Natural Language Generation Proposal - Lil A.I.

Mike Nelhams
oa18502@bristol.ac.uk

Joshua Ukairo Stevens
tx18073@bristol.ac.uk

Matthew Forster
zc18711@bristol.ac.uk

James Groeneweg
kr18544@bristol.ac.uk

December 15, 2020

1 Introduction to the Problem

Originating from 1970s Hip-Hop culture in the Bronx, New York [1], rap music has become an extremely popular and established genre of music. In essence, rap may be described as music that assimilates *"rhyme, rhythmic speech, and street vernacular"* [2]. Thus one may describe rap as being a form of 'street poetry'. The rhythmic structure found in rap makes the creation of new rap lyrics an interesting task and distinguishes it from other genres of music.

Our intrigue in the task stemmed from a group interest in investigating the possibilities of natural language generation using machine learning. As such, artificially generating rap lyrics seemed a natural fit.

In this paper, this proposal will be investigated and its findings presented. Firstly, existing literature will be reviewed, to look at work already done in this area and identify key successes and failures in these approaches to learn from. Next we will consider the methods taught in the Fundamentals of AI Module anticipated to be required in the modelling and some initial experimentation will also be carried out and analysed. Other ideas considered will be mentioned and a discussion of how the project has gone so far will be carried out. Finally, planned future work will be covered, showing our next steps in proceeding with this problem.

2 Literature Review

2.1 DopeLearning

The report *DopeLearning: A Computational Approach to Rap Lyrics Generation* [3] encapsulates what is defined to be two of the main features of rap: complex rhyme patterns and meaningful lyrics. The writers develop an algorithm `DeepBeat`, which in essence, is a rap lyric generator in which these outlined features are achieved.

The algorithm `DeepBeat` explored in the paper uses an *information retrieval* approach. From a database of over 104 different rap artists lyrics, accounting to over half a million lines, the algorithm finds what is termed a *relevant next line*. In doing so it is assumed that each lyric line from the database of over half a million can be pieced in a way that gives a relatively meaningful lyric structure. This involves the supposition that each line of a rap song's lyrics will contain some partial meaning. If two lines do not necessarily match up with continuous meaning, then this can also lead to what may be deemed as *computational creativity* or novelty.

The rhyme type utilised within `DeepBeat` is the *assonance* rhyme, which may be defined as: *"resemblance of sound between syllables of nearby words, arising particularly from the rhyming of two or more stressed vowels, but not consonants (e.g. sonnet, porridge), but also from the use of identical consonants with different vowels (e.g. killed, cold, culled)"* [4]. For implementing this, a rhyme density measure is used in which consonants and

spaces are ignored and vowel sequences are found. This method is built as a deep learning neural network model in which after each line is classified and they are ranked using the RankSVM algorithm to choose the next line of the rap.

The paper explores an algorithm that produces convincing rap lyrics with assonance rhyme structure and often a conception of meaning. As assonance rhyme is the most commonly used rhyming method within rap [5] it is unlikely that the only use of rhyme within the lyrics being assonance rhyme will be a large limitation.

The method used, however, may lack in some senses for having true novelty. As the database used counts over half a million lines of lyrics, one should not notice this, however the algorithm does not come up with truly original lines of lyrics. In order to improve the method explored, further work implementing a similar method to each word in the lyrics could be utilised.

2.2 The Case of Rap Lyrics Ghostwriting

Evaluating Creative Language Generation: The Case of Rap Lyric Ghostwriting [6] has the goal of being an artificial 'ghostwriter' or so to speak. Ghostwriting has always been prevalent in the creative industry and music is no exception. For years, major artists have had their songs written for them, but the goal is to create lyrics that the target artist would write, whilst also remaining unique.

NLG has grown over the past few years, but the difficulty has always been in evaluating the generated text because there is no 'perfect' metric for doing so (see Chris van der Lee et al. 2019 [7]). Using a wide range of different rappers and styles, the main goal of this paper is to develop a way to evaluate the generated lyrics, through fluency, coherence and style matching, ensuring that generated verses are similar in areas such as rhyme frequency and style. This is done so using both manual and automatic methods. At line level, annotators are given the task of measuring level of fluency (how fluent is the line), and coherence (how well does the line match the previous line), with different levels of fluency and coherence given different weightings. In the automated sense, generated verses are analysed against training verses to check for novelty.

