Poetic Press: A Machine Approach to Render News Articles in Poetic Style

Shrey Prakash Sahgal, Soo Hyun Ryu, Brandon Hardy, Chayce Baldwin
University of Michigan, Ann Arbor, MI
{shreyps,soohyunr,bkhardy,cbwin}@umich.edu

Abstract

Recent advances in natural language processing have effectively generated novel poetry with specific style and form. However, to date, models have not been developed to generate poetry while retaining the semantic content of input text. In this paper, we present the Poetic Press, which uses a state-of- the-art encoder-decoder model to summarize news articles and applies "poetic priors" to transfer news articles to the style of poetry while preserving semantic meaning. The "poetic priors" include a rhyme prior, which enforces a rhyme scheme, and word poeticness prior, which transforms the diction to be more poetic and less dry. The results show that the abstractive model provided with prior knowledge about poetry can render news articles in poetic style. This is an important advance in that it enables text style transfer even without supervised learning that requires a parallel corpus where a particular text is represented in multiple styles. The Poetic Press code can be found here: github.com/shreysahgal/news_poetry_cluster.

1 Introduction

Can the news be poetic? In this project, we aim to use machine learning approaches to render the news in poetic style.

While there is extensive work applying natural language processing (NLP) to automatically generating text [8], generating poetry has proven difficult. Unlike everyday prose, specific aesthetic form and requirements are important in poems, in addition to content. Moreover, poems may incorporate features such as rhythm, rhyme, and specific line breaks and punctuation not common in everyday prose. Recently, deep learning and generative modeling has been utilized to generate new poetry that conforms to a specific poetic form or style in English [3, 11], Chinese [19, 20], and French [16]. However, this work in poetry generation has focused almost exclusively on generating new poems with unique new content [11], rather than transferring a poetic style to already existing texts.

Why transfer poetic style to already existing texts? While AI-generated novel poetry may be interesting, we often might have a specific target that we want to be poetic, while preserving its original meaning. For example, we may want to make a public service announcement catch people's attention, get a fun new spin on famous speeches, or make a news article more interesting. Each of these applications would utilize machine learning to create poetry, but would require retaining the semantic meaning of the original text. With existing poem generation models, this sort of poetic style transfer is not possible.

There are a number of reasons why, to date, there is a lack of semantic-preserving poem generation technology. One reason may be because there are no parallel corpora that include prosaic text and the corresponding poetry that could be used as training data in a supervised model. Similarly, what text would you train on: prosaic

text or poetic text? Without both, these issues make learning a supervised model to render prosaic text in a more poetic form untenable. More practically, much pre-existing text is probably too long to effectively render matching poetry. Specific forms and restrictions in poetry (like a rhyme every number of syllables) would make transferring content directly from long texts infeasible, particularly if trying to transfer poetic style while retaining line-by-line content. These poetry-specific issues make transferring poetic style onto prosaic text a nontrivial problem.

In this project, we aimed to address these issues by developing a novel machine learning model that can render prosaic text in a poetic style. While most work generating poems has trained generative models on poetic text to create novel poems, we used a novel two-part approach that trains using only prosaic text to generate poems. In our approach, to retain the original meaning of input text, we first used an abstractive summarization encoder-decoder model that summarizes semantic content from existing text. Importantly, abstractive summarization methods can summarize the substantive semantic content of text using novel words that are not drawn directly from the input text–as the extractive summarization method does. This provides us with the ability to constrain this new, summarized text to match a poetic form and style. We then imposed priors onto the resulting summaries to constrain the text to a poetic style. Using this approach, we are able to generate poems, while retaining semantic content, even from long input text, and training exclusively on non-poetic text.

In this paper, we will use news articles to train our model and use as input. News articles are ideal content for testing this model, because they are dense with semantic meaning and also generally represent especially prosaic text. Thus, our objective in this project is as follows: Can we imbue prosaic, semantic-rich text like news articles into poetic style, while retaining their meaning?

To our knowledge, this is the first study that generates poems while preserving the semantics of prosaic input text.

