Computational Creativity

Student Name: Abhinav Gudipati
Roll Number: 2019227

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering

on 20th December 2022

BTP Track: Research

BTP Advisor
Prof. Ganesh Bagler

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

I hereby declare that the work presented in the report entitled **"Computational Creativity:- Song Lyrics Generation"** submitted by me for the partial fulfilment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Applied Mathematics Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Prof. Ganesh Bagler**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

..... **Place & Date: IIIT Delhi, 12th December 2022**
Abhinav Gudipati

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

..... **Place & Date: IIIT Delhi, 16th December 2022**
Dr. Ganesh Bagler

Abstract

Text Generation has been among the growing and popular sub-fields of Natural Language Processing. In particular, invoking creativity and poetic character, subjected to constraints like style, content, and uniqueness while trying to mimic a human reference in machine-translated texts, has produced exhaustive developments in the last several years. From RNNs, LSTMs, and GAN architectures to transformer models like T5 , the computational creativity domain has made steady progress in just a matter of a couple of years. For our endeavour, we would like to present a corpus of 20 song artists' lyrics compared against their model-generated counterparts, tested for BLEU scores.

Important Keywords in Report: Computational Creativity, Songs, Lyrics, Machine translation, BLEU, natural language processing, metrics

Acknowledgments

I would like to express my convey my sincere gratitude and thanks to my advisor Professor Dr. Ganesh Bagler who has guided and supported me in the development of my BTech Project. He was always understanding of the situation and provided constructive feedback throughout the research that helped in the enhancement of the project. I learned a lot as his advisee, and it was through his mentorship and among my peers working under him that I learnt most about the way research was supposed to be conducted. I am deeply grateful to some of my seniors who've imbued within me a zeal to pursue research during the course of my BTech. I would also like to thank my friends who've helped me a lot in various different capacities towards making sure that I get tangible results in this project.

Work Distribution

- August 29th : **Registered for BTP**
- September 1st - September 27th : **Problem Statement definition and its refinement along with Literature Survey**
- September 28th - October 12th : **Dataset Overview**
- October 13th - November 14th : **Worked on Baseline Model**
- November 15th - December 7th : **Results Analysis, Exploring future plan of action.**

This report describes the work done by me during the Monsoon Semester of 2022.

Contents

Chapter 1

Introduction

1.0.1 Motivation

What differentiates and distinguishes man from other forms of life is his raw ability to create unique compositions in the various seven domains of arts like cinema, paintings, architecture, sculptures, literature, theatre, and music. What makes this even more fascinating is that arts have, from time immemorial, been the manifestation of entirely man's imagination. The foundation for us to delve into this domain of computational creativity was to train and generate art with human-like creativity. This makes it a perplexing problem to solve as we are attempting to replicate the same degree of uniqueness, novelty, and creativity that a human brain can create out of skill and imagination with an algorithmic bent from patterns learnt from previously created human art. Our motive with this endeavour to create computationally creative models is not to compete with human-like creativity but to enhance our own capabilities in exploring how we can approach art creation algorithmically.

1.0.2 Creativity in Linguistics

As human knowledge is conversed and expressed in language, creating language models is among the main research sub-directories of Computational Creativity. Freedom to explore creativity within this space is plentiful, as within a language, there are different forms of dialogue exchanged between humans that accompany with it some creative element. For example, songs lyrics, jokes, puns, rhymes, similes, analogies, metaphors, novel sentences etc., that mimic the same degree of creativity implicit within humans through learning patterns which humans find most emotionally reactive to would be useful. One of the hardest problems to tackle in this is the process of evaluating machine translations. Comparing a machine-generated text against a human reference using evaluation metrics would become insufficient because of the potential inefficiency of models to create creatively novel results that can mimic humans.
Humans themselves are predisposed to using and expressing language creatively. With novel social media norms and pop culture, there are nuances that only keep getting added to the way humans are subjectively expressing themselves.
One of the main roadblocks of language models in recent developments has been largely to invoke the creative element of creating more unique machine translations. The output generated can be repetitive and generic, and most responses might not be of interest. Hence within creative text generation, the objective has been rebranded towards creating more diverse, unique and authentic samples.

