English and Chinese poetry generation
Software project: Deep Learning for the Processing and Interpretation of Literary Texts

Svetlana Novikova, Sangeet Sagar, Peilu Lin, Meng Li and Pavle Marković
Universität des Saarlandes
{svno00001,sasa00001,peli00002,meli00001,pama00002}@stud.uni-saarland.de

Abstract

Automatic text generation has always fascinated story-writers, and poets, for its ability to generate text inheriting similar properties as parent text. This work attempts to study and analyse one such task- poetry generation using recent state-of-the-art deep-learning approaches. We approach our task in two languages, English and Chinese, using models trained from scratch, e.g. recurrent based models (RNNs), and adopt transfer learning methods like generative pre-trained transformer model (GPT) to show how well poems can be generated by little fine-tuning. We apply multiple setups and strategies like varying encoding-decoding schemes, multiple modeling strategies like gated recurrent unit (GRU), long short-term memory (LSTM) for each language to arrive to the best setup that generates the best quality poems. We observe that fine-tuning generates meaningful poems with high coherence and fluency for English. While, transformer-based encoder-decoder models produce fine quality Chinese poems. Apart from this, we also trained a topic-prediction model to study how well a machine-generated poem is interpreted by a system trained on human-written poems.

1 Introduction

Deep neural networks (DNNs) have been the subject of research since last decade for their ability to solve complex tasks with ease in computer vision and pattern recognition. They have been found to robustly learn underlying patterns in data. But a question that has always been floating around the research community- how can DNNs be used for creative purposes like text or image generation? How can learning be influenced to generate an output that resembles the input? Writing articles, stories, poems, etc., have always been done by professionals who gracefully express their ideas, emotion, and cultural heritage in their writing. However, it's a challenge when a machine does the same. Machine-generated texts do not always carry the richness in the language and meaning. Hence special approaches and methods must be used to learn the hidden structures in text data to generate a rich quality text. This work studies one such creative application: poem generation using DNN.

NLG task in general Natural language generation (NLG) automatically generates intelligible text using textual representations by leveraging the state-of-the-art DNN approach. Features underlying the generated text like fluency, grammatical correctness, syntactical structure etc., highly reflects the data on which it is trained. Often the goal in NLG systems is to robustly capture the data distribution to generate a meaningfully rich, coherent and non-repeated text. NLG can be a text to text generation where the input is linguistic data like human-written poems and stories for literary text generation, parallel data for machine translation or it can be a data to text generation where the input is non-linguistic data like speech to perform automatic text summarization (Rott and Červa, 2016). A few applications include text summarization, story and poetic style text generation, image captioning, building chatbots etc.

Popular DNN approaches to NLG tasks includes conditional language models (CLMs) and encoder-decoder based architectures. In CLMs, the output is generated based on a probability distribution conditioned on an input word or a representation. The encoder-decoder architectures can perform sequence prediction given variable-length input-output prediction wherein encoder encodes a sequence of text into vector representations that form the context vector. The decoder maps the representation back to a variable-length output sequence. While these systems need to be trained from scratch depending on our goals, these methods often need large data to perform the task robustly. In absence of large data pre-trained models like BERT (Devlin et al., 2019a) and GPT-2 (Radford et al., 2019) are preferred to perform transfer learning. In this work, we have used these techniques to perform a specific task in NLG, i.e. poetry generation for Chinese and English languages.

Poetry generation task Poetry is a fine literary work consisting of a few lines of text that carry intense meaning and emotions, often expressed in a rhythmic style. Often poets use little words to express an idea or thought, indicating their mastery of the language. Automatic poetry generation is thus a challenging task in NLG because it requires a deeper understanding of language and its form. It is difficult to capture such fine language structure within given text data to generate an artificial poem with a natural touch. This work focuses on a specific style of poetry- free verse that allows the number of stanzas and rhyming patterns in a poem to be flexible. This work analyses several deep-learning approaches that generate free-verse style Chinese and English poems conditioned on mainly two parameters, i.e. topic and keywords. The goal is to generate a poetry style text based on a distribution conditioned on given a topic and a set of keywords. That is to say, the context of the generated poem should revolve around the given topic, and it must contain at least one keyword from the given set.

Overview of our work The current paper focuses on the poetry generation task in a multilingual setting. Thus, we propose several models for generating poetry in both English and Chinese. We approached the task of poetry generation as a controllable NLG. In our models, we controlled for the content by specifying a topic or keywords.

For the English language, we propose three models – LSTM Language Model, Encoder-Decoder and a fine-tuned Conditional GPT-2 model. Chinese poetry is generated using the Encoder-Decoder model with several model sub-types, including Seq2seq-GRU, -LSTM and -Transformer. We also implement a CNN topic classifier which is later used to evaluate the results of poetry generation.

To evaluate generated poems, we use several metrics – perplexity (PPL), keywords usage (KU) and topic relevance (TR) for English poems, and weighted average precision (WAP) and BLEU score for Chinese poems.

Overview of the following sections Section 2 conducts a detailed survey on recent work on automatic poetry generation and controllable NLG. In Section 3 we discuss at length the English and Chinese datasets that we use in this work. Further, in Section 4 we elaborate on the different models that we use to perform poetry generation like encoder-decoder models, Long Short Term Memory (LSTM) and the most recent area of research- transformer-based models. In Section 4.4 we talk about the model used to perform topic-prediction on generated poems followed by evaluation metrics in Section 5. In Section 6 we present our experimental setup. We discuss the proposed model architecture for both English and Chinese poetry at length. Moving on, we perform a detailed comparison of results in Section 7 and discuss our analysis in Section 8 and end our report with concluding remarks in Section 9. Our code is open-sourced and can be found on `https://github.com/palla-lin/SoProPoetry`.

2 Related work

2.1 Poetry generation

Poetry generation models receive prompts with different levels of language features from rhythm, lexical choice to syntax and semantics, and produce creative texts with aesthetic value. The task gets more attention until 2000 and the approaches of works on poetry could be mainly distributed into 3 categories: rule/template-based methods, probabilistic methods, and neural network based methods. The rule/template-based methods heavily rely on predefined steps and explicit linguistic structures like POS tags (Manurung, 1999; Gervás, 2001; Franky, 2013). Probabilistic language models also have been used to generate poetic text (Barbieri et al., 2012). The deep neural networks techniques in machine translation and summarization tasks also are transferred to poetry generation. Given a sequence of words, or with the guiding features of rhyme, structure and semantics, recurrent neural networks (RNN) could be used to generate new lines (Zhang and Lapata, 2014; Ghazvininejad et al., 2016; Yan, 2016). Our work compares mainstream neural network architectures (GRU, LSTM, and Transformer) in recent years and also adopts representations from the latest pre-trained models like BERT series model and GPT-X model.

The common poetic features used to produce poems usually include form and content features. Tikhonov and Yamshchikov (2018) extend language models with phonetic embedding for poetry generation and shows that phonetic information is very important for English and Russian language. Meter and rhyme are also considered in a few English poetry generation models (Colton et al., 2012; Tobing and Manurung, 2015). Some models are designed to produce specific forms. Ghazvininejad et al. (2016) deals with the sonnet, a classic form of poem with 14 lines and each line with 10 syllables typically. Zhang and Lapata (2014) and Yan (2016) focus on quartrains, a form of classic Chinese poem, 4 lines of 5 or 7 characters with rigid rhyme patterns. As for content features, keywords, lexical-syntactic patterns, or semantic relations are explored. Keywords could set the semantic domain and constrain the topics of generation (Wong et al., 2008; Oliveira, 2012; Zhang and Lapata, 2014; Yan, 2016). Tobing and Manurung (2015) extract dependency relations from input documents to constrain generated poems. In addition, some works exploit emotional lexicon to affect or control the sentiment of generated poems (Gervás, 2000; Misztal and Indurkhya, 2014; Oliveira et al., 2017). In our experiments, we use keywords (and general topic words) to generate English poems, and exploit formal patterns and keywords to generate classical Chinese poems.

