# Poem Generation using Transformers and Doc2Vec Embeddings

Marvin C. Santillan
*College of Computer Studies*
*De La Salle University*
Manila, Philippines
marvin_santillan@dlsu.edu.ph

Arnulfo P. Azcarraga
*Colloge of Computer Studies*
*De La Salle University*
Manila, Philippines
arnulfo.azcarraga@dlsu.edu.ph

***Abstract*—Poems are sequences of words that express ideas and emotions in an imaginative style, some following strict literary syntax or form. They are artistic expressions, and their generation requires in-depth knowledge and mastery of language. As such, poem generation is considered a very challenging task in Natural Language Processing and has been attracting research interest in the recent decade. We propose a method of generating poems using transformers, coupled with doc2vec embeddings in order to assess the automatically generated poems. In this method, we first train a transformer and a doc2vec model using a poem dataset. Then the trained transformer takes an input text and produces several generated poems. To have an objective basis for assessing the generated poems, we present a preliminary attempt at measuring the quality of a machine-generated poem by computing the cosine similarity score, referenced to the trained doc2vec model. This score is used as a basis for choosing the final output poem. The results show that this method ensures good cohesion between the machine-generated poem and the given input text. We then also explore the implication of the transformer training to the doc2vec embeddings of its output poems, which are shown to be more similar to poems (documents) in the train set as training progresses. Finally, we demonstrate how transformers can learn some poetry styles by exposing them to poems of specific poets.**

***Index Terms*—poem generation, transformers, doc2vec embedding, pre-training.**

## I. Introduction

A poem is a sequence of words that expresses ideas or emotions in a powerfully vivid and imaginative style, often subject to some rules on syntax and literary form. Poetry, indeed, is a literary art work whose content sometimes reflects the ideological underpinning of society and culture. There are various types of poems. They may follow metrical and rhythmic patterns while others do not have strict structure but all are considered as artistic expressions [1].

Poetry requires in-depth knowledge and mastery of language in order to convey the intended message via semantic and aesthetic form. Automatically creating poems is considered as one of the challenging tasks in Natural Language Processing (NLP) because it requires content and form understanding. Automated poem generation is typically a subfield of Natural Language Generation (NLG) together with applications and problems like dialogue system, text summarization, and text paraphrasing [2].

Automatic poem generation had been attracting interest in the recent decade. Approaches vary from applying grammatical and semantic template [2], [3]; statistical machine translation [4], [5] together with summarization techniques [6]; use of finite state machine and unsupervised learning [7], [8]; deep learning like recurrent neural network [9]–[11]; and generative adversarial training [12].

In the works of [5]–[8], [10], poem generation is facilitated by a context word or topic to which the poem to be generated must be related. In [8] word2vec embeddings were used to capture words related to the given topic and these words were filtered and eventually made part of the final poem output. Recurrent Neural Networks (RNN) based approaches leverage on the RNN's ability to learn content and style from the dataset that is fed to them. Recognizing the fact that RNNs tend to forget historical input quickly, [11] attention-based LSTM model to capture dependencies between sentences and thereby facilitating cohesion in the output prose. The attention mechanism allows the network to memorize long distance patterns making it an important ingredient in transduction systems like poem generation.

Reference [13] exploited the use of attention mechanism and proposed transformer architecture. It is like other transduction models that is implemented via encoder-decoder structure where the encoder maps sequence of representation *($x_1$, $x_2$, $x_3$,.. $x_n$)* to continuous representation *($z_1$, $z_2$, $z_3$,.. $z_n$)* and given that sequence of $z_i$, the decoder generates output sequence *($y_1$, $y_2$, $y_3$,.. $y_m$)*. However, the transformer uses stacked self-attention and point-wise, fully connected layers to implement its encoder and decoder. Aside from the advantage of learning long-range dependencies effectively, transformers are also noted to be computationally simpler than other architectures, and its operation requires less sequential computation because most of its computation can be parallelized.

This research explores generating poems using a transformer and doc2vec embedding. We first discuss our methods in the next section, then we present our experiments and results, and the last section talks about our conclusion. The main contributions of this research are as follows:

- Propose a method of generating poems leveraging from a pre-trained transformer model and retraining the model using various poem dataset.

- Explore the use of doc2vec embedding in assessing poems generated by a transformer, which can also be utilized in similar text generation system.

## II. Methods

In this section, we present our methods. We first describe our problem formulation and present our Poem generation approach that uses a transformer and doc2vec embedding.

