Grounded Natural Language Generation via Interpretable Hierarchical Operations

Harsh Jhamtani

CMU-LTI-21-021

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Taylor Berg-Kirkpatrick (Chair), University of California San Diego and Carnegie Mellon University
Graham Neubig, Carnegie Mellon University
Alan W. Black, Carnegie Mellon University
Julian McAuley, University of California San Diego

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Abstract

Much recent work in natural language generation has relied on deep learning, often using neural networks with soft attention mechanisms to select salient aspects from data and then construct fluent natural language text. However, in naturally occurring descriptions of data, humans often refer to higher-level patterns which may require complex computations on data. In many cases, neural models using soft attention mechanisms alone struggle to extract such patterns. Moreover, users might often find such models to be difficult to interpret and control. In this thesis, I propose methods for inducing certain types of discrete hierarchical operations on data and text for grounded natural language generation. Compared to using attention alone, such hierarchical operations can better model complex patterns in data, expose interpretable intermediate computations, and enable controllable generation. In the first half of the thesis, I will discuss adding specific discrete hierarchical operations to neural models for different grounded natural language generation tasks, such as image and table captioning, dialog response generation, and constructing reasoning chains for multi-hop question-answering. These tasks span various data modalities (including images, tabular data, numerical data, and knowledge bases). In the second half, I will describe hierarchical methods for content planning in text decoders, studying rhyming patterns in poetry generation and discrete plans for coherent narrative text generation.

Table of Contents

1 Introduction

We live in an era of big data, with billions of devices constantly collecting and storing raw data. However, such data is useless without an expert to analyze it and communicate the findings to relevant stakeholders. For example, there has been rapid digitization of personal medical data, including test reports and data from sensors on wearable devices such as digital watches. However, such data is of not much use if people can't understand it. An automated system can potentially unlock valuable insights by, for instance, detecting and giving an early warning for potential health disorders. Given the exponentially increasing volume of data that is becoming available, there has been an increased effort in building smart systems to assist people in making sense of such data. Importantly, natural language is a key medium for such smart machines to effectively deliver useful insights since users might not have the expertise to interpret a graph or understand a spreadsheet. Thus, extracting useful insights from data, and communicating them to users through automatically generated natural language descriptions, has emerged as an important application area for artificial intelligence (AI) technologies. However, real-world applications of such systems need to be configurable to user preferences and should be as transparent and/or interpretable as possible. Therefore, there is a need to enable some degree of user-control over system outputs as well as expose some working of such systems.

But building such technologies presents a challenge - how can we build systems that reason about data and present their results effectively in natural language? Can we leverage natural language to help systems learn useful abstractions of data similar to what humans often do? Can we learn models that can demonstrate controllable and interpretable use of data on output?

1.1 Motivation

This thesis deals with certain aspects of Grounded NLG (Natural Language Generation). I'll discuss the motivation of the work presented in this thesis in light of the framework proposed in Reiter (2007) which describes four stages in grounded Natural Language Generation as follows: Signal Analysis, Data Interpretation, Document Planning, Micro-planning, and Realization. While much recent work involves neural models trained through end-to-end learning, the framework is still relevant to conceptually understand and analyze how different aspects of models for grounded NLG.

1. **Signal analysis** involves information extraction typically not extending beyond shallow pattern extraction. In contemporary neural models, this would typically be neural data encoders, often including self-attention mechanisms.

2. **Data Interpretation** typically refers to higher-level complex patterns and should be employed when the final text communicates more than basic patterns.

3. **Planning** typically involves high-level text planning involving content selection, which would also govern aspects such as overall coherence and structure planning of text. Contemporary attention mechanisms could be considered as a popular means to achieve this.

4. **Micro planning and realization** for fluent text generation, taking into account any stylistic aspects of the text.

In the rest of this subsection, I will briefly summarize how many of the contemporary techniques are often not able to adequately model *Data Interpretation* and *Planning* aspects of NLG. Towards this goal, I will be discussing two major existing issues: (1) Failure to model useful abstractions for complex patterns (2) Failure to expose interpretable intermediate decisions, and enabling controllable generation. Modeling complex patterns on data can be considered to fall into *Data Interpretation* and *Planning* buckets, while modeling useful structures on text can be viewed to fail into *Planning* aspects (i.e. *Planning* encompasses both data and text aspects.).