2.2 Controllable NLG

Applying neural methods to NLG tasks helps with producing richer and more diverse textual outputs. However, a downside of these methods is lack of control over generated output, which is a crucial property for real-world applications. Given the importance of the previously mentioned property, it became an active area of research how to control generation process for factors such as content (Fan et al., 2018), style (Ficler and Goldberg, 2017), or politeness (Sennrich et al., 2016a). So far, the most successful approaches are decoding-based (See et al., 2019), learning-based (Ficler and Goldberg, 2017) and input-based (Gupta et al., 2021), applied to various NLG tasks like text summarization (Fan et al., 2018), machine translation (Sennrich et al., 2016a), dialogue response generation (See et al., 2019). Our focus in this work is on the poetry generation task where we experimented with different models to produce content-constraint output using the input-based approach.

3 Datasets

3.1 English

The dataset used for the English poetry generation is the Neural Poet dataset (Agarwal and Kann, 2020). The original dataset contains a total of 27131 on 144 topics. The topics include 'success', 'childhood', 'anger', 'animal', 'freedom', etc. The poems were collected from the web and therefore differ in genre and style. Most of the poems varied in length and were limited to 5 lines during data preparation. Further preprocessing steps included removing special characters, adding beginning and end of line symbols, and limiting the number of tokens. Since several models were proposed for the task, the input to each model was different and will be described in more detail in the subsequent sections.

3.2 Chinese

THU Chinese Classical Poetry Corpus (CCPC) contains classical Chinese poems from Nanbei dynasty (A.D. 386) to Qing dynasty (B.C. 1912) (Guo et al., 2019) . The numbers of poems in training, validation and test dataset are 109727, 7979, 9976. The genre of poems in this corpus is quatrain, 4 lines of 5 or 7 characters with specific rhyme patterns. The type that each line with 5 characters is called Wuyan(五言), and each line with 7 characters is called Qiyan(七言). According to the types and dynasties, a detailed distribution of the dataset is illustrated in Table 1.

Each poem in this corpus is also annotated with 3 or 4 keywords. To give the training model more format features, we also add a format template that differentiate characters and placeholders in each poem and provide information of segmentation and length of each line. For example, a poem with 4 lines of 5 characters, like "清浅白沙滩/ 绿蒲尚堪把/ 家住水东西/ 浣纱明月下", has a template "CCCCCPCCCCCPCCCCCPCCCCC", and we use different id to annotate them. Therefore, the input from each poem consists of 3 items: the text, keywords, and a format template.

4 Models

4.1 LSTM based model

Long short-term memory, or LSTM, was introduced by (Hochreiter and Schmidhuber, 1997) to

Dynasty	Train		Valid		Test	
	5-char	7-char	5-char	7-char	5-char	7-char
Nanbei	2	-	-	-	1	-
Sui	56	3	10	-	9	1
Tang	1863	6872	368	284	435	339
Song	7460	43887	1499	1946	1799	2373
Liao	-	4	-	-	-	-
Jin	149	1159	41	52	48	64
Yuan	1260	6688	234	286	344	402
Ming	8968	32152	1828	1430	2340	1821
Qing	-	4	1	-	-	-
Sum	19758	89969	3981	3998	4976	5000

Table 1: Distribution of classical Chinese poems in different dynasties and genre types.

address the problem of vanishing and exploding gradients and preserving the information for a longer-term. Since it is designed to capture long-term dependencies in the sequential data, the network is suitable for many NLP tasks, including language modeling.

The core idea of the network is that it has a memory cell and special mechanisms called input gate, forget gate and output gate to filter the information. A memory cell, or cell state, is designed to collect the information about the input and runs along the entire input sequence. The input gate addresses what new information will be passed to the memory cell. The forget gate is used to regulate the amount of information that will be retained from the previous memory cell. Lastly, the output gate is used to output the information from the current cell.

The input gate (i_t) , forget gate (f_t) and output gate (o_t) are computed with the following equations:

$$\begin{aligned} i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \end{aligned} \tag{1}$$

For the input sequence $x_1, x_2, ..., x_n$, at each time step t, the network computes the hidden state (at each time step) as an elementwise multiplication of the output gate and the cell state. **??**

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot tanh(W_c x_t + U_c h_{t-1} + b_c) \\ h_t &= o_t \odot c_t \end{aligned} \tag{2}$$

Previous research on poetry generation using LSTM networks (Agarwal and Kann, 2020; Lau et al., 2018; Wen et al., 2015) has shown promising results and thus was the motivation for using it as one of the models for this project.

4.2 RNN: Encoder-Decoder Architecture

Encoder-decoder is a well-known, general-purpose architecture used in deep learning. The overall idea is to first encode an input into a contextualised representation which will encapsulate all important information about the input. Then, the contextualised representation, possibly with additional input, is used by a decoder to produce the final output relevant to the task. Both an encoder and a decoder can be modelled using standard deep learning mechanisms (MLP, CNN, RNN). Furthermore, they can consist of the same or different mechanisms, and their parameters can be shared as well.

In the area of natural language processing (NLP), this architecture is frequently used as a sequence-to-sequence (Seq2Seq) model (Sutskever et al., 2014) with the idea to transform a variable-length input sequence into a variable-length output sequence. Although seq2seq model was successfully applied to many NLP tasks, it turned out not to be well-suited for longer input sequences. Specifically, the contextualised representation obtained from the encoder would prefer later information in an input sequence while forgetting those from the beginning. The solution for this problem is to use the attention mechanism, which allows the decoder to attend all input tokens during each decoding step instead of a single contextualised representation. There are various types of this mechanism, but the overall idea is to calculate input tokens' attention scores for each decoding step individually. These scores, ideally, tell the decoder which information is relevant for the current step.

Our encoder-decoder model is based on (Bahdanau et al., 2016), applied for the machine translation task. The intuition behind using this model is to consider the poetry generation task as a translation (decoding) task - from a set of useful keywords into an actual poem. Specifically, the steps of the model are as follow:

- Tokens are embedded using a learnable embedding matrix (shared by encoder and decoder).
- Encoder encodes input tokens (keywords) using bi-directional GRU.
- Attention mechanism is applied over encoded tokens and previously generated ones to decide which input tokens are relevant for the current decoding step.
- Calculated attention weights and previously generated tokens are used by the decoder (unidirectional GRU) to generate the next token.
- Greedy and top-k approaches are used for the final decoding.

Keywords used by our model are extracted from examples (poems) using RAKE algorithm (Rose et al., 2010). The RAKE algorithm is a popular tool for feature extraction in the NLP area, e.g. for relevant keywords extraction (Xu et al., 2020). The desirable property of this algorithm is that it does not consider stopwords as possible candidates for keywords. Also, this algorithm might extract bigrams, but after a manual inspection, only the first word is useful; thus, only the first one would be kept in such cases.

