Greek Lyrics Generation

Orestis Lampridis, Athanasios Kefalas, Petros Tzallas

Greek Lyrics Generation

Orestis Lampridis[1†], Athanasios Kefalas[1†] and Petros Tzallas[1†]

[1] School of Informatics, Aristotle University of Thessaloniki,
University Campus, 54124 Thessaloniki, Greece
{lorestis, kefalasa, ptzallas}@csd.auth.gr
[†] These authors contributed equally to this work

Abstract. This paper documents the efforts in implementing lyric generation machine learning models in the Greek language for the genre of Éntekhno music. To accomplish this, we used three different Long Short-Term Memory Recurrent Neural Network approaches. The first method utilizes word-level bi-directional network models, the second method expands on the first by learning the word embeddings on the initial layer of the network, while the last method is based on a char-level network model. Our experimental procedure, which utilized a high sample of human judges, shows that texts of lyrics generated by our models are of high quality and are not that easily distinguishable from actual lyrics.

Keywords: Lyrics Generation, Natural Language Processing, Machine Learning.

1 Introduction

Natural Language Generation (NLG) is the process of generating text or speech with the use of structured data. Although precisely defining NLG has been proven difficult, a definition often used has been given in [20]. “NLG is a subfield of artificial intelligence and computational linguistics that is concerned with the construction of computer systems than can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information”.

NLG as a whole and more specifically the specialized task of creative writing is a task that humans can be quite effective at, while computational intelligence may find it rather difficult to generate creative and good quality text [4]. To this end, many attempts have been made in the literature to try and mimic human creativity, including automatic generation of poetry [5, 8, 16], metaphors [23], slogans [22] and others.

The domain of song lyrical writing has not been explored as much [17]. This problem can be more apparent due to the fact that song lyrics also have a musical aspect and may have a different style based on the genre of the song as well as secondary constraints of the task such as rhyming words and thematic tone definition.

To evaluate the performance of our models, we used human judges in order to evaluate the fidelity and credibility of our model when it comes to accurately imitating the lyrics of a real Éntekhno song. In other words, how likely it would be for the text outputted by our model to be the lyrics of an actual song.

This paper is structured as follows: In section 2, we introduce various computational approaches which have been used to successfully generate either poetry or song lyrics. In section 3, we demonstrate our dataset. In section 4, we present our different approaches to solve this problem. In section 5, we evaluate our different approaches with the help of human judges. Finally, in section 6, we offer our conclusions and discuss several future directions for our work.

2 Related Work

While the study of text generation for creative writing purposes, such as poetry and song lyrics, is of great interest in academic areas such as linguistics and music, it is also of great importance for many subfields of computer science. Relevant literature can be found in research areas such as computational creativity, information extraction, natural language processing and machine learning.

In the work of Graves [9], Recurrent Neural Networks (RNNs) [21] were used for text generation and their high effectiveness was showcased. Graves used a variation of RNNs called Long Short-Term Memory (LSTM) [10] architecture to create a language model in the character level, which has a higher success at text generation than a regular RNN model. The results are impressive, as the network created was able to learn various grammatical rules, while also being able to accurately reproduce a considerable amount of vocabulary of words in the English language.

As already mentioned, automatic text generation for poetry and song lyrics, has also been explored in the literature. Language models have been used to generate poetic text, constrained by both a target style and a predefined form. These include Markov models as in [2] and models based on Support Vector Machines (SVMs) as in [5]. As far as rap lyrics are concerned, Wu et al. [1] present a system generating rap lyrics that outputs a single sequence of rap lyrics that are a response to a particular input. A different approach was given by Malmi et al. [14] by using a Deep Neural Network (DNN) and the RankSVM [11] algorithm. By utilizing full lines of lyrics from rap songs, they created 16-line verses.

The work which is most similar to ours is that of Potash et al. [19]. Given a sequence of rap lyrics and a specific rapper, an LSTM RNN model is trained and used to predict the next word. Their goal is to produce rap lyrics that are similar in style but not identical to already existing rap lyrics; a task known as ghost writing in the music industry.