1.0.3 Pop Music

Modern pop music gate-kept by major song artists has been more of a collaborative effort of an entire troop of professional songwriters that reflected the greater interests of all the contemporary pop cultural cues. This would make artificially trained language models as this would demand a system that is subject to multiple constraints in order to generate as catchy lyrics as possible. [7]

1.0.4 Research Problem

For the current semester, I have taken Ghostwriting song lyrics of artists as the research direction. Song artists invoke several different elements of creativity like novelty, emotion, sentiment, poetic character etc. in their lyrics. With our research we are attempting to mimic a song artists' previous compositions by trying to generate alike but unique lyrics. Further, the veracity of these machine translated lyrics would be cross attested against the human reference (which the text file with all the songs of the artist) for similarity. We perform this using evaluation metrics as would be discussed further in the Evaluation Metrics section. [6]

Chapter 2

Literature Survey

2.0.1 Survey Approach

During the first half of the semester, a meticulous and detailed literature survey was done to explore the full length of all recent developments in the following sub-fields. These sub-fields were directly relevant for refining my research direction based on existing progress in this space. As previously reflected, there are seven different forms of art, and subjecting my research inclination towards one art form while knowing previous developments was critical.

- Computational Creativity
- Computational Poetry
- Text Generation
- Evaluation metrics for Machine Translated models.

2.0.2 Related Work

Computational Creativity in the text generation space has been making steady progress from recurrent neural networks (RNNs), general adversarial networks (GANs), and long short-term memory (LSTMs) all the way till recent developments in transfer learning. Transfer models like BERT, auto-regressive models, GPT-2, GPT-3, sequence-to-sequence models and BART. [7]

One of the main distinctions behind generative art models between computational poetry and those of computational lyric generation is that music lyrics are not restricted to a specific rhyme scheme constraint. For example, in previous related works in generating Shakespearean prose and poetry, more emphasis was always given towards organically capturing and generating rhyming schemes and patterns. Shakespearean poetry can be bifurcated further into sonnets, each with 14 lines. And further broken down into quatrains (four-line verses), with each quatrain having an ABAB rhyming scheme.[10] In Andrej Karpathy's blog on the Unreasonable Effec-

tiveness of Recurrent Neural Networks, he emphasizes heavily on the effectiveness of character RNNs for generating Shakespeare prose. [4]

Generative Adversarial Networks (GANs) have been widely used for text generation tasks. A GAN architecture comprises of two neural networks, a generator and a discriminator, which are both simultaneously trained in an adversarial process. Unique synthetic text is produced by the generator which is indistinguishable from the real text. While the discriminator is expected to differentiate between real and the synthetic text.

Among the very first implementations of GAN architecture within the context of text generation was the use of character-level GAN for language modelling, introduced by Zhang et al. in 2016 [11]. This approach used a convolutional neural network (CNN) as the generator and a long short-term memory (LSTM) network as the discriminator.

There have been several advances and modifications since to the basic GAN model for the pupose of text generation. One among many modifications was the use of transformer-based architecture for the generator and discriminator networks, as introduced by Yang et al. in 2019. This allows the model to capture long-range dependencies within the text and produce more coherant and diverse text. [3]

Further developements has been devoted to improving the quality of generated text and include using adversarial training objectives, such as the Wasserstein distance or the maximum mean discrepancy, and using additional loss terms, such as reconstruction loss or mutual information loss. [9]

GAN architectures are heavily restricted in text generation. This is because, within the GAN architecture, the feedback given by the discriminator belongs to the entire sequence, and most of the generations are continuous data. This makes GANs very reliable for continuous distributions like image generations, unlike in our case for language modelling, which is more discrete in nature. CreativeGAN model was implemented with the premise that a discriminator which gives feedback to the generative model through a cost function that encourages the sampling of unique creative tokens. [8] Further, RankGAN model uses a novel approach wherein, it is able to analyze and rank a collection of human-written and machine written sentences with the help of a reference group. [5]

A short summary of the kind of approaches used in recent computational text generation papers.

