Syllable Neural Language Models for English Poem Generation

Danielle Lewis
Department of Electrical and Electronic Engineering
City, University of London
London, UK EC1V 0HB
Danielle.Lewis@city.ac.uk

Andrea Zugarini
DINFO
University of Florence
Via di S. Marta, 3 - 50139 Florence, Italy
andrea.zugarini@unifi.it

Eduardo Alonso
Department of Electrical and Electronic Engineering
City, University of London
London, UK EC1V 0HB
E.Alonso@city.ac.uk

Abstract

Automatic Poem Generation is an ambitious Natural Language Generation (NLG) problem. Models have to replicate a poem's structure, rhyme and meter, while producing creative and emotional verses. The lack of abundant poetic corpora, especially for archaic poetry, is a serious limitation for the development of strong poem generators. In this paper, we propose a syllable neural language model for the English language, focusing on the generation of verses with the style of a target author: William Wordsworth. To alleviate the problem of limited available data, we exploit transfer learning. Furthermore, we bias the generation of verses according to a combination of different scoring functions based on meter, style and grammar in order to select lines more compliant with the author's characteristics. The results of both quantitative and human evaluations show the effectiveness of our approach. In particular, human judges struggle to recognize real verses from the generated ones.

Introduction

Automatic poetry generation is an evolving area at the crossroads of computational creativity and NLG (Gatt and Krahmer 2018). NLG is a well-established sub-field of Artificial Intelligence (AI), with the goal of "generating understandable texts in a human language based on non-linguistic communication" (Ehud and Robert 2000). NLG is a challenging problem which has seen huge advances and contributions to Natural Language Processing (NLP) over the last 20 years: producing various types of texts, from biographies (Kim et al. 2002) to weather and financial forecasts (Reiter et al. 2005; Plachouras et al. 2016), as well as creative texts such as story narratives (Gervás et al. 2006), jokes (Ritchie et al. 2007) and poetry (Zugarini, Melacci, and Maggini 2019). Automatic poetry generation is an incredibly interesting topic for artificial intelligence and especially challenging due to the complex language features involved; like syntax, semantics, phonetics and lexical choice (Oliveira 2009).

Poem generation requires consideration to both content and form. Not only is poetry an expression of language, but an expression of the artist themselves. Automatically generated poetry must capture the linguistic features that can characterize a poet; the rational and semantic qualities of the poet's works, naturally influenced by their personal experiences, beliefs, and literary background.

Modern techniques for poetry generation have largely utilized neural architectures with a post-processing stage to generate well-formed verses. More often this research has used the poetical works of several authors, as opposed to a target poet, to tackle the need for large quantities of data.

(Zugarini, Melacci, and Maggini 2019) proposed a simple neural network model to generate verse in the Italian language, explicitly from the poet Dante Alighieri. What distinguished this approach from previous language models was the use of syllables as input tokens. This intuition for using sub-word information was based on the dependency of poetry using syllables to regulate form; meter and rhyme. The syllable-based approach proved successful for the Italian language, which has a rich morphology. However, it is unknown if the technique could be applied to other languages, specifically to a less phonetic language like English.

In this paper, we extend the syllable-based language model proposed in (Zugarini, Melacci, and Maggini 2019) to the case of English language for a target author, namely the romantic poet William Wordsworth. The model consists of a Recurrent Neural Network that outputs one syllable at each time step, conditioned to the previously generated text. The model is trained using William Wordsworth's work: *The Prelude*, composed in blank verse, i.e. unrhymed lines of iambic pentameters. By virtue of the syllable-based approach, the proposed model can learn several properties of the input and has large flexibility in its potential generation. To account for this, generations which resemble The Prelude and Wordsworth's style are favoured. Neural networks trained on a single author can lead to low generalization due to small training data. To combat this, a multi-transfer learning system is proposed. A transfer of information is learnt by the language model, using Wordsworth's non-poetic prose, a selection of his production of poems and lastly, the autobiographical epic, The Prelude.

