactly, but rather get the model to associate the sentences of the comments to verses in the song. We can use the PEGASUS large model, which has

performed well at this task, as a model for our application.

2.6 INTERPRETING SONG LYRICS WITH AN AUDIO-INFORMED PRE-TRAINED LANGUAGE MODEL

To the best of my knowledge, this is the only other known paper [7] that has attempted the task of lyric interpretation. It proposes "BART-fusion", which uses a linguistic PTM and audio samples from a song. The intuition behind the use of audio is that more information can be derived about the song, such as its "emotion."

A second encoder (see Figure 9) is added to the traditional transformer model to allow for the extraction of an audio representation of the given lyrics. It is able to extract the audio features using a convolutional neural network and also incorporates a multi-head attention sub-layer similar to the base transformer architecture (with a feed forward neural network and normalization sub-layers above it). It differs in that it features multi-head "cross-attention." Introduced in [8] as an improvement on the concept using in [3]. Cross-attention is calculated the same as self-attention, but it can combine two different input sequences (i.e.: the audio and lyrics). It has also been applied to machine translation and image generation tasks, where we need to map a source to target sentence and a prompt to an image respectively.

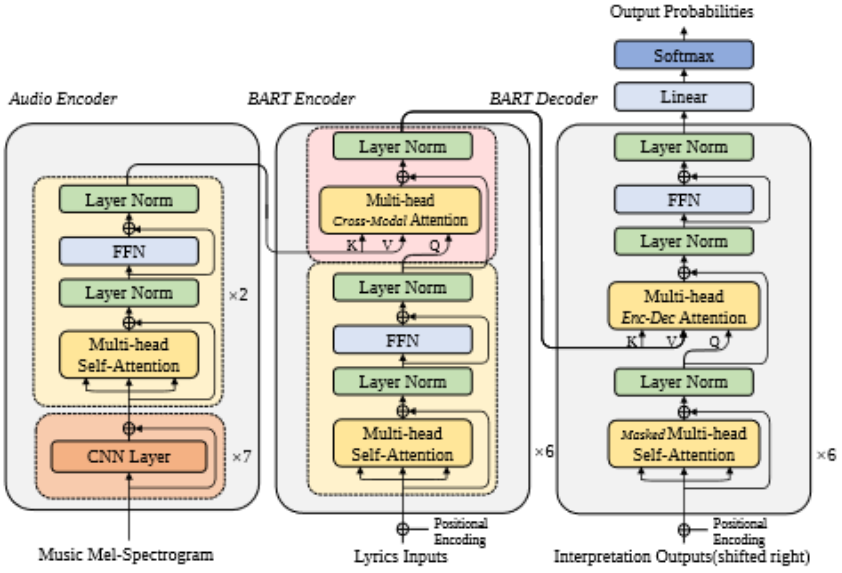

Figure 9: architecture of the BART-fusion model [7]

The other important contribution of this paper is their impressive "Song Interpretation Dataset". It contains the audio and lyrics of over 20,000 songs from a variety of genres and almost half a million human interpretations. It also includes the number of votes for each interpretation. They have preprocessed this dataset to remove short interpretations (which were often found to have nothing to do with the song) and ensure that only quality interpretations are used by taking those having the highest number of votes. For testing purposes, they

have created a smaller dataset by removing 800 rows from the original dataset. Note that, unlike the other two known song and interpretation datasets, this has been made available to the public and features much more data (approximately 10,000 and half a million more songs and interpretations respectively).

For evaluation, their BART-fusion model was compared to and outperformed the baseline BART model on the ROUGE and METEOR metrics and more or less matched it on the BERT-score. It has been noted that when the lyrics revolve around difficult subjects that carry complex undertones, both models fail to generate a real, analytical interpretation.

We can utilize the Song Interpretation Dataset and BART-model approach to song interpretation.

2.7 Evaluation metrics

There is no definitive metric for the specific task of lyric interpretation, but we can use some that are closely related, which are explored here.