### A. Problem Formulation

The goal of this project is to generate poems using transformers. The system will accept an input text to direct the generation of the poem. The output poem and the input text must be related in some way. That is:

$$T(s) \Rightarrow p \quad (1)$$

where *T(s)* is the transformer *T* taking input poem seed *s* and producing output poem *p*.

### B. Approach

The task requires the training of a transformer *T*, which will be used to generate the output poem *p* with the provided text input *s*. For this purpose, we will use the transformer model of [14]. However, due to memory considerations, we opted for the 117 million parameter version. This model is not trained with a poem dataset but with WebText. However, advantages can be gained from using a pre-trained model on a similar NLP task as stated in [15]. Transformer *T* will then be further trained using a poem corpus to fine tune its poem generating capability.

Another requirement for this poem generation task is, the input text *s* must be related to poem output *p*. To accomplish this, a doc2vec model of [16] will be used. Doc2vec model projects documents into a vector space where documents that are similar will be represented by vectors which are close to each other as stated in [17]. Fig. 1 visualizes a sample doc2vec projection. To capture the most related poem output *p*, Transformer *T* will then be allowed to produce several output $p_i$, then each pairs *(s, $p_i$)* will be referenced against doc2vec model, whichever $p_i$ whose projection has the highest cosine similarity with the projection of *s* will be the final output *p*.

Fig. 2 depicts this overall approach of generating poems with the Transformer and Doc2Vec referring to the trained models using the poem dataset. Preprocessing tokenizes and prepares input text before it can be accepted by both models. Scoring rates the candidate output poems, as described in the previous paragraph.

### C. Dataset

Poem corpus used for training the transformer and doc2vec models is taken from UniM-Poem dataset of [12]. UniM-Poem was refined by excluding poems that are less than 50 characters in length. This is further divided into two dataset, those which are 50 to 200 characters in length (short_poem dataset) and those which are above 200 characters in length (long_poem dataset), each with 58955 and 32381 number of poems respectively. Detailed information about these datasets is listed in Table I.

Another dataset used for the experiments in this research is the poem dataset from poetryfoundation.org downloaded from kaggle.com. This dataset is composed of 506 poems. However, only those from poet Sir Thomas Wyatt (22), Sir Philip Sidney (42), Edmund Spenser (34), and John Donne (41) were used. Added to this pool is the sonnets of Shakespeare (154). This dataset (poet dataset) has a total of 293 poems.

All datasets used for training any of the transformers have their individual document delimited by "<endoftext>" token, this will also be used to indicate a poem chunk in the output text of the transformers.

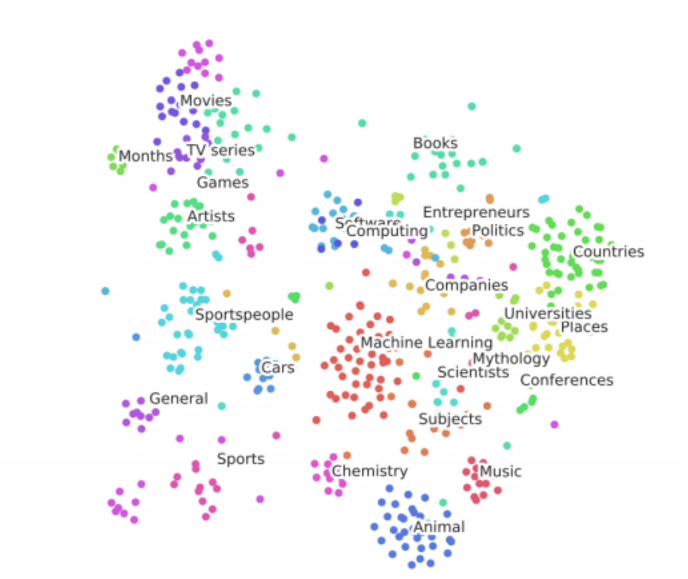


Fig. 1. Sample doc2vec document vector projection showing similar documents projected closer to each other.

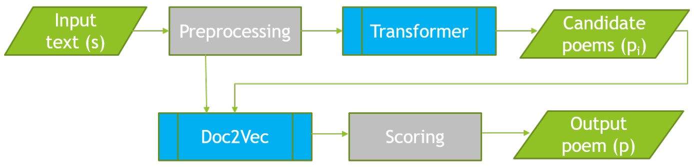


Fig. 2. Diagram of generating poems using transformer and doc2vec embedding.