Experimentation demonstrates that exploiting Wordsworth's production improves the perplexity of the language model, suggesting that the model's ability to capture the language and contents of The Prelude is enhanced. A qualitative analysis of the generated verses using human evaluation in a Turing style test was carried out. It was found that the generated verses were considered to be real by the judges, even more frequently than the genuine Wordsworth verses. To encourage further

Proceedings of the 12th International Conference on Computational Creativity (ICCC '21)
ISBN: 978-989-54160-3-5

research and the reproduction of the syllable poetry generation model, the open-source code can be downloaded at `https://gitlab.com/danielle.evalewis1/poetrygeneration_william_syl-worth`.

The paper is structured as follows. The next Section gives an overview of advances and state-of-art approaches to poetry generation. Then, we describe the proposed model and the generation mechanism, we report the results of the experiments, and finally we draw the conclusions.

Related work

According to the literature, the problem of poetry generation has been tackled often using machines which are programmed to generate poetry or approaches which utilize machine learning. Earlier methods relied on rule-based solutions, while more recent state-of-art techniques have employed learnable language models to tackle flaws of previous systems and go beyond template filling. Language Modelling predicts which word comes next, given a sequence of previous words. Neural language models have been the dominant class of algorithms applied to NLG in the last few years. Neural networks learn representations at increasing levels of abstraction through backpropagation (LeCun, Bengio, and Hinton 2015; Goodfellow et al. 2016). These representations are dense, low-dimensional and distributed, which complements the task of natural language processing by capturing grammatical and semantic generalizations (Gatt and Krahmer 2018). Whilst a feed-forward neural network has been found to be successful to address language modelling (Bengio et al. 2003), recurrent neural networks (RNN) are the much-preferred approach (Mikolov et al. 2010).

RNNs were designed to improve sequence modelling and retain information from sequences of text by introducing memory loops within the network. (Hochreiter and Schmidhuber 1997) developed a more sophisticated RNN architecture called the LSTM (Long Short-Term Memory), designed to retain information for an extended number of timesteps and as a solution to the vanishing gradient problem. Compared to other language models, LSTMs can handle various sequence lengths and avoid data sparseness as well as the explosion of the number of parameters. Several researchers have used LSTMs to produce state-of-art generation systems to generate sonnets from topics (Ghazvininejad et al. 2016), automatic rap lyrics (Potash, Romanov, and Rumshisky 2015), and target author-stylized poetry (Tikhonov and Yamshchikov 2018; Zugarini, Melacci, and Maggini 2019)).

There are many different approaches to poetry generation and the use of language models. (Lau et al. 2018) used a joint architecture consisting of a word-level language model with both word and character representations, where generations of quatrains are selected based on a pre-processing step. Another combinatory architecture using a language model and topic modelling was proposed by (Van de Cruys 2020) in which the system was exclusively trained on non-poetic text, with a post-processing step to constrain poetic verse. Word-based language models usually involve large vocabularies, storing all the most frequent words in a large textual corpus. They also cannot generalize to never-seen-before words. To overcome these issues, some approaches have exploited sub-word information: (Hwang and Sung 2017) proposed a character-level solution and (Miyamoto and Cho 2016) combined word embeddings with character-level representations. Exploiting sub-word knowledge is crucial to regulate and capture the poem's form. It has been shown in (Marra et al. 2018) that character-based models can produce powerful word and context representations, capturing both morphology and semantics. However, all of these solutions learned language models from large corpora, based on the works of multiple authors or texts. (Zugarini, Melacci, and Maggini 2019) took the intuition of sub-word information and chose syllables to tokenize text, and trained a syllable-based language model from a single Italian author. Syllables are naturally well suited for poetry, since they govern several aspects of the poem's form. As the poetical production of a single target author is commonly insufficient to train a deep neural network, they proposed a multi-stage transfer learning solution with other artist's production and a publicly available modern Italian corpora to capture syntax and grammar.

We follow this approach and extend the syllable-based language model to the English language. Whilst there are partial sets of English syllabification rules, there is no definitive set of rules to follow. Automatically splitting words into syllables is a challenging task, especially because the syllable is difficult to define. Even so, most people agree they can count how many syllables there are in each word or sentence. (Marchand, Adsett, and Damper 2007) state there is a general consensus that a syllable comprises of a 'nucleus' which is nearly always a vowel combined with zero or more preceding consonants (known as the onset) and zero or more proceeding consonants (known as the coda). However, for multisyllabic words it is difficult to define which consonants belong to which syllable. A modern alternative to English syllabification is a data-driven or corpus-based approach which tries to deduce syllabifications from previous syllabified words, using a dictionary or lexicon (Marchand, Adsett, and Damper 2007). Since (Liang 1983) formulated his TEX hyphenation algorithm, it has been a standard in the field (Adsett and Marchand 2009).

