Sonnet Generation by Training on Non-poetic Texts with Discourse-level Coherence and Poetic Features

Anonymous ACL submission

Abstract

Poetry generation, and creative language generation in general, usually suffers from the lack of large training data. In this paper, we present a novel framework to generate sonnets that does not require training on poems. We design a hierarchical framework which plans the poem sketch before decoding. Specifically, a content planning module is trained on non-poetic texts to obtain discourse-level coherence; then a rhyme module generates rhyme words and a polishing module introduces imagery and similes for aesthetics purposes. Finally, we design a constrained decoding algorithm to impose the meter-and-rhyme constraint of the generated sonnets. Automatic and human evaluation show that our multi-stage approach without training on poem corpora generates more coherent, poetic, and creative sonnets than several strong baselines.[1]

1 Introduction

A sonnet is a fourteen-line poem with rigorous meter-and-rhyme constraints. In this paper, we aim at generating full-length sonnets that are logically and aesthetically coherent, without training on poetic texts. There are several challenges for this ambitious goal.

First, there are limited number of sonnets available to train a fully supervised model. The only resource is a mere 3,355 sonnets collected by Lau et al. (2018) in Project Gutenberg (Hart, 2004), one of the largest free online libraries for English literature. While it is possible to train on related corpus such as general poems or English lyrics (Ghazvininejad et al., 2016), such approaches are not applicable to other languages if sizable poetry/lyrics data do not exist. Moreover, even if large-scale creative texts exist, learning from and mimicking existing corpora is *not* creative by definition and is unlikely to result in novel content.

[1]Our code and data will be released upon acceptance.

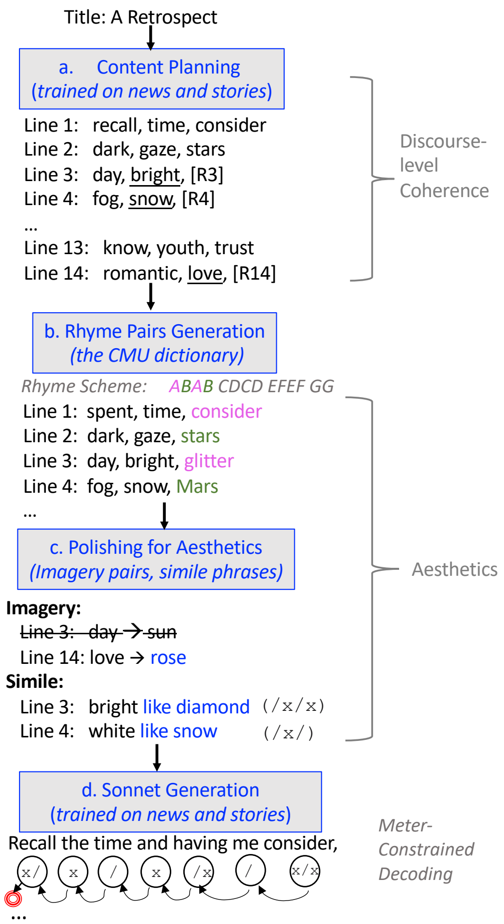

Figure 1: An overview of our approach. The content planning module generates keywords while maintaining discourse-level coherence. The second module form rhyming pairs and the polishing module enrich the imagination and add poetic flavor. Finally, we generate the sonnet with a meter-constrained decoding algorithm. Note that all four steps do not require poem/sonnet data.

Second, coherence remains a known issue among previous works on poetry generation. Existing works mainly focus on conforming to the format constraints (i.e., meter-and-rhyme), or generating a small stanza with a typical length of four (Lau et al., 2018; Liu et al., 2019; Yi et al., 2020). For full-length sonnets, Ghazvininejad et al. (2016) propose to use topical words as rhyme words to achieve topical relatedness, but the generated sonnets are not discourse-level coherent. They later generate discourse-level coherent English sonnets through French-English translation (Ghazvininejad et al., 2018). Generating logically and aesthetically ordered poems without relying on content translation from other languages remains a challenge.

With all these in mind, we propose to *generate sonnets without any poetic data*. Our framework, **SONG**, is shown in Figure 1, and consists of four components: content planning, rhyme pairing, polishing for aesthetics and final decoding. The first three steps provide salient points for the sonnet as a sketch. The last step is responsible of "translating" the sketch into well-formed sonnets.

Specifically, the content planning step is trained on a combination of news and stories to generate keywords per sentence, which aims at equipping the model with *general world knowledge to construct a coherent text world*. However, the language used by poems is different from that of standard texts because it follows certain rhetorical rhythm and is full of vivid descriptions that appeals to readers' senses and imagination (Gibbs Jr et al., 1994). To this end, in the polishing step we leverage external knowledge and incorporate two figurative speeches (i.e., simile and imagery) into the planned keywords to boost vividness and imagination. Step rhyme and final decoding steps are designed to impose the meter-and-rhyme constraints.