TABLE I
DETAILED INFORMATION OF DATASETS EXTRACTED FROM UNIM-POEM DATASET.

| Dataset | #Poem | #Line/Poem | #Char/Line | #Word/Line |
|---|---|---|---|---|
| long_poem | 32381 | 7.85 | 37.82 | 6.91 |
| short_poem | 58955 | 4.57 | 28.61 | 5.64 |

### D. Model Training

*a) Transformer:* Several transformers were trained using different datasets at end with the experiments conducted on this research. All transformer training used the same hyperparameters, such as learning rate of 0.00002 and batch size of 2. Adam was used as optimizer. A copy of the model was saved every 100 steps. The number of training steps for each model varies and is specified in each experiment conducted.

*b) Doc2Vec:* Gensim Python library was used for implementing the doc2vec models. All doc2vec models used Paragraph Vector Distributed Bag-Of-Words (DBOW) algorithm during training. Vector size was set to 300, words that appear only once on the training corpus were ignored, and window size was set to 15, as suggested in [18]. The number of training epochs varies per model and is mentioned for each experiment conducted. Other hyperparameters used the default values as provided by the Gensim library.

## III. Experiments and Results

### A. Transformer Generated Poems

Three transformers were trained using long_poem dataset, short_poem dataset, and a dataset composed of a single poem from long_poem dataset (single_poem) for 1500 steps, 1900 steps, and 100 steps respectively. The models were allowed to generate 10 samples of up to 50 tokens each. Then samples were split if a delimiter token appears as a substring. Corresponding doc2vec models were also trained using each mentioned datasets for the assessment of the output poems. Both doc2vec models for the long_poem and short_poem were trained for 1000 epochs, while for the single_poem was only trained for 400 epochs. Using the input poem seed in Fig. 3, the corresponding output poem for each transformer appears in Fig. 4, 5 and 6. Since the transformer model is pretrained (base_model) and already capable of generating text, its output was also generated for reference. Fig. 9 shows the base_model's output. All outputs appear truncated as a consequence of the generation set-up.

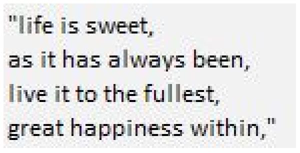
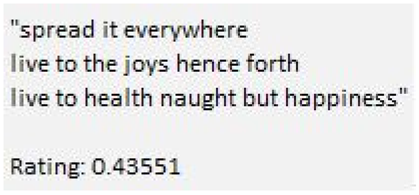
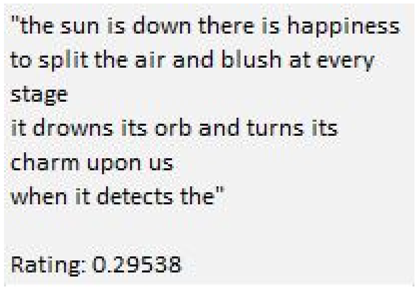

"life is sweet,
as it has always been,
live it to the fullest,
great happiness within,"

Fig. 3. Input poem seed.


"spread it everywhere
live to the joys hence forth
live to health naught but happiness"

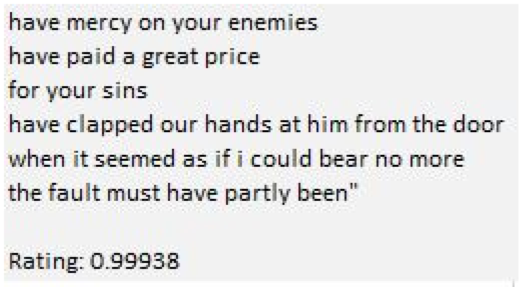
Rating: 0.43551

Fig. 4. Output of the long_poem transformer.


"the sun is down there is happiness
to split the air and blush at every
stage
it drowns its orb and turns its
charm upon us
when it detects the"

Rating: 0.29538

Fig. 5. Output of the short_poem transformer.


have mercy on your enemies
have paid a great price
for your sins
have clapped our hands at him from the door
when it seemed as if i could bear no more
the fault must have partly been"

Rating: 0.99938

Fig. 6. Output of the single_poem transformer.

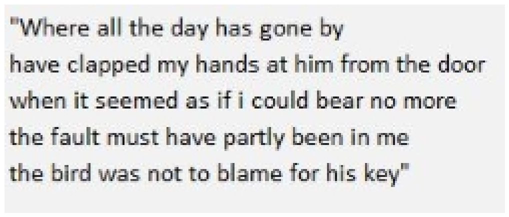

"Where all the day has gone by
have clapped my hands at him from the door
when it seemed as if i could bear no more
the fault must have partly been in me
the bird was not to blame for his key"

Fig. 7. Another output of the single_poem transformer using a different input poem seed.