We show in the Experiments that a syllable solution can indeed be applied to the English language by generating verses with correct form, and share characteristics in the style of the target author.

The Model

The proposed model consists of two blocks: a hyphenation module and a syllable-based Language Model, which processes an input sequence of syllables.

Hyphenation Module

The hyphenation module is responsible for splitting the text into a sequence of syllables. This module is language dependent, because each language has its own rules. As discussed earlier, as opposed to Italian, English does not have a precise set of hyphenation rules. Therefore, we were unable to implement a module similar to (Zugarini, Melacci,

and Maggini 2019), and we relied instead on an implementation of (Liang 1983) algorithm using the python package hyphenate[1], that exploits a TeX approach for finding legitimate hyphenation points. Each verse in the text is converted into a sequence of syllable tokens $\mathbf{x} := x_1, \ldots, x_T$ which belong to the syllable dictionary V_{sy}. A word-separator is inserted between words in each sequence, to distinguish breaks between words <sep>, begin-of-verse <go> and end-of-verse <eov>.

Language Model

The syllable-based language model learns to estimate the conditional probability of a token given the previous tokens. At each time step t, it outputs the token in the vocabulary V_{sy} with highest probability:

$$\hat{y}_t = p_\theta(x_t | x_1, \ldots, x_{t-1}) \tag{1}$$

where θ are the network's weights and $\hat{y}_t$ indicates the syllable associated with highest probability. Each element of V_{sy} is encoded into a one-hot representation of size $|V_{sy}|$. The model learns a latent dense d-dimensional representation of each token, called syllable embedding. The sequence of syllable embeddings is provided as input to the RNN, collected row-wise in the embedding matrix, one element at each time step. As V_{sy} is the set of all syllables and special tokens, its cardinality is smaller than traditional word-based vocabularies, which means the embedding matrix has significantly less trainable parameters than word-level representations. The internal state of the RNN at time step t is indicated with $\mathbf{h}_t$, and computed as follows:

$$\mathbf{h}_t = r(\mathbf{e}_t, \mathbf{h}_{t-1}), \tag{2}$$

computed by updating the previous state $\mathbf{h}_{t-1}$ combining it with the current syllable embedding $\mathbf{e}_t$ through the recurrent cell r. In our language model, r is an LSTM cell. The hidden state is further projected with a non-linear layer, into a d-dimensional vector $\mathbf{z}_t$:

$$\mathbf{z}_t = \sigma(W\mathbf{h}_t + b), \tag{3}$$

and finally a dense layer back-projects $\mathbf{z}$ into the syllables vocabulary space ($\mathrm{R}^{|V_{sy}|}$), that followed by the softmax activation function yield the probability distribution of Equation 1:

$$\mathbf{o}_t = W'_{s_y}\mathbf{z}_t, \tag{4}$$

$$\hat{\mathbf{y}}_t = \mathtt{softmax}(\mathbf{o_t}). \tag{5}$$

The language model is trained by minimizing the cross-entropy loss function between $\hat{y}_t$ and the ground truth, i.e. the actual tokens retrieved from Wordsworth's poetry. It encourages the model to assign high probability to the observed data, pushing toward 1 the element of $\hat{y}_t$ associated to the t-th syllable of the current line in observed data. Figure 1 visualizes the structure of the language model through an example.

[1] https://pypi.org/project/hyphenate/

Multi-stage Transfer learning

Neural language models are usually trained on large textual corpora. When focusing on a single author's work, such as in the case of *The Prelude: Growth of a Poet's Mind* of William Wordsworth, a language model struggles to learn, leading to poor generalization capabilities. To alleviate the lack of resources, we adopt a multi-stage transfer learning technique to pre-train the model on additional data and then fine-tuning it on *The Prelude*. We choose this transfer learning approach to mimic the approach used in (Zugarini, Melacci, and Maggini 2019), in generating poetic text in the style of Dante's The Divine Comedy. Highlighting the ability to produce desirable results from a relatively small corpus of a single author's work by exploiting syllables. In particular, we consider a large selection of Wordsworth's poetry production (excluding The Prelude) and *The Guide through the District of the Lakes in the North of England*, a book written in prose. For simplicity, in the rest of the paper the 3 corpora are referred to as *The Prelude*, *Production* and *The Guide*, respectively. We choose Wordsworth's non-poetic text (The Guide) to grasp the syntax and grammar of the English language as well as the author's style. The 30 poems from Wordsworth's production were selected from a bibliography.

