POETRY FROM IMAGE DESCRIPTIONS USING A GENERATIVE TRANSFORMER

Abhiramon R.

Master of Technology Thesis

June 2020

International Institute of Information Technology, Bangalore

POETRY FROM IMAGE DESCRIPTIONS USING A GENERATIVE TRANSFORMER

Submitted to International Institute of Information Technology,
Bangalore
in Partial Fulfillment of
the Requirements for the Award of
Master of Technology

by

Abhiramon R.
(IMT2015002)

International Institute of Information Technology, Bangalore
June 2020

Dedicated to

my parents and brother

Thesis Certificate

This is to certify that the thesis titled **Poetry from Image Descriptions using a Generative Transformer** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Technology** is a bona fide record of the research work done by **Abhiramon R.**, **(IMT2015002)**, under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma. The thesis conforms to plagiarism guidelines and compliance as per UGC recommendations.

Dinesh Babu Jayagopi

Bengaluru,

The 23rd of June, 2020.

POETRY FROM IMAGE DESCRIPTIONS USING A GENERATIVE TRANSFORMER

Abstract

Poetry Generation is a challenge that has piqued the interest of researchers for many years. Good poetry requires a combination of meaningful lines, correct grammar and a poetic form such as a regular metre or rhyme [1]. Recent developments in language modelling using generative transformers have shown promise in generating coherent text. A few methods have also been explored to control aspects of generation such as sentiment and topic. However, there is a dearth of research applying these methods to poetry generation. We fill this gap by experimenting with these new advances on controlled poetry generation.

Specifically, we choose the problem of generating poetry from images, represented by their caption and tags. Evaluation in terms of fluency, relevance (with caption and tags) and rhyme density show the effectiveness of our proposed adaptations of TransferTransfo [2] and PPLM [3] models that achieved a 7% improvement in fluency and a 17% improvement in rhyme density respectively over the baseline GPT-2 [4] model.

Acknowledgements

I would like to firstly thank my advisor Dr. Dinesh Babu Jayagopi for his continuous guidance not only during my Master's Thesis, but throughout my time in college. I'm really grateful for his support from my first machine learning project at Aagna Labs to the more recent course and industrial projects which shaped my interests and convinced me to pursue a career in Machine Learning.

I've also enjoyed all the exciting discussions and learnt a lot from the healthy feedback I regularly got from my fellow members at the Multimodal Perception Lab, IIIT Bangalore. I would also like to thank MINRO whose funding made this project possible.

Lastly, I would like to thank all the Professors at IIIT-B whose classes and interactions helped me build a solid foundation and appreciation for research in Computer Science.

Contents

List of Figures

List of Tables

List of Abbreviations

BOW . . . . . Bag of Words

CNN . . . . . Convolutional Neural Network

CTRL . . . . . Conditional Transformer Language Model

DenseCap . . . . . Dense Captions

GPT . . . . . Generative Pre-trained Transformer

GPT-2 . . . . . Generative Pre-trained Transformer 2

LM . . . . . Language Model

LSTM . . . . . Long Short-Term Memory

MS-API . . . . . Microsoft Computer Vision API

PPLM . . . . . Plug and Play Language Model

PPLM-Rhyme . Plug and Play Language Model - Rhyme

RNN . . . . . Recurrent Neural Network

Transfo . . . . . TransferTransfo

VAE . . . . . Variational Auto-encoder

CHAPTER 1

INTRODUCTION

In this chapter, we introduce the recent transformer architecture and discuss the generative transformer model. We later outline the challenges in generating poetry and how generative transformer fits into the picture. We follow this by formalizing our problem statement.

1.1 Background

1.1.1 The Generative Transformer

In a time when Recurrent Neural Networks (RNN) and Convolutions Neural Networks (CNN) were the popular approaches in sequence-to-sequence tasks, Vaswani et. al. (2017) proposed the transformer architecture which solely uses attention mechanisms (Figure FC1.1). In the paper, they demonstrate how their model overcomes the problem of long-range dependencies while being parallel and able to train in a fraction of time taken by RNNs. They also record a state of the art performance of the transformer in machine translation task, which even beats the previous ensemble models by a large margin [5].

Using this successful method based on self-attention, Radford et. al. (2018) at OpenAI proposed a Generative Pretrained Transformer (GPT) model. The model is trained with

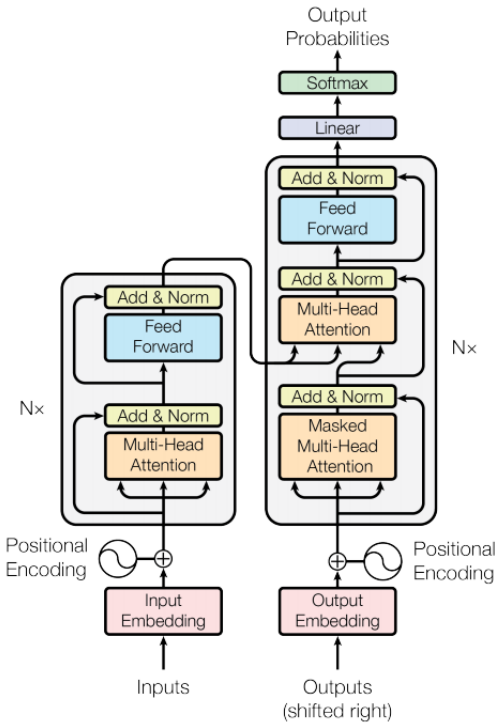

Figure FC1.1: The Transformer model architecture [5].

language modelling objective, to maximize the likelihood of a given corpus of tokens $U = [u_1, u_2, \cdots]$ as given in Eqn 1.1 [10].

$$L(U) = \sum_i logP(u_i|u_{i-k}, \cdots, u_{i-1}; \Theta) \quad \text{(Eqn 1.1)}$$

Just like the transformer decoder, the model comprises of transformer blocks (multi-headed self-attention followed by a position-wise feedforward layer). The output of the last block, h_n (Eqn 1.2) is used to compute a probability distribution over the target tokens [10].

$$h_i = transformer\_block(h_{i-1}) \forall i \in [1, n] \quad \text{(Eqn 1.2)}$$

Following a similar design, with minor architectural modifications, Radford et. al.