1.1.1 Modeling Complex Patterns for *Data Interpretation* and *Planning* aspects

Recently, there has been interest in automatic description of tabular data to generate biographies from tables of biographical information (Lebret et al., 2016; Wiseman et al., 2018), or recipes from ingredient lists (Kiddon et al., 2016a). In many of these tasks, the main focus is on designing systems that aggregate information into a low dimensional vector or a set of such vectors and using neural attention mechanisms to select entries. However, in many naturally occurring descriptions of data, humans often refer to higher-level patterns. For example, consider the following stock market description 'the stock prices peaked towards the end of the day'. To be able to generate such descriptions from stock market data would require a model to be able to effectively identify the notions of 'peak' and 'towards the end'. As I will discuss in more detail later, simply using popular neural encoder-decoder frameworks often leads to inferior results and are less interpretable.

Much prior work on text generation relies on recurrent and transformer neural networks trained to maximize the likelihood of data. However, such models, including more recent large pre-trained models such as GPT2 (Radford, 2018), often fail to capture the overall structure and coherency in multi-sentence or long-form text (Bosselut et al., 2018; Holtzman et al., 2018; See et al., 2019). To rectify this, prior work has proposed loss functions that encourage overall coherency or other desired behavior (Li et al., 2016b; Zhang and Lapata, 2017; Bosselut et al., 2018). However, most of these approaches rely on manually provided definitions of what constitutes a good or suitable structure, thereby limiting their applicability. In this thesis, I propose and discuss methods to induce such patterns from data itself, such as learning rhyming constraints from poetry data without being provided any external phonetic information.

1.1.2 Interpretable and Controllable Generation

While recent progress has achieved great success in being able to generate fluent text, there still exist many issues. For instance, consider the task of difference description between a pair of images. A model for this task would need to screen out noisy isolated changes, group together related changes, and describe one change at a time. A straightforward method for this task would be to pass the images through a deep neural encoder and train it to predict the difference captions. However, such a model performs rather poorly. Firstly, it is difficult to interpret if the model actually screens out noise and groups pixels as expected. Attempts to use attention weights as explanations can be unreliable (Jain and Wallace, 2019). Secondly, since there can be multiple differences between two images, the above model is opaque with respect to which particular one is being described. Issues of such nature are observed across a wide spectrum of text generation models and tasks and include lack of effective means of controlling the output, hallucinating content which is not present, lack of generalization to unseen contexts, and so on. While past work with neural module networks (Andreas et al., 2016a) has explored such goals for tasks such as question answering, there has been a lack of work in such direction for data-to-text tasks.

Much prior work on text generation has relied on directly generating the target text, without exposing or generating interpretable plans for a generation. Some recent work has shown the usefulness of adding the correct level of planning (Yao et al., 2019; Fan et al., 2019). However, most prior techniques rely on first tagging data with desired structure or plan, and then using such codes directly for training and/or decoding. While this provides a certain level of explanation and control, naturally occurring data often lacks such tags. Thus there is a need for a new set of models and training techniques for inducing such useful structures or plans from the data itself.

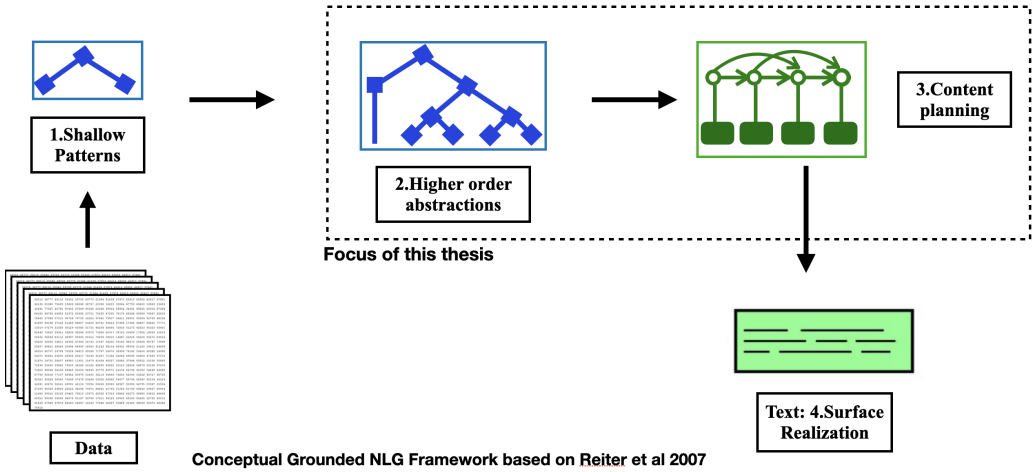


Figure 1.1: Overview: The figure highlights the focus of this thesis in context of a conceptual framework for grounded natural language generation proposed by Reiter (2007). Reiter (2007) outlined four major conceptual stages in grounded natural language generation. In this thesis, I propose and discuss how models with induced hierarchical operations for (data interpretation and planning aspects of) grounded natural language generation often generate better quality outputs, expose patterns to effectively control the output, and are much more interpretable compared to many contemporary methods relying on neural attention alone.