Poem Generation Mechanism

Once the language model has been trained, we can exploit it at inference time to generate verses.

Decoding. After training, new verses can be generated directly from the model. We start with h_0 set to zeros, and we feed the system with the start symbol, then we auto-regressively feed the network by sampling the next token at each step. Sampling has proven to be essential for the generation of diverse, creative and free generations (Holtzman et al. 2019). There are several different sampling strategies. We explored two popular sampling techniques: multinomial sampling with temperature (Ackley, Hinton, and Sejnowski 1985; Ficler and Goldberg 2017; Fan, Lewis, and Dauphin 2018) and top-p (Holtzman et al. 2019) sampling.Sampling with temperature regulates the crispness of the probability distribution p through a parameter t, namely temperature:

$$p(x_i | x_{<i}) = \frac{exp(h_i/t)}{\sum_{j=1}^{|V_{sy}|} exp(h_j/t)}. \tag{6}$$

Setting $t \in [0, 1]$ skews the distribution towards high probability events, which implicitly lowers the mass of the tail distribution. Top-p sampling, also known as Nucleus Sampling, was instead proposed in (Holtzman et al. 2019). Such stochastic decoding technique achieved higher quality text from neural language models than greedy search and temperature sampling. The approach avoids sampling from the tail of the probability distribution by truncating it dynamically, such that the remaining tokens contain most of the probability mass, at least more than a value P. Formally:

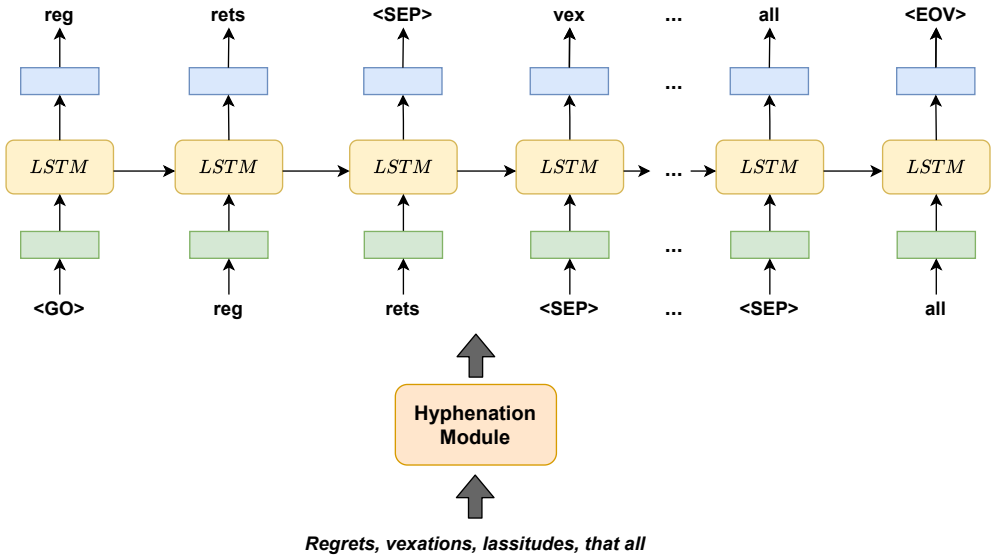

Figure 1: Sketch of the syllable-based Language Model. The hyphenation module is responsible for the tokenization of text in syllables (enriched by some special tokens that account for word separation, end of verse etc..). Green rectangles represent syllables' embeddings, yellow ones the LSTM cell unfolded through time, and the light blue blocks the computations of equations 3 and 4.

$$p'(x_i|x_{<i}) = \begin{cases} \frac{p(x_i|x_{<i})}{\sum_{x_j \in V_{sy}^{(P)}} p(x_j|x_{<j})}, & \text{if } x_i \in V_{sy}^{(P)} \\ 0, & \text{otherwise,} \end{cases} \quad (7)$$

where $V_{sy}^{(P)}$ is the set of tokens constituting the nucleus that as mass probability greater or equal to P.