2.7.1Recall-Oriented Understudy for Gisting Evaluation / ROUGE

This set of metrics [10] measure the amount of critical information retained in the generated summary when compared to a human based one. Each metric is divided into recall, precision, and f1-measures. These represent how well a generated sentence matches the target sentence, but each paints a slightly different picture. Consider the sentences "*the project was found to be a success*" and "*the project was a success*" as the generated and target/human sentences respectively. Then:

Recall is number of overlapping words divided by total words in target sentence (in our case: 5/5 = 1). Recall tells us how much of the target sentence was in our generated one but overlooks how many unnecessary words the model has added.

Precision refers to the ratio of the number of overlapping words to the total words in generated summary (we have 5/8 = 0.625). Precision indicates how much of the generated summary was relevant – which is important as summaries are intended to be short and to the point.

$$F - measure = 2 * \frac{(precision * recall)}{(precision + recall)}$$

This incorporates both recall and precision to give us a more complete view of how good the summary is.

ROUGE has several different scores. We consider ROUGE-1, ROUGE-2 (measures unigram and bigram overlap respectively) and ROUGE-L (measures the longest subsequence of words).

2.7.2 METEOR score

Mainly used for evaluating machine translation based on the idea of unigram/singular-word matching [11]. Note that these matches are calculated in various forms such as surface, stemmed and most importantly meaning (in other words, synonyms can also be matched). A singular score is then calculated from the recall, precision, and f-measure of these matches.

In the context of translation, it designed to indicate how well the word order of a translation matches the target sentence. However, we can leverage its ability to capture matches for similar words – in contrast to ROUGE, which only considers the given words.

2.7.3 BERTScore

The BERTScore [12] matches words in the generated and human sentences via a cosine similarity (note that it also utilizes recall, precision, and f-measure values). The idea is to provide an insight into as to the semantic similarity of the sentences. It has been demonstrated to be close to human-level judgement in terms of evaluation.

Having a good interpretation is not worth much if it is not a natural and fluent example of the English lexicon. Hence, we consider BERTScore as one of the main metrics – considering it equally important to those metrics evaluating the quality of the summary.

3 METHODS AND TECHNIQUES

3.1 Preparation

I have decided to use the Hugging Face platform to get the pre-trained models, namely the **facebook/bart-large-cnn** and **google/Pegasus-large** as shown in the previous section. They offer a framework to train and deploy models easily and provide an online repository from which users can store and load models from scratch or pre-trained ones (for the purpose of making the machine learning community more environmentally friendly). In the way of a community, there are numerous tutorials and forums to assist in AI development.

For the primary software, I have used python and the Google Colab environment to write Jupyter notebooks that can build and demonstrate the models. It also provides access to computing resources such as GPUs, TPUs and extra disk space. It should be noted that, while these are free, they come with an undefined usage limit that is refreshed after an (also undefined) amount of time.

Additionally, the choice between Pytorch and Tensorflow as python libraries is influenced by the Colab environment, as it comes pre-installed with some TensorFlow packages. It also allows for the use of Accelerated Linear Algebra (XLA), which massively speeds up the compilation of our transformer models.

3.2 Implementation details

With regard to preprocessing the dataset there are further steps that are taken in addition to those done by the team by BART-fusion:

1. Some of the text can contain encoding errors, which occurs when characters are meant to have a UTF-8 encoding but were decoded as another type of encoding. These can be fixed using the "fix text" method of the ftfy library [17] on the given JSON file.
2. The JSON file is then converted into a Pandas Dataframe to allow for the following easy manipulations (as I have worked with Pandas previously and am familiar with its functionality).
3. Comments often include an appended username and date such as "xxletsxrockon January 17, 2007\xa0\xa0\xa0 Link Reply" which is likely a result scraping from websites. This can be fixed by applying the following algorithm with a lambda apply function on the Dataframe:

Algorithm 1 : Clean comment

```
date pattern ← regular expression to match a date of format dd mmmm yyyy
matches ← find matches (date pattern, comment) (find all strings that match regex in comment)
 if no matches: return comment (return comment as is)
 else:
  comment array = comment.split() (create array of words from comment)
try:
  while not comment array[-1] != "on:
    pass
except:
  return comment (in the event the user intentionally included a data in their interpretation)
return comment
```

4. Lyrics that exceed 2048 words in length (prior to any tokenization) are removed.
5. We also enforce a limit on the number of rows loaded due to memory and compute restraints.
6. Lastly, we convert the Dataframe into a DatasetDict object to integrate with the Hugging Face methods and for easier management of the training and validation sets.