(2019) introduced another model called the Generative Pretrained Transformer 2. GPT-2 is trained on a data-set called WebText, comprising of millions of web-pages (filtered to contain only good quality text). Another interesting addition is the Byte Pair Encoding to embed its input. This mechanism uses word level inputs for frequent words and breaks down infrequent words into groups of characters to overcome the limitations of its vocabulary [4].

On release, the large GPT-2 model (of 1.5 Billion parameters) achieved state of the art performance on 7 out of 8 varied language modelling data-sets (such as LAMBADA and CBT) without even using the data-sets' text for training or fine-tuning (zero shot) [4].

1.1.2 Poetry Generation

What separates Poetry Generation from regular text generation problems is that Poetry requires a poetic form along with a coherent message. The features which create a poetic form, such as metre (stress patterns, number of syllables in a line), rhyme (matching vowel or consonant sounds) and figurative language often are interdependent and influence the way the meaning is conveyed [11].

Among a broad range of approaches well documented by Oliveira (2017) [11] and those explained in Section 2.2, the recent methods often involve the use of Recurrent Neural Networks (RNN) as a language model. However, as explained by Lau et. al. (2018), while the task of maintaining rhyme and metre is relatively easier, conveying a message which is readable and elicits emotion is a challenge [7]. Manurung (2003) [1] explains the need for poetry to satisfy the qualities of meaningfulness, grammaticality and Poeticness (rhyme and metre) at the same time.

1.2 Problem Statement

In this thesis, we tap into the strength of the Generative Transformer model (GPT-2) and apply it to Poetry Generation which clearly requires a fluent and coherent language model. Similar to stimuli-based models explained by Oliveira (2017) [11], we set our task of Poetry Generation as an Image to Poetry problem. We use image captions and tags to represent the image's content.
Formally, given an image I with a caption C and tags T, we apply a Generative Transformer model $M(C,T)$ to produce a poem P.

$$P = M(C,T) \tag{Eqn 1.3}$$

We adapt the models GPT-2, TransferTransfo and PPLM as explained in Section 3.2 and apply them to our problem. The evaluation of the efficacy of these models in terms of rhyme density, fluency and relevance to caption and tags is explained in Chapters 4, 5 and 6.

CHAPTER 2

RELATED WORK

2.1 Text Style Transfer

Just as we try to generate poetry using the content of image descriptions, the objective of unsupervised text style transfer is to transfer the input text into a target style without changing its meaning. The central idea in some of the latest research in this domain is to use an auto-encoder based architecture with the objective of separating style from the content [12].

Hu et. al. (2018) [13] attempt to do this using a modified Variational Auto-encoder (VAE). To the hidden representation which encodes the content, they add control codes which are trained to exert control over the style of text generated, such as sentiment and tense. Using VAEs however can be computationally complex, as explained by Shen et. al. (2017) [14], since they try to make the content representation simple. Instead, other recent approaches [12, 14, 15] try to bring the content representations of the two styles closer using adversarial networks. Fu et. al. (2017) use two adversarial networks, one to predict the correct style and another one to maximize the entropy making the two styles indifferentiable [15]. Shen et. al. (2017) use an adversarial network to differentiate samples from the content distributions of the two styles of text [14]. In these methods the adversarial loss is added to the auto-encoder loss of reconstructing the original sentence from the hidden representations. In a different approach, Yang et. al. (2018)

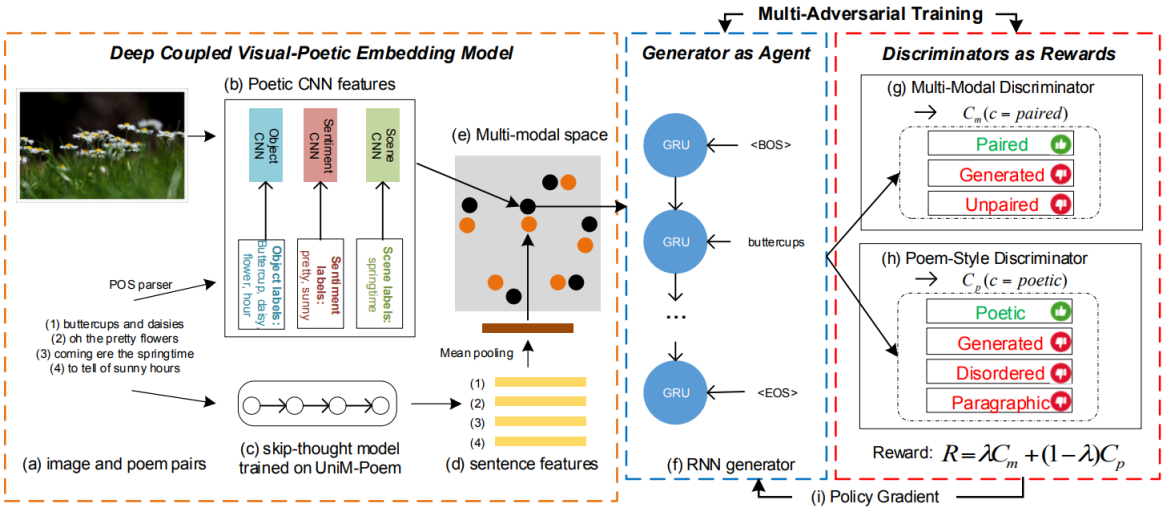

Figure FC2.1: Image-to-Poem framework of Liu et. al. (2018) [6].

