DIT869 - Deep Machine LearningProject report: Generating Metal lyrics

Nathalie Gocht
Gothenburg university
Applied data science master
Email: gusgochna@student.gu.se

Gleb Bychkov
Gothenburg University
Applied data science master
Email: gusbycgl@student.gu.se

***Abstract*—Natural Language Generation (further NLG) can be perceived as a sub-field of artificial intelligence and machine learning that is concerned with creating understandable texts in human languages. In this project, we focus on lyrics generation, particularly Metal genre generation lyrics. We used the dataset '380,000+ lyrics from MetroLyrics' provided by Kaggle, after cleaning and language detection procedure, we left with 21338 lyrics of the metal genre. We built two text generators GPT-2 and LSTM. GPT-2 as one of latestmodels created for text generation and LSTM as one of the simplest models. Then we evaluated the models using survey and BLUE score. Results show that text, generated by GPT-2 is harder to recognise for people from the real lyrics, than text generated by LSTM while the BLEU score shows partly different results than the survey.**

I. Introduction

NLG is a powerful tool to be used in various tasks in NLP. [8] Nowadays, the quality of the state-of-the-art text generators has reached the point, where developers do not post their models in free access with the fear of using this model in unethical issues. A recent example: OpenAI did post the full gpt-2 model with 1.5 billion parameters and 'smaller' with 700 million parameters, saying that they concern that the models will be used to generate fake news for example. In this project, we are implementing the 'smallest' version of gpt-2 with 117 million parameters to generate metal lyrics. Besides that, we also implement LSTM model in an attempt to compare the results of gpt2 and LSTM. In other words, trying to use LSTM as a benchmark.

II. Background theory/Related Work

NLG has a big way from the applications like generate textual weather forecasts from representations of graphical weather maps [4] and helping customer service representatives write letters for customers [2] to generating poetry and the whole texts.

Albert Gatt and Emiel Krahmer assume that Content determination (Deciding which information to include in the text under construction), text structuring (Determining in which order information will be presented in the text), sentence aggregation (Deciding which information to present in individual sentences), Lexicalisation (Finding the right words and phrases to express information), Referring expression generation (Selecting the words and phrases to identify domain objects), Linguistic realisation (Combining all words and phrases into well-formed sentences) are the main tasks of NLG. [5]

Also, Reiter and Dale [8] identified the stages of the NLG:

- Document planning: deciding what is to be said and creating an abstract document that outlines the structure of the information to be presented.
- Micro planning: generation of referring expressions, word choice, and aggregation to flesh out the document specifications.
- Realisation: converting the abstract document specifications to a real text, using domain knowledge about syntax, morphology, etc.

Following these stages, we would like to say, that on a document planning stage, we decided to create new metal lyrics, using already written ones. Microplanning and realization were materialized by using gpt-2 and lstm model.

There are also two main approaches to language generation: using templates and dynamic creation of documents. [8]

Examples of templates: Gap-Filling Approach, Scripts or Rules-Producing Text, Word-Level Grammatical Functions. Even though these models are not perceived as generative models in our modern understanding, they still play a role in the development of NLG topic. Most tasks of these models are now performed by dynamic NGL anyway.

The dynamic approach can be also divided into 2 parts: Dynamic Sentence Generation and Dynamic Document Creation or micro and macro generation. Sentence level's aim to generate the sentence with the desired grammatical structure without implicitly coding all the borderline cases, which can be counts as thousands in the language.

Document creation level task produces a document which is relevant and useful to its readers, and also well-structured as a narrative.

We could name several main models that are used in NLG: Markov chain, RNN, LSTM [6], Transformer [9].