1.2 Thesis Overview

Thesis statement: *Models with Induced Hierarchical Operations for (data interpretation and planning aspects of) Grounded Natural Language Generation often generate better quality outputs, often expose patterns to effectively control the output, and are much more interpretable compared to many contemporary methods relying on neural attention alone.*

Hierarchical Interpretable Operations: I use the term Hierarchical Interpretable Operations to refer to a variety of discrete (and often of hierarchical nature) operations including but not limited to discrete content selection, sequence of deterministic programs of increasingly abstract nature, alignment between portions of data and text, sparse hierarchical grouping, and so on. For example, many of the proposed methods in this thesis leverage discrete latent random variables for data selection and alignment. This is in contrast to the more popular way of using neural soft attention mechanisms on the entire input. As we will demonstrate in various chapters, the proposed hierarchical interpretable operations are 1) more interpretable to humans 2) often acts as useful inductive biases leading to improved perplexity, and other metrics 3) often exhibit desirable characteristics such as controllable text generation. [1]

Part 1: Learning Interpretable Hierarchical Operations on Data: In the first part of the thesis, I discuss applying the notion of hierarchical operations in model architectures for modeling complex patterns in data for different tasks such as image captioning, knowledge-based dialog response generation, constructing reasoning chains for multihop question-answering, etc. spanning across various data modalities (such as images, structured data, numerical data, knowledge bases). The proposed approach is in contrast to popular contemporary techniques of encoding entire data input using neural soft attention instead of learning and exposing more interpretable operations.

1. **Chapter 2: Difference Description Generation via Interpretable Hierarchical Operations:** In **Chapter 2**, I discuss models for difference descriptions in structured data (Jhamtani et al., 2018)

[1] A note on Interpretability: In context of machine learning models, interpretability can refer to a variety of notions and corresponding objectives (Lipton, 2016). For example, a high level English language explanation of a model for an end user versus a gradient based visualization of model parameters for debugging purposes. In this thesis, I do not limit to a single global definition of model interpretability, and instead separately describe the contours of interpretability being evaluated for the experiments and models in question.

and images (Jhamtani and Berg-Kirkpatrick, 2018) via latent interpretable operations. We release two new datasets, one for chess game move commentary generation, and the second one for our proposed task of captioning differences between a pair of similar images. Through experiments, we demonstrate that the proposed models using latent operations perform better than just using neural soft attention as per various automated and human evaluation studies. Our methods for difference description generation have applications in computer assisted tracking of changes in media assets, automated game commentary generation, and generating summaries of changes in documents.

2. **Chapter 3: Fine-grained Reasoning for Knowledge Base Grounded Text Generation:** In this chapter, we propose novel modeling approaches using latent reasoning chains over knowledge base for explanation generation in multi-hop question answering (Jhamtani and Clark, 2020) and incorporating common-sense knowledge in a dialog system (Majumder and Jhamtani et al. (2020)). We compare proposed methods against prior work which encodes a subset of relevant knowledge instead of performing fine-grained selection. We observe that the proposed modeling techniques expose interpretable reasoning chains over knowledge base, lead to improved output quality over baselines, and are more robust to certain types of perturbations to the input. Our proposed methods for explanation generation has applications in building tutoring systems for children. Additionally, proposed techniques of incorporating commonsense knowledge in dialog system can enable more natural conversations between people and dialog agents.

3. **Chapter 4: Inducing Neuro-Symbolic Rules for Numerical Data:** In this chapter, we propose methods to induce modules that detect useful trends such as peak or dip in time series numerical data, being guided only by accompanying natural language descriptions (Jhamtani and Berg-Kirkpatrick, 2021). We propose a novel *truth-conditional* method to learn modules that combine in a latent computation graph, which outputs the truth value of whether the feature represented by the composed computation graph holds true for a given data point or not. Outputs from the proposed model demonstrate higher precision and diversity compared to various baselines, and can potentially be extended for use in other natural language generation setups to improve on factual correctness of machine generated text.

Part II: Latent Discrete Plans for Long-form Text Generation: In the second part of the thesis, I discuss how related techniques of hierarchical operations can be leveraged to induce latent global plans for text generation. Many neural language models often fail to capture higher-level structures present in text: for example, rhyming patterns present in poetry or narrative plan for coherent long-form text generation[2]. Prior work has heavily relied on the injection of external knowledge such as rhyming knowledge, or the use of external tools to tag narrative plans, to effectively model such long-range patterns. In the second part of the thesis, I focus on investigating whether such long-range patterns or structures can be treated as latent variables, and be learned from the data, resulting in better quality outputs.