use a language model applied on the transferred sentences as their discriminator loss, which is then added to the reconstruction loss of the auto-encoder. The assumption they make is that a sentence which is not of the target style will have a high perplexity when a language model trained on the target style is applied on it [12].

The models discussed in this section typically use a recurrent neural network (RNN) as their encoder and decoder. In our approach, we use transformer-based models which are much better at capturing long-range dependencies [5]. Also, the constraints on content matching is crucial here, whereas our problem only uses the image captions to guide the direction of poetry generation.

2.2 Poetry Generation

Generating poetry using a stimulus such as semantic predicates, keywords, a sentence or a newspaper article have been explored for many years. However, earlier approaches to Poetry Generation generally used only a knowledge base of words and relations extracted from sources such as WordNet (a lexical database), ConceptNet (a common sense knowledge base) or even a dictionary [11].

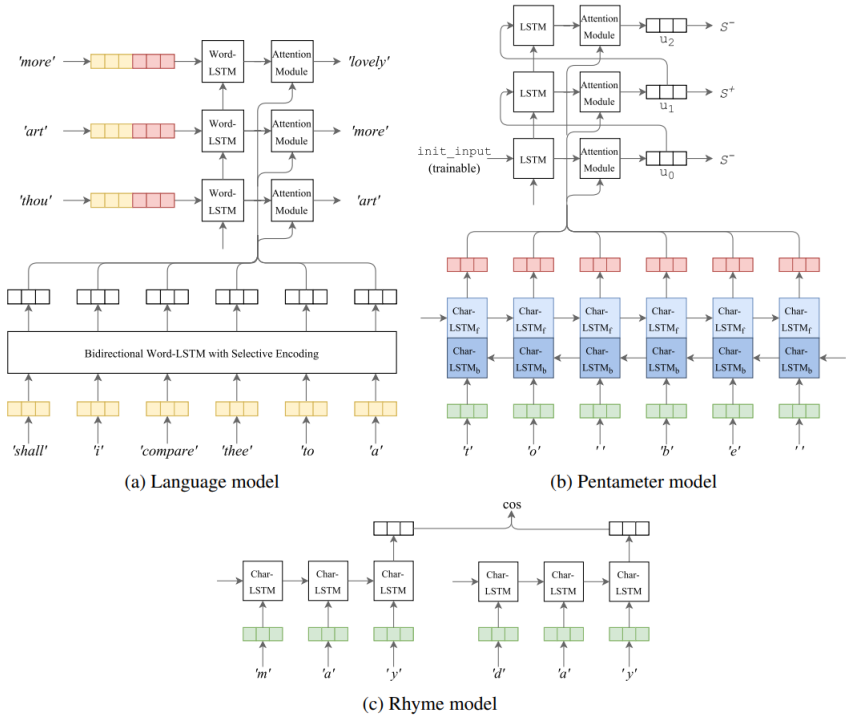

Figure FC2.2: Model architecture of Lau et. al. (2018) [7]

The more recent research have demonstrated the success of models based on RNNs applied to poetry generation. These RNNs are often used as a language model [11]. Potash et. al. (2015) use an LSTM as a language model to produce rap lyrics. They add endVerse and endLine tokens to capture metre and rhyme and demonstrate the superiority of the approach over a baseline n-gram model [16]. Yan et. al. (2016) generate poetry using character-level RNNs guided by given keywords (to set the theme, topic or subject). They encode keywords (using character-level RNNs) and use this as input to a generation RNN. In addition, they re-use the final hidden state of the RNN a few times, which they claim helps refine wording and make the output more poetic. They call this method iterative polishing schema [17]. In a more complex approach, Lau et. al. (2018) use attention-based LSTMs for a language model (to predict the next word in the poem), a pentameter model (to predict stress of the characters) and a rhyme model (to compute rhyme of word pairs using character LSTM) for poetry generation (Figure FC2.2) [7].

Liu et. al. (2018) [6] tackle the same broad problem as ours, of generating poetry from images. They use an end-to-end model (Figure FC2.1), which converts an input image embedding into poetry using an RNN. To embed the image, they use a CNN trained for object, scene and sentiment detection, which are deemed important information for generating poetry. They then embed the image and the paired poetry text (embedded using skip-thought features) together in the same space by minimizing a pairwise ranking loss. To decode the poem from the embedding, an RNN is used in conjunction with a poem-style and a multi-modal discriminator network [6]. They also release two data-sets, one with a large collection of poems (to train skip-thought vectors) and another with image-poem pairs to train the end-to-end model. We use these data-sets to train our models, more details on these are provided in Section 3.1.