Though these models might be pretty different in their implementations, dynamic text generation has one common way of generation. Almost all dynamic models assume that text is a sequence, so text generation in its essence is the reconstruction of the sequence: given the input (the input might be empty, the model makes the prediction and then uses the predicted + previous input as another prediction). Markov chain model assumes that each word is a 'state', given the

previous 'states', it can a word before the desired forecast, two words (bigram), three words and so on. During the training, the model is estimating transitional probabilities and generates the text according to the transitional probabilities. And this approach is still developing, but in more narrow language topics, like in the example of Steganographic Text generation. [11] Big drawback of the Markov chain model lies in the assumption that there exists a state that summarizes all the previous states and only the last state is needed for the forecast. This has consequences in identifying how many words are needed to forecast the next one and also the model has exponentially increasing number of parameters even if we want to catch short - term dependencies.

RNN has developed from classical feed-forward-NN with the extension to have not-fixed-size input and with the ability to remember the past and make decisions that are influenced by what it has learnt from the past. The vanilla RNN was introduced in 1986 by David Rumelhart [10]. The model can work with longer-term dependencies compared to Markov models, though it has a problem of 'vanishing gradients', which limits to identify 'the long-term' dependencies.

LSTM was discovered by Hochreiter and Schmidhuber in 1997 [6] and set accuracy records in multiple NLG domains. The main issue that was solved by LSTM is exponential vanishing gradient problem of CNN. The problem was solved by introducing new units, that can retain the information in memory for long periods. LSTM was modified with gated units, which further developed what now is perceived as a separate model: gated recurrent unit (2014, Kyunghyun Cho) [1] Later attention layer was developed to improve the performance. Which gradually leads us to the introduction of current 'state-of-art' NLG model - transformer.

The Transformer was proposed in the paper Attention Is All You Need [9]. To keep things simple for now (we will be rigorous in the METHOD/PROPOSED SOLUTION section, as we used gpt-2 transformer in our task), we would citate the authors of "Attention is all you need" The Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution".

III. Method/proposed solution

We decided to address the text generation by constructing and comparing two NLG models: LSTM and gpt-2 transformer. The data that we are using to train the model is publicly available data '380,000+ lyrics from MetroLyrics', provided by kaggle. In order to clean the data, we first get rid of all the songs, that have no lyrics, run the language detector to leave only English texts. And then we decided to focus only on Metal genre.

A. LSTM

As was discussed in the previous section, LSTM model is a modification of the RNN that has a memory unit to store the long-term dependencies. The compact form of the LSTM:
Input gate: $i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1})$
Forget gate: $f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1})$
Output: $o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1})$
New memory cell: $\tilde{c}_t = tanh(W^{(c)}x_t + U^{(c)}h_{t-1})$
Final memory cell: $c_t = f_t \circ c_{t-1} + i_t \circ c_t$
Final hidden state: $h_t = o_t \circ tanh(c_t)$

As an input to the model, we use word embedding, embedding size is 64, LSTM size is also 64. We run the training of the model for 10 hours on the google cloud.

B. GPT-2

This approach is probably the most successful one currently. GPT-2 is a transformer architecture based model, proposed by Vaswani et al [9]. The transformer is based solely on self-attention mechanism, dispensing with recurrence and convolutions entirely. Self-attention is used to represent 'the correlation' between words is a sentence. General formula of attention can be described as:

$$attention = softmax(f(K, Q)) * V$$

, where $K-$is a key, $Q-$ query and V is value. The $f(K, Q)$ in the proposed transformer architecture is $\frac{QK^T}{\sqrt{d_k}}$. The dot-product is scaled by d_k. Besides dot-product $f(K, Q)$ might be content-base, additive, location-base etc. So, to stay clear, the attention mechanism in the transformer computed by:

$$attention = softmax(\frac{QK^T}{\sqrt{d_k}}) * V$$

The transformer has 2 parts: encoder and decoder. The encoder computes self-attention for the source input, Decoder computes for target one. After Encoder has computed self-attention, the linear layer is applied to return Keys and Values, while decoder returns Queries for another self-attention layer. After attention layer that uses Keys and Values from encoder and Queries from decoder, the output is fed to the feed-forward Neural Network. Transformer computations can be performed in parallel, as it does not have recurrent computations.