While previous works on *plan-and-write* rely on training data from the target task domain (e.g., stories, dialogue conversations) for improved performance, we on the other hand adopt content planning to disentangle the training from the decoding step and circumvent the shortage of training data. We summarize our contributions as follow:

- We propose **SONG**, a sonnet generation framework that does not require training on poem data, by disentangling training from decoding. Specifically, we first learn to predict context and rhyme words from news and story dataset, and then polish the predicted keywords to promote creativity. A constrained decoding algorithm is designed to impose the meter-and-rhyme constraints while incorporating the keywords.

- We develop two novel evaluation metrics to measure the quality of the generated poems: automatic format checking and novelty evaluation (i.e., diversity and imageability).

- Human evaluation shows that **SONG** generates more discourse-level coherent, poetic, creative, and emotion-evoking sonnets than its baselines.

2 Background

In this section, we introduce the characteristics of sonnets in terms of structure, meter and rhyme. We then define important terminologies.

2.1 The Structures of Sonnets

We aim to generate the two most representative sonnets: *Shakespearean* and *Petrarchan*. Sonnets make use of rhymes in a repeating pattern called **rhyme schemes** as shown in Table 1. For example, when writing a Shakespearean sonnet, poets usually adopt the rhyme scheme of ABABCDCDEFEFGG. Although all sonnets have 14 lines, a Petrarchan sonnet consists of an 8-line stanza called an octave followed by a 6-line stanza called a sestet. On the other hand, a Shakespearean sonnet consists of three 4-line quatrains and a 2-line rhyming couplet which leaves the reader with a lasting impression.

	# of Lines	**Iambic Penta**	**Structure**	**Rhyme Scheme**
Shakespearean Sonnet	14	Yes	3 quatrain 1 couplet	ABAB CDCD EFEFGG
Petrarchan Sonnet	14	Yes	1 octave 1 sestet	ABBA ABBA CDECDE

Table 1: Comparison between a Shakespearean sonnet and a Petrarchan sonnet.

2.2 Meter Constraints

Most sonnet conform to iambic pentameter, a sequence of ten syllables alternating between unstressed (`x` or `da`) and stressed syllables (`/` or `DUM`). Strictly speaking, each line reads with the rhythm ($\texttt{da-DUM}$)^5, which enhances the tone for the poem and operates like an echo. In reality, there are many rhythmic variations. For example, the first foot is often reversed to sound more assertive, and can be written as ($\texttt{DUM-da}$ * ($\texttt{da-DUM}$)^4). Another departure from the standard ten-syllable pattern is to append an addition unstressed syllable to the end, forming feminine rhymes which can be written as (($\texttt{da-DUM}$)^5*$\texttt{da}$).

2.3 Rhyme Words, Couplets and Patterns

A pair of **rhyme words** consists of two words that have the same or similar ending sound. A **rhyming couplet** is a pair of rhymed lines. For example, Line 1&3, 2&4 in Figure 1 are two pairs of rhyming couplets. From the CMU pronunciation dictionary (Weide, 1998), we know that "fall" and "thaw" in Figure 1 are *strict* rhyming pairs because they have exactly the same phonetic endings: `"AO L"`. "Leaves" (`"IY V Z"`) and "trees" (`"IY Z"`) are *slant* rhymes, because they have the same stressed vowels, while the ending consonants are similar but not identical.

2.4 Terminology

We formally define the following terms:

- Sketch: The sketch of a poem contains three aspects: 1) content words that cover the key concepts or main ideas, 2) the rhyme words to appear at the end of each line, and 3) the modification of keywords for aesthetics.
- Keywords $\mathcal{K}$: content words and rhyme words combined. They contain main ideas of a poem and define the rhyming pattern.
- Content words $\mathcal{C}$: keywords that do not appear in the end of each line. We target at predicting 2 context words per line, C_{i1} and C_{i2}.
- Rhyme words $\mathcal{R}$: words in the end of each line. For example, in a Shakespearean sonnet with the rhyme scheme ABABCDCDEFEFGG, there are seven pairs of rhyme words: R_1R_3, R_2R_4, ..., and $R_{13}R_{14}$.
- Initial rhyming lines $\mathcal{I}_{Init}$: index of the lines that the first rhyme word in a rhyming couplet appears (e.g., $\mathcal{I}_{Init}$ = [1, 2, 5, 6, 9, 10, 13] for a Shakesperean sonnet and $\mathcal{I}_{Init}$ = [1, 2, 9, 10, 11] for a Petrarchan sonnet).