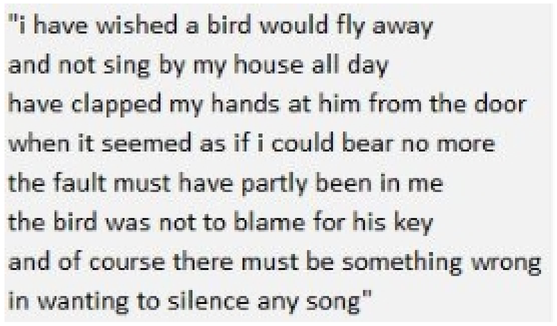
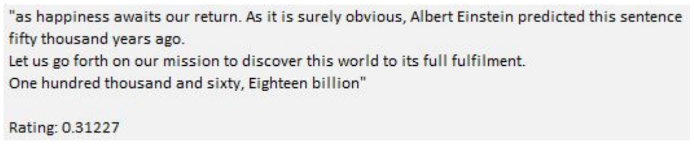

"i have wished a bird would fly away
and not sing by my house all day
have clapped my hands at him from the door
when it seemed as if i could bear no more
the fault must have partly been in me
the bird was not to blame for his key
and of course there must be something wrong
in wanting to silence any song"

Fig. 8. Lone poem on single_poem dataset.


"as happiness awaits our return. As it is surely obvious, Albert Einstein predicted this sentence fifty thousand years ago.
Let us go forth on our mission to discover this world to its full fulfilment.
One hundred thousand and sixty, Eighteen billion"

Rating: 0.31227

Fig. 9. Output of the base_model transformer.

The input poem seed in Fig. 3 is about life and contains the word “happiness”. All transformer models managed to generate poem, with the base, short_poem, and long_poem models also mentioning the word “happiness” signifying relatedness between the input and output poems. The base_model’s output, however, made reference to “Albert Einstein” which is doubtful to be part of a poem text.

The single_poem transformer, on the other hand, seems to be talking about an entirely different topic. A closer look at its outputs shows that they are alike regardless of the input poem seed used. This suggests that it cannot generalized and that it is tied to a variation of a particular output. Fig. 7 shows another output of the single_poem transformer using a different seed. A review of its training statistics show that at step 100 of its training, the loss value plateaus at 0.01150 signifying overfitting, and indeed, all of its outputs are very similar to the sole poem in its training dataset as shown in Fig. 8.

Examining the length of the transformers’ output, it doesn’t necessarily follow that the long_poem transformer will produce longer poems than that of the short_poem transformer, as evidenced by the sample. This might be because of the relatively small number of training iteration steps for both of the transformers and that it is not enough for the transformers to learn the average number of characters per poem as reflected by the corpus on their respective dataset.

It is noticeable that the structure of the output of the base_model is different from the other models. Each line is lengthy and sentence-like, whereas most poem text has a phrase-like structure. Arguably, the transformers retrained using poems are more poetic than the base_model.

Table II shows the top five outputs of the long_poem transformer on the same input poem seed in Fig. 3. The scores are based on the cosine similarity of the input to each respective output based on their projection to the doc2vec model of the long_poem dataset. All output relates to the input since most of them are about life and happiness. The top output has a large margin compared to the other outputs. This is primarily because of the word ‘happiness’ which is common to the input and the output poem.

TABLE II
LONG_POEM TRANSFORMER’S SAMPLE OUTPUT AND RATING.

| Output Poem | Rating |
|---|---|
| spread it everywhere<br>live to the joys hence forth<br>live to health naught but happiness | 0.43552 |
| to suffer not of frost-cold<br>much in heaven’s care<br>will follow life and shall gird<br>it enfold this<br>in a garden of foliage | 0.27648 |
| across all three forms of life<br>what can be done for the good which you have received<br>what can be left aside to say<br>she’s gone to sleep | 0.26391 |
| the bliss endless beyond<br>oh dawn incarnate hearts of wine<br>where the exultation of life seems to hail<br>oh dawn incarnate hearts and hearts and hearts | 0.23559 |
| we do not speak for truth or created being<br>but these words that i say to you<br>is orthodox wisdom of man’s nature<br>of man his true nature | 0.23451 |

## B. Transformer Training Progress

A good language model captures the probability of the words in the target language. Just like other transduction models, a transformer should generate text that is more similar to their training dataset as the training progresses. This indicates that the model is indeed learning from its dataset. To show this in our poem transformer, a copy of the model was evaluated at every 500 steps of training. Sample outputs were generated using three different input poem seeds. Then, each output was referenced to its respective doc2vec model. The cosine similarity score of the output to the most similar document in the dataset was calculated and averaged over the number of each input poem seeds. The graph of the result appears in Fig. 10. The similarity score on step zero represents the score of the base_model. The graph shows a trend of increased output to dataset similarity as the training progresses.