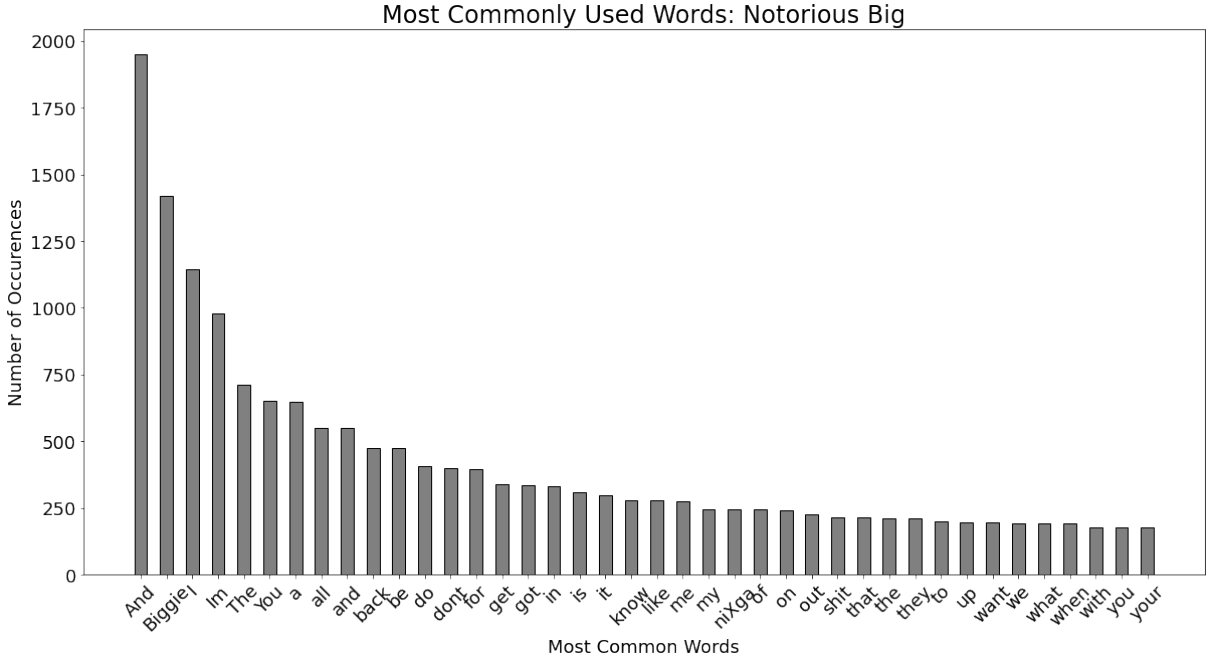
Language	**Approaches**
English	RNN, GRU, LSTM, GANs, Bi-RNN, Bi-LSTM GPT-2, GPT3, GPT-J, Transformer, BERT Seq-GAN, Tran-GAN, Rel-GAN AWD-LSTM, TransformerXL GumbleGAN, CreativeGAN

[3]

Chapter 3

Datasets and EDA

Data from already available open sources from Kaggle of popular song artists was collected. From the corpus of 48 song artists, analysis for the current semester was performed on 20 song artists. The aim is to expand our corpora of data from here towards generating more creative results by working on short formats like jokes and longer proses as well. Vocabulary Richness or Lexical richness was computed to check for the lexical diversity of the data corpus. This metric becomes particularly helpful because the NIST metric calculated better scores for those machine-translated results that had a higher degree of **lexical diversity**.



Number	Song Artist	Verses + Chorus	Unique Vocabulary	Vocabulary Richness
1	Beatles	1846	954	21.2610
2	Kanye	3799	3474	35.1674
3	Lil Wayne	3155	2104	45.8146
4	Lady Gaga	3807	1295	19.9330
5	Rihanna	3895	962	21.9726
6	Bieber	3715	1103	23.4024
7	Bob Marley	2218	1126	17.8511
8	Eminem	6812	4218	53.6201
9	Johnny Cash	1935	1575	42.7832
10	Al Green	2130	677	19.8759
11	Alicia Keys	2897	923	26.2518
12	Cake	2023	886	26.1347
13	Bruce Springsteen	2413	1468	35.6234
14	Britney Spears	3848	925	21.0929
15	Disney	2499	1753	36.6877
16	Dickinson	10053	5459	116.2324
17	Dj Khaled	5717	2731	29.3837
18	dr Seuss	1006	581	23.5274
19	Biggie smalls	5283	2582	46.9805
20	Nirvana	5717	690	20.8705