1. **Chapter 5: Structured Discriminators for Modeling Long-Range Latent Patterns:** Much prior work on poetry generation uses manually defined rhyming and rhythm constraints. We propose a novel *structured discriminator* in a generative adversarial setup that operates on a matrix of self-similarity values of all pairs of line-ending words. The proposed discriminator (1) induces an accurate rhyming metric, and (2) guides a generator to learn rhyming patterns in data without being provided with phonetic information (Jhamtani et al., 2019). We successfully apply similar techniques for modeling self-repetition in music generation (Jhamtani and Berg-Kirkpatrick, 2019), which induces pitch and rhythm-based musical measure similarities. Such models have applications in assisting creative professionals and students for applications such as music generation.

[2] In such cases, the generated text can be considered as grounded in some latent data such as a specific rhyming scheme.

2. **Chapter 6: Latent Discrete Generation Plans for Controllable and Coherent Generation:** Prior work has shown that long-form text generation can benefit by first creating a rough sketch or plan for the content and then generates text conditioned on the plan. However, naturally occurring data is not tagged with such plans. Compared to rhyming scheme constraints, this type of structure is less restricted and much less formal. We propose a deep generative model which uses a hierarchical discrete latent plan realized via a sequence of keywords. To train the model (Jhamtani and Berg-Kirkpatrick, 2020), we propose a constrained inference network which (1) learns to identify useful keywords from sentences (2) guides the model in learning to generate sequences of keywords in the generation plan.

3. **Chapter 7: Retrieved Snippets as Discrete Plans for Guided Generation** Outputs of many existing dialog models are limited by the 'knowledge' available to the models at training time. In this chapter, I discuss methods to introduce relevant additional 'knowledge' at decoding time without the need to re-train the models. In Majumder et al. (2021), we equip persona grounded dialog models with 'background stories' related to a persona by retrieving fictional narratives from existing story datasets (e.g. ROCStories). In Jhamtani et al. (2021), we equip dialog models with dictionary *translations* of figurative English expressions to their literal counterparts to improve dialog model handling. Such proposed techniques of incorporating commonsense knowledge in dialog system can enable more natural conversations between people and dialog agents.

2 Difference Description via Hierarchical Interpretable Operations

In this chapter I discuss inducing useful hierarchical abstractions on data guided by accompanying natural language annotations contrasting the differences between two states or a pair of data points. Modeling the changes or edits is important as our thinking is often grounded in relative states or conditions. In addition to being a useful task in itself, this can be a useful approach to learn fine-grained classifiers (Khosla et al., 2011), and as a tool for eliciting variety of lexicon (Maji, 2012). A straightforward approach to such difference description generation tasks can be to simply apply popular encoders on the data. We however are interested in learning useful and interpretable hierarchical abstractions on data, which can hopefully outperform baselines as well. First I discuss Spot-the-diff (Jhamtani and Berg-Kirkpatrick, 2018), where we propose a model to expose salient groups of pixels to describe difference between two similar images. Our approach begins with pixel-level difference for removal of uninteresting isolated changes, followed by clustering of pixels for object grouping and spatial similarity, and finally followed by cluster alignment to text description. We observe that the proposed approach acts as a useful inductive bias leading to improved performance, and leads to controllable and interpretable caption generation. I conclude the chapter with a brief discussion on chess commentary generation (Jhamtani et al., 2018), where we discuss models for a chess move commentary by comparing successive chess board states through operations which elicit which piece has moved, potential piece interactions, and game score changes.

2.1 Introduction

The interface between human users and collections of data is an important application area for artificial intelligence (AI) technologies. Can we build systems that effectively interpret data and present their results concisely in natural language? One recent goal in artificial intelligence has been to build models that are able to interpret and describe visual data to assist humans in various tasks. For example, image captioning systems (Vinyals et al., 2015b; Xu et al., 2015; Rennie et al., 2017; Zhang et al., 2017) and visual question answering systems (Antol et al., 2015; Lu et al., 2016; Xu and Saenko, 2016) can help visually impaired people in interacting with the world. Another way in which machines can assist humans is by identifying meaningful patterns in data, selecting and combining salient patterns, and generating concise and fluent 'human-consumable' descriptions. For instance, text summarization (Mani and Maybury, 1999; Gupta and Lehal, 2010; Rush et al., 2015) has been a long standing problem in natural language processing aimed at providing a concise text summary of a collection of documents.