4.3 Transformers

Transformer models have proved successful in natural language processing and computer vision tasks in recent years. The attention mechanism was proposed to improve the representation of dependencies between source and target in machine translation and later plays a crucial role in Transformer models. Content-based, addictive, general, and dot-product attentions are a family of popular attention mechanisms (Graves et al., 2014; Bahdanau et al., 2014; Luong et al., 2015). The scaled dot-product attention, or multi-head attention, is the foundation of Transformer models (Vaswani et al., 2017).

A basic form of self-attention involves three steps. First, derive weights of similarity between the current item and all other items in the sequence. Second, normalize the weights with the softmax function. Third, use the weights and corresponding sequence items to compute attention scores. A scaled dot-product attention introduce three additional weight matrices to help optimize models during training and produce query, key, and value. The multi-head part allows the model looking all aspects of the input at all levels.

The original transformer model consists of two main blocks: an encoder and a decoder. The encoder takes the input sequence and encodes contextual representations with a multi-head attention module. The decoder receives the processed input and produces sequences with a masked self-attention. The encoder and decoder part is inherited respectively by two landmark transformer-based pre-trained models, BERT and GPT (Radford et al., 2018; Devlin et al., 2019b). The success of these two models largely depends on the integration of self-supervised learning and Transformer.

4.3.1 BERT and BERT-CCPoem

Bidirectional Encoder Representations from Transformer (BERT), is an autoencoding language model based on the encoder Transformer stack. The pre-trained language models could provide a better understanding of target languages and are therefore applied to a wide range of downstream tasks.

Pre-training The data representation of BERT involves three embedding layers: token embedding, segment embedding, and position embedding. The [CLS] token is added only at the beginning of the first sentence, and [SEP] token is added at the end of every sentence. Segment embedding is used to distinguish the two given sentences. Position embedding provides information on word order since transformer models process all words in parallel and do not use any recurrence mechanism. The BERT model is pre-trained on two tasks: masked language modeling and next sentence prediction. In the masked language modeling task, 15% of the words are masked, and the model is trained to predict words in both directions. Next sentence prediction is a binary classification task, and the model is trained to predict whether sentence B follows sentence A. During the pre-training, subword tokenization algorithms like Byte Pair Encoding (BPE) are adopted to handle OOV words.

BERT-CCPoem The success of pre-trained language models inspire variants of BERT recently. BERT and ELMO are trained on general domain corpora such as news articles and Wikipedia. To handle NLP tasks in specific domains like biomedical or scientific texts, variants of BERT models are trained on corresponding specific corpora. Here, we use BERT-CCPoem, a BERT-based pre-trained language model especially for classic Chinese poetry, developed by Tsinghua University [1]. It is trained on CCPC-Full v1.0, a collection of classic Chinese poems consisting of 926,024 classical poems. It takes Chinese characters as a basic unit, and characters with frequency less than 3 are treated as [UNK]. The pre-training implementation is based on Hugging Face transformer library. The experiments in Section 6 shows BERT-CCPoem improves the understanding of classical Chinese poetry.

4.3.2 GPT-2

Generative Pre-trained Transformer or GPT-X model series, as is popularly known, is an unsupervised pre-trained transformer decoder based language model tasked to predict next words given some text. GPT-1 (Radford et al., 2018) first introduced in 2018 as a semi-supervised approach for language modeling is trained on a huge BookCorpus dataset. It is built upon a stack of 12 transformer decoder layers with the self-attention mechanism. This helps to learn long-range dependencies in a text that otherwise is not efficient in RNN based models. The goal is to perform transfer learning by learning a universal representation that requires little fine-tuning or adaptation to different NLP tasks. It is shown to perform exceptionally well on tasks like natural language inference, question-answering, sentence similarity and classification (Radford et al., 2018).

Let us take a deeper dive into how unsupervised pre-training works. Given a sequence of unlabeled tokens $\mathcal{U} = \{u_1, u_2, \cdots u_n\}$ which forms a context, the goal is to predict the next word. This is done by maximizing the likelihood

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \cdots, u_{i-1}; \theta) \quad (3)$$

where k is the size of context window and θ is some model parameter. Next, we have the transformer decoder given as

$$\begin{aligned} U &= (u_{-k}, \cdots, u_{-1}) \\ h_0 &= UW_e + W_p \\ h_l &= \text{transformer\_block}(h_{l-1}), l = 1, \cdots, L \\ P(u) &= \text{softmax}(h_L W_e^T) \end{aligned} \quad (4)$$

[1]https://github.com/THUNLP-AIPoet/BERT-CCPoem

Where U is the context vector, W_e is the token embedding matrix, W_p is the position embedding matrix, and L is the number of layers. We use both token and position embeddings since we process an entire sentence in one pass. To sum up, the text and position embeddings are passed through 12 layers of stacked transformer decoder blocks, and output is mapped to softmax to get prediction probabilities.

Next, we perform supervised fine-tuning for our target task using labeled data. Given a labeled dataset $\mathcal{C}$, input tokens $x^1, \cdots, x^m$ and output labels y, we take the last block of the transformer decoder and select the last entry of the sequence h_L^m. This is further passed through a linear layer to get probability y given some input tokens:

$$P(y|x^1, \cdots, x^m) = \text{softmax}(h_L^m W_y) \quad (5)$$

Next, we optimize a loss function corresponding to the above classification objective by maximizing:

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \cdots, x^m) \quad (6)$$

Finally, we also optimize the unsupervised loss L_1 on the labeled data with some weight λ.

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda \times L_3(\mathcal{C}) \quad (7)$$

The unsupervised loss helps achieve faster convergence even when we don't have enough data to train on. This is what we optimize during fine-tuning.

In our work, we use GPT-2 (Radford et al., 2019) which is a successor to GPT-1. It is trained on a larger dataset, 10X, that was used to train GPT-1 with over 1.5 billion parameters, almost ten times that of GPT-1. One of the drawbacks of GPT-1 was it's fine-tuning step in which the model needed to be adapted to every downstream task. GPT-2 overcomes this issue by eliminating the need to fine-tune. It differs from GPT-1 in a way the conditioning is done to model the probability of the next word. GPT-2 models the probability of the next word based on some context vectors as well as task information. The basic architecture is still a transformer-decoder. GPT-2 follows byte-pair encoding (BPE) (Sennrich et al., 2016b) system, which uses corpus statistics to decide how to segment a text into tokens. It has the advantage that this encoding mechanism requires a vocabulary of size only 256.

For our specific task to poetry style text generation, we needed to fine-tune GPT-2 since it is pre-trained to perform text generation but not of poetry style. The generated poems are conditioned upon a unique topic and a set of keywords. The keywords used by our model are extracted by exploiting contextual BERT embeddings of input text. To begin with, we use TF-IDF vectorizer to list important words or word phrases that capture the theme of each poem. This is further pipped through a spacy model, which only outputs words that are noun or noun phrases. The next step generates BERT embeddings of these output words and given poems and uses cosine similarity to obtain top word candidates that are most similar to embeddings of the poem.

4.4 CNN

Convolutional neural networks (CNN) have been very popular since their first use for computer vision years ago. Since then it has been very common among researchers for obvious reasons. Over the recent years, CNN has been a hot topic among computational linguists. Apart from being popular and outperforming other commonly used models in NLP like Long-Short Term Memory (LSTM) or Bidirectional Recurrent Neural Networks (BiRNN), CNNs have been shown to achieve greater accuracy. (Kim, 2014) reported significant improvement in results using CNNs compared to then used traditional methods for sentence classification. Since then CNN based models are effective for most natural language processing tasks like sentiment analysis, entity recognition, sequence labeling, etc.