The paper definitely has its strengths. The evaluation methods are very original and offer what so many NLP projects lack, a robust evaluation method that can be used in many different contexts. The authors realise the need for some way to analyse the creativity and distinctiveness of the generated text, rather than just analysing how accurate the generated text is. The wide range of different rappers also mean that there is some ability to be distinguished between the generated verses, that are in turn based off certain metrics.

However, there are certainly some drawbacks to the method. Not much consideration is given to the length of verse or line, which can be key to certain rappers. Furthermore, there could be an argument to be had about age/experience. No rapper keeps the same stylistic patterns throughout their career so there could be some way to break this up and weight certain verses. Some of the newer rappers who are coming up also use *adlibs* and this could be an interesting idea to add into a similar project. In this sense, common adlibs could be singled out and be implemented into generated verses for each artist.

2.3 Automatic Neural Lyrics and Melody Composition

Automatic Neural Lyrics and Melody Composition [8] looks at the process of generating both melody and lyrics with a neural network model consisting of a lyric generator, a lyric encoder and a melody decoder. They trained several of these encoder-decoder models to find correlations between lyrics and melodies. Overall, their model can generate lyrics and corresponding melodies from a seed lyric inputted by the user, or can generate a melody for complete lyrics the user inputs.

The proposed model included a lyrics generator which worked by predicting the next lyrical token given the previous lyrical token, starting with a seed lyric inputted by the user. A recurrent neural network (RNN) was used to do this, and a non-linear function was applied to map the network states over

time. This function can be a simple sigmoid function or a more advanced function such as a long short-term memory (LSTM) unit. The function is trained on a dataset of 12,023 MIDI files of "popular English songs" and looks at the relationship between the lyrics and the melody on the syllable, word and sentence level.

The paper uses a fairly basic of lyric inference. At each step of lyric generation the paper uses a softmax probability distribution to obtain the most probable next lyric given the previous lyrics (a user inputted token is used as the first lyric). Furthermore, a freezing function is applied to the possible lyric tokens using a controllable parameter - this introduces noise to predict more robust and diverse music and is something we could consider in our model.

Perhaps the biggest problem with this model is that there is no attention paid to the structure of the song - there are no choruses/verses etc. generated, instead just a string of words and sentences. This also means that the piece of music as a whole makes little musical sense, as the notes are tied to the syllables and words generated or inputted with no attention paid to either the structure or the key of the piece. Finally, only 11 responses to the model were gathered, which is unlikely to be representative of the wider populace.

3 General Proposed Methods

The topic of natural language processing (NLP) divides into two strongly coupled subcategories of natural language understanding (NLU) and natural language generation (NLG). The generation of coherent lyrics is a popular NLG problem, which can be tackled through the use of various different machine learning algorithms, each algorithm with its own strengths and limitations [9]. The procedure of generating convincing language is difficult to produce an optimal solution for, and it is equally difficult to introduce an algorithm that can optimally evaluate potential lyrics and determine if they are accurate or not. Subsequently, for this reason, each model only produces 'good' solutions, which is opposed to producing optimal solutions, because it is far easier to produce and evaluate 'good' solutions.

The first model is a very procedural and simplistic approach to generating natural language. A template approach can be applied to producing text, where an algorithm simply fills in the blanks with reasonable words to produce a grammatically structured rap that vaguely has a logical structure. The principle issue with this initial approach is that the template language lacks the linguistic nuances and dynamics which are commonplace in the English language [10]. Furthermore, the time complexity for templatic models increases exponentially as the number of grammatical rules are applied to attempt to increase the accuracy.

Discrete Markov models are an early attempt at state-based machine learning. A Markov chain is a graphical web, representing how likely each word is to follow from a previous word. This model predicts the following word in a sentence by using the current word and considering how common each next word is and then randomising the outcome, based on likelihoods. The word linguistic dependencies of the English language are well-described by Markov chains and they are able to produce accurate seeming short sentences, because each word is strongly related to the previous word. A powerful example of Markov models includes the early 'suggested word' predictors on smartphones [11]. Markov models are incredibly fast and take up very little space for small models, the issue however becomes that Markov models struggle to connect longer sentences together and lose more and more meaning as more words are generated. The loss of accuracy over time is due to the fact that model system has no memory of the previous words, beyond the most recent.