3 Approach

Overview As is shown in Figure 1, our sonnet generation model can be divided into four steps. At step a, we train a title-to-outline module by finetuning T5 (Raffel et al., 2019) on keywords extracted from news reports and stories. During inference time, we generate a fourteen-line sonnet sketch that contain those content words $\mathcal{C}$ (Section 3.1). At step b, we aim at forming the correct rhyming pairs. We first select the initial rhyme words from $\mathcal{C}_i$ for $i \in \mathcal{I}_{Init}$, and then generate the remaining rhyme words (i.e., for $i \in \overline{\mathcal{I}_{Init}}$) by forcing the decoder to sample from a vocabulary pool that contains strict and slant rhyme words (Section 3.2). At step c, we infuse imagery and simile as two figurative devices to $\mathcal{C}$ (Section 3.3). In the last step, we leverage a fine-tuned language model with constrained decoding algorithm to impose the meter-and-rhyme constraints (Section 3.4).

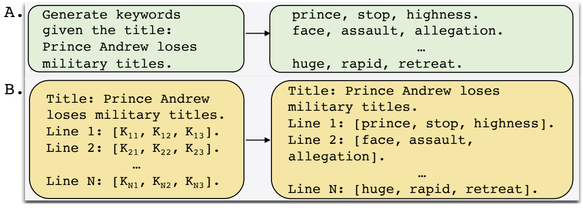

Figure 2: A comparison diagram of two input-output formats to train the first module. While format A is most straight-forward, there is no control over the output structure. Therefore, we purposefully design the prompt shown in format B to control the number of keywords and the number of lines to be generated. $\mathcal{K}_{ij}$ represents the mask tokens at the i-th sentence.

3.1 Content Planning

For each piece of news or stories, we train a title-to-keywords framework that predicts the outline. To this end, we first extract three most salient words per line using the RAKE (Rose et al., 2010) algorithm, which is a domain-independent keyword extraction technique.

Controllable Text Formatting We then leverage the task adaptability of the pretrained T5 (Raffel et al., 2019) to predict the keywords of the whole body. As a unified framework that treats every text processing task as a "text-to-text" problem, T5 can be easily adapted to our task as shown in Figure 2.A, where the input is an instruction to generate the sketch given the title, and the outputs are multiple keywords for each line. However, we need a mechanism to specify the number of lines and keywords to be generated, since we train on prosaic texts with varying formats but infer only on the 14-line sonnets.

To solve this problem and gain control over the poem structures, we format the input and output as shown in Figure 2.B. Specifically, we use [MASK] tokens as placeholders for the keywords. Now that one [MASK] token on the input side corresponds to exactly one word on the output side, we are able to specify the number of lines and keywords during the inference time.

3.2 Generating Rhyme Words

Our title-to-outline model is trained to generate keywords, regardless of the rhyme constraints. In this section, we describe the procedure to generate rhyme pairs. Specifically, we force the model to generate a 14-line outline, with two or three content words for each line depending on whether the line

```
"Title: The Four Seasons.
Keywords: [plums, autumn, leaves].
Keywords: [trees, quivering, fall].
Keywords: [tangled, branches, R3].
Keywords: [fog, snow, R4].

                  …
Keywords: [blossoming, lemon, fell].
Keywords: [swirl, air, R14]."
```

Figure 3: An example input to query the remaining rhyme words during the inference time. Rhyme words in the same background color form a rhyming pair.

is an initial rhyming line:

$$\text{Keywords}_i = \begin{cases} [K_{i1}, K_{i2}, K_{i3}]\,, \text{ if } i \text{ in } \mathcal{I}_{Init} \\ [K_{i1,} K_{i2}]\,, \text{otherwise.} \end{cases} \tag{1}$$

where K_{ij} represents the j-th keyword in the i-th line. Among the three keywords in the initial rhyming lines, we select the last word as the initial rhyme word.

Rhyme Pairs Generation Given the initial rhyme words, we then retrieve all the possible rhyme words $\mathbb{R}$ based on their phonetics information from the CMU prounounciation dictionary (Weide, 1998). This include strict rhymes and slant rhymes. For instance, in Figure 3, the retrieved rhyme word candidates $\mathbb{R}$ for 'leaves' are ['achieves', 'believes', 'Steves', 'trees', ...]. The probability distribution for generating the rhyme word w_R from the candidate list $\mathbb{R}$ is modified as:

$$P'(w_R) = \begin{cases} \frac{p(w_R|\text{context})}{\sum_{x \in \mathbb{R}} p(x|\text{context})} & , \text{if } w_R \in \mathbb{R} \\ 0 & , \text{otherwise.} \end{cases} \tag{2}$$

where $p(w_R|\cdot)$ is the original word probability yielded by the title-to-outline decoder.