3 Dataset

The task of generating song lyrics using machine learning algorithms requires the use of a proper dataset of song lyrics for training these algorithms. The dataset was downloaded mainly from three sources, stixoi.info, greeklyrics.gr and kithara.to by using a web scraper utility program. The scraper utility targeted the URL of the song details web page of each source by injecting an id that was generated from a randomly shuffled sequence of Long type numbers. The Long datatype was selected, because Long is an appropriate type for mapping a numeric primary key id in most widely used

relational databases and therefore would have the highest probability of targeting an existing id in the source's database. The scrapper was setup to use a random back-off scheme with a 20 second minimum await time, to avoid interfering with the website's normal traffic as much as possible.

In cases where the scrapper requested a URL that was not a miss, the retrieved HTML document was queried using XPath to get the textual content of HTML DOM elements that contained the song title, lyrics, artist and other optional information such as the songs composer or lyricist. The raw textual data was then sanitized by removing special characters, duplicate whitespace characters, new line characters and symbols. Finally, after gathering an appropriately large amount of songs for the scope of our research, the dataset was exported as a csv.

The size of the raw dataset we have gathered for training the text generating language model, consisted of about 18000 entries of unprocessed songs of various artists and genres. Each song in the dataset consists of the song title and lyrics for each song and optionally in cases where it the data were available, the information of the artist, the composer, the music producer and lyricist. All entries in the raw dataset are strictly Greek songs and short poems or limericks that had been used in some traditional folk songs. The very first preprocessing step was to prune all entries in the dataset that had mixed language lyrics because the context switch between different languages along with the small number of samples with language context switches would impede the text generation model from effective learning and would negatively affect character-based models.

As different genres of music can have different styles of lyrics and specific phraseology, rhythm and even vocabulary, using the entirety of the dataset would only produce incoherent results regardless of the quality of the text generation model. For the purposes of our research, we selected songs of various artists from only the Greek Éntekhno genre. After dropping songs from all other genres the dataset was then pruned to a final size of 1150 songs.

The entries in the dataset were preprocessed to replace all whitespace characters in the lyrics of song with a space character. Furthermore, all remaining symbols and non-alphanumeric characters expect for punctuation were removed from all songs in the dataset. Additionally, alternate types of single and double quote characters were replaced by the respective standardized English counterpart. Finally, the last preprocessing step for the dataset was to correct all the disfluencies and typographical errors within the songs by hand and additionally expand words with apostrophes to their full variant form. Shortened word forms contained in the song lyrics as localisms and idioms were replaced by hand with the full variant or normalized form. Lastly, some explicit or offensive words were replaced in the corpus by alternates that had either a similar meaning or a similar phonetic rhythm or letter similarity to retain rhyme within the song.

4 Implementation

As seen in the literature, RNNs are appropriate for the modeling of natural language sequences as the RNN cells are able to retain contextual information about a sequence of tokens. As basic RNN units can present a variety of issues like the exploding or vanishing gradient problem [18], we settled on using LSTM units as the primary component of the neural networks. Sequences of word tokens may have long running dependencies in the context of a sentence that may be directed either from the beginning to the end or vice versa. The use of bidirectional LSTM units allows for the modeling of these forward and backward dependencies of a token in a given sequence. Deriving from the baseline architectural components succinctly described above we followed three different approaches that all utilize the bidirectional LSTM - RNN models and compared their results. For the first two models, we based our approach in [6, 7], while for the final model we drew influence from [15]. The code that was developed is available on our GitHub repository[1].

4.1 Word level Bi-LSTM

The RNN LSTM model used in the first approach is based on the idea of training it with a large sequence of words and then trying to predict the next word by using in a one-hot vector representation of the sequences. As a first step, we had to read our lyrics in text form, convert to lowercase, to have fewer words and split the sentences into tokens. We chose to treat the newline as an individual word. The thought process behind this is that we are giving the LSTM the capability to decide when to start a new line. On this first approach, we didn't use any further pre-processing (i.e. punctuation removal), because we wanted to see if the network could also learn from these features and apply them when creating the new lyrics.