The models discussed so far rely on recurrent neural networks. As explained in Section 1.1.1, we use a transformer-based model which is much more powerful, especially in capturing long-range dependencies [5]. A recent paper, which also uses a generative transformer for poetry is by Bena et. al. (2020) [18]. In this paper, they use GPT-2 fine-tuned on 8 data-sets drawn from the GutenBerg Corpus, each representing a particular emotion such as sadness, anger, joy, etc. They show that GPT-2 is able to capture and replicate the emotions in the output generated [6]. Though they generate poetry according to a given emotion, their control is limited to using different data-sets for fine-tuning. Our problem, on the other hand, requires a lot more control since the poetry needs to match any given image description. The additional goal of rhyming only adds to the challenge we take on.

2.3 Controlled Generative Transformer

As discussed in Subsection 1.1.1, a transformer-based language model such as GPT-2 is very powerful. But, controlling the attributes dictating style and content of the text

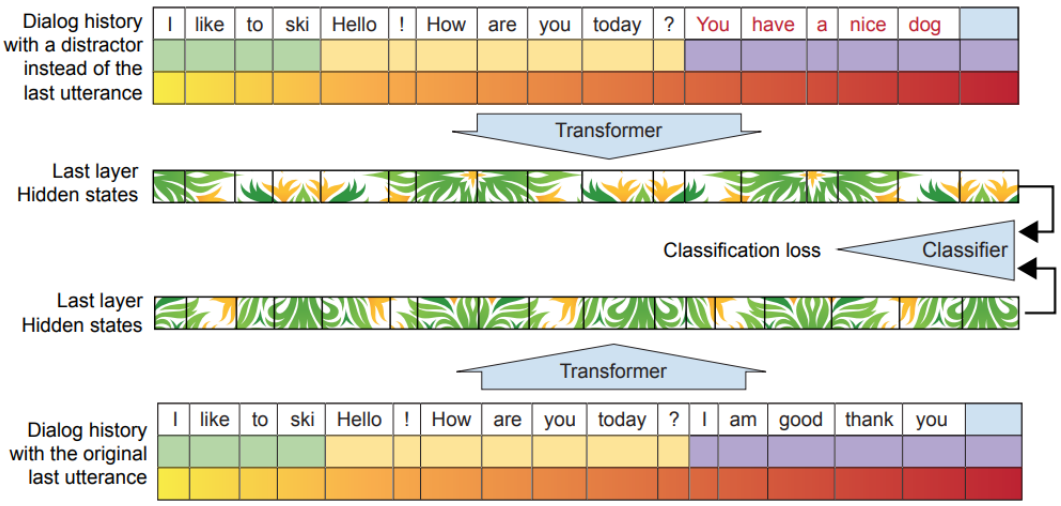

Figure FC2.3: Example of next sentence prediction for TransferTransfo [2]

generated is a challenging task. Some recent research which approach this problem are discussed here.

2.3.1 Conditional Transformer Language (CTRL) Model

CTRL is a large transformer-based language model (1.63 billion parameters) provided by Salesforce. They enable control of aspects such as domain (horror, review, legal, etc.), style (question, translation, etc.) and source (website, Reddit thread, etc.) using control codes. They also provide an option to analyse the most likely source for each generated output. They provide the option of using multiple codes together to mix different aspects of the generated text [19].

The model is pre-trained along with the control codes. The control codes are learnt by prepending them to the input text during training [19]. Though the model works well, this mechanism limits the control over generation to the control codes available. For our problem, a more flexible mechanism of control is needed since each poem to be generated needs to be conditioned on its paired image caption.

2.3.2 TransferTransfo

TransferTransfo is a method proposed by Wolf et. al. (2019) [2] as an adaptation of the GPT model [10] to build a personality-based conversation agent. The model was originally built for the PERSONA-CHAT data-set which was a part of Conversational Intelligence Challenge 2 (held during NIPS 2018 conference). The objective of the contest is to use a given set of persona and a conversation history to generate an appropriate response. The TransferTransfo model uses two main ideas: augmented input representation and multitask learning [2].

The persona sentences and chat history lines are concatenated to form the input representation. In addition to positional embedding, they add a dialogue-state embedding which helps differentiate between tokens in the personality, utterance of person 1 and utterance of person 2. They refer to this method as the augmented input representation [2].

Multitask learning refers to the additional objective of next utterance classification. Both the classification loss and the language modelling loss are then optimized together. Next utterance classification is implemented using a linear layer which uses the hidden state from the transformer to pick the correct next utterance, mixed with a list of random distractors (Figure FC2.3) [2].

Using this model, Wolf et. al. (2018) claim a State of the art performance in the challenge with an improvement of 45% in perplexity, 46% in Hits@1 and 20% in F1 scores over the next best model [2]. The design of this model makes it a great match and we adapt it for our problem statement as explained in Section 3.2.