2 Related Work

A few other notable papers have used recurrent neural networks (RNN) with encoder-decoder models to generate poetry. Models using encoder-decoders to generate poetry build a context vector from one line of poetry using the encoder, and then generate a new line of poetry, word-by-word, using the decoder. One of the earliest papers to do so was Zhang and Lapata [20], who used an RNN with an encoder-decoder model to generate novel Chinese poetry. Wang et al. [18] used a similar method, but improved the model by incorporating an attention mechanism into a bi-directional LSTM encoder. While these models use encoder-decoders to create hidden context vectors that are used to produce novel poetry, we use an encoder-decoder model to retain the semantic content of the input text, independent of any poem generation.

Additionally, our model is similar to two state-of-the-art encoder-decoder abstractive summarizer models. Sutskever et al. [15] used deep neural networks with sequence-to-sequence learning to approach abstractive summarization, and later, Nallapati et al. [4] was first to apply the encoder-decoder RNN to abstractive summarization, after it had been primarily used for machine translation. However, both of these models have been implemented exclusively to summarize text, and have not utilized priors to constrain the summarized output to a novel form or style.

To date, only one other model has trained on prosaic, non-poetic text, and applied priors to constrain output to poetic style. Van De Cruys [16] also used an encoder-decoder RNN, but with gated recurrence units and using a general attention mechanism. In this model, the encoder encodes the current verse, and then the decoder predicts the next verse word-by-word in reverse, so that the rhyme (i.e, the last word of the line) is generated first and the rest of the line is predicated on matching it. Because this model is trained on generic text, Van De Cruys implemented two priors, a rhyme constraint and a topical constraint which are applied to the decoder output to constrain it to poetic style.

All existing models for poem generation have been built to exclusively generate novel poetry. While Van De Cruys [16] used prosaic text to train their model and priors to constrain the output to a poetic style, no model has 1) summarized semantic content to create poems or 2) generated poems while retaining the semantic content of the input text.

3 Approach

The Poetic Press is composed of three main parts: a summarization model, poetic priors, and an inference model. The summarization model consists of an encode/decode network which aims to summarize input news articles; the poetic priors are constraints which capture certain aspects of poetry; the inference model takes news article text as input then passes the text through the summarization model and applies the poetic priors to generate the output poem.

3.1 Summarization Model

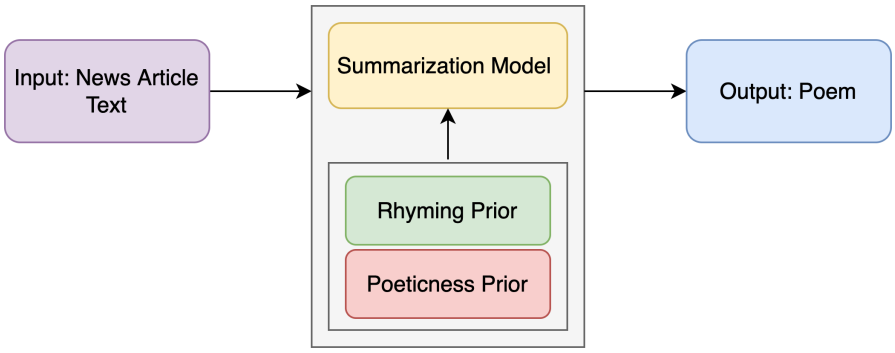

Figure 1: Pipeline for Poetic Press inference model: text input is passed into the summarization model and the poetic priors are applied, producing the output poem.

Abstractive summarization is a relatively well-studied area of natural language processing, and the Poetic Press summarization model is similar to many state-of-the-art abstract summarization models in the literature [5] [13].

Method. Figure 3 shows a diagram of summarization model. The model follows a sequence-to-sequence architecture and uses an encoder network to represent the input text as a latent context vector model and a decoder model which uses the latent vector to predict a summary [15]. The summarization model utilizes three long short-term memory (LSTM) networks to encode the input as a latent vector and a single LSTM network to decode this latent vector into a summary. Not shown in the diagram are two embedding layers, one each for the encoder and decoder models. While we experimented with various pre-trained word embeddings models such as GloVe [7], we found the model worked best when it learned the word embedding weights. The encoder embedding layer learned the vector representations of the input articles' vocabulary while the decoder embedding layer learned to latent vector representation to vector representations of the output summary vocabulary.