3.3 Polishing Context Words for Aesthetics

Now, we have the generated context words and rhyme words that are discourse-level coherent yet less vivid. To this end, we use external knowledge to incorporate two figurative devices into the planned keywords: imagery and simile.

Imagery We leverage the <symbol, imagery> pairs (e.g., <love, rose>) in the ConceptNet dataset (Liu and Singh, 2004) and finetune a imagery generation model from a pretrained model called COMmonsEnse Transformer (Bosselut et al., 2019) (COMeT). It is trained on imagery pairs to generate the imagery word given the symbolism word as input. At inference time, we randomly sample multiple nouns from the sketch to predict their imageries, and only make replacement for the two most confident generations. For example in Figure 1, both <day, sun> and <love, rose> are generated, yet we only replace 'love' with 'rose', because the probability of generating the latter pair is much higher than the former pair.

Algorithm 1 Gen Valid Tokens

function GEN(gen_t, $stress_t$)
Parameter: Int - t ▷ *current time step*
Parameter: Int - N ▷ *num of return samples*
Parameter: List - CW ▷ *context words yet to include*
Input: List of strings - gen_t, $stress_t$ ▷ *generated beams at time step t and corresponding 0/1 stress series*
Output: List of strings - gen_{t+1}, $stress_{t+1}$
Initialize gen_{t+1}, $stress_{t+1}$ to empty
for gen, $stress$ in zip(gen_t, $stress_t$) **do**
▷ *repeat topk sampling N times and return all generations*
    $tokens$ = generate_next(gen, N).to_set()
    for c in CW **do**
        if c not in tokens **then**
            $tokens$.append(c)
    for t in $tokens$ **do** ▷ *check for meter constraints*
        if satisfy(t, $stress$) **then**
            update gen_{t+1}, $stress_{t+1}$, CW
        else
            continue
    return gen_{t+1}, $stress_{t+1}$ ▷ call recursively until 10 or 11 syllables are generated and disregard the metric line unless all three keywords are incorporated.

Simile A simile phrase consists of two parts: the adjective and the figurative vehicle. For example, 'sudden like a flash' is a simile phrase where 'a flash' is the figurative vehicle of 'sudden'. We leverage the simile generation model by Chakrabarty et al. (2020) as an off-the-shelf tool to generate simile vehicles from adjectives and calculate the possibility.[2] At inference time, we randomly sample multiple adjectives from the sketch to predict their figurative vehicles, and only keep the most confident ones. In addition, we also make sure the generated simile phrase conforms to the iambic-meter constraint. For example in Figure 1, the phrase 'bright like diamond' (/x/x) follows the iambic meter, whereas another phrase such as 'shining like diamond' (/xx/x) will be disregarded.

3.4 Sketch to Sonnet Generation

Generating the full sonnet requires more powerful pretrained model than generating the outlines. We fine-tune GPT-Neo-2.7B on the same combination of news and stories data as a language model to generate the sonnet. In order to write fluent and poetic languages that meet the meter-and-rhyme constraints, we make the following adaptations.

[2]https://github.com/tuhinjubcse/SimileGeneration-EMNLP2020

First, to effectively incorporate the rhyme words at the end of each line, we follow previous methods (Ghazvininejad et al., 2016; Van de Cruys, 2020) and generate the whole sonnet line-by-line *in reverse*, starting from the final rhyme word to the first word. That is to say, our language model is finetuned to generate right to left. Second, we include the sketch in the prompt, so that the decoder will learn to give higher probability for these keywords. We also include the previously generated lines in the prompt so that the full sonnet will be more coherent. A simile phrase in the sketch is considered fixed that cannot be modified. Namely, we force to generate the whole phrase when the first word in the phrase is decoded. Third, we design a decoding strategy which modifies the beam search algorithm to impose the meter-and-rhyme constraint. Algorithm 1 displays the skeleton of our decoding strategy. At each decoding step, we apply rhythm control, so that only those tokens that satisfy the iambic-pentameter and its two variations (listed in Section 2.2) are kept in the beams. We recursively generate the next token until 10 or 11 syllables are generated and make up a metric line where all the context words are incorporated.

4 Experimental Setup

4.1 Dataset

Our approach does not require poem data. The training dataset for the content planing module and the decoding module is a combination of 4,500 CNN news summary (Hermann et al., 2015) and 16,000 short stories crawled from Reddit[3]. We remove those articles that contain conversations, urls, or are too long (>50 lines) or too short (<8 lines). During decoding, we generate sonnets using top-k sampling and set no_repeat_ngram_size to 3 to promote creativity and avoid repetition.