Chapter 4

Approach

To supplement my research work done on the literature survey, I have implemented a baseline LSTM-based model inspired from the Kaggle repository [1]. With respect to the style of the songs and poems given as inputs, RNNs and Markov Chains help in generating new verses. The markovify python library helps in building coherant unique sentences with respect to word1 to word2 probabilities. Further the keras.lstm functions were used to predict the properties of the follow line of the verse of the song with respect to number of syllables and its rhyme scheme. This implementation is inspired from the research approach followed in where the authors emphasized towards creating unique and original poetic text as opposed to it being semantically logical. [6] [2]

4.0.1 About Markov Chains

Python's Markovify module was applied, which replicates how Markov Chains work. Markov chains are mathematical systems which transfer from one state to the other. Within a list of states that could form all the possible states or the state space of that particular system. In addition Markov Chains also specify the probability of transitioning to another state. It is this property which helps us identify suitable words and characters to predict word1-¿word2 probability. Formally Markov Chains can be defined with the following mathematical expression. For any positive integer n and possible transitionable states of random variables.

$$P(X_n = i_n | X_{n-1} = i_{n-1}) = P(X_n = i_n | X_0 = i_0, X_1 = i_1, ...X_{n-1} = i_{n-1})$$

To simplify the intuition behind this mathematical expression, we'd need the knowledge of the previous state to predict the current state. We can address the same property being applied in the context of predicting the following word-to-word predictions. In our corpus, there would be a probability associated with each word state against other words states except its own word state. For example if the current word state is "Thou", there might be a 70 per cent probability that the following word might be "shalt" and maybe a 20 percent probability that the word is "may" and so on. Based on these probabilities of the previous word the current word state is

predicted. It is also redundant to have repeated the word state once after already being repeated as it would semantically make no sense, for which reason the probability attached to the word state against it's own word state would be 0.

4.0.2 Implementation

Most the dataset was already well-cleaned and ready to use, therefore the need to perform any additional preprocessing to the training dataset was not required. Some irrelevant words occupying the text files like [Chorus] and [Verse] which served as unnecessary noise to the generated output was cleaned.
I have run my baseline model for 10 epochs and further computed respective BLEU scores for each of the song artists. This was done by comparing the result against the human reference, as will be elaborated further in the Evaluation Metrics section. Further improvements to the baseline by parameter tuning and increasing the number of epochs as well, but the model returned the same BLEU scores. This could be owing to overfitting, when the models cannot generalise and fits too closely to the training data.
Important parameters which were included in our implementation.

- **Max syllables**:- This parameter was important to specify the maximum number of syllables per line.
- **Maximym Overlap Ratio**:- To differentiate between how similar or different the output must be to the human reference (our input).
- **tries**:- the number of times it can take to build a line of the verse.
- **number of epochs** :- The total number of iterations for the LSTM Neural Network to pass over the data. In our case I have specified with 10 epochs.

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_40 (LSTM)               (None, 2, 256)            265216
_________________________________________________________________
lstm_41 (LSTM)               (None, 2, 512)            1574912
_________________________________________________________________
lstm_42 (LSTM)               (None, 2, 512)            2099200
_________________________________________________________________
lstm_43 (LSTM)               (None, 2, 512)            2099200
_________________________________________________________________
lstm_44 (LSTM)               (None, 2, 512)            2099200
_________________________________________________________________
lstm_45 (LSTM)               (None, 2, 512)            2099200
_________________________________________________________________
lstm_46 (LSTM)               (None, 2, 512)            2099200
_________________________________________________________________
lstm_47 (LSTM)               (None, 2, 512)            2099200
_________________________________________________________________
lstm_48 (LSTM)               (None, 2, 512)            2099200
_________________________________________________________________
lstm_49 (LSTM)               (None, 2, 2)              4120
=================================================================
Total params: 16,538,648
Trainable params: 16,538,648
Non-trainable params: 0
```

A basic lstm model is comprised of a single hidden LSTM layer which is followed by a standard feedforward output layer. In my implementation, I have set an appropriate depth and increased the number the hidden LSTM layers in order to improve the performance of the model. We do this because LSTM models operate over sequential data, and in our case textual data, adding multiple layers to the model would add additional levels of abstraction.

An example of the list of 2-letter rhyme endings the model generates out of a song's text file.