We keep sampling and generating tokens until the <eov> symbol is generated, or the number of syllables reaches a fixed maximum limit. Numerous different sequences can be generated by sampling from the model's distribution learned from the training data.

Poem Selection. We generate 100 verses for each sampling strategy and we assign a score $\mathbf{S}(\mathbf{x})$ to each generated verse. $\mathbf{S}$ is an averaging of three different scoring functions: namely $\mathbf{S}_1(\mathbf{x})$, $\mathbf{S}_2(\mathbf{x})$, $\mathbf{S}_3(\mathbf{x})$, based on meter, style and grammar, respectively. To promote verses with an iambic pentameter meter, $\mathbf{S}_1$ counts the number of syllables (excluding special tokens) in a verse, as follows:

$$\mathbf{S}_1(\mathbf{x}) = abs\left(\frac{|\mathbf{x}|}{10} - 1\right), \quad (8)$$

where $|\mathbf{x}|$ indicates the number of syllables in the verse $\mathbf{x}$ and $abs(\cdot)$ is the absolute value function. In this way, verses differing from the 10 syllable target are penalized. To monitor the generation of words in the author's style, we considered the subset $V_{sy}^{(k)}$ of top-k ($k = 2000$) most frequent words (stop words excluded) used by the author in The Prelude. In the attempt to better mimic the artist's style, we measured with $\mathbf{S}_2$ the proportion of tokens in the verse that belongs to $V_{sy}^{(k)}$:

$$\mathbf{S}_2(\mathbf{x}) = abs\left(\frac{|x_i \in V_{sy}^{(k)}|}{|x_i \in V_{sy}|} - 1\right), \quad (9)$$

where $|\cdot|$ is the counting function.

Score $\mathbf{S}_3$ was used to account for grammar, in the hope of highlighting non-sensical words and repetitions within the line. Python's *LanguageTool*[2] was used to count the number of "mistakes" which appear per line, with all generations receiving at least 1 count. *LanguageTool* is a popular open source proofreading software developed by (Naber and others 2003) checking for grammar, style and spelling. The scores were normalized, resulting in values between 0 and 1 – with 0 being the ideal score.

$$\mathbf{S}_3(\mathbf{x}) = \frac{1}{|\mathbf{x}|} \sum_{x_i \in \mathbf{x}} f(x_i), \quad (10)$$

$$f(x_i) = \begin{cases} 0, & \text{if } x_i \in V \\ 1, & \text{otherwise} \end{cases} \quad (11)$$

where V is the vocabulary of words accepted by *LanguageTool*. For all three scoring functions, better performance is demonstrated with a value closer to zero. An average of the three scores was calculated for the 100 generations. The verses from the top 20 highest scores were selected for the human evaluation.

[2] https://pypi.org/project/language-check/

	Train	Test	Text style	Syllables
The Prelude	$7,134$	793	poetry	5083
Production	$17,469$	$1,941$	poetry	6990
The Guide	948	106	prose	3958

Table 1: Number of verses/sentences and syllables for each corpus.

Experiments

In the following section we report our quantitative analysis of the syllable language model generations using scoring functions, showing also the benefits of transfer learning. Further, we outline the results of the model's generations using a human evaluation in a Turing like test.

Datasets. The syllable-based language model was trained with three different corpora: *The Prelude*, *Production* and *The Guide*. Corpora statistics are outlined in Table 1.. The number of syllables are large in comparison to (Zugarini, Melacci, and Maggini 2019) which highlights the differences in the English and Italian languages. Perplexity (PPL) was measured to choose the best hyper-parameters for the neural network from numerous configurations using validation and test sets from Production. The Hyper-parameter tuning explored different learning rates (with and without decay), batch size, gradient clipping, dropout probabilities and different network sizes.