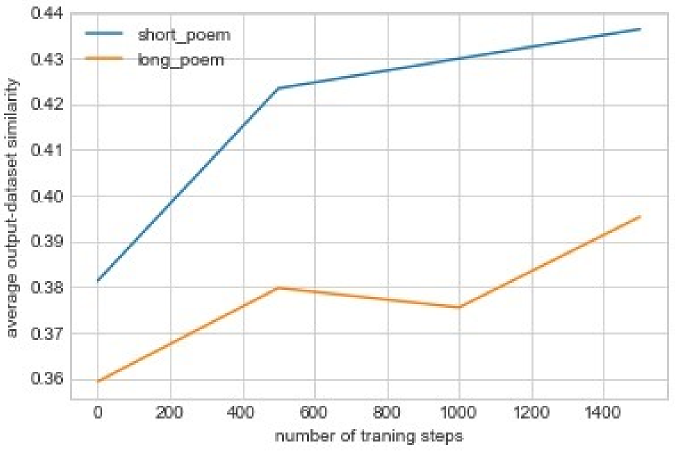
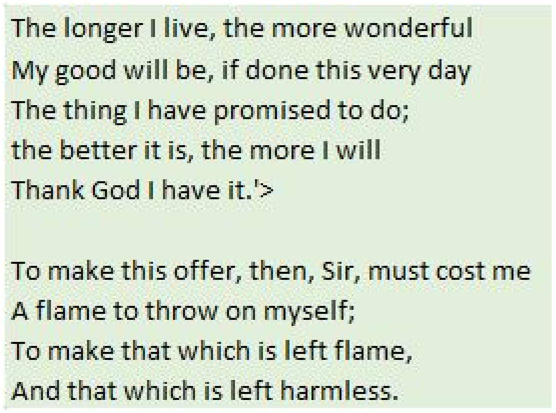


Fig. 10. Transformer’s Output-Dataset similarity scores.

## C. Transformer Poet

Can a transformer learn style from a specific poet? To check, poems made by several poets were extracted from the kaggle poem dataset and were used to train a transformer. The top poets with the largest number of poem on the poem dataset were used; these are Sir Thomas Wyatt, Shakespeare, Sir Philip Sidney, Edmund Spencer, and John Donne. One transformer was trained for each poet for 300 steps and respective doc2vec model for 400 epochs. Then, a poem was generated for each using three input poem seeds. The output poem of each poet transformers using one of the input poem seed appears in Fig. 11, 12, 13, 14 and 15. Also, a transformer and a doc2vec model were trained using poems from all of the poets for 1500 steps and 1000 epochs, respectively. Its output sample appears in Fig. 16.


The longer I live, the more wonderful
My good will be, if done this very day
The thing I have promised to do;
the better it is, the more I will
Thank God I have it.'>

To make this offer, then, Sir, must cost me
A flame to throw on myself;
To make that which is left flame,
And that which is left harmless.

Fig. 11. Wyatt transformer's sample output.

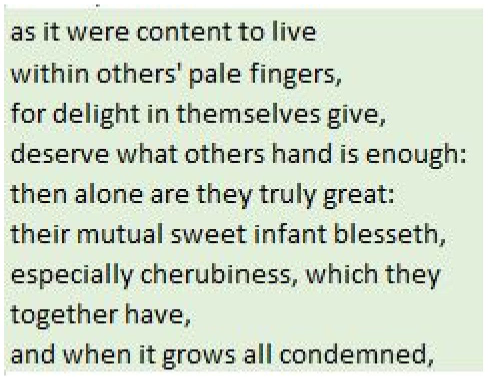

as it were content to live
within others' pale fingers,
for delight in themselves give,
deserve what others hand is enough:
then alone are they truly great:
their mutual sweet infant blesseth,
especially cherubiness, which they
together have,
and when it grows all condemned,