The PEGASUS and BART tokenizers and models are loaded using Hugging Face model checkpoints **google/pegasus-large** and **facebook/bart-large-cnn** respectively. The tokenizers are then used to process the

dataset, which is accelerated by using a data collactor, which implements batch processing (having a batch size of 2) with the function Dataset.map()

These are then used to fine-tune the loaded pre-trained models using an AdaFactor optimizer, employing a custom learning rate scheduler that changes it from 6x10-4 to 6x10-5 after 7 epochs. Thanks to XLA, the optimizer can benefit from a reduced compilation time. Every version of the model is subject to training of 10 epochs. A type of callback is used to push the model to my Hugging Face repository at the end of each epoch (does not end training early).

Testing is done using the first 100 songs and interpretations from test set of the Song Interpretation Dataset. This was created such that are no overlaps between it and the training set – ensuring that no data leakages occur (there is no chance the models observe or learn from any of the test data). To speed up the generation process, the predictions are also done using an XLA function, this time to increase the rate at which the model can generate tokens for a given input. The metrics are calculated from the resulting predictions and interpretations (which must be decoded using the tokenizer).

Note that the research teams behind BART had access to 64 TPUs and their hardware could train their models over several days. When compared to using an environment like Google Colab - although a powerful – is ultimately meant for use by students and casual members of the AI community. Thus, the results are predictably inferior to those obtained by research teams. However, it can be demonstrated that this project can achieve similar results using extrapolation. That is, I train the models on increasingly larger subsets of the dataset and show that the results and metrics are directly proportional.

4 RESULTS AND DISCUSSION

4.1 Main results

The following table shows the resulting metrics achieved by the PEGASUS and BART models (rounded to 5 decimal places). Note that only the f-measure scores are shown (as they are a more accurate representation compared to recall and precision measures).

Table 1: PEGASUS and BART metrics comparison

Songs Loaded	Model	R1	R2	RL	METEOR	BERTScore
500	P	0.01941	0.0	0.01942	0.05859	0.7522
	B	**0.21730**	**0.04721**	**0.20868**	**0.26145**	**0.8394**
1000	P	0.03448	0.00794	0.03448	0.06893	0.8012
	B	**0.17908**	**0.04118**	**0.17908**	**0.23786**	**0.8413**
1500	P	0.05313	0.00781	0.05313	0.09982	0.7948
	B	**0.26075**	**0.05235**	**0.22205**	**0.28753**	**0.8381**
2000	P	0.12327	0.03704	0.11409	0.10804	0.8089
	B	**0.23168**	**0.06490**	**0.22306**	**0.26824**	**0.8482**
2500	P	0.12183	0.03731	0.11119	0.11153	0.8088
	B	**0.24407**	**0.07073**	**0.24407**	**0.27642**	**0.8392**
3000	P	0.15942	0.03681	0.15109	0.18278	0.8305
	B	**0.24900**	**0.06452**	**0.24060**	**0.28510**	**0.8422**

At first glance, we can notice that BART outperforms the PEGASUS model across all metrics. But consider the following:

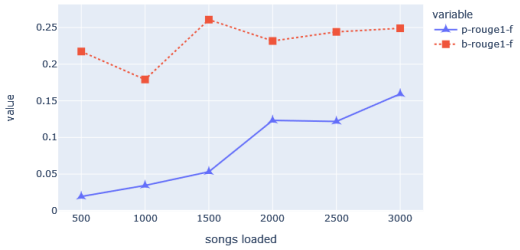

Figure 10: F-measures of PEGASUS and BART ROUGE-1 scores

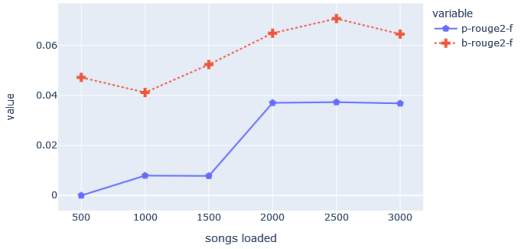

Figure 11: F-measures of PEGASUS and BART ROUGE-2 scores

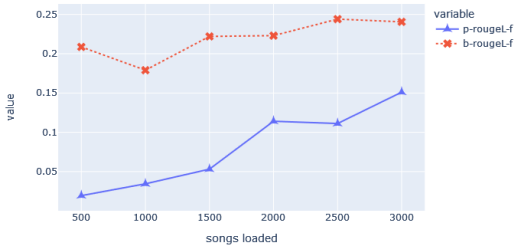

Figure 12: F-measures of PEGASUS and BART ROUGE-L scores

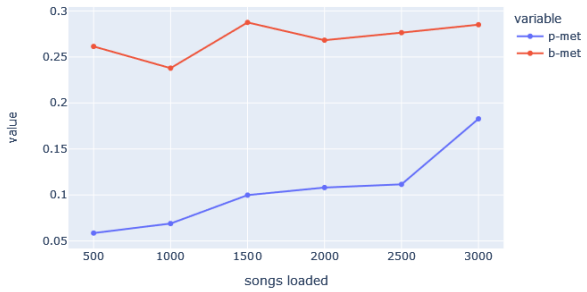

Figure 13: METEOR score of PEGASUS and BART

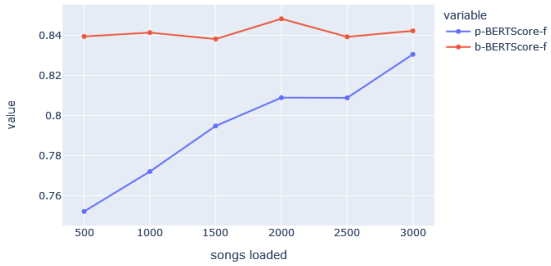

Figure 14: BERTScore of PEGASUS and BART

From Figures 10 to 14, we can observe that the PEGASUS model has a steeper gradient across all metrics. This implies that, it learns more from the increases in data compared to the BART model.

Ultimately, the models do not hold a candle to the results from BART-fusion, which outperforms us by a large margin. This includes even the BART model used as a baseline to test their BART-fusion model. This is a direct consequence of attempting a task of this magnitude at honors level (with the resources to match).

However, we need not despair. As hypothesized, we can notice a steady increase in the metrics as more data is fed into the models. The data displays evidence of bring a polynomial of degree one, i.e.: it has a (near) linear relationship between the number of songs loaded and the metrics. If we perform extrapolation to find the values when the full dataset is loaded, we get:

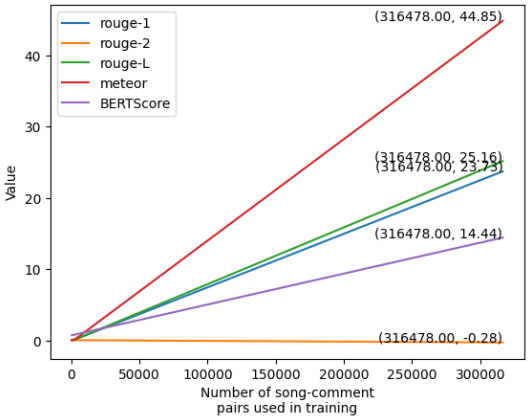

Figure 15: Extrapolating PEGASUS metrics

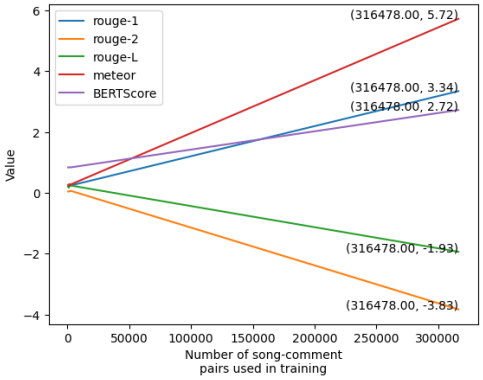

Figure 16: Extrapolating BART metrics

Trying to extrapolate the metrics when 300,000 songs are loaded from data has only 3000 is widely out of scope (observe that some metrics are decreasing as more songs are added!) and it should be noted that it is the nature of extrapolation to be less accurate the further we want to predict. Hence, these graphs and values are not accurate and do not represent any conclusive results that would have been obtained if the models were able to train fully.