```
----------------------------------------------------------------
Building list of rhymes:
List of Sorted 2-Letter Rhyme Ends:
['', "s'", 'a', 'aa', 'ba', 'ca', 'da', 'ga', 'ha', 'ia', 'ja', 'ka', 'la', 'ma', 'na',
'pa', 'ra', 'sa', 'ta', 'va', 'ya', 'za', 'ab', 'eb', 'ib', 'ob', 'ub', 'ac', 'fc', 'ic',
'oc', 'ad', 'ed', 'id', 'ld', 'nd', 'od', 'rd', 'be', 'ce', 'de', 'ee', 'fe', 'ge', 'he',
'ie', 'ke', 'le', 'me', 'ne', 'oe', 'pe', 're', 'se', 'te', 've', 'ye', 'ze', 'af', 'ef',
'ff', 'if', 'lf', 'of', 'ag', 'eg', 'gg', 'ig', 'ng', 'og', 'rg', 'ug', 'ah', 'ch', 'gh',
'hh', 'oh', 'ph', 'sh', 'th', 'uh', 'ai', 'ci', 'gi', 'ii', 'ki', 'li', 'mi', 'ni', 'ri',
'ti', 'zi', 'pj', 'tj', 'ck', 'ek', 'lk', 'nk', 'ok', 'rk', 'sk', 'al', 'el', 'hl', 'il',
'll', 'ol', 'rl', 'tl', 'ul', 'xl', 'am', 'bm', 'em', 'hm', 'im', 'lm', 'mm', 'om', 'rm',
'sm', 'um', 'ëm', 'an', 'en', 'gn', 'in', 'mn', 'nn', 'on', 'rn', 'un', 'wn', 'yn', 'ao',
'co', 'do', 'eo', 'fo', 'go', 'io', 'ko', 'lo', 'mo', 'no', 'oo', 'po', 'ro', 'to', 'vo',
'zo', 'ap', 'ep', 'ip', 'lp', 'mp', 'op', 'pp', 'rp', 'up', 'ar', 'cr', 'er', 'ir', 'jr',
'or', 'pr', 'rr', 'ur', "'s", 'as', 'bs', 'cs', 'ds', 'es', 'gs', 'hs', 'is', 'js', 'ks',
'ls', 'ms', 'ns', 'os', 'ps', 'rs', 'ss', 'ts', 'us', 'vs', 'ws', 'ys', 'ös', 'at', 'ct',
'et', 'ft', 'ht', 'it', 'lt', 'nt', 'ot', 'pt', 'rt', 'st', 'tt', 'ut', 'au', 'cu', 'ru',
'uu', 'av', 'lv', 'ov', 'ew', 'lw', 'ow', 'ww', 'ax', 'ex', 'ox', 'ay', 'by', 'cy', 'dy',
'ey', 'fy', 'gy', 'hy', 'ky', 'ly', 'my', 'ny', 'oy', 'ry', 'sy', 'ty', 'vy', 'zy', 'ez',
'iz', 'oz', 'tz', 'yz', 'zz', 'öâ', 'àö']
```

An example of the metrics returned after running the model. These scores were further compiled and tabulated for all song artists.