Fig. 12. Shakespeare transformer's sample output

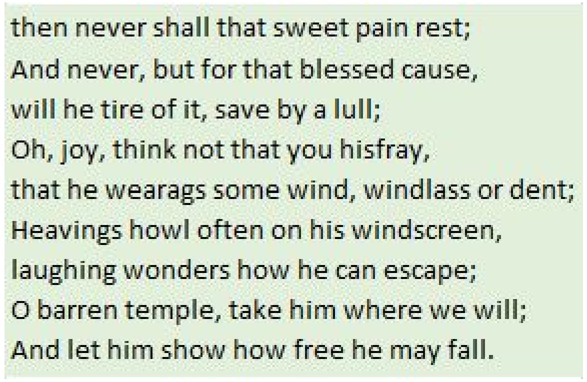
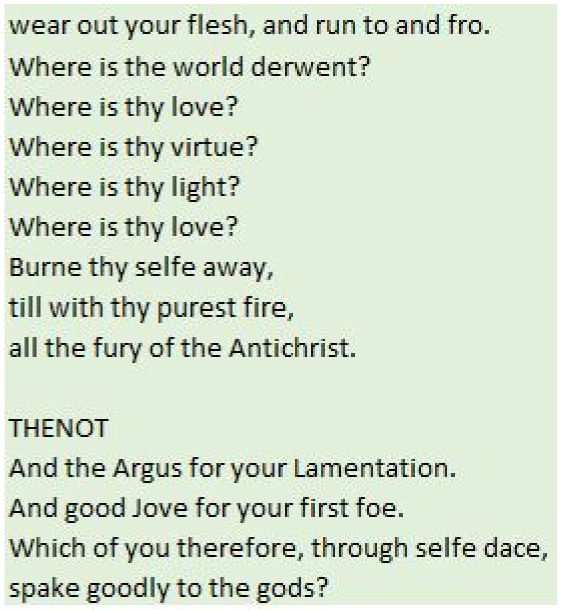

then never shall that sweet pain rest;
And never, but for that blessed cause,
will he tire of it, save by a lull;
Oh, joy, think not that you hisfray,
that he wearags some wind, windlass or dent;
Heavings howl often on his windscreen,
laughing wonders how he can escape;
O barren temple, take him where we will;
And let him show how free he may fall.

Fig. 13. Sidney transformer's sample output.

All poet transformers were able to generate output that is poem-like. They were able to learn how to use new lines and punctuation. It is interesting to note that certain poem style transpired in some of the output. In Fig. 12, a rhyming words at the end of the first and third lines appeared. This is a typical characteristic of Shakespearean sonnets. Also, in Fig. 14, the word 'THENOT' appeared. Thenot is actually a


wear out your flesh, and run to and fro.
Where is the world derwent?
Where is thy love?
Where is thy virtue?
Where is thy light?
Where is thy love?
Burne thy selfe away,
till with thy purest fire,
all the fury of the Antichrist.

THENOT
And the Argus for your Lamentation.
And good Jove for your first foe.
Which of you therefore, through selfe dace,
spake goodly to the gods?

Fig. 14. Spenser transformer's sample output.

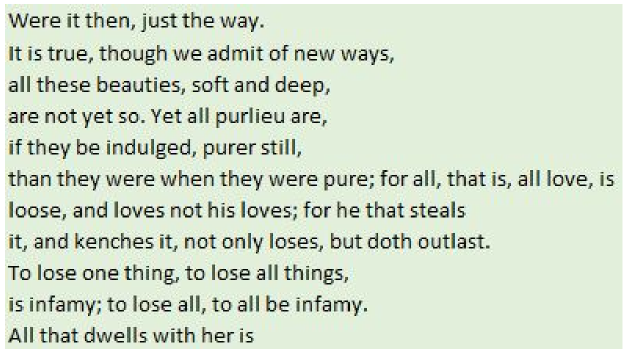
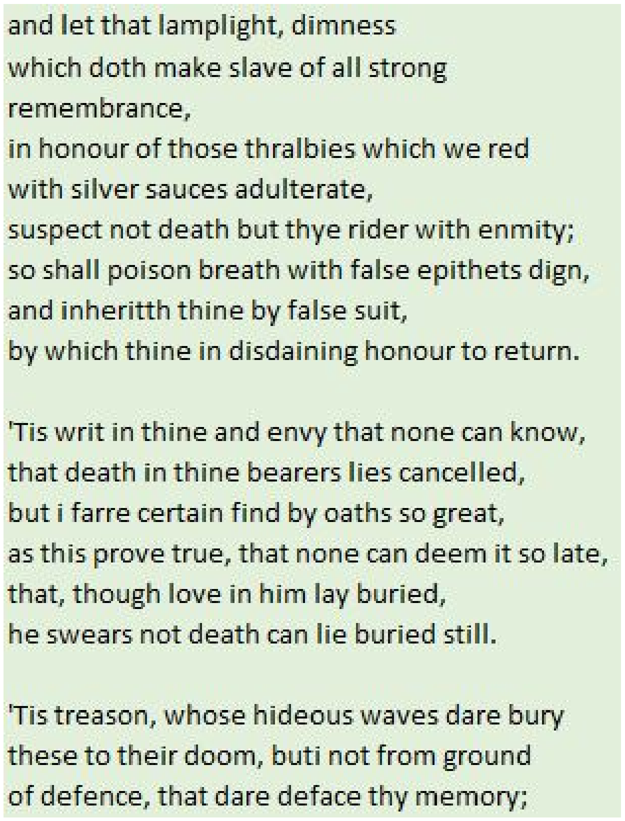