Recurrent neural networks (RNNs) and convolutional neural networks (CNNs) are different classes of neural networks with recurrent and convolutional layers respectively. Neural networks are modern models which attempt to mirror the complex processes which the brain uses to learn adaptively. Through weights and biases, neural networks pass input information through hidden network layers and then at the output stage, all of the weights and biases are adjusted to minimise the error. Recurrent neural network layers maintain a degree of short-term and even long-term memory for a given system and are exceptionally useful for the cate-

gory of generating rap lyrics, since there should be a continuous flow of meaning, which Markov and templatic models will be unable to track. Convolutional neural network layers are layers which extract specific meaning out of an input. They are traditionally used for image processing, since they can look for objects such as eyes or noses in a given picture. By combining CNN layers, you can classify any given object if done correctly. This can be applied to analysing rap songs for structural differences, such as a decrease in the syllable length or CNNs. RNNs will definitely feature in any neural network which plans on generating artificial language, since coherence is necessary for artificial compositions, otherwise the output would be clearly distinguishable by any human.

A further specialised denomination of recurrent neural networks are the Long-Short-Term Memory models. Mirroring RNNs, the LSTM models include a four-layer neural network and have the ability to retain information for a specified amount of time. Information comes into the neural network and the information can be lost, which is similar to the way a human brain operates, if the information is not frequently utilised. A unit named the 'Forgotten Gate' [9] will recognise the context of the sentence being generated may change and the system will quickly adapt to a new context. Subsequently, the model will exclusively track current state information and minimises the 'vanishing gradient problem' [12], therefore effectively producing a more effective and stable model.

The most cutting-edge current model focused on using the impressive accuracy of a neural network in conjunction with the pure computational power of Google's many supercomputers [13]. GPT3 is an incredible transformer model, which effectively is able to focus on every word in the sentence will almost a full grasp on the memory of each word. GPT3 has been trained rigorously on millions upon millions of text samples accessible through their search engine and acts as a two-way encoder, known as OpenAI, which can accurately reproduce paragraphs from merely a sentence description of what you want it to say. Despite this, it would be infeasible to consider using any transformer model for the proposed project, firstly due to the computational complexity requirements and secondly, because of the lack of big data that we have readily available to the public.

The aforementioned categories of models for generating natural language are not necessarily complete, but they are the primary divisions, which can be further subdivided themselves. Furthermore, the algorithms and models themselves are not mutually exclusive and can be combined to various extents to produce more effective results. It would be overly simplistic and dissimilar to machine learning to tackle a task as interesting as NLG with only a templatic model, especially with its numerous drawbacks. It is highly suggested to make use of the modern and well-researched, yet still leading recurrent neural networks, especially ones that include memory, since recalling the already used vocabulary is a prerequisite for song lyrics. Perhaps the inclusion of a structural Markov chain to decide the topic for example could even be advantageous for this machine learning model, so it is expected that many various prototype algorithms will be devised.

4 Initial Experimentation and Analysis

4.1 Pipelining and preprocessing the data

The first procedure for the report will be gathering rap lyrics data from various artists, preproccessing the data to a manageable format and then censoring the language used by the majorities of rappers. Using a collaborative GitHub repository [14] with lyrics gathered from the following websites [15] and [16], it is possible to pipeline saved text files for use with python. All of the text files have been formatted as to remove all of the unnecessary punctuation, leaving only commas and periods to maintain the expression of pauses within the lyrics and the hyphens to join certain words together. All of the upper casing within the texts have been kept, for the purpose of allowing the model to understand more fluently which words will commonly start sentences. Furthermore, there are sometimes sections of rap with 'shout rap', which will be written as exclusively upper case letters, which would be lost if the text was formatted to lower case.

The next course of action is to have the ability to censor profanities within the text. This is greatly important, because the majority of rap has various slurs and profanities which are likely to offend many given audiences. To circumvent this issue, an optional censoring processing phase is included to the main data preproccessing algorithm for censoring the language. By compiling a '.csv' file of all the profanities and their substitute, it becomes trivial to loop through the text and substitute profanities for their censored counterparts.