Data. The summarization model was trained on a dataset of over 100,000 news articles, with the target summaries being the article headlines. The input article texts were first cleaned by removing punctuation, URLs, and superfluous whitespace. The cleaned texts were then tokenized and padded to a maximum length of 62 tokens. These vector inputs were passed through the encoder embedding layer, which represents each word as a 200-dimensional vector. These sequences were then pass through the encoder network and encoded into a 300-dimensional latent vector. The latent vector was passed through the decoder embedding layer and this latent sequence was finally passed through the decoder network, producing at every timestep a softmax distribution of the same size as the output vocabulary, indicating the probability that each word should be next in the summary. The model was trained using a sparse categorical cross entropy loss function applied to the predicted summary and the article headline, both represented as a sequence of indices of the output vocabulary. The model was trained for twenty-three epochs; the train and validation loss over training epochs is shown in figure 2.

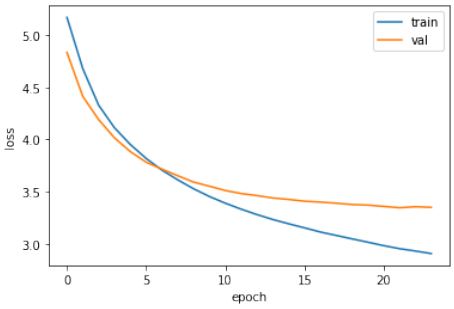

Figure 2: Summarization model train and validation loss over training epochs.

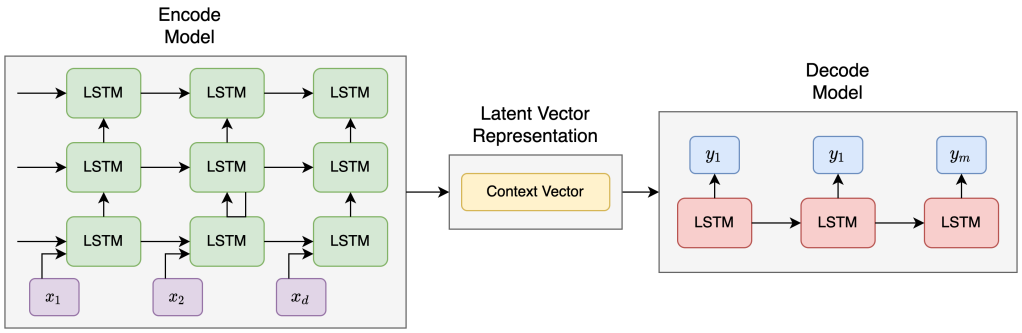

Figure 3: Poetic Press summarization model architecture: three stacked encode LSTM networks and one decode LSTM network. $x_1, ldots, x_d$ represents the tokenized input news article padded to length d, and $y_1, \ldots, y_m$ represents the output summary of length m.

3.2 Poetic Priors

Given there is no parallel corpus for pairs of news articles and poems, we follow Van de Cruys's [16] *priors* approach in poem generation. To be specific, we provide the abstractive summarizer model with prior knowledge about poeticness - knowledge of *rhyme* and of *poeticness of each word* - to render prosaic text summary in poetic style.

3.2.1 Rhyme Prior

The rhyme priors were utilized in order to enforce a rhyme scheme on the output from our abstractive summarizer. One of the most apparent differences between prosaic text and poetic text is the incorporation of rhyme, so we considered the inclusion of a rhyme prior as a key element in our goal of poetic summarization.

Method. The first technical challenge in calculating the rhyme prior was quantitatively determining the perfect rhymes and slant rhymes of a particular word. Once a target rhyme word is specified, such as "calculation," we must first determine its phonetic structure. The phonetic breakdown of "calculation" is "K AE2 L K Y AH0 L EY1 SH AH0 N." For another word to be a perfect rhyme, it must match our target word's phonetic pattern for the stressed syllable as well as for every phoneme following the stressed syllable (i.e. the "EY1 SH AH0 N" component must match). An example of a perfect rhyme with "calculation" would be "calibration." In order to avoid overly constraining our rhyme choices, we also incorporated slant rhymes, which we defined as rhymes in which the stressed syllable does *not* match, but every phoneme following it does (i.e. "SH AH0 N"). An example of a slant rhyme with "calculation" under our definition would be "abolition."