2 types of transformers are recognised as 'state of the art' nowadays: google's BERT [3] and OpenAI's gpt [7]. The BERT model is a bidirectional transformer, while the unique feature of gpt that it combines unsupervised pre-training and supervised fine-tuning. Unsupervised performed using standard language model: given the corpus of words, the model has to predict the following word. Or maximize the likelihood function:

$$L_1(U) = \sum_i logP(u_i|u_{i-k}, ..., u_{i-1}; \theta)$$

, where $U = u_1, ..., n$ - corpus of tokens, k is the size of the context window, and the conditional probability P is modelled using a neural network with parameters θ.

After unsupervised pre-training, comes fine-tuning as supervised, unsupervised part. Another difference of gpt from the model described in Attention is all you need is that it has 12 attention layers in the decoder. Classical gpt was trained on 7000 books.

On February 14, OpenAI announced gpt-2 model, the main difference of gpt-2 from gpt is that it has a larger number of parameters (the largest version is 1.5 billion parameters) and was trained on 8 million web pages. It outperformed all previous models in NLP and NLG tasks.

GPT-2 has 4 versions with different number of parameters: 124M, 355M, 774M and 1,5B parameters. 124M model was published in February. 335M in May and 774M in August. In this project, we are using 124M parameter model. To fine-tune our Metal lyrics generator, we are using 21338 metal lyrics in an unsupervised manner for gpt-2.

IV. Results/Discussion

To evaluate both models we trained we used a survey where people had to guess if song texts were generated by our models or if they were written by a human. As a second evaluation method, we used the BLEU score.

A. BLEU score

We used the BLEU score calculation from the *nltk* package. As references, we took the lines from the file *metal_to_feed.txt*. Since lyrics have a similar form like poems where we can see a vers or even a line as a unit of meaning. Therefore we decided to use each line as a reference sentence.

For both models, LSTM and gpt-2, eleven texts were generated. Firstly, we calculated the BLEU score for each line of each generated paragraph. In the results, some lines showed a perfect matching score of 1.0, which can be explained by the number of reference sentences that were used. Taking the whole generated paragraph as the hypothesis might cause a misleading BLEU score since the hypothesis and the references sentences wouldn't be of similar length. Therefore, we take the BLEU score of each line, so that we have multiple BLEU scores for each generated text and take the mean from these scores to balance out the perfect scores of 1.0.

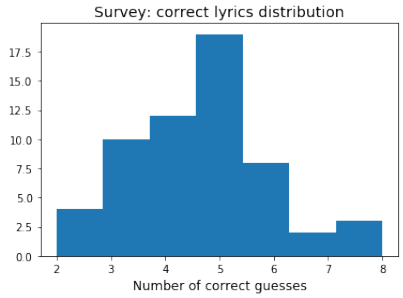
For the LSTM model, the BLEU scores range from 0.420373826 to 0.75111272, where six out of the eleven songs have a higher score than 0.6. Looking at the survey, however, even the generated paragraph with the highest BLEU score couldn't fool our participants in the survey.

For the GPT-2 model, the BLEU score ranges from 0.370900076 to 0.982321824, where five songs have a score above 0.8. The generated text with the lowest BLEU score here scored highest in our recognition survey, where the text could convince around 58% of the participants into thinking this text was written by a human. As well as in the BLEU scores of the LSTM model we see in the BLEU scores for GPT-2 that this metric only does a precision-based evaluation according to the references that are given the BLEU score. It usually doesn't correlate with the actual human perception.