Before building the LSTM RNN model, we split our data into 98% for the training set and 2% for the test set. Several different architectures were explored for our network. After carefully examining the quality of the resulting lyrics and taking into account metrics such as accuracy and validation accuracy we ended up on using the following architecture: The first layer in the network consisted of a bidirectional RNN layer with 256 LSTM units. The second layer was a dropout layer with dropout=0.2. Finally, there is a dense layer with softmax as activation function. The output of this layer is a vector of size equal to the number of words in the corpus which contains the probabilities for each available word in the corpus. The next word is then predicted by using the multinomial distribution to sample on these probabilities.

In order to fit the model, we had to use a data generator. We do this because in this approach we used one-hot representation of the sentences. One-hot vectorization results in vectors of 0s with a single 1 in the column of the used word. Thus, each sentence is represented by an array of size the length of the sequences times the number of unique words in our corpus. Our total training set would have size equal to the number of sentences times the length of the sequences times the number of unique words. This

[2] https://github.com/orestislampridis/Greek-Lyrics-Generation

number ends up being enormous. Using data generators, we feed the model with chunks of the training data, one for each step, instead of feeding everything at once. The generator function gets the list of sentences, the list of next words, and the size of the batch. For training the model, a shuffled set of the training sequences was used. The loss function used in this model is the categorical cross entropy function. For validation, we send another generator with the test data, so it gets evaluated every epoch. Finally, the optimizer used was adam [12].

4.2 Word level Bi-LSTM with trainable embeddings

The model used in this approach uses a trainable Embedding layer before feeding the embeddings of a word sequence to a bidirectional LSTM. The raw text of the dataset is further preprocessed by removing all punctuation from the song lyrics and all tokens were converted to lowercase. By applying case folding to lowercase, we limit the size of the vocabulary the embedding layer will have to learn. After processing the raw text of a song, the lyrics were segmented in the word level and were consequently converted to sequences of tokens. Sequences with size smaller than the pre-configured sequence size were padded with a synthetic padding token. Along with the synthetic word used for padding an additional two synthetic words were introduced to the sequences, one marking the beginning of a song and one denoting the end of a song. The inputs of the embedding layer and the model as a whole are non one-hot encoded words, meaning that instead of vectors representing words the model accepts integers that map to a specific token in a predefined vocabulary that was extracted from the dataset.

The first layer in the model's neural network was a trainable embedding layer. The size of the output vector was set to 1024 dimensions. The next layer is a bidirectional RNN layer with 256 LSTM units. The output of this layer is a collection of logit values that are intercepted by a dropout layer. The final layer is a dense or regression layer with a softmax activation. The output of this layer is a vector with a number of dimensions equal to the size of the vocabulary. The next token is predicted by using the multinomial distribution to sample on the probabilities contained in the output vector.

For training the above model a shuffled set of all available sequences was used. In contrast with the word level approach, the loss function used in this model is the sparse categorical cross entropy function that is used on sparse categorical data. For validation, the entirety of available sequences were also used, in essence forcing the model to overfit on all of the sequences, while simultaneously increasing the dropout rate to 0.2. This method essentially forces the model to overfit on the dataset but still retain some generalization when predicting sequences. In later epochs of training, some of the data in the train set should be held out to increase the generalization of the model's predictions. The validation function used was sparse categorical accuracy while the optimizer used was adamax [12] for the first few epochs and later manually switched to adam. The use of adamax in the first few epochs is a minor optimization on training because that specific optimizer is the best suited for training models with embedding layers.