Were it then, just the way.
It is true, though we admit of new ways,
all these beauties, soft and deep,
are not yet so. Yet all purlieu are,
if they be indulged, purer still,
than they were when they were pure; for all, that is, all love, is
loose, and loves not his loves; for he that steals
it, and kenches it, not only loses, but doth outlast.
To lose one thing, to lose all things,
is infamy; to lose all, to all be infamy.
All that dwells with her is

Fig. 15. Donne transformer's sample output.

character in one of Spenser's eclogues; a poem usually cast as pastoral dialogues. However, all outputs seem less related to the input poem seed which is about "life and happiness". Most talked about love and god. This might be because of the small training dataset used that may only contain poems that are about specific category of topics.

We wanted to check if each transformer is indeed producing poems that are similar to the poems of the poet used in its training. That is, can we identify the output of each transformer as belonging to the poet in their respective datasets? For this purpose, we used the doc2vec model trained using poems of all the poets. Using three input poem seeds, we allowed each poet transformer to produce several output, then checked which among the document in the doc2vec model has the highest cosine similarity with each output. Then we counted to which poet that particular document belongs to. We tallied the results and computed for the hit-rate by dividing it with the total number of output generated by each poet transformer. We then normalized the hit-rate by dividing it with the total


and let that lamplight, dimness
which doth make slave of all strong
remembrance,
in honour of those thralbies which we red
with silver sauces adulterate,
suspect not death but thye rider with enmity;
so shall poison breath with false epithets dign,
and inheritth thine by false suit,
by which thine in disdaining honour to return.

'Tis writ in thine and envy that none can know,
that death in thine bearers lies cancelled,
but i farre certain find by oaths so great,
as this prove true, that none can deem it so late,
that, though love in him lay buried,
he swears not death can lie buried still.

'Tis treason, whose hideous waves dare bury
these to their doom, buti not from ground
of defence, that dare deface thy memory;

Fig. 16. All_poet transformer's sample output.

TABLE III
NORMALIZED HIT-RATE OF POET TRANSFORMER'S OUTPUT WITH THE POET'S POEM IN THE DATASET.

| | Normalized Hit-Rate | | | | |
|---|---|---|---|---|---|
| Transformer | Sidney | Donne | Spenser | Wyatt | Shakespeare |
| Sidney | **0.00595** | 0.00136 | 0.00408 | 0.00505 | 0.00289 |
| Donne | 0.00325 | **0.00499** | 0.00134 | 0.00103 | 0.00384 |
| Spenser | 0.00384 | 0.00000 | **0.01328** | 0.00000 | 0.00251 |
| Wyatt | 0.00397 | 0.00136 | 0.00408 | **0.01768** | 0.00162 |
| Shakespeare | 0.00106 | 0.00108 | 0.00196 | **0.00606** | 0.00462 |
| Shakespeare2 | 0.00000 | 0.00000 | 0.00151 | 0.00117 | **0.00599** |

number of document that each poet has in the training dataset of the doc2vec model. The results appear in the Table III.

Table III shows that the majority of the poet transformers are indeed producing output that can be identified as belonging to its own dataset than to other poets (results in bold). However, Shakespeare transformer output seems to be more similar to Wyatt's poem. Looking at the training details of the poet transformers, all are uniformly trained for 300 steps, but Shakespeare transformer has the highest number of training poems. To compensate, we further trained the Shakespeare transformer for another 300 steps and rerun the experiment. The result also appears in Table III with the label of "Shakespeare2", and this shows that the transformer is now producing more similar output with the Shakespeare poems. Over-all, it can be said that the transformers are producing poems that are similar to the poems of the poet used in their respective training, therefore their output can be identified as belonging to their corresponding poet.