In our topic-prediction system, we make use of the work proposed by (Kim, 2014) that uses a single-layered CNN on top of pre-trained word vectors. It demonstrated a conventional approach for sentence classification using CNN and reported that a single layered CNN with pre-trained word embeddings word2vec (Mikolov et al., 2013) outperformed traditional models not just in sentence classification but also sentiment analysis and question classification.

The proposed architecture consists of single CNN layered model on top of pre-trained static word embeddings fasttext (Mikolov et al., 2018).The implementaion follows the steps:

- Tokens are passed through an embedding layer learned on static word embeddings to obtain vectored representations.
- Convolution filters are employed for each ngram representations of a sentence.
- Each of these filters are mapped to non-linear activations followed by max-pooling to reduce possible overfitting and computational costs.
- The output to each layer is concatenated to obtain a flattened out single dimension tensor which are further fed into a full-connected layer. This gives a learned non-linear representation for each. It can be interpreted as sequence of probabilities for each topic that is used to perform classification.

5 Evaluation metrics

Evaluating an NLG model might be more challenging than designing and training it. The main reason is the nature of the task itself. In most other tasks there are a finite number of possible output values and one or more correct (e.g. text classification). In the NLG task, the output should be fluent, coherent, and on the topic natural language sequence, which leaves room for an infinite number of valid outputs. To best evaluate mentioned properties of our models, we selected the following metrics:

- **Perplexity** (**PPL**) for fluency. It is a standard measure for evaluating language models in NLP, and it tells how perplexed a model is predicting the next word. Formally, it is calculated as $PPL(W) = P(w_1, w_2, ..., w_N)^{-\frac{1}{N}}$, where N is number of tokens in the sequence W.
- **Keywords usage** (**KU**) for coherence. This metric evaluates how effective a model is in using provided keywords in a poem generation. Furthermore, it is calculated as $KU = G/T$, where G stands for number of generated provided keywords and T represents the total number of provided keywords.
- **Topic relevance** (**TR**) for on the topic evaluation. This metric aims to evaluate if a model generated a poem relevant to the provided topic and/or keywords. For this metric, we trained a topic prediction which will be explained in detail in the next section.
- **Weighted average precision** (**WAP**) for the topic prediction model. Average precision

(AP) lies between 0 and 1 and is computed from the prediction scores. AP is defined as AP $= \sum_n (R_n - R_{n-1})P_n$ where P_n and R_n are the precision and recall at the nth threshold. Weighted average computes a weighted sum of the precision score for each class and takes average across all classes. A higher WAP signify a better classifier.

- **BLEU** for adequacy and fluency. The BLEU score is a metric widely used in the open evaluation of machine translation (Papineni et al., 2002). It is also used in the previous studies of poem generation (Zhang and Lapata, 2014; Wang et al., 2016; Yan, 2016). We follow their settings and report sentence BLEU scores in a range from 0 to 1.

6 Experiments

6.1 English poetry

6.1.1 LSTM language model

The model consists of two single-layer unidirectional LSTM blocks. The first LSTM learns topic representation of the poem, which is then concatenated with the word embedding vector. The second LSTM block is used as a LM for poem generation. The LM is conditioned on the topic and is defined by the following equation:

$$p(x) = \prod_{i=1}^{n} p(x_i | \{x_0, ..., x_{i-1}\}, t) \qquad (8)$$

where x_i is a poem, and $x_0, ..., x_n$ is the sequence of the words in that poem.

For training, the input to the model consists of a 4-line poem with a maximum of 50 tokens and a single word topic of the poem. For shorter poems, a padding token was used. Each poem also contains BOP and EOP tokens, indicating the beginning and the end of the poem. The topic is represented as a sparse vector padded to the size of the poem. Word embeddings are used for both topic and poem vectors. The output of the LSTM layer is passed to the fully connected linear layer.

During generation, the model gets the topic word and BOP token as the input, both represented as single-digit tensors. The data split of 70%, 15% and 15% was used for training, test and validation parts, respectively. Total number of 22735 training examples were used.

A series of experiments were run varying the embedding size and the size of the hidden layer for both topic and words, as well as the learning rate. The initial parameters for the model were 100 hidden units for both topic and word-level LSTM, and embedding size of 300 and 100 units for topic and words, respectively. As seen in the Figure 1, increasing the learning rate had a negative impact on the training loss. The only significant difference in the training loss is observed for increased hidden size of the word-level LSTM.

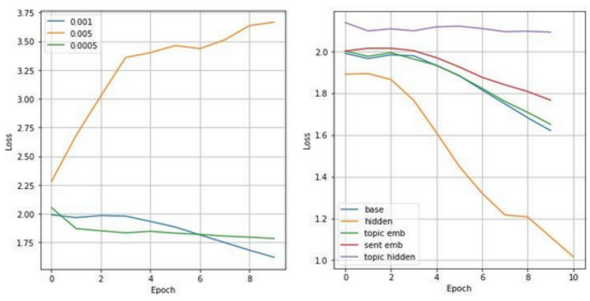

Figure 1: Training loss per epoch.

The experiments were ran on a minimally preprocessed dataset. However, due to immense overfitting, for the training with final hyperparameters further preprocessing was done. The final model setup has a hidden size of 256 units, all other parameters are the same as in the initial model. The model is trained during 17 epochs using Adam optimizer with the learning rate of 0.001 and a batch size of 1 on Nvidia K80 GPU, with one training epoch taking approx. 7 mins to execute.

6.1.2 Encoder-decoder

Initial testing was run with the base model (single layer bi-directional GRU with dropout 0.2, embedding size 126, hidden size 256, batch size of 32 examples, input size of 5 keywords and maximum poem length of 100 tokens). For all experiments, Adam optimizer was used, and experiments were conducted to establish the best parameters setup for the final model training. Dramatical overfitting of the model can be seen in the figure 2 - right with the minimal (e.g. lowercasing) dataset preprocessing. This suggested a poor mismatch between training and validation set distributions. After a detailed manual inspection, it was noticed that the whole corpus is overwhelmed with non-English characters (e.g. Chinese letters, special characters like $, etc.). Therefore, a strict cleaning is performed, leaving only English alphabet characters and lowercasing the obtained corpus.

Results of experiments on the cleaned corpus can

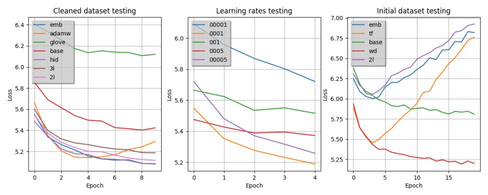

Figure 2: Validation losses for encoder-decoder model experiments.

be seen in figure 2 - left. It can be observed that increasing embeddings size (**emb**), hidden size (**hid**) or number of layers (**2l** - 2 layers and **3l** - 3 layers) improved performance consistently. On the other hand, changing Adam optimizer with AdamW led to overfitting again. More interestingly, applying pre-trained Glove embeddings (**glove**) showed worse performance compared to learning embeddings from scratch. Lastly, results of experiments for establishing the initial learning rate for the optimizer are presented in figure 2 - center.