In this paper, we propose a new task and accompanying dataset that combines elements of image captioning and summarization: the goal of 'spot-the-diff' is to generate a succinct text description of *all* the salient differences between a pair of similar images. Apart from being a fun puzzle, solutions to this task may have applications in assisted surveillance, as well as computer assisted tracking of changes in media assets. We collect and release a novel dataset for this task, which will be potentially useful for both natural language and computer vision research communities. We used crowd-sourcing to collect text descriptions of differences between pairs of image frames from video-surveillance footage (Oh et al., 2011), asking annotators to succinctly describe *all* salient differences. In total, our datasets consist of

Man by yellow poles in after pic wasn't there before.
There are two people in middle of court that were not there earlier.
Person crossing crosswalk is no longer there

The blue truck is no longer there.
A car is approaching the parking lot from the right

Figure 2.1: Examples from Spot-the-diff dataset: We collect text descriptions of all the differences between a pair of images. Note that the annotations in our dataset are exhaustive wrt differences in the two images i.e. annotators were asked to describe all the visible differences. Thus, the annotations contain multi-sentence descriptions.

descriptions for 13,192 image pairs. Figure 2.1 shows a sample data point - a pair of images along with a text description of the differences between the two images as per a human annotator.

There are multiple interesting modeling challenges associated with the task of generating natural language summaries of differences between images. First, not all low-level visual differences are sufficiently salient to warrant description. The dataset presents an interesting source of supervision for methods that attempt to learn models of visual salience (we additionally conduct exploratory experiments with a baseline salience model, as described later). Second, humans use different levels of abstraction when describing visual differences. For example, when multiple nearby objects have all moved in coordination between images in a pair, an annotator may refer to the group as a single concept (e.g. 'the row of cars'). Third, given a set of salient differences, planning the order of description and generating a fluent sequence of multiple sentences is itself a challenging problem. Together, these aspects of the proposed task make it a useful benchmark for several directions of research.

Finally, we experiment with neural image captioning based methods. Since salient differences are usually described at an object-level rather than at a pixel-level, we condition these systems on a first-pass visual analysis that exposes clusters of differing pixels as a proxy for object-level differences. We propose a model which uses latent discrete variables in order to directly align difference clusters to output sentences. Additionally we incorporate a learned prior that models the visual salience of these difference clusters. We observe that the proposed model which uses alignment as a discrete latent variable outperforms those that use attention alone.

2.2 'Spot-the-diff' Task and Dataset

We introduce 'spot-the-diff' dataset consisting of 13,192 image pairs along with corresponding human provided text annotations stating the differences between the two images. Our goal was to create a dataset wherein there are meaningful differences between two similar images. To achieve this, we work with image frames extracted from VIRAT surveillance video dataset (Oh et al., 2011), which consists of 329 videos across 11 frames of reference totalling to about 8.5 hours of videos.

2.2.1 Extracting Pairs of Image Frames

To construct our dataset, we first need to identify image pairs such that some objects have changed positions or have entered or left in the second image compared to the first image. To achieve this, we first extract a certain number of randomly selected image frame pairs from a given video. Thereafter, we

Total number of annotations	13,192
Mean (std dev.) number of sentences per annotation	1.86(1.01)
Vocabulary size	2404
Frequent word types (>=5 occurrences)	1000
Word tokens that are frequent word types	97%
Mean (std dev.) number of words in sentence:	10.96(4.97)
% Long sentences (> 20 words)	5%