B. Survey method

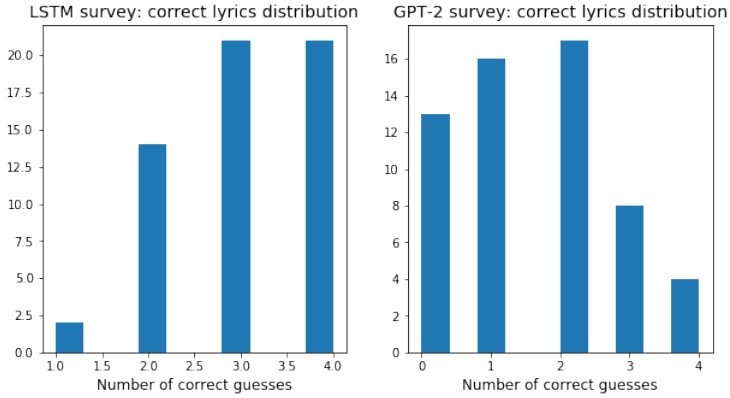
In the article by OpenAI GPT-2: 6-Month Follow-Up was said: people find GPT-2 synthetic text samples almost as convincing (72% in one cohort judged the articles to be credible) as real articles from the New York Times (83%).

We also decided to evaluate the model using survey (link). In the survey participants have to guess what is the TRUE, not model generated lyrics. Each pair starts with the same word, 4 questions have LSTM - generated fake lyrics and 4 questions have GPT-2 - generated fake lyrics. As of Thursday 24th of October 2019, 15:00 CET, We have 57 respondents. The average score of the whole quiz is 4.6, median 5.



Pic 1. Distribution of the correct answers in the whole survey

But we think that it is more insightful to plot distributions separately.



Pic 2. Distributions of the correct answers by model

As can be seen from the picture 2, the distribution of answers of the LSTM model is skewed to the right, while the distribution of answers of the gpt-2 model is skewed to the left, which implies that LSTM generated texts are easier to guess than the gpt-2 ones.

V. Conclusions

In this project, we decided to build lyrics generator, to perform the task. We used the dataset '380,000+ lyrics from MetroLyrics' provided by Kaggle, focusing on Metal lyrics, which has 21338 songs. We trained two models: LSTM and GPT-2 transformer. To evaluate the models we computed BLUE-score and used a survey to ask respondents if they can identify fake lyrics from the real ones. The survey was done by 57 people (state: Thursday 24th 3 pm), survey score for GPT-2 39%, LSTM - guessed score 76%, meaning it was harder to guess fake GPT-2 text than LSTM generated one. Therefore, the results are as expected. We took gpt-2 as one of the state-of-the-art models and even the smaller model of gpt-2 outperformed the LSTM model. We also found that

the BLEU score indeed is not as reliable as a survey with human participants. A generated vers by gpt-2 could fool our participants in $\approx 59\%$ of the time while it achieved a BLEU score of under 0.4.

It also might be the case, that our LSTM model specification might be not optimal for this particular task or it was not trained enough. The great advantage of transformers is that they can be used as transfer models, while recurrent neural networks are not transferable.

We would like to point out, that our survey has only 8 samples from both models (4 generated texts by LSTM and 4 generated text from gpt-2). It might lead to a biased estimate as we do not compare enough samples from the model. It might be a good idea to create a user-friendly interface, where the models sample new texts all the time we ask a person to evaluate the results.

Another step we would like to try is to have bigger gpt-2 models to generate the texts. As of August 2019, OpenAI published 700M parameter model in public access.

We would also like to generate folk lyrics and try other languages besides English that we have in the project.

REFERENCES

[1] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. 2014.

[2] J. Coch. Evaluating and comparing three text production techniques. *In Proceedings of COLING*, 1996.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Goldberg E., Driedgar, and Kittredge R. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9:pp. 45–56, 1994.

[5] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research (JAIR)*, 61:65–170, 2018.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[7] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018.

[8] Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 1, 1995.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[10] Williams, Ronald J., Geoffrey E. Hinton, and Rumelhart David. Learning representations by back-propagating errors. *Nature*, 323((6088)):533–536, 1986.

[11] Zhongliang Yang, Shuyu Jin, Yongfeng Huang, Yujin Zhang, and Hui Li. Automatically generate steganographic text based on markov model and huffman coding, 11 2018.