The rhyme prior is only applied every n syllables, where n can be specified before running the inference model. This makes it more likely (but not a certainty) that each line will have the same number of syllables, which creates a more poetic "flow" compared to simply enforcing the same number of words per line. Once

the target number of syllables have been reached, we find all rhymes as indicated above, and create a vector over our entire vocabulary with the scheme:

$$\begin{aligned} P(rhyme|token = j) &= 0 \text{ for non-rhymes} \\ P(rhyme|token = j) &= 1/Z \text{ for slant rhymes} \\ P(rhyme|token = j) &= 4/Z \text{ for perfect rhymes} \\ Z &= 4N_{perfect} + N_{slant} \end{aligned} \tag{1}$$

where Z is a normalization factor to ensure a proper probability distribution. The output from the abstractive summarization model is then element-wise multiplied with our rhyme prior, scaling each token in the original output by its ability to rhyme with our specified target word. This procedure is different from [16], where a uniform probability was assigned to every rhyme word. Our method is a novel way to expand the potential rhymes to draw from, while still taking the quality of the rhymes into account.

Data. We utilized the Python Pronouncing library [6] as well as the Carnegie Mellon Pronouncing Dictionary [10] to produce the phonetic breakdown of the tokens in our model's vocabulary. Each time the rhyme prior was invoked, rhymes were found only across our model's vocabulary in order to avoid selecting rhymes from outside of our model's scope. This is another reason we found it useful to add slant rhymes; our model's limited vocabulary further constrains the number of words that can be selected per timestep.

3.2.2 Word Poeticness Prior

Patterns of word usage significantly vary depending on text genre [14]. This must be also true for news and poetry, especially given the inherent difference of their purpose of writing. In this regard, we provide the abstractive summarizer with prior knowledge about each word's poeticness using the relative frequency of each word appearing in poems compared to appearing in news articles. Put differently, we added the prior knowledge of how more likely a token would appear in poems than in news is used as *poeticness* of each word.

Method.

$$\begin{aligned} P(token = j|poem) &= \frac{N_j^{poem}}{\sum_i N_i^{poem}} \\ P(token = j|news) &= \frac{N_j^{news}}{\sum_i N_i^{news}} \\ Poeticness(token_j) &= \frac{P(token = j|poem)}{P(token = j|news)} \end{aligned} \tag{2}$$

We only include tokens that are contained in the output dictionary of the abstractive summarizer to make sure that the poetic dictionary has the same token set as the abstractive summarizer.

After computing how likely each word would show up in poetry and in news articles separately, we computed the relative probability by dividing the two probabilities to represent poeticness of each word. The process is shown in Equation (2).

The computed poeticness then were normalized before being added to the abstractive summarizer in order to prevent the poeticness prior from completely overriding the effect of word prediction by the summarizer. Specifically, we prevented huge variation of the poeticness distribution by having upper and lower limits of poeticness as 1.1 and 0.9, respectively. In so doing, we were able to control the influence of the poeticness prior on the word prediction if it becomes too strong to also reflect the prediction of the summarizer [1]. The distribution of the poeticness over the entire dictionary is shown in Figure 4.

Data. Word frequency in news was counted using the same corpus that we used to train the abstractive summarizer. Word frequency in poetry was counted using the poem dataset downloaded from Kaggle. The total number of tokens in poetry corpus and in news corpus were 3,127,769 and 5,283,184, respectively. To match the size of the two corpus we only used a subset of the news corpus.

[1] When implemented, it was found that the word poeticness prior changes the next word selection by the model about 24% of the time.

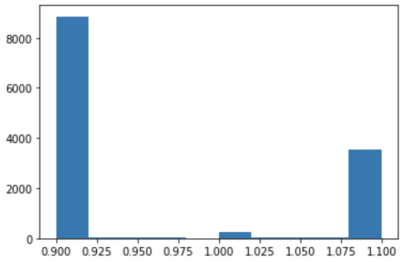

Figure 4: The distribution of word poeticness over the entire dictionary

3.3 Inference Model

The Poetic Press inference model combines the summarization model and the poetic priors together to generate a poem given news article text.

Method. Figure 1 shows the inference model pipeline. Given some news article text, the inference model first cleans, tokenizes, and pads the text as described in Section 3.1. The text is then passed through the summarization model encoder network, producing the latent vector representation of the text. The latent vector is passed through the decoder embedding layer, and this sequence is finally passed through the decoder network. At every timestep, the decoder model generates a softmax distribution which indicates the probability of each word being next in the summary. The inference model multiplies this probability by the poetic priors in order to scale the probabilities of each word based on poetic constraints. The inference model then selects the word which maximizes this posterior distribution, and proceeds to the next timestep. We can then manually tune constants by which the priors are scaled by to strike a balance between transferring to a poetic style and preserving the intended meaning of the summary.