Table 2.1: Summary statistics for spot-the-diff dataset

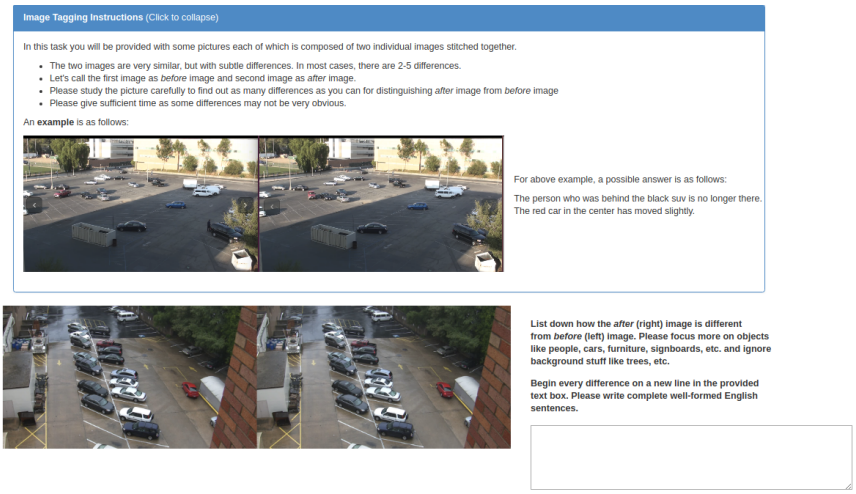


Figure 2.2: AMT (Amazon Mechanical Turk) HIT (Human Intelligence Task) setup for data collection. We provide the annotators with detailed instructions, along with an example showing how to perform the task. We request the annotators to write complete English sentences, with each sentence on a separate line. We collect a total of 13,192 annotations.

compute the L_2 distance between the two images in each pair (under RGB representation). Finally, we set a lower and a upper threshold on the L_2 distance values so calculated to filter out the image pairs with potentially too less or too many changes. These thresholds are selected based on manual inspection. The resulting image pairs are used for collecting the difference descriptions.

2.2.2 Human Annotation

We crowd-sourced natural language differences between images using Amazon Mechanical Turk. We restrict to annotators from primarily Anglophone countries: USA, Australia, United Kingdom, and Canada, as we are working with English language annotations. We limit to those participants which have lifetime HIT $> 80\%$. We award 5 cents per HIT (Human Intelligence Task) to participants. We provide the annotators with an example on how to work on the task. We request the annotators to write complete English sentences, with each sentence on a separate line. We collect a total of 13192 annotations.

2.2.3 Dataset statistics

Table 2.1 shows some summary statistics about the collected dataset. Since we deal with a focused domain, we observe a small vocabulary size. On an average there are 1.86 reported differences / sentences per image pair. We also report inter-annotator agreement as measured using text overlap of multiple

Dataset	BLEU-1/2/3/4	ROUGE-L
Spot-the-diff ($A = 3$)	0.41/0.25/0.15/0.08	0.31
MS-COCO ($A = 3$)	0.38/0.22/0.13/0.08	0.34
MS-COCO ($A = 5$)	0.66/0.47/0.32/0.22	0.48

Table 2.2: Human agreement for our dataset: We report measures such as BLEU and ROUGE when 'evaluating' one set of human generated captions against the remaining sets. $A = k$ represents k captions per data point, out of which 1 is chosen as hypothesis, while remaining $k-1$ act as references.

annotations for the same image pair. We collect three sets of annotations for a small subset of the data (467 data points) for the purpose of reporting inter-annotator agreements. We thereby calculate BLEU and ROUGE-L scores by treating one set of annotations as 'hypothesis' while remaining two sets act as 'references'(Table 2.2). We repeat the same analysis for MS-COCO dataset and report these measures for reference. The BLEU and METEOR values for our dataset seem reasonable and are comparable to the values observed for MS-COCO dataset.

2.3 Modeling Difference Description Generation

We propose a neural model for describing visual difference based on the input pair of images that uses latent alignment variable to capture visual salience. Since most descriptions talk about higher-level differences rather than individual pixels, we first perform a visual analysis that pre-computes a set of difference clusters in order to approximate object-level differences, as described next. The output of this analysis is treated as input to a neural encoder-decoder text generation model that incorporates a latent alignment variable and is trained on our new dataset.

2.3.1 Exposing Object-level Differences

We first analyze the input image pair for the pixel-level differences by computing a *pixel-difference mask*, followed by a local spatial analysis which segments the difference mask into clusters that approximate the set of object-level differences. Thereafter, we extract image features using convolutional neural models and use these as input to a neural text generation model, described later.

Pixel-level analysis: The lowest level of visual difference is individual differences between corresponding pixels in the input pair. Instead of requiring our description model to learn to compute pixel-level differences as a first step, we pre-compute and directly expose these to the model. Let $X = (\boldsymbol{I_1}, \boldsymbol{I_2})$ represent the image pair in a datum. For each such image pair in our dataset, we obtain a corresponding pixel-difference mask $\boldsymbol{M}$. M is a *binary-valued* matrix of the same dimensions (length and width) as each of the images in the corresponding image pair, wherein each element in the matrix is 1 (active) if the corresponding pixel is *different* between the input pair, and 0 otherwise. To decide whether a pair of corresponding pixels in the input image pair are sufficiently *different*, we calculate the L_2-distance between the vectors corresponding to each pixel's color value (three channels) and check whether this difference is greater than a threshold δ (set based on manual inspections).