The final dataset consists of more than 40000 examples. Validation and test sets have 1000 examples each, uniformly represented among topics (6 or 7 per topic), while the rest of the examples are used for the training set. Each example has 3, 4 or 5 lines of poem and for each one of the 3, 4 or 5 keywords are extracted.

The final model setup is a 2-layer bi-directional GRU encoder and uni-directional GRU decoder. Both, embedding and hidden, have a size of 256. Adam optimizer is applied with an initial learning rate 0.001 and linear decreasing after every 2 epochs with no validation loss improvement. Additionally, teacher forcing is applied during training. Training is run for 50 epochs with early stopping after 10 epochs without validation loss improvement. All other parameters are the same as the base model. A single Nvidia GeForce RTX 2060 GPU was used for training (average training time per epoch was 14 minutes).

6.1.3 Conditional GPT-2

As discussed in Section 4.3.2 the model takes in a sequence consisting of poem-topic, set of keywords and the poem. This is tokenized using GPT-2's tokenizer with special tokens BOS, EOS, PAD, SEP and UNK added to get contextual embeddings. The tokenizer employs BPE encoding scheme to split words into a sequence of characters rather than words. We fixed the maximum sequence length to 124 with padding enabled. This parameter is important as this is the maximum length that GPT-2 can process. The dataset is split into 80-20 train-validation ratio and is ready to feed the model.

We load the pre-trained GPT-2 model using a model configuration that has information of all special tokens added during context encoding and resize the token embeddings. We use Adam optimized with 10^{-4} learning rate and perform a gradual decay of learning rate using a scheduler. This helps optimize the training process and reduce training time. The fine-tuning procedure is run for 8 epochs that take approx. 22 minutes for each epoch on a single RTX6000 24 GB GPU card.

We employ top-k sampling and beam search to generate poems by supplying two conditioning parameters: (1) topic (2) set of five keywords extracted from that topic in the training set. Keywords are extracted using `WordWise` [2]. We compute PPL, KU and TR scores for each generated poem and report their mean across all generated poems.

6.1.4 Topic Prediction Model

Sentences were tokenized using NLTK's word tokenizer and pre-trained word embeddings from (Mikolov et al., 2018) were used to convert word into vector representations. It consists of 1M word vectors trained on various Wikipedia articles, UMBC web base corpus, and stamt.org news dataset, constituting about 16B tokens. When defining the model architecture, we load these pre-trained embeddings of dimension 300 in the Embedding layer. Words in the input absent in the pre-trained embeddings are initialized randomly and can be updated accordingly during training using appropriate arguments.

The based model was trained on all 11467 poems with 144 topics. It used convolution filters of three sizes 2, 3, 4, and 5 (each mapped to a ReLU activation) that emulate the bi-gram, tri-gram, four-gram and five-gram models to obtain multiple feature representations. All these features are further passed through the max-pooling layer of stride 3 to reduce the dimension of the input feature representation. These are further concatenated, flattened and a dropout probability of 0.15 is applied to prevent any overfitting. The output features are passed through a fully connected sigmoid layer to distribute the output probability over 144 topics. We used Adam optimizer with a cross-entropy loss function. This setup performed poorly in capturing

[2] https://github.com/jaketae/wordwise

the context of poems and labelling them into appropriate topics. Overfitting was observed both on validation and test set where the loss decrease in train and validation set were inconsistent and became constant after a few epochs (Figure 3). Similar behaviour was observed with the F1 score (Figure 4). We believe that samples in NeuralPoet dataset (each poem mapped to one topic) are not closely related (e.g. truth and justice, joy and happy), and the model fails to make a clear distinction. Further, we also found that the dataset was unclean and full of illegal non-English characters, which also led to decreased performance. The training was performed for 50 epochs on a single NVIDIA GeForce RTX 2060 GPU that took 1.53 mins to execute.

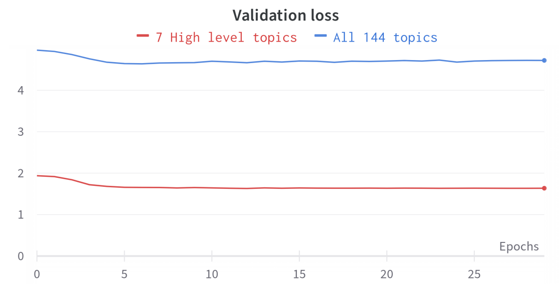

Figure 3: Validation losses for topic-prediction model. Losses became almost constant after 20 epochs. For model trained on all 144 topics, losses can be seen to increase slightly after 8 epochs.

We trained a second model with only seven high-level topics (*love, nature, life, romantic, freedom, culture, suicide*) to inspect model behaviour when target labels are not closely related. Model parameters were kept the same as the base model for comparison. As evident from Figure 3 validation loss showed a significant decrease indicating an increase in performance results. F1 score too showed a drastic improvement (Figure 4).

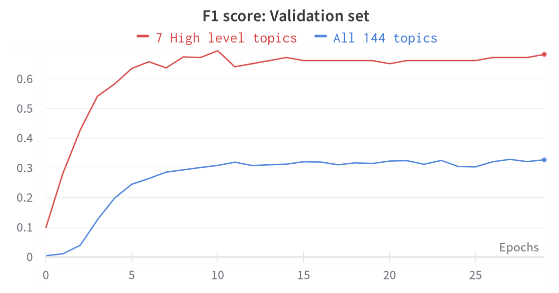

Figure 4: F1 score on validation set for topic prediction model. Significant improvement can be seen when model was trained on seven high level topics.

In the final step, we performed topic classification on each of the generated poems via encoder-decoder based model and GPT-2 transformer based model. Since the generated poems were collectively based on all 144 topics, we use the base model for topic prediction to study its performance on machine-generated poems. To no surprise, the prediction accuracy was worse than the test data. We analyse possible reasons in Section 8.

6.2 Chinese poetry

6.2.1 Experimental Design

In the experiments of Chinese poetry generation, we control three factors (language representations, types of neural networks, and decoding strategies) and answer the following questions:

- How language representations affect the performance of neural network models in generating poems? What is the difference between word embeddings learned through the embedding layer and loading pre-trained contextual representations from BERT models?
- Given the general sequence to sequence architecture, which type of neural network performs better? GRU, LSTM or vanilla Transformer?
- Among greedy, top-k, and beam search decoding strategies, which one provides more fluent and elegant poems?

Language representations have two levels: word2vec and BERT. Types of neural networks have three levels: GRU, LSTM and vanilla Transformer. Decoding strategies have three levels: greedy, top-k, and beam search. Therefore, we get an experiment of $2 \times 3 \times 3 = 18$ conditions.

6.2.2 Experimental Settings

We adopt a general encoder-decoder architecture and three types of neural network models, so we have three sub-types: Seq2seq-GRU, Seq2seq-LSTM, Seq2seq-Transformer.

For all models, we use the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.98$ and $\epsilon = 10^{-9}$. The learning rate is 0.0001. We also take dropout as a regularization technique to each layer of encoder and decoder. The dropout rate is 0.1. Early stop strategy is applied to all models after observing 5 worse validation scores. The embedding size of word2vec is 768. The word2vec embedding initializes the

Model	PPL	KU	TR
LSTM LM	4805.970	-	?
Enc-Dec	131.543	0.24	0.09
Cond. GPT-2	15.672	0.20	0.21

Table 2: Overall comparison of LSTM, encoder-decoder architecture and conditional GPT-2 models for English poetry generation. As a general rule PPL scores can be considered when deciding over quality of generated poem. To our observation, GPT-2 generated high quality poems.

weight matrix from normal distribution $N(0,1)$. BERT-CCPoem v1.0 is a pre-trained model with 8 layers, 512 hidden size, and 8 heads. We train these models on a server with 4 GeForce GTX TITAN X GPUs.