4.3 Character level Bi-LSTM

In this approach, the data are left in their raw form. The vocabulary now consists of all the characters in the text data, including punctuation. Since we are working at character level there is no point in finding the most common words. Additionally, the data are not converted to lowercase in the final approach, because we noticed that the results were worse in this scenario. The only data processes are the mapping of the characters to integer numbers and the creation of sequences to feed the RNN model. The input value in each sequence is a 100-character string and the target value is simply the next character which is encoded by the One Hot Encoding method. This way we can give RNN a greater power to learn a probability for each possible target value. When a one hot encoding is used for the output variable, it may offer a finer set of predictions than a single label. The input is reshaped from a one-dimensional array to a two-dimensional array to feed the first layer of the RNN.

The model uses a trainable Embedding layer which works in the same fashion as the one in the Word level Bi-LSTM with Embeddings. It then feeds the embedding sequences of characters to the first layer of the RNN, which a bidirectional CuDNNLSTM layer with 512 nodes. The CuDNNLSTM is the same as a simple LSTM layer, the only difference is that it uses GPU for training, which makes it much faster. The hidden layer is another bidirectional CuDNNLSTM, again with 512 nodes. There is a Dropout layer set to 0.2 to avoid overfitting. Finally the output layer is a dense layer and the activation function is the softmax function as we want a distribution over the outputs. We use the adam optimize and the categorical cross-entropy loss.

Callbacks are made monitoring the train and validation accuracy. We train the model for 50 epochs. We use the validation split set to 0.2, which randomly splits up the data into a training set and test set and evaluates the model. The maximum validation accuracy achieved by this model is around 50%, which is an acceptable value for an NLG model. In the generation process we use both a random seed, sampling the sequences made before and a specific set seed. The seed is then mapped and reshaped and a prediction is made using the argmax function.

5 Evaluation

In evaluating the results of our approaches, we decided to use human judges for extrinsic evaluation of the three proposed neural language models. We were inspired by the work of Belz and Reiter [3] to set up our experimental procedure. We created a survey that contained 10 different texts in total. Out of these texts, 9 of them were from our 3 different models (3 texts for each model) and the last text was from an actual song. We did this to see how humans would rate an actual (i.e. not automatic generated) Éntekhno song. The participants didn't know that a text from an actual song would be shown. We expected this song to have the highest score and if this was the case, we would be certain about the validity of our survey. Then, we asked 76 volunteers to participate in the survey. A high percentage of them (96.1%) stated that they were familiar with the genre of Éntekhno.

The texts were presented in a random fashion and after each text, the volunteers were called to rate the accuracy of the generated lyrics on a scale of [1, 5] when it comes to imitating the lyrics of a real song. In other words, how likely it would be for the text they were reading to be the lyrics of an actual Éntekhno song. We call this accuracy the fidelity of our models. An example of the texts that were shown to the participants is shown in Table 1. To make the results more understandable to the international community, we tried to offer the best possible translation in the English language for the texts in Table 2. Note, that these translations do not accurately represent the capability of our models since each language has different ways to express creativity and our models were only trained using the Greek language. Lastly, average fidelity for each model according to the survey and after taking into account the three texts for each of our models is reported in Table 3.

Table 1. Sample texts shown to the participants during our experiment

	text shown
word_level	πως έγινε και αλλάξαμε πορεία και μένει το όνειρο ξανά στην ψυχή μου. παλίρροια, παλίρροια, μπήκες και η ζωή μου. πιες από τα μάτια μου φωτιά, και κάψε την ψυχή μου. σαν ξαφνική παλίρροια, μπήκες και τη ζωή μου. πιες από τα μάτια μου φωτιά, και κάψε την ψυχή μου. σαν ξαφνική παλίρροια, μπήκες μες τη ζωή μου
embeddings	να αράξω και τον καημό για να σε δω να σε αγκαλιάσω και θα μου πεις φοβάμαι μη σε χάσω δεν είσαι εδώ το όνειρο μου ξανά ζήσω και να φοβάμαι μη χαθείς όταν ξυπνήσω
char_level	Χρόνια γυρίζεις θεατής στου κόσμου τις κερκίδες μα πες μου τι αγκάλιασες σφιχτά από όσα είδες. Χρόνια καρδιές πολλές και σε χρόνια να παίζεις το παραθείο και στην καρδιά μου ποιημούρα αγάπη μου είπες μου πονάς τα μάτια σου κι εσύ το χρώμα της καρδιάς σου μια κορφή εκεί που πάει εσύ είσαι εγώ εγώ είσαι εσύ. Πάρε με μια καρδιά μου που σε αγαπώ σε αγαπώ το χρόνο να παίζει το φως μου και με παγιδεύεις
Actual	Από τα ίδια πάλι να γεμίσω το κεφάλι. Φέρε να πιω για να μου φύγει το βάρος γιατί απόψε θα πεθάνει ο χάρος. Μια γυναίκα με έχει κάνει το πιοτό να μη με πιάνει. Θέλησε να με πληγώσει μα πικρά θα το πληρώσει. Φέρε να πιω