4.2 Rhyme Choice

One of the main features of rap lyrics is rhyme structure. The most common place being assonance rhyme [3]. Thus we may be concerned with the syllables that are within our text, this can be across words. If we generate different lines of text that may be our possible lyric lines we look to choosing the best next line by the assonance rhyme between line i and line $i + 1$. A simple way to do this is by randomly assigning one of the generated lines as line 1, we then go about choosing the best line 2 from the available lines we have generated -without choosing line 1 again. As rhyme is concerned with the end of each line and not the beginning we reverse the list of syllables that we extract. The GitHub program is only concerned with lower case syllables. As we are looking to run a pipeline on our text this should be of no concern.

Now comes the question, how do we decide what is the best assonance rhyme? Initially we experimented by ranking our next possible rhymes by the assonance rhyme throughout the whole of each possible line. This works by working through the reversed extracted syllables of each line and by ranking them using a counter where we have $+1$ if syllable[i] of line 1 is equal to syllable[i] of line i and -1 if syllable[i] of line 1 is not equal to syllable[i] of line i. As rhyme structure is mostly concerned with having an end that rhymes more than moderate rhyme throughout each line this is not optimal.

Another method looked at was by again using a counter where the counter will stop as soon as syllable[i] of line 1 is not equal to syllable[i] of line i. This method is thus purely concerned with the end of each line and possibly depending on the syllables found within the line, the final word. Lastly we may also set a number n in which our counter will only be concerned with the last n syllables of each line and run a counter as described in the first method. The advantage of this is that if there are two similar line endings that yield the same rank as with the stop counter method this should choose find the better of the two depending on the choice of n. Code for the stop count method is included on the following github [14].

5 Discussion

As a limitation to our rhyme density rank we may have that our choice of rank model (being a stop count as shown or looking at specifically the last n syllables of a line) may not choose what may be regarded as the best rhyme choice for the next line. This is mostly down to personal opinion and so we may alter this as and when we think it is needed going forward.

Alongside this there are other forms of rhyme not considered by out assonance rhyme density ranker. We may need to look to improving this. We can also improve our rank choices by including similar sounding assonance rhyme which does not include exactly the same vowels. As an example of this ***h**ei**r*** and ***b**ea**r*** have assonance rhyme but not the same vowels, such we could look to including *ei* and *ea* as the same phoneme.

The principal issue with censoring text is that using word substitutions will lose the meaning and it will lose the rhyming and phonetic information. Losing the meaning is acceptable, since it is the meaning which would cause offense, but losing the phonetics and the rhyming information will deeply impact the model accuracy and bias. A possible alternative, would be to censor using asterisks '*'. However by doing this, the meaning is still maintained, since it is possible to count the number of letters and guess which profanities have been replaced.

6 Future Work (i.e., work you intend to do in TB2)

Once our pipeline and rhyme rank systems are organised we will move onto the generation of rap

lyrics. The main ideas we plan to look and use from the course are mentioned in the Discussion on each covered unit topic section. After we have created a system that is able to generate rhyming lyrics from our chosen vocabulary we may then look to finding how we may test how 'good' our lyrics are.

One of our main questions as to how to test our lyric generator is how good lyrics are defined. Factors could include meaning, rhyme, number of phonemes and syllables in each line, and also fluency of the language. We could use statistical measures and human input to evaluate how good the lyrics are. Methods of testing are considered in the Discussion on each covered unit topic section.

As a side note to testing our lyrics, we may also look to creating a simple beat in which we use a text-to-speech generator to rap our lyrics and see how this performs.

7 Other project ideas considered

With the exciting new possibilities possible with the emergence of NLP, there was a desire to explore the possibilities of undertaking a task in NLG. Having a look at the other kinds of projects currently being examined in the field, as well as reading a number of reports to understand the potential avenues and limitations that could arise with the project. A list was put together of projects that interested the group. These included Speech to Text, a Text Simplifier, Text to Speech and Translation.

Ultimately, as a group there was an interest in the English language, and how computers and machines can take words and phrases, which it has almost no information or understanding of, and begin to learn from the way these words are used and the order and syntax. How can a computer act in a way similar to that of a human brain? This led us to decide on our current project, as our other ideas either didn't fit this criteria, or were already pretty commonplace in NLG (text to speech, text simplification etc.).