4.2 Baselines

Hafez A program that is trained on lyrics data and generates sonnets on a user-supplied topic (Ghazvininejad et al., 2018). It combines RNNs with a finite state automata to meet the meter and rhyme constraints. Hafez is the state-of-the-art model that generates full-length sonnets but it does not train on standard, non-poetic texts.

Few-shot GPT-3 We utilize the most capable model in the GPT-3 family (Brown et al., 2020), *GPT3-davinci*[4], as a strong baseline to follow instructions and generate sonnets. In the prompt, we provide two examples of standard sonnets and then instruct the model to generate a sonnet given the title. We force the output to be exactly 14 lines.

Prosaic An stronger version of *nmf*, the first (and only) model to generate rhyming verses from prosaic texts (Van de Cruys, 2020) by modifying the word probability of rhyme and topical words. For fair comparison, we replace the original encoder-decoder with the more powerful GPT2 and force the output to be 14 lines.

SONG *w/o fig* The model consisting of step a, c, and d as illustrated in Figure 1, but without the polishing the sketch for figurative devices. Our full model consisting of 4 modules is called SONG.

4.3 Automatic Evaluation

It is difficult and thus uncommon to automatically evaluate the quality of poems. For example, Ghazvininejad et al. (2016) and Van de Cruys (2020) exclude automatic evaluation, with the later stating “Automatic evaluation measures that compute the overlap of system output with gold reference texts such as BLEU or ROUGE are of little use when it comes to creative language generation." Here, we propose to evaluate the generated poems in two aspects: format and novelty.

Format Checking For rhyme checking, we count the percentage of rhyme pairs that belong to strict or slant rhymes. For meter checking, we consider the following most common scenarios mentioned in Section 2.2: the standard Iambic Pentameter; the first foot reversed; and a feminine rhyme. In all scenarios, words that are monosyllables can serve as both stressed and unstressed syllables. For a looser standard, we also calculate the percentage of valid lines that contain either 10 or 11 syllables.

Novelty We follow the settings in exsiting works Yi et al. (2018, 2020) and calculate the Distinct-2 scores (Li et al., 2015) to measure the diversity of generated poems. Besides, imagery is another important feature of poems as pointed out by linguistic studies Kao and Jurafsky (2012); Silk (2006). Here, we calculate *Imageability* score to assess how well a poem invokes mental pictures of concrete objects. Specifically, we extracted the features from the resource by Tsvetkov et al. (2014), who use

[3]https://www.reddit.com/r/shortscarystories/

[4]https://beta.openai.com/docs/engine

Model Name	Format Checking			Novelty	
	Rhyme	Meter	Syllable	Dist-2	Img
Hafez	98.3%	76.8%	95.7%	84.8	0.44
Fewshot GPT-3	14.0%	17.6%	30.9%	85.3	0.48
Prosaic	<u>100%</u>	10.1%	19.0%	84.9	0.46
SONG *w/o fig*	<u>100%</u>	<u>77.7%</u>	<u>98.6%</u>	<u>86.6</u>	0.49
SONG	<u>100%</u>	75.6%	98.4%	<u>86.6</u>	<u>0.51</u>
Human	94.6%	70.7%	81.8%	0.87	0.52

Table 2: Automatic evaluation results for rhyme, meter, syllable checking, distinct scores, and imageability (Img in the table). Best machine scores are underlined.

a supervised learning algorithm to calculate the imageability ratings of 150,114 terms. For each poem, we average the ratings of all its words after removing the stop words.

4.4 Human Expert Judgement

Considering the expertise required to appreciate sonnets, we recruit 6 professionals that hold a bachelor's degree in English literature or related majors as domain experts to annotate the generated sonnets. We provide detailed instructions and ask them to evaluate the each poem on a scale from 1 (not at all) to 5 (very) on the following criteria: **1) Discourse Coherence**: whether the sonnet is well organized, with the sentences smoothly connected and flow together logically and aesthetically, **2) Originality/Creativity**: the usage of original ideas in the poem, including imagination, rhetorical devices, etc., **3) Poetic in language**: how well the poem adopts descriptive and vivid language that often has an economical or condensed usage, **4) Emotion Evoking**: if the poem is emotionally abundant and make the readers emphasize with the writer. At last, we ask the annotators to judge if the sonnet is written by a poet with *serious* goals to write a poem. In total, we evaluate 50 sonnets for each baseline and the gold standard (human) model. Each sonnet is rated by three professionals. The average inter-annotator agreement (IAA) in terms of Pearson correlation is 0.61 with p-value <0.01, meaning that our collected ratings are highly reliable. We also conduct paired t-test for significance testing. The difference between our best performing model and the best baseline is significant.