Table 2. Sample texts shown to the participants translated in the English language

	text shown
word_level	how it happened and we changed course and the dream stays in my soul again. tide, tide, you came in and my life. drink fire from my eyes, and burn my soul. like a sudden tide, you came into my life. drink fire from my eyes, and burn my soul. like a sudden tide, you came into my life
embeddings	to seize the misery to see you hug you and you will tell me i'm afraid i will lose you you're not here my dream live again and i'm afraid you get lost when i wake up
char_level	For years you have been watching the stands in the world, but tell me what you hugged tightly from what you saw. Years many hearts and in years to play the paratheo and in my heart my love poem you said you hurt my eyes and you the color of your heart a peak where it goes you are me I am you. Take me with a heart that I love you I love you time to play my light and you trap me
Actual	From the same again to fill the head. Bring me to drink so that my weight will go away because tonight the Reaper will die. A woman has made me stop drinking. She wanted to hurt me but she will pay dearly. Bring me a drink

As we can see in Table 3, the word level model with pre-trained embeddings is the one with the highest fidelity, while the word level model is a close second. This is mostly because the embeddings model uses a more complex model than the word level model one, as it uses an additional embedding layer. The character level approach suffers from grammatical and syntax errors; therefore, it has been rated lower. Also, we noticed that the longer the lyrics generated the more difficult it is for each model to generate realistic lyrics.

Table 3. The average fidelity for each model

	word_level	embedings	char_level
fidelity	3.022	3.053	2.338

Looking at the highest fidelity score in Table 4, it was interesting to see that the actual song scored a fidelity of 4.092, not that close to the perfect score of 5. Also, its score was not that far ahead from the second highest scoring text, which was generated by our word level embeddings model and had a fidelity score of 3.815. The third highest scoring text was also not that far behind with a score of 3.671.

Table 4. The highest fidelity achieved by each model along with the fidelity of the actual lyrics

	word_level	embeddings	char_level	actual
fidelity	3.671	3.815	3.026	4.092

6 Conclusions and future work

With regards to future work, a larger dataset could be very beneficial. In our approach, we used a dataset of only 1150 songs, which is a compensatory number, but a larger dataset may produce better results. To this end, the Greek Music Dataset [13] could also be explored as it includes songs of the Éntekhno category. Also, even if we tried many different combinations of layers and tunings in our models, there are a lot more that can be used and bring a better output. Furthermore, an expansion to the whole process could be attempted. One idea could be to add a function to compute the style and tempo of each song and use that information to the generation process, this could produce more realistic lyrics imitating the style of actual songs. In addition, more complex features could be added to the RNN, such as the part of speech of each word, the phonetic representation of each word or the frequency of each word or character. In the character level approach, a function could be made to correct the syntax and grammatical errors or investigate if the words outputted from the network are correct by cross-checking them with a Greek dictionary of words. Finally, another approach to help with the rhyme of the song generated could be to create two RNN models, one that generates

the song from start to finish and another one that does the opposite and takes as seed the end of the previous lyric.

Concluding, we are obligated to acknowledge the difficulty of the lyrics generation task. With all the progress that is made it can only be used for inspiration by a lyricist and not for immediate commercial use. Poetry and music are arts and have a high degree of creativity that emanate from each artist's soul and it is quite difficult to train a machine learning algorithm to copy them.