2.3.3 Plug and Play Language Model (PPLM)

Dathathri et. al. (2020) propose the Plug and Play Language Model as a method to control the generation of the powerful GPT-2 model, without having to change the

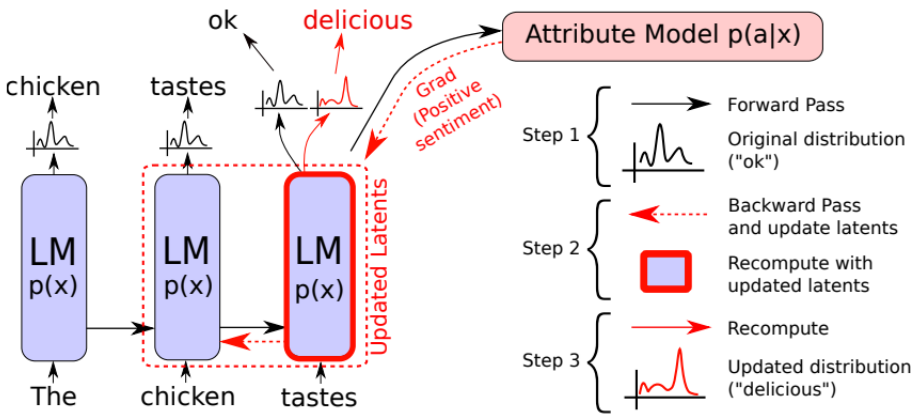

Figure FC2.4: Example of latent update using a PPLM attribute model [3]

architecture or even fine-tune to any data-set. They use an attribute model, $P(a|x)$ (a is the controllable attribute and x is the generated sample) to control topic or sentiment of a language model $P(x)$ output. The two attribute models they experiment with are bag-of-words (BOW) and a single layer learnt as a classifier [3]. The main idea is to use the attribute model to compute a gradient and alter the hidden activations of the language model. Figure FC2.4 gives an example of how the hidden activations are changed to output "delicious" instead of "ok" in order to better fit the attribute model (single layer trained to identify positive sentiment) [3].

The equation for the bag-of-words attribute model (Eqn 2.1) is just the sum of probabilities of the words in BOW for the LM [3].

$$log(p(a|x)) = log(\sum_{i}^{k} p_{t+1}[w_i]) \qquad \text{(Eqn 2.1)}$$

For the single layer classifier, the attribute model is defined as in Eqn 2.2, where f is a function trained to predict a target label using the mean token embedding of the generated tokens, for each attribute [3].

$$log(p(a|x)) = log(f(o_{:t+1}, o_{t+2})) \qquad \text{(Eqn 2.2)}$$

The flexibility of control this model allows is a great fit to our problem and we adapt it as explained in Section 3.2.

CHAPTER 3

EXPERIMENTS

3.1 Data-set

To fine-tune our models, we use the two data-sets provided by Liu et. al. (2018) [6]. The Uni-modal data-set consists of 92,265 poems and Multi-modal data-set consists of 8,292 human-annotated image-poem pairs (Figure FC3.1). We convert the images into captions and tags using Microsoft Computer Vision API (MS-API) [9] to generate caption-poetry pairs, which we split to obtain train and test sets. The test set consists of 670 caption-poem pairs. We also experimented with Dense Captions model released by Johnson et. al. (2016) [8], we enclose the results of its comparison with MS-API in Appendix A. Due to better caption quality and fit, we use MS-API for generating all captions and tags.

3.2 Models

To conduct our experiments we adapt the Transfo and PPLM models, explained in the Subsections 2.3.2 and 2.3.3 respectively, to Controlled Poetry Generation. We use GPT-2 as our baseline. All the models are first fine-tuned using the Uni-modal data-set explained in Section 3.1.

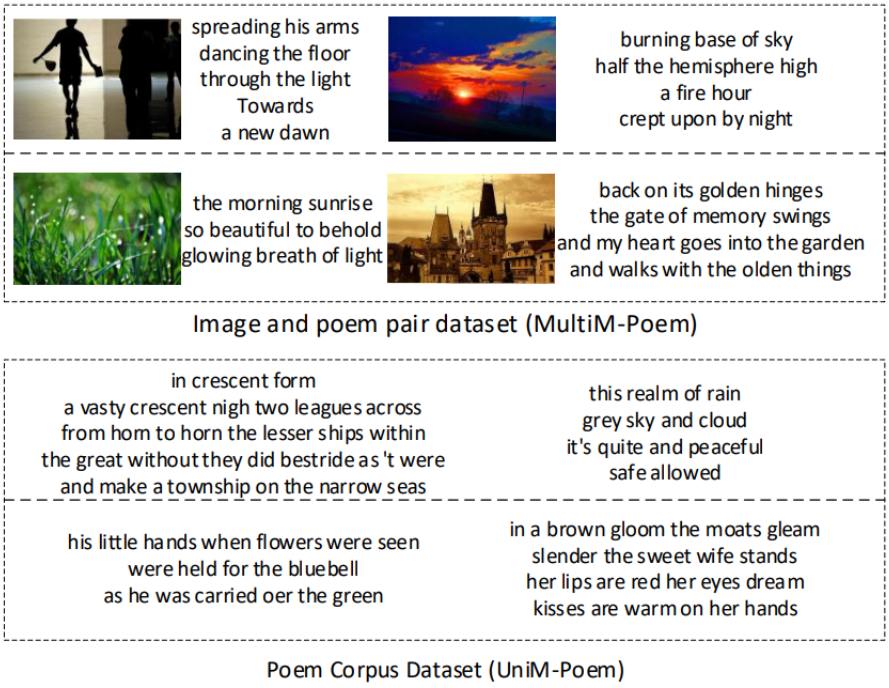

Figure FC3.1: Data-sets used for fine-tuning [6].

3.2.1 Adapted GPT-2

Once GPT-2 is fine-tuned for poetry, we simply use the output generated by giving caption as the starting text. After fine-tuning, the outputs acquire a poetic form and are always split into short lines as seen in the data-set. We generate an output of 50 tokens and all lines except the first (which is the caption) and the last (which is often incomplete) are considered as the poem.