The poeticness priors are applied to every word selection at every timestep but the rhyme priors are only applied every n syllables. While we want every word in the poem to maximize poeticness (as defined above), we naively only want to generate a rhyme at the end of the line. As each word is being generated at each timestep, we keep a counter of how many syllables have been added since the last application of the rhyme priors (or since the start of the poem) and reset the counter to zero when we exceed n syllables.

Data. In our final implementation of the inference model, we fine-tuned parameters for the poetic priors to produce reasonable output poems. For the rhyming priors, we scaled the probabilities of words with perfect rhymes by 500 and slant rhymes by 300. While this seems large, we found that smaller numbers did not produce the desired affect. This could be improved by using a larger rhyming dictionary which could find more rhymes within the news dataset vocabulary. Since the outputs produced by the summarization model are relatively short, we chose to enforce the rhyming prior every 6 syllables. For the word poeticness priors, we clipped the ratios between 0.9 and 1.1; from experimentation we found that a larger range compelled the inference model to output poems which, while abundantly poetic, did not adequately preserve the semantic meaning of the text.

Since the inference model generates novel poems for given article text, it cannot be directly compared to the original article text or the original headline in the same way that the predicted summary is in the summarization model. Instead, we tested the inference model on a random selection of test data (some examples can be seen in section 5) and assessed the model using various evaluation metrics which are further discussed in the following section.

4 Evaluation

In order to evaluate performance of the model in rendering news article in a poetic style, we used two evaluation criteria that can be computed without human ratings: *transfer strength* and *semantic preservation*.

4.1 Transfer strength

To confirm that Poetic Press generates texts conform to the poetic style, it should be examined how likely the poems generated though the Poetic Press are recognized as poems rather than news articles. To achieve this goal, we built a simple text classification tool using deep neural network models, which can classify whether an input text is a news article or it is a poem.

Method. First, we represented the input text into a 768 dimensional vector using SentenceTransformer [9] implemented with BERT [2]. By doing so, we are able to let a deep neural network deal with texts having different lengths. Then, we trained a 3-layer neural network using Keras [1] to predict whether input texts are poetry or news. Each hidden layer had 150, 50, 30 units with *tanh*, *relu*, *relu* as activation functions, respectively. The final layer had only a single unit with *tanh* as the activation function. The objective function was the binary cross-entropy function. After training, the accuracy of classifying texts into news articles or poems reached 99% with the test set from the data described above.

4.2 Semantic Preservation

Method. One of the key properties of Poetic Press is that it preserve the semantics of the original input text. To ensure that poems generated by Poetic Press preserve the same semantic meanings as the input news article, we measured similarities between input news articles and the generated poems.

First, we rendered news articles and the corresponding poems that Poetic Press generate in 768 dimensional vector using the same way we described in the previous section. Then, we computed the similarity between the two vectors in terms of the cosine similarity, using Equation (3).

$$Similarity(In, Out) = \frac{\sum_{i=1}^{768} In_i \cdot Out_i}{\sqrt{\sum_{i=1}^{768} In_i^2}\sqrt{\sum_{i=1}^{768} Out_i^2}} \tag{3}$$

, where *In* is input news article and *Out* is the corresponding poem that Poetic Press generate; and *i* represents the index of the value in a vector.

As for the baseline, we simply used the similarity between random pairs of news article and the generated poems.

Data We used 2,864 number of poems from Kaggle Poem dataset and also used 2,864 news articles from the same corpus that we used for summarizer. In order to prevent any confound effect from the difference in length, We made sure that texts from both corpora are in the similar range of length (the mean number of tokens in texts of poetry corpus and news corpus were 62.98 and 68.48, respectively).

5 Results

5.1 Output Example & Qualitative Analysis

The typical output from Poetic Press is described in Table 1.As samples show, the semantics of news articles is considerably preserved after being transformed into poems. More interestingly, we can observe the generated poems seem to be consistent with our prior knowledge about poetic style.

5.2 Transfer Strength

In order to test transfer strength, we used 1,000 poems generated by Poetic Press. Tested with the classification model described in section 4.1, 87.5% of generated poems were correctly classified.