5 Results and Analysis

5.1 Results of Automatic Evaluation

Table 2 summarizes the format checking and novelty scores of our model compared to the baselines.

	DC	O	P	E	WH
Hafez	3.09	3.01	3.05	2.95	41.3%
Few-shot GPT3	3.43	3.10	2.86	3.11	52.7%
Prosaic	3.25	2.95	2.97	2.98	46.0%
SONG *w/o fig*	<u>3.57*</u>	3.25	3.35	3.13	58.7%
SONG	3.52	<u>3.41*</u>	<u>3.66*</u>	<u>3.22*</u>	<u>62.0%*</u>
Human	**3.82**	**3.54**	**3.68**	**3.56**	**83.3%**

Table 3: Expert ratings on several criteria to assess sonnet quality: discourse-level coherence (DC), originality/creativity (O), poeticness in language (P), emotion evoking (E), and written by human (WH). We show average scores with 1 denoting the worst and 5 the best. We boldface/underline the best/second best scores. $*$ denotes that paired t-test shows that our model variations (**SONG** *w/o fig*, and **SONG**) outperform the best baseline in all aspects with statistical significance (p-value < 0.05).

We can see that human poets tend to incorporate more variations and do not strictly follow the meter and rhyme constraints, which computers are good at. GPT-3 fails to learn the sonnet formats through massive pretraining and few-shot learning despite its gigantic size. Prosaic falls short of meter-checking because is only trained to generate rhyming verses. Since we utilize the the phonetics information provided in the CMU dictionary, **SONG** achieves 100% success in rhyme words pairing. As for novelty, **SONG** generates most diversely and is best at that arousing mental pictures of concrete objects among machines.

5.2 Results of Human Evaluation

Table 3 presents the performance of the aforementioned evaluation criteria: coherence, originality, poeticness, and emotion-evoking. Our models (**SONG** *w/o fig*, and **SONG**) outperform the baselines in all aspects by a large margin. Compared with Prosaic which also generates poems from non-poetic texts, our models generates more coherent sonnets with great statistical significance (p-value < 0.01), showing the superiority of explicit sketch planning over generating from scratch (i.e., end-to-end generation).

Comparison between our own models. **SONG** *w/o fig* generates more coherently than **SONG** (p-value < 0.10). However, **SONG** achieves high scores in originality, poeticness by a large margin (+0.2). Hence, we still consider it as our best model. It is also noteworthy that **SONG** is the most *emotion-evoking* system among all machines even though we do not have explicit sentiment control. Poem theories have shown that emotion appeals lie in the following aspects: the general topic, the word choice, vivid descriptions, figurative language, insights and experience (Scheub, 2002). We posit that aesthetic features in the SONG arouse emotion appeals.

Analysis for high poeticness. SONG is on par with humans in terms of poeticness score, meaning that our models generate highly descriptive, vivid, and condensed text. With manual examination, we attribute such high poeticness to three aspects. First, the imagery and similes clearly represents traits of poems. Second, in keyword-planning we ensure that at least three concepts will be presented per line, and thus the generation module naturally become economical in word usage to include all the information. Lastly, with the constraint decoding algorithm to insert keywords, we inevitably become less natural (e.g., miss conjunctions and auxiliary verbs). While this can be a drawback in other generation tasks, the occasional omission of such auxiliary words is just opportune for sonnets, and adds to the flavor of a poem. The examples in table 4 helps demonstrate these points.

6 Qualitative Analysis

6.1 Case Study

We conduct case study to better understand the advantages of our model over the baselines. Table 4 lists the generated sonnets by Hafez, Prosaic and SONG given the same title: “The Four Seasons".

Problems with the Baselines Hafez chooses words that are related to the title as rhyme words. However, topically related rhyme words are not sufficient for overall coherence. While it is locally understandable, the sonnet generated by Hafez is divergent and disconnected when sentences are put together. On the other hand, Prosaic mimics the rhyme and topical properties of poems, but still generate highly prosaic and colloquial sentences that are not poetic at all.

Advantages of Our Model Thanks to content planning, SONG *w/o fig* generates a well-organized sonnet that describes the four seasons from winter to autumn in a logical order. Despite minor grammar errors, the full model SONG benefits from vivid descriptions and natural imagery such as ‘whispers rumors of a winter coming’, ‘blossom of the season’, and ‘sudden like a flash’.

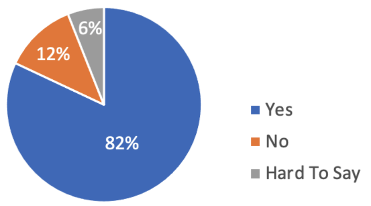

Figure 4: Pie chart showing whether the generated sonnet be condensed into the planed keywords.