Training Details. The best performing size d for the syllable embeddings was 300 and the size of the LSTM state was $1,024$. Neurons were dropped out with probability 0.3 and the gradient was clipped at norm equal to 4. The size of V_{sy} was set to $8,000$, including all the syllables in the three datasets and the special tokens. Regarding the learning, the best results were obtained with batch size 32 and learning rate 0.002. The best parameters were used for pre-training on the *Guide* and *Production* and then fine-tuned on *The Prelude*. Learning rate was tuned to 0.0002 to achieve the best perplexity scores on *The Prelude*. After a qualitative examination of the best generated verses (i.e. the ones having the highest score $\mathbf{S}$) with both multinomial temperature sampling and top-p using different t and p parameters, we chose multinomial temperature sampling (t=0.7) to produce the verses for the human evaluation.

Transfer Learning Results. Table 2 reports the PPL results for validation and test sets from the transfer learning procedure for each dataset.

As anticipated, the language model trained on *The Prelude* benefits from pre-training on additional datasets. The most significant improvement in PPL is given when pre-training on *Production*, showing a positive transfer of information from Wordsworth's poetry production, reducing perplexity by $\sim 53\%$ and $\sim 54\%$ relative improvement in validation and test sets, respectively. The additional pre-training stage on *The Guide*, was instead not beneficial, probably because of its limited size.

Datasets	Val PPL	Test PPL
Prelude	25.36	26.27
Production$\rightarrow$ Prelude*	**11.92**	**11.80**
Guide $\rightarrow$ Production $\rightarrow$ Prelude*	17.04	18.09

Table 2: Validation and Test PPL for multi-transfer learning. $A \rightarrow B$ means that we train on data A first, and then we train on data B. * indicates that the Prelude was fine-tuned using 0.0002 as learning rate.

Author	Real-Mark
LM	56%
Poet	52%

Table 3: Percentage of participants who judged the generations to be real.

Human Evaluation. The standard evaluation metric for automatic poetry generation uses human evaluation. Human judgement was enrolled to further assess the generations. Judges were recruited in a Turing style test to judge the language model's generations compared to the author's real examples. We recruited 15 graduate students, with a mixed background. We refer to them as "non-expert" judges, since they were not specialized in Wordsworth's production, but had heard of William Wordsworth. They were asked to judge if a given verse was authored by William Wordsworth or not (i.e., generated by our language model). Each judge evaluated 10 verses, 5 of which were from Wordsworth and 5 from our model, as shown in Table 4. The example verses from Wordsworth were from The Prelude. The poem was split into verses, with 5 verses randomly chosen by an algorithm. The 5 verses representing the syllable language model were manually chosen from the top 20 scoring verses from the combined scoring functions. The same ten verses were shown to all evaluators.

Table 3 reports the number of times (percentages) that verses were judged to be authored by Wordsworth. Generated verses from the language model are considered as real 56% of the time, more so than the real examples authored by Wordsworth, with a relative difference of 7.4%.n These results match the work from (Van de Cruys 2020) where about half of the generated poems were judged to be written by a human and (Zugarini, Melacci, and Maggini 2019) who achieve a similar 56.25

Conclusion

In this paper, we extended the syllable-based approach proposed in (Zugarini, Melacci, and Maggini 2019) to the English Language. We focused on the generation of verses written in blank verse with the style of the poet William Wordsworth. Regardless of differences between the phonetic Italian language and English, the results show the method can be generalized to English, thus proving its potential applicability to other languages, even those with loose hyphenation rules. The adoption of a transfer-learning approach was crucial for alleviating the lack of textual resources necessary to train the neural language model to learn

Example	Source	Real Mark %
and when the shadow of the gentler sleep	LM	**80**
the intellect of men and hope was theirs	LM	73
endowed by nature in the midst of airs	LM	60
beneath the mountain clouds of our two cheeks	LM	47
for thy own life the errors of the first	LM	**20**
and sallying forth we journeyed side by side	Poet	80
if mid indifference and apathy	Poet	60
and from his work this moment had been stolen	Poet	53
and in our dawn of being constitute	Poet	40
even files of strangers merely seen but once	Poet	27

Table 4: Examples of verses submitted to judges, some real (Poet), some generated by our model (LM). We also report the percentage of participants who marked each verse as "real". Marks of the best and worse generated lines are highlighted in bold.

the poetry of one author. Despite its simplicity and the absence of large-scale collections of data from the target author, our model produced verses that were marked as "real" by human judges over 56% of the time. Apart from transfer learning, such performances were achieved thanks to the poem selection mechanism, which evaluated the generations for meter, style and grammar, and also due to a multi-transfer learning procedure which improves the quality of the model, exploiting a large collection of the poet's production and prose.