Acknowledgments. We would like to express our gratitude to prof. Grigorios Tsoumakas for assigning us this project and for his valuable help and guidance. We would also like to thank all 76 volunteers for the time and effort they took to participate in the survey used for evaluating our three proposed models along with the comments and suggestions they provided about possible improvements. This work is supported by the Data and Web Science MSc program of the School of Informatics at the Aristotle University of Thessaloniki.

References

1. Addanki, Karteek, and Dekai Wu. "Unsupervised rhyme scheme identification in hip hop lyrics using hidden Markov models." *International conference on statistical language and speech processing*. Springer, Berlin, Heidelberg, 2013.
2. Barbieri, Gabriele, et al. "Markov Constraints for Generating Lyrics with Style." *Ecai*. Vol. 242. 2012.
3. Belz, Anja, and Ehud Reiter. "Comparing automatic and human evaluation of NLG systems." *11th Conference of the European Chapter of the Association for Computational Linguistics*. 2006.
4. Colton, Simon, and Geraint A. Wiggins. "Computational creativity: The final frontier?" *Ecai*. Vol. 12. 2012.
5. Das, Amitava, and Björn Gambäck. "Poetic Machine: Computational Creativity for Automatic Poetry Generation in Bengali." *ICCC*. 2014.
6. Enrique A.. Word-level LSTM text generator. Creating automatic song lyrics with Neural Networks. https://medium.com/coinmonks/word-level-lstm-text-generator-creating-automatic-song-lyrics-with-neural-networks-b8a1617104fb, Jun 2018. Accessed on 2019-27-02
7. Enrique A.. Automatic song lyrics generation with Word Embeddings. https://medium.com/coinmonks/word-level-lstm-text-generator-creating-automatic-song-lyrics-with-neural-networks-b8a1617104fb, Jun 2018. Accessed on 2019-27-02
8. Gervás, Pablo. "An expert system for the composition of formal spanish poetry." *Applications and Innovations in Intelligent Systems VIII*. Springer, London, 2001. 19-32.
9. Graves, Alex. "Generating sequences with recurrent neural networks." *arXiv preprint arXiv:1308.0850* (2013).
10. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.
11. Joachims, Thorsten. "Optimizing search engines using clickthrough data." Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. 2002.
12. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

13. Makris, Dimos, Katia Lida Kermanidis, and Ioannis Karydis. "The greek audio dataset." *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, Berlin, Heidelberg, 2014.
14. Malmi, Eric, et al. "Dopelearning: A computational approach to rap lyrics generation." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.
15. Mohammed Ma'amari. AI Generates Taylor Swift's Song Lyrics. https://towardsdatascience.com/ai-generates-taylor-swifts-song-lyrics-6fd92a03ef7e, Sep 2018. Accessed on 2019-27-02
16. Oliveira, Hugo. "Automatic generation of poetry: an overview." *Universidade de Coimbra* (2009).
17. Oliveira, Hugo Gonçalo. "Tra-la-lyrics 2.0: Automatic generation of song lyrics on a semantic domain." *Journal of Artificial General Intelligence* 6.1 (2015): 87-110.
18. Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "Understanding the exploding gradient problem." *CoRR, abs/1211.5063* 2 (2012): 417.
19. Potash, Peter, Alexey Romanov, and Anna Rumshisky. "Ghostwriter: Using an lstm for automatic rap lyric generation." *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015.
20. Reiter, Ehud, and Robert Dale. "Building applied natural language generation systems." *Natural Language Engineering* 3.1 (1997): 57-87.
21. Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *nature* 323.6088 (1986): 533-536.
22. Tomašic, Polona, M. Znidaršic, and Gregor Papa. "Implementation of a slogan generator." *Proceedings of 5th International Conference on Computational Creativity, Ljubljana, Slovenia*. Vol. 301. 2014.
23. Veale, Tony, and Yanfen Hao. "A fluid knowledge representation for understanding and generating creative metaphors." *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. 2008.