6.2 Impact of Keywords

By comparing SONG *w/o fig* versus Prosaic, our human evaluation results already show that content planning contributes to discourse-level coherence. In addition, we provide the keywords along with the sonnet generated by SONG, and ask human annotators to judge if the sonnet can be condensed into those keywords. Results are shown in Figure 4. We observe that 82% of the time the planed keywords successfully guide the generation by providing salient points of the sonnet. We then conduct error analysis on the rest 18%. Top two reasons among the fail cases are: 1) the decoding step generates novel contents that are not represented by the keywords (8%), and 2) the polishing step alters the original meaning of planed keywords (6%).

6.3 Limitation and Future Direction

Sonnets are divided in to multiple stanzas. Lines within a stanza are more interlaced than across stanza, and the start of a new one usually indicates transition to another viewpoint. Our current approach could not capture such structural characteristics during planning and generation, and we hope to investigate these features in future work.

7 Related Work

Poetry Generation Automatic poetry generation before the deep learning age relies heavily on templates, norms, or rule-based approaches (Gervás, 2001; Manurung, 2004; Manurung et al., 2012). Neural approaches to automatic poetry generation pay little attention to the coherence issue of long poems. For example, Wang et al. (2016); Lau et al. (2018); Yi et al. (2018); Liu et al. (2019) merely target at generating the first stanza (four lines) of a poem. For longer poems such as sonnets, Ghazvininejad et al. (2016) propose to use related words as rhyme words to achieve topical relatedness, and later propose to generate discourse-level coherent English sonnets by French-English translation (Ghazvininejad et al., 2018). Van de Cruys (2020) propose a naive RNN framework to gener-

Hafez	Score	Prosaic	Score
Delightful hours over long ago, Succeed to hang the little hook and light. The darker still and more extended snow, The fresh invention bore the better night.	C: 2.7	The leaves fall and the air is cooler, Were we known with seasons? Spring is coming, Of course there isn't that much humor, And it is now blooming.	C: 3.3
Another party started getting heavy, And never had a little bit of summer. And got a trigger on the finger really, Or something going by the name of winter.	O: 3.0	Even when everyone else's warm, Someone would want to know what we could do in may, else we didn't known – It was the first storm, It's how they used to do every other day.	O: 2.7
Retreat the colors and fun in afternoon And bought the tickets loved a royal song A sacred person join and ride in moon Away the author go and winding long,	P: 3.0	What aren't you looking up Right behind us, no clouds, Just know no one's will ever see the clouds or thunder, And nobody other than the crowds, Before now, we had just started to wonder.	P: 3.0
Oh maybe get a little out of sight, And wounded victim suffered from the site.	E: 3.0	We all aren't sure what we can keep, they couldn't see me, like they were about to reap.	E: 3.3
SONG *w/o fig*	**Score**	**SONG**	**Score**
Falling snow, I feel a little cooler, Is a sign of stormy winter coming, Meet the honest friends and gather humor, Happy spring and just admire the blooming.	C: 3.7	The snow is falling, whether getting cooler, Whispers rumors of a winter coming, Gather honest, peace and love and humor, Blossom of the happy season blooming.	C: 3.7
The heat and ocean air becoming warm, Little birds and beautiful songs in May, Leaves and flowers probably just the storm, Among the lunar moons and summer day.	O: 3.3	Wind gets heat and the air becoming warm, The birds enjoy the little lovely may, Beneath the leaves, flowers survive the storm, The moon is shining on a summer day.	O: 4.0
Sudden rain and downpour from the thunder, And summer always fill hotels with crowds, Take a shower and give the spring a wonder, Watch the blue sky and far behind the clouds.	P: 3.3	Sudden like a flash comes rain with thunder, The summer vibes fill the running crowds, Because of shower, spring became a wonder, The sky is high and blue like sea with clouds.	P: 4.0
In months the future vegetables reap, The years and seasons never really keep.	E: 3.0	The coming months are watching future reap, Those years and seasons bring us all to keep.	E: 3.3

Table 4: An example of the generated sonnets from four systems with the same title: "The Four Seasons". The scores are average numbers of three human ratings on the following criteria: coherence (C), originality (O), poetic in language (P), and emotion evokingness (E). We underline the planed keywords and highlight the figurative languages in blue.

ate rhyming verses from prosaic texts by imposing a priori word probability constraints. We on the other hand achieve discourse-level coherence by learning from standard, non-poetic texts.

Other related works to boost the creativity of generated poems include adding rhetorical (Liu et al., 2019) and influence factors (e.g., historical background) as latent variables (Yi et al., 2020). To the best of our knowledge, we are the first to explore adding both figurative speeches and meter-and-rhyme constraints to poetry generation without relying on poetry data.