3.2.2 Adapted TransferTransfo

Similar to how Wolf et. al. (2019) [2], we fine-tune GPT-2 in a supervised fashion using the caption-poetry data-set explained in Section 3.1. We replace persona with image tags and caption. We also use previous poem lines in place of conversation history. To train next sentence classification, we create a data-set of true next line mixed

with random distractors sampled from the poetry data-set. The classifier has to use the caption, tags and previous poem lines to pick out the true next line. The remaining procedure remains the same as the original paper.
Just as the original model induced elements of persona into the response, we hope that the poem generation content is controlled by the image caption and tags provided.

3.2.3 Adapted PPLM (PPLM-Rhyme)

We modify the PPLM [3] explained in Subsection 2.3.3 to generate poetry. Using caption as the starting text and bag-of-words (BOW) attribute model containing the image tags, we generate the poem. This helps us control the content of the poem and make it stick with the caption and tags.
In addition to content, we also attempt to add rhyming to the poetry. To reach this goal, we make a simple modification to the PPLM bag-of-words approach. The main hypothesis is that including rhyming words in the bag-of-words should increase the probability of them being generated in the subsequent poem lines.
To generate a poem P using caption C and tags T, we use the PPLM-Rhyme algorithm $PPLMR(C,T)$. Let $PPLM(S,B)$ be the PPLM model, where, S is the start text given to the model and B is the bag of words used for the attribute model. Let $Rh(w)$ be function which returns a list of rhyming words for the word w. Let $Vow(w)$ be a function which returns the number of vowel phonemes in the word w. We break the poem generation task into N lines, $P = [L_1, L_2, ...]$ and use Algorithm 1.

To compute the rhyming words $Rh(w)$ and vowel phonemes $Vow(w)$ of the words, we use Python's pronouncing library [20].

Algorithm 1 PPLM-Rhyme algorithm

procedure PPLMR(C, T)
    $P \leftarrow \{\}$
    $L_1 = PPLM(C, T)$
    $P.insert(L_1)$
    for $i \leftarrow 2 <= N$ **do**
        $W_r \leftarrow \{\}$
        for $w \in P$ **do**
            if $Vow(w) >= 2$ **then**
                $W_r.extend(Rh(w))$
            end if
        end for
        $L_i = PPLM(P, W_r)$
        $P.insert(L_i)$
    end for
    return P
end procedure

CHAPTER 4

EVALUATION

We evaluate the three models described in Chapter 3 using the metrics: rhyme density, relevance and fluency. The poems used for evaluation are generated for the test-set explained in Section 3.1.

4.1 Rhyme Density

Introduced by Malmi et. al. (2015) [21], rhyme density measures the amount of matching vowel sounds (multisyllabic assonance rhyme). It is computed using the maximum matching vowel phoneme sequence for each word in the poem. For a poem P, let V_w represent the vowel phonemes of a word w in P, i.e.

$$V_w = [v_1, v_2, \cdots], w \in P \quad \text{(Eqn 4.1)}$$

Let V be the concatenation of V_w for all words in P, i.e.

$$V = [V_{w_1} V_{w_2} \cdots] \quad \text{(Eqn 4.2)}$$

The we compute the rhyme density using the function, $RD(P)$ as:

$$RD(P) = \frac{\sum_{i=1}^{n} MaxMatch(V_{w_i}, V \setminus V_{w_i})}{n}, w_i \in P \qquad \text{(Eqn 4.3)}$$

Where, $MaxMatch(A,B)$ computes the length of maximum matching sub-sequence of arrays A and B.
We compute the vowel phonemes of every word using Python's pronouncing library [20].

4.2 Relevance

We compute the relevance of each image caption and tags with the poem generated both manually and using an automated metric.

4.2.1 Manual Relevance

For each image caption and tag, a human annotation score of 1, 2 or 3 is given to the poems generated by the three models.

- **1:** The poem is not related to the image caption or tags.
- **2:** The poem is moderately related to the image caption or tags (Some references to the content in caption/tags is made).
- **3:** The poem matches the image caption and tags well (The main point of the poem revolves around the caption/tag content).

4.2.2 Automated Relevance

Using the average Glove word-embedding [22] for the image caption and poem, we compute the cosine distance between them. Let $E(L)$ be a function which returns the

average Glove word-embedding of words in a sentence L. Let P_i denote the i^{th} line in the poem P. Then, we can compute the relevance score as:

$$Rel(P,C) = \frac{\sum_{i=1}^{n} CosDist(E(P_i), E(C))}{n}, P_i \in P \quad \text{(Eqn 4.4)}$$

Where, $CosDist(A,B)$ is the cosine distance of vectors A and B, given as:

$$CosDist(A,B) = \frac{A \cdot B}{|A||B|} \quad \text{(Eqn 4.5)}$$

4.3 Fluency

Similar to Li et. al. (2015) [23], we use Distinct-1 and Distinct-2 scores to compute fluency of our model. These metrics are simply the ratio of distinct unigrams and bi-grams respectively. The GPT-2 and its variants that we use are notorious for repeating the same words many times in the output [3]. Using the distinct scores allows us to assess this repetition and accordingly interpret their fluency.