```
BLEU Scores
BLEU score with No Smoothing function -> 0.32044369350271595
BLEU score with smoothing function 1 -> 0.32044369350271595
BLEU score with smoothing function 2 -> 0.33325085978041163
BLEU score with smoothing function 3 -> 0.32044369350271595
BLEU score with smoothing function 4 -> 0.32044369350271595
BLEU score with smoothing function 5 -> 0.404834996895921
BLEU score with smoothing function 6 -> 0.3194167972390196
BLEU score with smoothing function 7 -> 0.404834996895921
NIST score -> 2.8454664110701593
GLEU score -> 0.35585585585585583
```

Chapter 5

Evaluation Metrics

5.1 Introduction

For the purpose of evaluating the machine-generated translations I have implemented the following scoring metrics to aide my analysis of the quality of machine translated text. The goal of any evaluation metric is to adjudge the quality, performance, and similarity of the machine translation in comparison to the professional human translation. Within, BLEU there exist a family of metrics which make use of various weighting schemes. As per recent text generation, BLEU metric is most reliable to adjudge the quality of machine translated text. More than 80 percent of the research studies relied on BLEU metrics for checking the veracity of the machine generated models. About 8 percent relied on other metrics like ROUGE, perplexity, and the rest on cosine similarity, diversity score and WER (word error rate).

- BLEU
- NIST
- GLEU

5.2 BLEU

For all deep learning models, BLEU has been so far the de facto standard Machine Translation evaluation metric. This is because of the following reasons,

1. Ease of computation time. It does not take much time to compute BLEU scores
2. BLEU is a language independent.
3. Compared to other evaluation methods, BLEU corresponds highly with Human level translation.

One of the pitfalls of using BLEU metric is that it correlates poorly with human translations as it computes a geometric mean of n-gram precisions on a sentence level. Say for example the geometric mean of n-gram precisions of a sentence is 0. Then the BLEU score of the entire sentence would translate to 0, no matter how many number of 1-grams or 2-grams are matched. To counter this issue, I have made use of several smoothing techniques in my analysis to achieve better scores. Let us first formally define how BLEU scores are being computed. The BLEU score is essentially a geometric mean of n-gram precisions. Let's take a translation T, compared against its reference R, BLEU is calculated with the help of precision P(N,T,R) and brevity penalty BP(T,R). Here P(N,T,R) is the geometric mean of n-gram precisions

$$BLEU(N,T,R) = P(N,T,R) \times BP(T,R)$$

$$P(N,T,R) = \left(\prod_{n=1}^{N} p_n \right)^{\frac{1}{N}}$$

$$p_n = \frac{m_n}{l_n}$$

In this, m_n denotes the number of matched n-grams with respect to our machine translation T and its corresponding reference R, further l_n is the total number of n-grams in the translation T. The brevity penalty is exercised to punish the score if the translation length $len(T)$ is shorter than the referencce length $len(R)$ as per the given equation.

$$BP(T,R) = min(1.0, exp(1 - \frac{len(R)}{len(T)}))$$

The issue with BLEU scores is that while computing the geometric mean of n-gram precisions, it is hard to deduce which particular n-gram has contributed least to the score. As per the NLTK documentation, there are seven smoothing techniques listed. As per the table, we can clearly find that the smoothing techniques from 1 to 6 have reproduced more or less the same results, while Smoothing Function 7 has produced the best results.

5.3 NIST and GLEU

- It is important to note that both BLEU and NIST metrics are based on the modified N-gram precision. [12]
- The NIST metric is derived from the same way BLEU is computed, except that it gives more weight to how unique, informative or rare the word is. In BLEU there is equal weight devoted to each word, whereas in NIST, there is more weight given to that particular n-gram which has more information. This is because it is less likely to occur as frequently. [12]

- The NIST metric also exercises a different Brevity Penalty compared to BLEU.