5.3 Semantic Preservation

The same set of samples was used to test semantic preservation. The average cosine similarity between news article vectors and generated poem vectors was 0.42, which is greater than the baseline average cosine

Table 1: Sample output from Poetic Press

Input	Output
(...) state law barred local governments from enacting ordinances banning discrimination based on sexual orientation or gender identity but (...)	united court rules **legislation** can be used to carry **discrimination**
(...), May warned that the failure to strike a deal on any future relationship would have consequences for security (...)	uk chief calls for **situation** with security **violation**
President Donald Trump said reports that members of his campaign team (...) during his presidential campaign are "fake news" (...)	**fake news** **fake news** taken by **fake news** in us

similarity between random pairs of news article vectors and poem vectors 0.36. Although we might need a baseline to draw a robust conclusion about the performance of Poetic Press in preserving the semantics, the initial analysis shows that it is capable of retaining a considerable amount of the semantics of the original text.

6 Conclusion and Future Work

In this work, we introduced a novel method to build an automatic poem generator that preserves the semantics of original texts. The model was enabled by providing prior knowledge about poetic style -*rhyme* and *poeticness of words* - to the abstractive summarizer. The evaluation that is conducted both qualitatively and quantitatively provided evidence that the integration of abstract summarizer and prior poetic knowledge does enable generation of poems that retain the meaning of the input text. This study contributes to the field of NLP, especially by showing a method to build a text style transfer model even without supervised learning that necessitates parallel corpora.

There are a number of directions in which different aspects of this research can be taken in the future. While the summarization model is relatively state-of-the-art, there are two particularly interesting possible improvements. The first would be to add an attention model, which would help in capturing the meaning in larger news articles and flexibly determine the most important parts of an article to summarize [17]. The second would be to implement a pointer-generator network which would learn to copy parts of the original text via pointing while retaining the ability to produce novel words via the generator, as done in See, et. al. in 2017 [12]. Selectively choosing to reproduce the original text would allow for the summarization of articles related to people, places, and ideas which are not found in the training news dataset. An advantage of the Poetic Press is its modularity: since the summarization training is done separately from the inference and application of the poetic priors, another future direction of this research would be to replace the existing summarization model with a model which summarizes some other form of prosaic text, for example novels or textbooks. Lastly, some form of unsupervised clustering could be performed on existing datasets of particular types of poems, such as limericks or quatrains, to generate poetic priors for specific poetic forms. The successful incorporation of any of these devices would mean a fascinating step towards more generally applying poetic style transfer on prosaic texts.

Author Contributions

All authors wrote paper. CB reviewed literature. SS implemented abstractive news summarization. SS and BH implemented rhyme prior. SR implemented word poeticness prior. SR constructed evaluation tools and tested the model. All authors contributed equally to this work.

References

[1] F. Chollet et al. Keras, 2015.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[4] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

[5] R. Nallapati, B. Zhou, C. N. d. santos, C. Gulcehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. 2016.

[6] A. Parrish. Pronouncing library, 2018.

[7] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[8] R. Perera and P. Nand. Recent advances in natural language generation: A survey and classification of the empirical literature. *Computing and Informatics*, 36(1):1–32, 2017.

[9] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[10] A. Rudnicky. Carnegie mellon pronouncing dictionary, 2014.

[11] A. Saeed, S. Ilić, and E. Zangerle. Creative gans for generating poems, lyrics, and metaphors. *arXiv preprint arXiv:1909.09534*, 2019.

[12] A. See, P. J. Liu, and C. D. Manning. Get to the point: Summarization with pointer-generator networks, 2017.

[13] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy. Neural abstractive text summarization with sequence-to-sequence models, 2018.

[14] E. Stamatatos, N. Fakotakis, and G. Kokkinakis. Text genre detection using common word frequencies. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*, 2000.

[15] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks, 2014.

[16] T. Van de Cruys. Automatic poetry generation from prosaic text. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 2471–2480, 2020.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

[18] Q. Wang, T. Luo, D. Wang, and C. Xing. Chinese song iambics generation with neural attention-based model. *arXiv preprint arXiv:1604.06274*, 2016.

[19] R. Yan. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *IJCAI*, volume 2238, page 2244, 2016.

[20] X. Zhang and M. Lapata. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, 2014.