5 CONCLUSIONS AND FUTURE WORK:

In this paper, I have proposed interpretating song lyrics using more advanced models than those applied by the team behind BART-fusion, by only using the lyric and comment text (whereas they included audio features). The models are pre-trained and obtained from the Hugging Face platform.

The project could not meet the standards of a typical language model one, namely due to hardware limitations. However, there is good evidence that these models can achieve comparable results in a less constricted environment.

In future work, while it is an important addition to the task of lyric interpretation, the dataset itself can be improved. Specifically, when it comes to the interpretations themselves.

As they are written by casual music enjoyers, they lack structure. Comparing these to the studies on poetry done at school level, we can note that they are much more comprehensive. They provide an in-depth view of various figures of speech and a breakdown of the lyrics at a line-by-line basis. If a dataset could be trained on poetry and its analysis, it would allow the models to gain a greater understanding by associating words to wildly different concepts that are not mentioned explicitly within the lyrics.

Similar to how the BART-fusion team has performed feature extraction on the audio, we can do the same for the lyrics. From the rhyme structure to orientation, there are many such features that can be extracted from just the text alone. The work done in [13] uses lyric features to classify music into genres – but these can be leveraged to provide models with a greater understanding of the song.

Lastly, the dataset could be expanded to include some information about the artist/s of the song themselves, as well as some information about the time period it was written in.

6 REFERENCES

[1] Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., Qiu, J., Yao, Y., Zhang, A., Zhang, L. and Han, W., 2021. Pre-trained models: Past, present and future. AI Open, 2, pp.225-250.

[2] Thrun, S. and Pratt, L., 1998. Learning to learn: Introduction and overview. In Learning to learn (pp. 3-17). Boston, MA: Springer US.

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems, 30.

[4] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[5] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. and Zettlemoyer, L., 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

[6] Zhang, J., Zhao, Y., Saleh, M. and Liu, P., 2020, November. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning* (pp. 11328-11339). PMLR.

[7] Zhang, Y., Jiang, J., Xia, G. and Dixon, S., 2022. Interpreting Song Lyrics with an Audio-Informed Pre-trained Language Model. *arXiv preprint arXiv:2208.11671*.

[8] Choi, K., Lee, J.H., Hu, X. and Downie, J.S., 2016. Music subject classification based on lyrics and user interpretations. *Proceedings of the Association for Information Science and Technology*, *53*(1), pp.1-10.

[9] Manco, I., Benetos, E., Quinton, E. and Fazekas, G., 2021, July. Muscaps: Generating captions for music audio. In *2021 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). IEEE.

[10] Lin, C.Y., 2004, July. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74-81).

[11] Banerjee, S and Lavie, A., 2005, June. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization (pp. 65-72).

[12] Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q. and Artzi, Y., 2019. Bertscore: Evaluating text generation with bert. arXiv preprint arXiv:1904.09675.

[13] [13] M. Fell, C. Sporleder, "Lyrics-based analysis and classification of music*", the 25th International Conference on Computational Linguistics: Technical Papers*, pages 620–631, Dublin, Ireland, August 23-29, 2014.

[14] Martins, R.M. and Gresse Von Wangenheim, C., 2023. Findings on teaching machine learning in high school: A ten-year systematic literature review. *Informatics in Education*, *22*(3), pp.421-440.

[15] Castaño, J., Martínez-Fernández, S., Franch, X. and Bogner, J., 2023. Exploring the Carbon Footprint of Hugging Face's ML Models: A Repository Mining Study. *arXiv preprint arXiv:2305.11164*.

[16] W. Buchanan, J Dodge, 2022. Measuring and Mitigating AI Carbon Intensity

[17] Robyn Speer (2019) "ftfy". Zenodo. doi: 10.5281/zenodo.2591652.

[18] Genius Lyrics. 2016. Retrieved from https://genius.com/11622137