In the decoding phase, we set $k = 32$ in top-k sampling and $beam = 5$ in beam search.

Seq2seq-GRU, Seq2seq-LSTM Both encoder and decoder parts have one bidirectional layer. The hidden size in feed-forward layers is 1024. The GRU models with word2vec embedding and the BERT embedding took 9 epochs (about 90 mins) and 10 epochs (about 90 mins), respectively to finish training. The LSTM models with word2vec embedding and BERT embedding spent 9 epochs (about 105 mins) and 9 epochs (about 90 mins) respectively in training.

Seq2seq-Transformer Both encoder and decoder parts have 6 layers. And feed-forward layers have a hidden size of 2048. The models with word2vec embedding stopped training after 17 epochs (about 57 mins). And the models with BERT embedding took 16 epochs (about 51 mins) to finish training.

7 Results

We present final results in Table 2 that shows a birds-eye view of important results obtained for this project at a glance.

English poetry We also PPL, KU and TR scores on poems generated using conditional GPT-2 model in Figure 3. We use two decoding strategies- top-k sampling and beam search and report the results on only seven high-level topics.

To understand the intricacies of the topic-prediction system discussed in Section 6.1.4 we train the model following two strategies: (1) first we train the model on all 144 topics (2) then we train the model on only seven high-level topics (we call it HLT) and report the weighted average precision score (WPA) and accuracy as observed on test set. The results reported can be found in Table 7

	Top-K sampling			Beam search		
Topics	**PPL**	**KU**	**TR**	**PPL**	**KU**	**TR**
love	**8.758**	0.22	0.2	11.423	0.12	0.2
nature	17.621	0.30	0.2	9.672	0.20	0.2
life	**10.937**	0.22	0.2	21.419	0.14	0.3
romantic	**9.054**	0.20	0.3	27.268	0.28	0.5
freedom	**7.814**	0.24	0.3	10.322	0.18	1.0
culture	**10.545**	0.14	0.1	11.838	0.16	0.5
suicide	31.53	0.12	0.1	12.469	0.14	0.2

Table 3: PPL, KU and TR scores on poem generated using fine-tuned GPT-2 based on top-k sampling and beam search. The results are reported for only seven high level topics. Poem generated using top-k sampling generates better quality poems than beam-search.

Chinese poetry We present BLEU-1, BLEU-2 and KU scores for Chinese poetry generation in table 4, table 5,and table 6. 5-Char and 7-Char indicate Chinese poems with 5 characters per line and 7 characters per line, respectively. GRU, LSTM, Transformer represent the model with GRU , LSTM and vanilla Transformer as backbone, respectively. Base and BERT denote the embedding initialized from $N(0,1)$ and pre-trained BERT, respectively. And greedy, top-k, and beam indicate three types of decoding strategies of greedy decoding, top-k decoding with k=32 and Beam search. In all 18 conditions, Transformer+base+beam and Transformer+bert+greedy both achieve the best BLEU-1 score (0.364), Transformer+base+beam achieve the best BLEU-2 (0.246), and Transformer+bert+beam achieves the best KU score (0.960), for all Chinese poems.

8 Discussion

LSTM Language Model (English) The results suggest that a proposed model architecture is insufficient for the task in question. First of all, the model failed to produce good quality results. Overfitting was observed even after more detailed pre-processing of the dataset was done. Increasing validation loss indicates the model does not generalize well to unseen data. Second of all, conditioning on the topic does not lead to the intended result – this could be either be due to the model architecture in general, or due to the choice of topic vector representation. Finally, the topic alone is not enough

Model	5-Char			7-Char			All		
	BLEU-1	BLEU-2	KU	BLEU-1	BLEU-2	KU	BLEU-1	BLEU-2	KU
GRU+base	0.328	0.219	0.691	0.291	0.182	0.735	0.310	0.200	0.713
GRU+bert	**0.368**	**0.241**	**0.792**	**0.322**	**0.207**	**0.861**	**0.345**	**0.224**	**0.826**
LSTM+base	0.325	0.217	0.683	0.279	0.177	0.710	0.302	0.197	0.697
LSTM+bert	**0.362**	**0.242**	**0.784**	**0.318**	**0.204**	**0.841**	**0.340**	**0.223**	**0.813**
Transformer+base	**0.390**	0.256	0.895	0.333	0.218	0.958	0.361	0.237	0.927
Transformer+bert	**0.390**	**0.260**	**0.917**	**0.337**	**0.219**	**0.971**	**0.364**	**0.240**	**0.944**

Table 4: Comparison on the impact of different embedding initialization on Chinese poetry generation. BLEU and KU scores of Chinese poems generated by greedy decoding strategy are reported here.

Model		5-Char			7-Char			All		
		BLEU-1	BLEU-2	KU	BLEU-1	BLEU-2	KU	BLEU-1	BLEU-2	KU
base+greedy	GRU	0.328	0.219	0.691	0.291	0.182	0.735	0.310	0.200	0.713
	LSTM	0.325	0.217	0.683	0.279	0.177	0.710	0.302	0.197	0.697
	Transformer	**0.390**	**0.256**	**0.895**	**0.333**	**0.218**	**0.958**	**0.361**	**0.237**	**0.927**
base+top-k	GRU	0.265	0.193	0.518	0.236	0.152	0.535	0.250	0.172	0.526
	LSTM	0.257	0.199	0.488	0.226	0.148	0.520	0.241	0.173	0.504
	Transformer	**0.371**	**0.248**	**0.854**	**0.312**	**0.205**	**0.919**	**0.342**	**0.227**	**0.886**
base+beam	GRU	0.363	0.253	0.878	0.314	0.213	0.934	0.338	0.233	0.906
	LSTM	0.359	0.250	0.846	0.309	0.206	0.900	0.334	0.228	0.873
	Transformer	**0.394**	**0.268**	**0.934**	**0.335**	**0.224**	**0.972**	**0.364**	**0.246**	**0.953**

Table 5: Comparison on the impact of GRU, LSTM and Transformer. BLEU and KU scores of different networks with the same decoding and embedding are reported here.

Model	5-Char			7-Char			All		
	BLEU-1	BLEU-2	KU	BLEU-1	BLEU-2	KU	BLEU-1	BLEU-2	KU
GRU_greedy	**0.368**	0.241	0.792	**0.322**	**0.207**	0.861	**0.345**	0.224	0.826
GRU_top-k	0.308	0.213	0.639	0.266	0.164	0.670	0.287	0.188	0.654
GRU_beam	0.349	**0.249**	**0.863**	0.300	**0.207**	**0.915**	0.325	0.228	**0.889**
LSTM_greedy	**0.362**	0.242	0.784	**0.318**	0.204	0.841	**0.340**	0.223	0.813
LSTM_top-k	0.301	0.212	0.622	0.263	0.170	0.649	0.282	0.191	0.635
LSTM_beam	0.356	**0.253**	**0.884**	0.299	**0.208**	**0.926**	0.328	**0.231**	**0.905**
Transformer_greedy	0.390	0.260	0.917	**0.337**	0.219	0.971	**0.364**	0.240	0.944
Transformer_top-k	0.372	0.252	0.872	0.318	0.211	0.937	0.345	0.231	0.904
Transformer_beam	**0.391**	**0.266**	**0.946**	0.334	**0.221**	**0.974**	0.362	**0.244**	**0.960**

Table 6: Comparison on the impact of different decoding strategies. BLEU and KU scores of Chinese poems generated by models with Bert embedding are reported here.

for the model to generate topic-relevant output. In the best case scenario, the output contains the topic word itself, while the rest of the words are not closely related to it. Overall, the model is able to learn to generate poems, however their quality is poor. Produced output resembles poetry, although it is mostly due to the vocabulary range. Generated poems lack coherence and grammaticality, which is an expected outcome according to high perplexity.