Content Planning Content planning for automatic text generation originates in the 1970s (Meehan, 1977). Recently, the *plan-and-write* generation framework has shown to be efficient in creative content generation (Wang et al., 2016; Martin et al., 2018; Yao et al., 2019; Gao et al., 2019). The framework employs a hierarchical paradigm and helps to produce more coherent and controllable generation than generating from scratch (Fan et al., 2019; Goldfarb-Tarrant et al., 2020). However, all existing works under this line learn the storyline/plot from the target domain for improved coherence. We on the other hand adopt content planning to disentangle the training from the decoding step which aims at circumventing the shortage of sizable creative contents for training supervised models.

8 Conclusion

We investigate the possibility of generating sonnets without training on poems at all. We propose a hierarchical planning-based framework to generate sonnets which first plans the high-level content of the poem, refine the predicted keywords by adding poetic features, and then achieve decoding-time control to impose the meter-and-rhyme constraints. Extensive automatic and expert evaluation show that our model can generate sonnets that use rich imagery and are globally coherent, poetic, and emotion provoking.

References

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Tuhin Chakrabarty, Smaranda Muresan, and Nanyun Peng. 2020. Generating similes effortlessly like a pro: A style transfer approach for simile generation. *arXiv preprint arXiv:2009.08942*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *ACL*.

Jun Gao, Wei Bi, Xiaojiang Liu, Junhui Li, Guodong Zhou, and Shuming Shi. 2019. A discrete cvae for response generation on short-text conversation. *arXiv preprint arXiv:1911.09845*.

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, Yejin Choi, and Kevin Knight. 2018. Neural poetry translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 67–71.

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.

Raymond W Gibbs Jr, Raymond W Gibbs, and Jr Gibbs. 1994. *The poetics of mind: Figurative thought, language, and understanding*. Cambridge University Press.

Seraphina Goldfarb-Tarrant, Tuhin Chakrabarty, Ralph Weischedel, and Nanyun Peng. 2020. Content planning for neural story generation with aristotelian rescoring. *arXiv preprint arXiv:2009.09870*.

Michael Hart. 2004. Project gutenberg.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701.

Justine Kao and Dan Jurafsky. 2012. A computational analysis of style, affect, and imagery in contemporary poetry. In *Proceedings of the NAACL-HLT 2012 workshop on computational linguistics for literature*, pages 8–17.

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*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Zhiqiang Liu, Zuohui Fu, Jie Cao, Gerard de Melo, Yik-Cheung Tam, Cheng Niu, and Jie Zhou. 2019. Rhetorically controlled encoder-decoder for modern chinese poetry generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1992–2001.

Hisar Manurung. 2004. An evolutionary algorithm approach to poetry generation.

Ruli Manurung, Graeme Ritchie, and Henry Thompson. 2012. Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence*, 24(1):43–64.

Lara J Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

James R Meehan. 1977. Tale-spin, an interactive program that writes stories. In *Ijcai*, volume 77, page 9198.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Harold Scheub. 2002. *The Poem in the Story: Music, Poetry, and Narrative*. Univ of Wisconsin Press.

Michael S Silk. 2006. *Interaction in poetic imagery: with special reference to early Greek poetry*. Cambridge University Press.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 248–258.

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

Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016. Chinese poetry generation with planning based neural network. *arXiv preprint arXiv:1610.09889*.

Robert L Weide. 1998. The cmu pronouncing dictionary. *URL: http://www. speech. cs. cmu. edu/cgibin/cmudict*.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.

Xiaoyuan Yi, Ruoyu Li, Cheng Yang, Wenhao Li, and Maosong Sun. 2020. Mixpoet: Diverse poetry generation via learning controllable mixed latent space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9450–9457.

Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Wenhao Li. 2018. Automatic poetry generation with mutual reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3143–3153.

Appendix

A Experimental Setup

Configurations We finetune the pretrained T5 for 10 epochs for the "content planning" component, and finetune GPT-Neo-2.7B for 6 epochs for the decoding component. We use one Nvidia A100 40GB GPU. The average training time is 5 10 hours for each experiment.

Decoding Strategy For decoding, we generate sonnets from our models using a top-k random sampling scheme. At each time step, the model generates the word probability and randomly sample from the k = 50 most likely candidates from this distribution. To avoid repetition and encourage creativity, we set no_repeat_ngram_size to 3 and use a softmax temperature of 0.9.

Human Evaluation Considering the expertise required, human evaluators are paid $25 per hour. We also explained that the collected ratings will be used for analysis and will be reported in scientific papers.