While the images are extracted from supposedly still cameras, we do find some minor shifts in the camera alignment, which is probably due to occasional wind but may also be due to manual human interventions. These shifts are rare and small, and we align the images in the pair by iterating over a small range of vertical and horizontal shifts to find the shift with minimum corresponding L_2-distance between the two images.

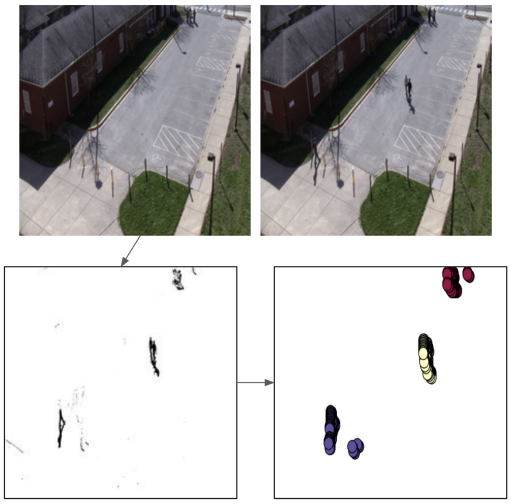


Figure 2.3: Exposing Object-level Differences: Before training a model to describe visual difference, we first compute pixel-level differences, as well as a segmentation of these differences into clusters, as a proxy for exposing object-level differences. The first row shows the original image pair. Bottom left depicts the pixel-difference mask, which represents extracted pixel-level differences. The segmentation of the pixel-difference mask into clusters is shown in the bottom right.

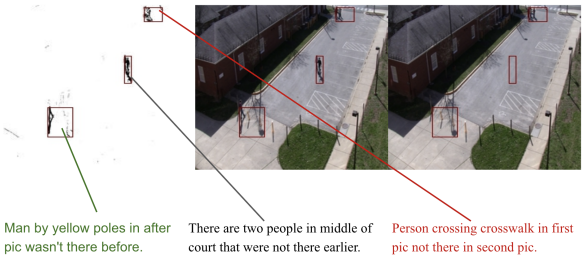


Figure 2.4: The figure shows the *pixel-difference* mask for the running example, along with the two original images, with bounding boxes around clusters. Typically one or more difference clusters are used to frame one reported difference / sentence, and it is rare for a difference cluster to participate in more than one reported difference.

Object-level analysis: Most visual descriptions refer to object-level differences rather than pixel-level differences. Again, rather than requiring the model to learn to group pixel differences into objects, we attempt to expose this to the model via pre-processing. As a proxy for object-level difference, we segment the pixel-level differences in the pixel-difference mask into clusters, and pass these clusters as additional inputs to the model. Based on manual inspection, we find that with the right clustering technique, this process results in groupings that roughly correspond to objects that have moved, appeared, and disappeared between the input pair. Here, we find that density based clustering algorithms like DBScan (Ester et al., 1996) work well in practice for this purpose. In our scenario, the DBScan algorithm predicts clusters of nearby active pixels, and marks outliers consisting of small groups of isolated active pixels, based on a calculation of local density. This also serves as a method for pruning any noisy pixel differences which may have passed through the pixel-level analysis.

As the output of DBScan, we obtain segmentation of the pixel difference matrix M into *difference clusters*. Let the number of *difference clusters* be represented by K (DBScan is a non-parametric clustering method, and as such the number of clusters K is different for each data point.). Now, let's define $\boldsymbol{C_k}$ as another binary-valued mask matrix such that the elements in matrix corresponding to the k^{th} difference cluster are 1 (active) while rest of the elements are 0.

X=(I_1,I_2)	:	Image pair in the datum
M	:	Pixel-difference mask is a binary-valued matrix depicting pixel-level changes
F_1, F_2	:	Image feature tensors for I_1 and I_2 respectively
K	:	Number of segments
C_k	:	Cluster mask corresponding to k^{th} difference cluster
T	:	Number of reported differences / sentences
z_i	:	Discrete alignment variable for the i^{th} sentence. $z_i \in \{1,2,...,K\}$
$S_1,..,S_T$	:	List of T Sentences

Table 2.3: Summary of notation used in description of the method.

2.3.2 Text Generation Model

We observe from annotated data that each individual sentence in a full description typically refers only to visual differences within a single cluster (see Figure 2.4). Further, on average, there are more clusters than there are sentences. While many uninteresting and *noisy* pixel-level differences get screened out in preprocessing, some uninteresting clusters are still identified. These are unlikely to be described by annotators because, even though they correspond to legitimate visual differences, they are not visually salient. Thus, we can roughly model description generation as a cluster selection process.