- Since the BLEU metric was designed to compute scores for a corpus measure, it is hard for BLEU to give reliable results for single sentences. Google BLEU was developed to counter this and it performs better on sentence level comparisons and can be applied for over a corpus as well.

TABLE of Scores of various song artists and their results.

					BLEU						
Song Artist	No	1	2	3	4	5	6	7	Avg	NIST	GLEU
Beatles	0.3525	0.3525	0.3645	0.3525	0.3525	0.4411	0.3512	0.4411	0.3793	2.7147	0.3762
Kanye	0.3204	0.3204	0.3333	0.3204	0.3204	0.4048	0.3194	0.4048	0.3462	2.8455	0.3559
Lil Wayne	0.2543	0.2543	0.2695	0.2543	0.2543	0.3400	0.2530	0.3400	0.2808	2.5982	0.2982
Gaga	0.2626	0.2626	0.2626	0.2765	0.2626	0.3471	0.2615	0.3471	0.2886	2.3208	0.2936
Rihanna	0.3349	0.3349	0.3475	0.3349	0.3349	0.4229	0.3334	0.4229	0.3616	2.7213	0.3619
Bieber	0.2803	0.2803	0.2949	0.2803	0.2803	0.3657	0.2789	0.3657	0.3066	2.5243	0.3155
Marley	0.2601	0.2601	0.2740	0.2601	0.2601	0.3451	0.2589	0.3451	0.2862	2.3652	0.2928
Eminem	0.3204	0.3204	0.3337	0.3204	0.3204	0.4076	0.3190	0.4076	0.3470	2.6472	0.3495
Cash	0.2921	0.2921	0.3046	0.2921	0.2921	0.3787	0.2909	0.3787	0.3184	2.4515	0.3186
Al Green	0.2531	0.2531	0.2683	0.2531	0.2531	0.3384	0.2516	0.3384	0.2794	2.5195	0.2944
Keys	0.0937	0.0937	0.1118	0.0937	0.0937	0.1718	0.0937	0.1718	0.1186	0.8680	0.1195
Cake	0.2891	0.2891	0.3040	0.2891	0.2891	0.3742	0.2878	0.3742	0.3154	2.5295	0.3232
Bruce	0.2537	0.2537	0.2658	0.2537	0.2537	0.3387	0.2530	0.3387	0.2796	2.1326	0.2800
Spears	0.3135	0.3135	0.3247	0.3135	0.3135	0.4018	0.3122	0.4018	0.3401	2.5019	0.3361
Disney	0.3204	0.3204	0.3337	0.3204	0.3204	0.4076	0.3190	0.4076	0.3470	2.6809	0.3495
Dickinson	0.3618	0.3618	0.3732	0.3618	0.3618	0.4514	0.3602	0.4514	0.3888	2.8196	0.3853
Dj Khaled	0.3036	0.3036	0.3170	0.3036	0.3036	0.3898	0.3029	0.3898	0.3300	2.7596	0.3394
dr Seuss	0.2881	0.2881	0.3016	0.2881	0.2881	0.3739	0.2869	0.3739	0.3144	2.3944	0.3178
Biggie	0.2151	0.2151	0.2301	0.2151	0.2151	0.3019	0.2143	0.3019	0.2419	2.1974	0.2564
Nirvana	0.3349	0.3349	0.3475	0.3349	0.3349	0.4229	0.3334	0.4229	0.3616	2.7213	0.3619

Chapter 6

Future Work

- Fine-tune with respect to our input data and implement pre-trained models like GPT-2, GPT-JT(open-sourced version of GPT-3), and GPT-3 and compare the results of each.
- On the evaluation metrics, introduce and interpret ROUGE precision scores, as well as perplexity scores to analyse the machine-translated text.
- Improve and expand the corpus of data if possible, to more creative domains like jokes and puns. Or maybe improve the corpus of data within song lyrics generation itself. As of now, we have data on song lyrics for 48 artists. I have done analysis for 20 song artists for this semester.

Bibliography

[1] Poetry generator rnn markov. *https://www.kaggle.com/code/paultimothymooney/poetry-generator-rnn-markov*.

[2] Robbie Barrat. https://github.com/robbiebarrat/rapping-neural-network/blob/master/model.py.

[3] Noureen Fatima, Ali Shariq Imran, Zenun Kastrati, Sher Muhammad Daudpota, and Abdullah Soomro. A systematic literature review on text generation using deep neural network models. *IEEE Access*, 10:53490–53503, 2022.

[4] Andrej Karpathy. http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[5] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation.

[6] Peter Potash, Alexey Romanov, and Anna Rumshisky. Evaluating creative language generation: The case of rap lyric ghostwriting. 12 2016.

[7] Naveen Ram, Tanay Gummadi, Rahul Bhethanabotla, Richard J. Savery, and Gil Weinberg. Say what? collaborative pop lyric generation using multitask transfer learning. pages 165–173. Association for Computing Machinery, Inc, 11 2021.

[8] Asir Saeed, Suzana Ilić, and Eva Zangerle. Creative gans for generating poems, lyrics, and metaphors. 9 2019.

[9] Sakib Shahriar. Gan computers generate arts? a survey on visual arts, music, and literary text generation using generative adversarial network.

[10] Stanley Xie, Ruchir Rastogi, and Max Chang. Deep poetry: Word-level and character-level language models for shakespearean sonnet generation.

[11] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. 12 2016.

[12] Ying Zhang, Stephan Vogel, and Alex Waibel. Interpreting bleu/nist scores: How much improvement do we need to have a better system?