8 Discussion on each covered unit topic

When it comes to NGL, the main methods as discussed include Markov chains and RNN's. As a result, the work done on MDPs will be useful in coming up with our first prototype of the model. MDP's are an extension of Markov Chains, but we can use the same ideas in our model. Due to their lack of 'memory' of previous words, this seems like a good place to start with the model, before expanding and looking at better ways to tackle the problem.

As with most AI systems and models, the ethics of the model must be looked at. To address this and avoid inappropriate language, at the text preprocessing stage a pipeline will be introduced to remove language deemed to be 'inappropriate'. This will include swear words and other offensive language. Furthermore, there needs to be a discussion over the rights of the generated lyrics. There is an argument that the source of the data retains the rights. However, it could also be argued that the creators of the model have the rights, or maybe even the AI itself [17].

With the other topics, there may not be an obvious way to involve them in our model. However, the validation process is where these topics could be used effectively. Decision Trees are an effective way to assess the accuracy and ability of our model and it is possible to create one manually that can be used to evaluate if our model meets certain criteria. Likewise, there is the potential to use classification methods, either in a binary sense or a multi-class sense. This could be to classify if the model is 'good' or 'bad' when it comes up against actual lyrics, or to classify whether it is deemed as fluent, cohesive or rhymes.

References

[1] M. Dyson. *Know What I Mean? Reflections on Hip-Hop*, chapter Introduction. Basic Civitas Books, 2007.

[2] C. Lynette Keyes. *Rap Music and Street Consciousness*, chapter Introduction. University of Illinois Press, 2004.

[3] E. Malmi; P. Takala; H. Toivonen; T. Raiko and A. Gionis. Dopelearning: A computational approach to rap lyrics generation [online]. *KDD'16*, June 2016. [Accessed (13/11/2020)].

[4] Michael Proffitt. *The Oxford English Dictionary*. Oxford University Press, 2020.

[5] P. Edwards. *How to Rap: The Art and Science of the Hip-Hop MC*. Chicago Review Press, 2009.

[6] P. Potash; A. Romanov; A. Rumshisky. Evaluating creative language generation: The case of rap lyric ghostwriting [online]. December 2016. [Accessed (13/11/2020)].

[7] C. van der Lee; A. Gatt; E. van Miltenburg S. Wubben; E. Krahmer. Best practices for the human evaluation of automatically generated text [online]. 2019. [Accessed (13/11/2020)].

[8] G. Reddy M; Y. Yu; F. Harscoët; S. Canales; S. Tang. Automatic neural lyrics and melody composition [online]. November 2020. [Accessed (23/11/2020)].

[9] Sciforce. A comprehensive guide to natural language generation [online]. Available from: https://medium.com/sciforce/a-comprehensive-guide-to-natural-language-generation-dd63a4b6e548. [Accessed (20/11/2020)].

[10] NLG vs Templates: Levels of Sophistication in Generating Text. Ehudreiter [online]. Available from: https://ehudreiter.com/2016/12/18/nlg-vs-templates/. [Accessed (20/11/2020)].

[11] Hidden Markov Models and Spelling Correction on the iPhone. Dr. jeffrey l. popyack [online]. Available from: https://www.cs.drexel.edu/~jpopyack/Courses/AI/Wi12/notes/iPhone_SpellingCorrection.pdf. [Accessed (20/11/2020)].

[12] Chi-Feng Wang. The vanishing gradient problem [online]. Available from: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484. [Accessed (26/11/2020)].

[13] Openai api [online]. Available from: https://openai.com/blog/openai-api/. [Accessed (26/11/2020)].

[14] M. Nelhams; J. Groeneweg; J. U. Stevens; M. Forster. ai-group-project-team-jmjm [online]. Available from: https://github.com/EMAT31530/ai-group-project-Team-JMJM. [Accessed (11/12/2020)].

[15] Paul Mooney. Poetry and lyrics [online]. Available from: https://www.kaggle.com/paultimothymooney/poetry. [Accessed (11/12/2020)].

[16] ohhla. The original hip-hop archives [online]. Available from: https://www.ohhla.com/. [Accessed (11/12/2020)].

[17] P. Dhariwal; H. Jun; C. Payne; J. Wook Kim; A. Radford; I. Sutskever. Jukebox: A generative model for music [online]. [Accessed (27/11/2020)].

2