## IV. Conclusion and Future Research

In this research, we have presented a method of generating poems from given input poem seed using a transformers and a doc2vec embeddings. This method leverages from a pre-trained model, which is then finetuned using poem dataset and uses cosine similarity score from a doc2vec model to assess generated poem output. The results show that this method ensures good cohesion between the output and the given input text. We also demonstrated that a transformer can capture the style of the poem eminent in its training set and that given enough training, transformers can produce output that can be identified as belonging to a particular poet.

In the future, we would like to apply this method of generating poems in a different language domain and consider more qualitative assessment, like meter and rhythm, of the transformers' output poems. It is also interesting to investigate the behavior of doc2vec embedding of transformers' output as an indicator of train overfitting.

## References

[1] C. Melin, "Between the lines: When culture, language and poetry meet in the classroom," *Language Teaching*, vol. 43, no. 3, p. 349–365, 2010.

[2] H. Gonçalo Oliveira, "Automatic generation of poetry: an overview," 2009.

[3] H. G. Oliveira and H. Gonc, "PoeTryMe : a versatile platform for poetry generation PoeTryMe : a versatile platform for poetry generation," *Computational Creativity, Concept Invention, and General Intelligence,*, vol. 1, no. November, pp. 1–7, 2015.

[4] M. Zhou, L. Jiang, and J. He, "Generating {C}hinese Couplets and Quatrain Using a Statistical Approach," in *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 1*. Hong Kong: City University of Hong Kong, dec 2009, pp. 43–52. [Online]. Available: https://www.aclweb.org/anthology/Y09-1006

[5] J. He, M. Zhou, and L. Jiang, "Generating Chinese Classical Poems with Statistical Machine Translation Models," in *AAAI*, 2012.

[6] R. Yan, H. Jiang, M. Lapata, S.-d. Lin, X. Lv, and X. Li, "I, poet: Automatic Chinese poetry composition through a generative summarization framework under constrained optimization," in *IJCAI International Joint Conference on Artificial Intelligence*, 2013, pp. 2197–2203.

[7] E. Greene, T. Bodrumlu, and K. Knight, "Automatic Analysis of Rhythmic Poetry with Applications to Generation and Translation," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA: Association for Computational Linguistics, oct 2010, pp. 524–533. [Online]. Available: https://www.aclweb.org/anthology/D10-1051

[8] M. Ghazvininejad, X. Shi, Y. Choi, and K. Knight, "Generating Topical Poetry," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, nov 2016, pp. 1183–1191. [Online]. Available: https://www.aclweb.org/anthology/D16-1126

[9] X. Zhang and M. Lapata, "{C}hinese Poetry Generation with Recurrent Neural Networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing ({EMNLP})*. Doha, Qatar: Association for Computational Linguistics, oct 2014, pp. 670–680. [Online]. Available: https://www.aclweb.org/anthology/D14-1074

[10] X. Yi, R. Li, and M. Sun, "Generating Chinese Classical Poems with RNN Encoder-Decoder," apr 2016. [Online]. Available: https://arxiv.org/abs/1604.01537

[11] Q. Wang, T. Luo, D. Wang, and C. Xing, "Chinese Song Iambics Generation with Neural Attention-based Model," apr 2016. [Online]. Available: https://arxiv.org/abs/1604.06274

[12] B. Liu, J. Fu, M. P. Kato, and M. Yoshikawa, "Beyond Narrative Description: Generating Poetry from Images by Multi-Adversarial Training," apr 2018. [Online]. Available: https://arxiv.org/abs/1804.08473

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," jun 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[14] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," 2018.

[15] A. Radford, "Improving Language Understanding by Generative Pre-Training," 2018.

[16] Q. V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," may 2014. [Online]. Available: http://arxiv.org/abs/1405.4053

[17] G. Fisher, M. Israni, and Z. Robert, "Exploring Optimizations to Paragraph Vectors," *CS224n: Natural Language Processing with Deep Learning*, pp. 1–7, 2017. [Online]. Available: https://web.stanford.edu/class/cs224n/reports/2760664.pdf

[18] J. H. Lau and T. Baldwin, "An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation," jul 2016. [Online]. Available: https://arxiv.org/abs/1607.05368