We compute the average of Distinct-1 and Distinct-2 scores as fluency of the poem. Let $NG_n(P)$ be a function which returns a list of K n-grams in a poem P. Let I be an indicator function which holds the value 1 if the condition is satisfied and 0 otherwise. Then, we can define Distinct-n as follows:

$$Dist_n(P) = \frac{\sum_{i=1}^{K} I(count(ng_i) = 1)}{K}, ng_i \in NG_n(P) \quad \text{(Eqn 4.6)}$$

Using $Dist_n(P)$, we can compute Fluency as:

$$Fluency(P) = \frac{Dist_1(P) + Dist_2(P)}{2} \quad \text{(Eqn 4.7)}$$

CHAPTER 5

RESULTS

The results obtained after running the evaluation metrics discussed in Chapter 4 on the poems generated by models explained in Chapter 3 on the test-set (Section 3.1) are given in Table TC5.1. The values reported are averages over the test-set.

Table TC5.1: Comparison of GPT-2, TransferTranfo and PPLM-Rhyme

	GPT-2	TransferTransfo	PPLM-Rhyme
Rhyme Density	0.884	0.859	**1.038**
Relevance (Manual)	1.507	1.542	**1.873**
Relevance (Automated)	**0.833**	0.820	0.813
Fluency	0.848	**0.910**	0.743

CHAPTER 6

OBSERVATIONS

While all the models are able to mimic the structure of the poems in the data-set, their performance varies in different metrics, which we discuss below. Figures FC6.1 and FC6.2 show a comparison of poems generated by the three models for the same image.

6.1 PPLM-Rhyme

From the results, we observe an improvement of about 17% in rhyme density of the poems generated by PPLM-Rhyme (compared to the baseline GPT-2). This supports our hypothesis that using rhyming words in the bag-of-words attribute model of PPLM leads to rhyme generation. As an example, Figure FC6.1 shows that PPLM-Rhyme uses words such as ”hour”, ”power” and ”tower” to rhyme with ”flower”.

In terms of relevance, the automated score is low, while the manual score is high. This is because the first one or two lines of most poems are relevant to the image. The subsequent lines, however, deviate from the context in order to fit in the rhyming words. This might be resulting in a low cosine similarity score for these lines taken with the image caption. For example, both the Figures FC6.1 and FC6.2 initially refer to a flower and rocky hill respectively, matching the image content. But, the next lines deviate from this context.

PPLM-Rhyme: *i was once a flower*
and all the flower of my own kind
was made of gold in a minute
but in that hour
the gold was all gone
and power and power was no more
i saw a new dawn
and the sky a golden tower

Transfo:
the roses are always pretty
like birds in the air
a golden little star
that was looked at by all
a rose i could see in the bough

GPT2:
the flowers are so beautiful looking
their beauty is so great
you can't help but love the life that blooms
Look let's see
the flowers when they are all blooming

Figure FC6.1: Comparing poems for a flower image

PPLM-Rhyme: *in the morning of a*
tree on a rocky hill
in the evening dew-blossoms in the
night- mourning
we are the children of God
the world's largest man is warning
the other kids
he knows that they're coming

Transfo:
trees in the forest
it's the best way to go
out of the wood
and see a new future
by following the path of time

GPT2:
the bird takes flight and goes down
into the darkness
and the tree on a rocky hill
has no more reason to cry
what is it that makes the world go
round
how to live through the dark
what are the

Figure FC6.2: Comparing poems for a rocky forest image

Another observation regarding the poems generated by PPLM-Rhyme is their tendency to repeat the rhyming words multiple times. For example, the word ”power” is repeated in the same line of Figure FC6.1's poem. Since fluency is designed to penalize repetition, the fluency recorded is the lowest for PPLM-Rhyme.

6.2 TransferTransfo

Though the model performs on par with if not slightly better than GPT-2 in terms of relevance and rhyme density, it is a clear winner in fluency (about 7% improvement compared to GPT-2).

Though the reason for this is not explicit, an observation of poems generated by this model show more adherence to the poetic style and frequent indirect references to the context, similar to the poems in the data-set. This improved poetic style might be due to the objective of next line classification that is added to language modelling. Since poems in the data-set use a wide range of vocabulary, a better poetic style of Transfer-Transfo might have translated to an improved fluency.

CHAPTER 7

CONCLUSION

In this thesis, we introduce the Generative Transformer, the challenge of poetry generation and define our problem statement in Chapter 1. After reviewing research in the related areas (Chapter 2), we explain the design of the models TransferTransfo [2], PPLM [3] and their adaptations (Chapter 3) to the image-to-poetry problem. We evaluate these models in terms of rhyme density, fluency and relevance (to image caption and tags) in Chapter 4 and report the results in Chapter 5. Using the results, we explain how our proposed model, PPLM-Rhyme demonstrates a superior rhyme density (improvement of about 17% over the baseline GPT-2) in Section 6.1.

Some of the future directions for this research problem could be:

- Using an attribute model which directly uses image embedding to compute $P(a|x)$ in the PPLM framework. This can allow the model to capture information (such as colour, scene, etc.) in the image which is not reflected in the caption or tags.
- Instead of using the BOW attribute model to generate rhyme, a rhyme classification model can be integrated into the PPLM framework. This could reduce the pressure that PPLM-Rhyme faces to accommodate the words in BOW, thus mitigating the fluency and relevance problems.

Bibliography

[1] Ruli Manurung. An evolutionary algorithm approach to poetry generation. 01 2003.

[2] Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. Transfertransfo: A transfer learning approach for neural network based conversational agents. *CoRR*, abs/1901.08149, 2019.

[3] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation, 2019.

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

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[6] Bei Liu, Jianlong Fu, Makoto P. Kato, and Masatoshi Yoshikawa. Beyond narrative description. *2018 ACM Multimedia Conference on Multimedia Conference - MM '18*, 2018.