Encoder-Decoder (English) Results obtained for Encoder-Decoder model on English corpus suggest following: *(i)* Better corpus is needed in order to properly train a model from scratch. Even after additional cleaning and preprocessing, the difference between validation and train losses was significant (approx. 3.0). This might be due to considerable differences between data distributions in two sets. *(ii)* Modeling poetry generation as a translation task might not be the best option. First of all, the encoder part could not represent keywords in a meaningful way for a decoder to learn to use them properly. A potential solution might be to use a non-sequential type of encoder (e.g. transformers). Next, the model struggled to be fluent. Since this is not a task-specific problem, it might be solved using a large pre-trained language model (our results with conditional GPT-2 are promising). Lastly, using only topic-related keywords is not enough for a model to generate poems on that topic. Additional control factors for a topic might help, but since the model is trained from scratch, it probably would not be sufficient. Therefore, a more sophisticated decoder component with a lot more data could be a solution.

Conditional GPT-2 (English) Results reported for fine-tuned GPT-2 model in Table 2 suggests that transfer learning generates the best quality poems. We conclude this by analyzing the PPL scores, where a lower score indicates more fluent and meaningful poems. Further investigation is done in Table 3 where we observe that the top-k sampling decoding scheme performed better than beam search. We also observed more word repetitions in poems decoded using beam search. We also find that top-k sampling better retains keywords in its poems by comparing the KU scores for both decoding strategies. Lastly, we computed TR scores that showed that the topic-prediction model could not predict correct topics to generate poems. We believe that although GPT-2 generated poems captured some intricacies of a mastered poet, the topic-prediction model was trained from scratch on poems from amateur poets, resulting in a low-performing classifier. Examples of generated poems can be found in the Appendix.

	WAP	**Accuracy (%)**
All topics	0.3993	28.81
HL topics	0.6063	58.77

Table 7: Weighted average precision (WAP) scores and accuracy on test set for topic-prediction model. We followed two strategies (1) train the model on all topics (2) train the model only seven high level (HL) topics

Topic prediction Model To study the performance of the topic-prediction model, we compute WAP and accuracy scores and report in Table 7. We first train the model on 144 topics and observe on test set that 28% of topics are classified correctly while the accuracy increases to 58% when trained on high-level topics. Higher WAP on HL topics indicates that the model makes more relevant predictions, i.e., we achieve around 51% improvement over WAP by eliminating closely related topics. We believe that these high-level topics are distinct from each other, and hence the model finds it easier to make predictions. Moreover, upon inspection, we conclude that these poems are not well-curated, poorly written and do not very well align with their topic. It was also unclean and overwhelmed with illegal characters that brought some noise in its model distribution. We believe that poems written by a mastered and experienced poet are suitable for performing such topic-classification tasks.

Encoder-Decoder (Chinese) In Table 4, we compare the effects of word embedding learned in the embedding layer and contextual representations from the pretrained BERT-CCPoem model on the performance of poetry generation tasks. The result shows all models with BERT-CCPoem representations shows a better performance in both 5-char and 7-char poem generation. It implies that contextual representations encode useful knowledge to generate poems in the downstream task since BERT-CCPoem was pretrained on a relevant domain-specific corpora.

In Table 5, we compare the performance of different neural network models in generating poems, and the result shows the transformer models deliver the best performance. Even the length of

Chinese poems is no longer than 28 characters, the transformer model still shows it performs better in capturing long-range contextual dependencies than RNNs.

In Table 6, we compare the impact of different decoding strategies. The greedy and beam search algorithms could deliver higher scores in some specific cases, respectively. It implies that there is no silver bullet for all models to generate fluent and elegant poems. In addition, we also notice that some keywords receive excessive 'attention' from models during training; therefore, models produce repeated keywords in a single line during the decoding phase. Therefore, how to add proper parameters of penalty automatically in decoding is a key question to explore in the future.

9 Conclusion and Future work

Based on all experiments that we conducted, we came up with the following future work suggestions:

- **Corpus-wise.** In case of English corpus, rigorous data collection and processing should be performed. There are a lot of non-English characters, and poems should be stored in a more structured way (e.g. clear distinction between an actual poem and additional information). Furthermore, keywords (and other factors) labelling would be nice to have. This is especially important for models trained from scratch, which also requires a larger corpus. For Chinese corpus, we do not have sufficient resources to train a word tokenizer for classical Chinese in the preprocessing, so we adopt a character-based representation. We believe an efficient word tokenizer would enrich more linguistic information for models in the future.

- **Model-wise.** According to our results, using large pre-trained yields the best results. Therefore, we think that transfer learning approach might be an interesting next step to try. This strategy achieves good performance on other tasks. Thus, designing a task-specific step is a reasonable way to go. This way, the problem with fluency should be overcome, while task-specific steps would guide a model to learn how to perform the task-specific operations (e.g. select useful keywords).

- **Decoding-wise.** In the English task, top-k shows better performance over greedy decoding, so it might be reasonable to try similar strategies. Top-k decoding introduces better diversity and less repetition during the generation process. In Chinese tasks, there is not one decoding strategy that could always achieve better performance. Penalty parameters are crucial for beam search decoding to eliminate repeated keywords, which is a question to explore further.

References

Rajat Agarwal and Katharina Kann. 2020. Acrostic poem generation. *arXiv preprint arXiv:2010.02239*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate.

Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. 2012. Markov constraints for generating lyrics with style. In *Ecai*, volume 242, pages 115–120.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full-face poetry generation. In *ICCC*, pages 95–102.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *NMT@ACL*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. Copenhagen, Denmark. Association for Computational Linguistics.

Franky Franky. 2013. A rule-based approach for karmina generation. In *Proceedings of the 2013 NAACL HLT Student Research Workshop*, pages 24–31.

Pablo Gervás. 2000. Wasp: Evaluation of different strategies for the automatic generation of spanish verse. In *Proceedings of the AISB-00 symposium on creative & cultural aspects of AI*, pages 93–100.

Pablo Gervás. 2001. An expert system for the composition of formal spanish poetry. In *Applications and innovations in intelligent systems VIII*, pages 19–32. Springer.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Zhipeng Guo, Xiaoyuan Yi, Maosong Sun, Wenhao Li, Cheng Yang, Jiannan Liang, Huimin Chen, Yuhui Zhang, and Ruoyu Li. 2019. Jiuge: A human-machine collaborative Chinese classical poetry generation system. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 25–30, Florence, Italy.

Prakhar Gupta, Jeffrey Bigham, Yulia Tsvetkov, and Amy Pavel. 2021. Controlling dialogue generation with semantic exemplars. Online. Association for Computational Linguistics.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification.

Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. 2018. Deep-speare: A joint neural model of poetic language, meter and rhyme. *arXiv preprint arXiv:1807.03491*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Hisar Maruli Manurung. 1999. A chart generator for rhythm patterned text. In *Proceedings of the First International Workshop on Literature in Cognition and Computer*, pages 15–19.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Joanna Misztal and Bipin Indurkhya. 2014. Poetry generation system with an emotional personality. In *ICCC*, pages 72–81.

Hugo Gonçalo Oliveira. 2012. Poetryme: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

Hugo Gonçalo Oliveira, Raquel Hervás, Alberto Díaz, and Pablo Gervás. 2017. Multilingual extension and evaluation of a poetry generator. *Natural Language Engineering*, 23(6):929–967.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. *Automatic Keyword Extraction from Individual Documents*.

Michal Rott and Petr Červa. 2016. Speech-to-text summarization using automatic phrase extraction from recognized text. In *Text, Speech, and Dialogue*, pages 101–108, Cham. Springer International Publishing.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? how controllable attributes affect human judgments. Minneapolis, Minnesota. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Controlling politeness in neural machine translation via side constraints. San Diego, California. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

Aleksey Tikhonov and Ivan P Yamshchikov. 2018. Sounds wilde. phonetically extended embeddings for author-stylized poetry generation. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 117–124.

Berty Chrismartin Lumban Tobing and Ruli Manurung. 2015. A chart generation system for topical metrical poetry. In *ICCC*, pages 308–314. Citeseer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. 2016. Chinese song iambics generation with neural attention-based model. In *IJCAI*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Martin Tsan Wong, Andy Hon Wai Chun, Qing Li, SY Chen, and Anping Xu. 2008. Automatic haiku generation using vsm. In *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering*, volume 7. Citeseer.

Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. 2020. MEGATRON-CNTRL: controllable story generation with external knowledge using large-scale language models. *CoRR*, abs/2010.00840.

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

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

A Appendix

A.1 Generated examples

LSTM LM: Top-K decoding (English)

1. **Topic**: *love*

   i'd force love more freezing
   a river once upon my
   in blest bearing pauses passion for another

2. **Topic**: *anger*

   searing spark up fast asleep
   with hopes lamp angered keep
   by thy numbers can fly these years
   will not turn loss

3. **Topic**: *trust*

   strength entering envelopes fill beneath
   christian kids beloved lips breathing
   water with mine making hand
   pride but not any human kind

Encoder-Decoder: Top-K decoding (English)

- **Keywords:** friend, feel, cheer, sister

  sister are my sister
  you are my sister
  you are me
  you feel you

Encoder-Decoder: Greedy decoding (English)

- **Keywords:** dare, love, compare, copper, soft

  when you have to see the heart
  i you taste lips
  on
  cheek
  i a was in its of love me

Conditional GPT-2: Top-K decoding (English)

1. **Topic**: *love*
   Keywords: roses, water, peacock, strands, puppy love

   You are beautiful
   and there is nothing wrong with you
   You are free
   and so are the puppy loves
   they don't want you
   Because you have nowhere else to go
   You may have a lot of money
   and you will have lots of shoes to put on
   But your so very own and so very happy
   Because you have seen the other side of things
   and the little puppy loves got away with it
   Because you have no other choice
   But you will stay and watch
   till the water comes!

2. **Topic**: *culture*
   Keywords: indian culture, monasteries, jester, meditation, distasteful gestures

   This is life as we know it -
   The first year was "Called "Ammurition"

   Then the land was mapped with wonders
   Of Indus and Sphinxes
   The Jura stone and the marble
   Made of silver, "All" seemed like gold
   To the people of this land;
   But what they didn't know was "India";
   How many things it made
   Even today?
   No one knows, but what we know
   As far back as the late eighteenth century
   As people in Hanukkah had built temples
   They knew the wonders untold
   But they didn't know at all what they'd come up with
   As a result,
   Of what they had created in our
   Indias and Sphinxes
   And of all our modern misadventures
   This is life as we know it.

Conditional GPT-2: Beam search (English)

1. **Topic**: *love*
   Keywords: key, pure bliss, pain, flame, peacock

   Love is pure bliss.
   It doesn't require a lot of work,
   just a little bit of love and it will go a long, long way!

   Love can be as simple as a key
   or as sweet as the heat of a candle
   and it can last for hours and hours
   until the end of the world.

2. **Topic**: *culture*
   Keywords: mysticism, prayers, sages, future fuse, ruin

   The Future fuse

   The mysticism of the future
   Will fuse with the sage tradition
   Of the past and the present
   And bring together
   An amalgation of mysticisms
   That will combine
   With modern conveniences
   To bring order to the world.

Seq2seq-GRU (Chinese)

- **Embedding type:**Bert embedding
  Decoding: greedy decoding
  keywords: 绿水，青山，醉眼，离愁

绿水青山绕郭城
离愁无处不相思
只今醉眼无人到
一夜风前一两声

- **Embedding type:**Bert embedding
  Decoding: top-k decoding
  keywords: 暮云, 春雨, 行人, 缥缈

  浮生缥缈暮云中
  一树黄花独倚丛
  数尽行人归未得
  几回春雨过溪风

- **Embedding type:** No pretrained embedding
  Decoding: greedy decoding
  keywords: 琉璃, 明河, 东去, 星

  东去明河水上天
  琉璃一片落星光
  不知何处人间世
  只有当年一寸无

- **Embedding type:** No pretrained embedding
  Decoding: top-k decoding
  keywords: 明月,清光,徘徊,终夜

  不识清光有
  徘徊终夜流
  独坐无语想
  犹自怯寒身

Seq2seq-Transformer (Chinese)

- **Embedding type:**Bert embedding
  Decoding: greedy decoding
  keywords: 江水,浮云,秋色,美人

  江水浮云秋色多
  美人何处不相过
  一声一笛惊相逐
  不似当年梦里过

- **Embedding type:**Bert embedding
  Decoding: top-k decoding
  keywords: 茅檐,长条,高楼,寒鸦

  茅檐翠雨度寒鸦
  一枕高楼醉似霞
  今日长条谁是伴
  半江晴色散梅花

- **Embedding type:** No pretrained embedding
  Decoding: greedy decoding
  keywords: 寒潭,兰桡,月出,携酒

  月出寒潭水
  兰桡泛钓船
  携酒看不足
  随意到江边

- **Embedding type:** No pretrained embedding
  Decoding: top-k decoding
  keywords: 孤舟,蓑笠,寒江,鸟

  云外寒江水
  沙平鸟飞还
  蓑笠孤舟外
  孤舟载月间

A.2 Word cloud of keywords

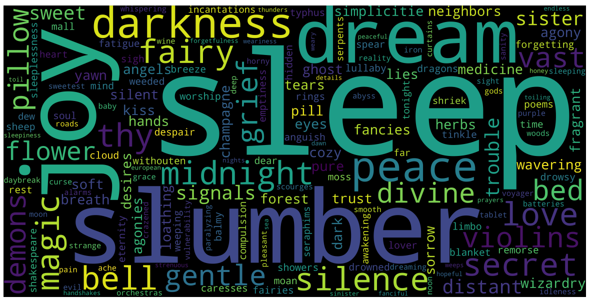

Figure 5: World cloud of topic- *sleep* poems. The keywords were extracted using scheme discussed in Section 4.3.2