However, the generations produced were small in length and the quality of the generations were not evaluated further for emotion and content. Future work would plan to increase the size of generations and engage expert judges to compare results based on emotion and content qualities, beyond a Turing-style test.

References

Ackley, D. H.; Hinton, G. E.; and Sejnowski, T. J. 1985. A learning algorithm for boltzmann machines. *Cognitive science* 9(1):147–169.

Adsett, C. R., and Marchand, Y. 2009. A comparison of data-driven automatic syllabification methods. In *International Symposium on String Processing and Information Retrieval*, 174–181. Springer.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Ehud, R., and Robert, D. 2000. Building natural language generation systems.

Fan, A.; Lewis, M.; and Dauphin, Y. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Ficler, J., and Goldberg, Y. 2017. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*.

Gatt, A., and Krahmer, E. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research* 61:65–170.

Gervás, P.; Lönneker-Rodman, B.; Meister, J. C.; and Peinado, F. 2006. Narrative models: Narratology meets artificial intelligence. In *International Conference on Language Resources and Evaluation. Satellite Workshop: Toward Computational Models of Literary Analysis*, 44–51.

Ghazvininejad, M.; Shi, X.; Choi, Y.; and Knight, K. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1183–1191.

Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Holtzman, A.; Buys, J.; Du, L.; Forbes, M.; and Choi, Y. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Hwang, K., and Sung, W. 2017. Character-level language modeling with hierarchical recurrent neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5720–5724. IEEE.

Kim, S.; Alani, H.; Hall, W.; Lewis, P.; Millard, D.; Shadbolt, N.; and Weal, M. 2002. Artequakt: Generating tailored biographies from automatically annotated fragments from the web.

Lau, J. H.; Cohn, T.; Baldwin, T.; Brooke, J.; and Hammond, A. 2018. Deep-speare: A joint neural model of poetic language, meter and rhyme. *arXiv preprint arXiv:1807.03491*.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. nature (2015). *May; 521 (7553): 436 10.1038/nature14539*.

Liang, F. M. 1983. Word hy-phen-a-tion by com-put-er. Technical report, Calif. Univ. Stanford. Comput. Sci. Dept.

Marchand, Y.; Adsett, C. R.; and Damper, R. I. 2007. Evaluating automatic syllabification algorithms for english.

Marra, G.; Zugarini, A.; Melacci, S.; and Maggini, M. 2018. An unsupervised character-aware neural approach to word and context representation learning. In *International Conference on Artificial Neural Networks*, 126–136. Springer.

Mikolov, T.; Karafiát, M.; Burget, L.; Černockỳ, J.; and Khudanpur, S. 2010. Recurrent neural network based lan-

guage model. In *Eleventh annual conference of the international speech communication association*.

Miyamoto, Y., and Cho, K. 2016. Gated word-character recurrent language model. *arXiv preprint arXiv:1606.01700*.

Naber, D., et al. 2003. A rule-based style and grammar checker.

Oliveira, H. 2009. Automatic generation of poetry: an overview. *Universidade de Coimbra*.

Plachouras, V.; Smiley, C.; Bretz, H.; Taylor, O.; Leidner, J. L.; Song, D.; and Schilder, F. 2016. Interacting with financial data using natural language. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 1121–1124.

Potash, P.; Romanov, A.; and Rumshisky, A. 2015. Ghostwriter: Using an lstm for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1919–1924.

Reiter, E.; Sripada, S.; Hunter, J.; Yu, J.; and Davy, I. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence* 167(1-2):137–169.

Ritchie, G.; Manurung, R.; Pain, H.; Waller, A.; Black, R.; and O'Mara, D. 2007. A practical application of computational humour. In *Proceedings of the 4th International Joint Conference on Computational Creativity*, 91–98.

Tikhonov, A., and Yamshchikov, I. P. 2018. Guess who? multilingual approach for the automated generation of author-stylized poetry. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, 787–794. IEEE.

Van de Cruys, T. 2020. Automatic poetry generation from prosaic text. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2471–2480.

Zugarini, A.; Melacci, S.; and Maggini, M. 2019. Neural poetry: Learning to generate poems using syllables. In *International Conference on Artificial Neural Networks*, 313–325. Springer.