[7] Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. Deep-speare: A joint neural model of poetic language, meter and rhyme.

In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1948–1958, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[8] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[9] Microsoft. Microsoft computer vision api.

[10] Alec Radford. Improving language understanding by generative pre-training. 2018.

[11] Hugo Gonçalo Oliveira. A survey on intelligent poetry generation: Languages, features, techniques, reutilisation and evaluation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 11–20, Santiago de Compostela, Spain, September 2017. Association for Computational Linguistics.

[12] Zichao Yang, Zhiting Hu, Chris Dyer, Eric P. Xing, and Taylor Berg-Kirkpatrick. Unsupervised text style transfer using language models as discriminators. *CoRR*, abs/1805.11749, 2018.

[13] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward controlled generation of text, 2017.

[14] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment, 2017.

[15] Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. Style transfer in text: Exploration and evaluation, 2017.

[16] Peter Potash, Alexey Romanov, and Anna Rumshisky. GhostWriter: Using an LSTM for automatic rap lyric generation. In *Proceedings of the 2015 Confer-*

ence on Empirical Methods in Natural Language Processing, pages 1919–1924, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[17] Rui Yan. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *IJCAI*, 2016.

[18] Brendan Bena and Jugal Kalita. Introducing aspects of creativity in automatic poetry generation, 2020.

[19] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation, 2019.

[20] Allison Parrish. Python pronouncing library.

[21] Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. Dopelearning. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016.

[22] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[23] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models, 2015.

APPENDIX A

DENSECAP VS MS-API

After initial experiments with the captioning model, DenseCap [8], we moved to Microsoft Computer Vision API [9] due to the better quality of captions and tags generated. Figure FA1.1 shows examples of the captions produced by both models for the same images. We also conducted experiments to compare the poems generated by the adapted TransferTransfo model (Subsection 3.2.2) using each captioning method.

We used the evaluation metrics: BLUE score, Novelty (similar to Liu et. al. (2018) [6]) along with the automated relevance score explained in Subsection 4.2.2. The BLEU score is computed between the original poem (poem given in the test-set for the image) and the poem generated using the mentioned captioning mechanism. Novelty-n is the proportion of number of n-grams not in the training set. The average of these metrics computed over the test-set (Section 3.1) for the original poem and the two captioning methods is given in Table TA1.1.

Due to its clear superiority, as evident from these results, we conducted all our experiments in the thesis using MS-API caption and tags.

Table TA1.1: Comparison of DenseCap and MS API

	Test-set	DenseCap	MS API
BLEU score	-	**0.469**	0.211
Novelty-2	0.001	0.138	**0.258**
Novelty-3	0.004	0.541	**0.729**
Relevance	0.785	0.819	**0.842**

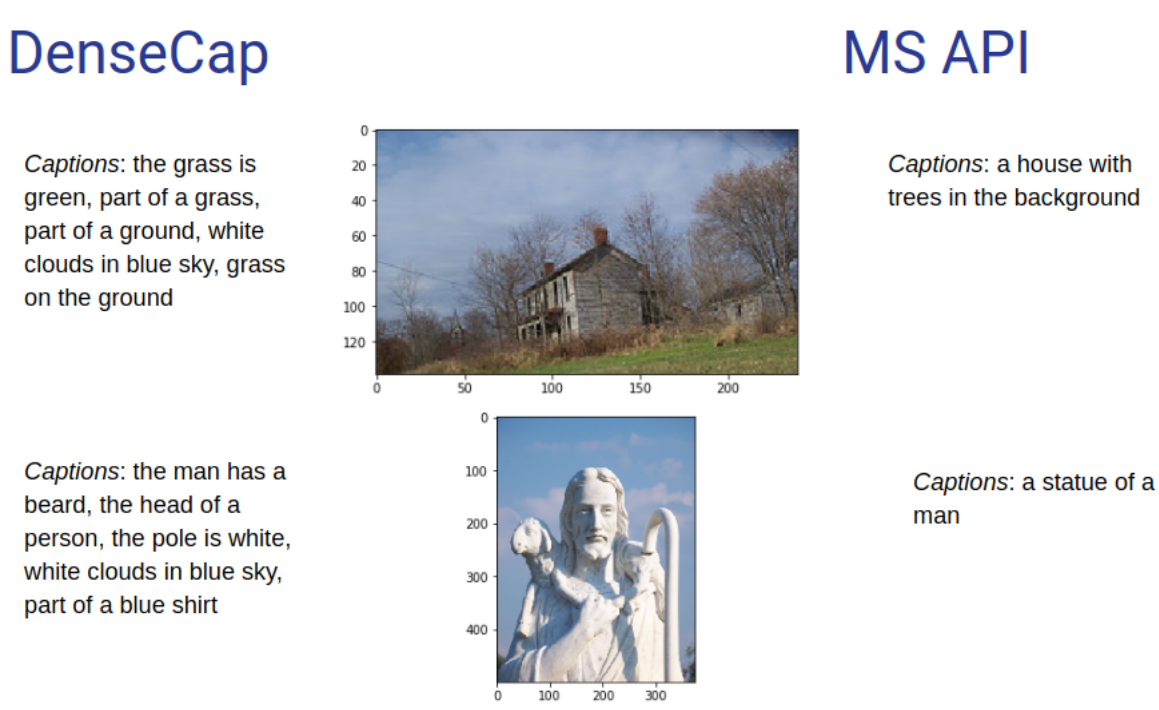

Figure FA1.1: Comparing DenseCap and MS API captions [8, 9]