In our model, which is depicted in Figure 2.5, we assume that each output description, which consists of sentences $S_1,\ldots,S_T$, is generated sentence by sentence conditioned on the input image pair $X = (I_1,I_2)$. Further, we let each sentence S_i be associated with a latent alignment variable, $z_i \in \{1,\ldots,K\}$, that chooses a cluster to focus on (Vinyals et al., 2015a). The choice of z_i is itself conditioned on the input image pair, and parameterized in a way that lets the model learn which types of clusters are visually salient and therefore likely to be described as sentences. Together, the probability of a description given an image pair is given by:

$$P(S_1,..,S_T|X) = \sum_{z_1,..,z_T} \prod_{i=1}^{T} \underbrace{P(S_i|z_i,X;\theta))}_{\text{decoder}} \underbrace{P(z_i|X;w)}_{\text{alignment prior}} \tag{2.1}$$

The various components of this equation are described in detail in the next few subsections. Here, we briefly summarize each. The term $P(z_i|X;w)$ represents the prior over the latent variable z_i and is parameterized in a way that lets the model learn which types of clusters are visually salient. The term $P(S_i|z_i,X;\theta)$ represents the likelihood of sentence S_i given the input image pair and alignment z_i. We employ masking and attention mechanisms to encourage this decoder to focus on the cluster chosen by z_i. Each of these components conditions on visual features produced by a pre-trained image encoder.

The alignment variable z_i for each sentence is chosen independently, and thus our model is similar to IBM Model 1 (Brown et al., 1993) in terms of its factorization structure. This will allow tractable learning and inference as described in subsection 2.3.3. We refer to our approach as DDLA (**D**ifference **D**escription with **L**atent **A**lignment).

Alignment prior: We define a learnable prior over alignment variable z_i. In particular, we let the multinomial distribution on z_i be parameterized in a log-linear fashion using feature function $g(z_i)$. Specifically, we consider the following four features: the length, width, and area of the smallest rectangular region enclosing cluster z_i, and the number of active elements in mask C_{z_i}. Specifically, we let $P(z_i|X;w) \propto \exp(w^T g(z_i))$.

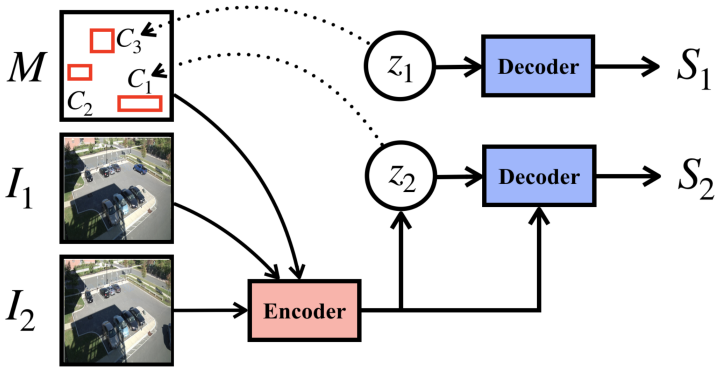


Figure 2.5: Model architecture for generating difference descriptions. We incorporate a discrete latent variable z which selects one of the clusters as a proxy for object-level focus. Conditioned on the cluster and visual features in the corresponding region, the model generates a sentence using an LSTM decoder. During training, each sentence in the full description receives its own latent alignment variable, z.

Visual encoder: We extract images features using ResNet (He et al., 2016) pre-trained on Imagenet data. Similar to prior work (Xu et al., 2015), we extract features using a lower level convolutional layer instead of fully connected layer. In this way, we obtain image features of dimensionality $14 * 14 * 2096$, where the first two dimensions correspond to a grid of coarse, spatially localized, feature vectors. Let F_1 and F_2 represent the extracted feature tensors for I_1 and I_2 respectively.

Sentence decoder: We use an LSTM decoder (Hochreiter and Schmidhuber, 1997) to generate the sequence of words in each output sentence, conditioned on the image pair and latent alignments. We use a matrix transformation of the extracted image features to initialize the hidden state of the LSTM decoder for each sentence, independent of the setting of z_i. Additionally, we use an attention mechanism over the image features at every decoding step, similar to the previous work (Xu et al., 2015). However, instead of considering attention over the entire image, we restrict attention over image features to the cluster mask determined by the alignment variable, C_{z_i}. Specifically, we project binary mask C_{z_i} from the input image dimensionality (224*224) to the dimensionality of the visual features (14*14). To achieve this, we use pyramid reduce down-sampling on a smoothed version of cluster mask C_{z_i}. The resulting projection roughly corresponds to the subset of visual features with the cluster region in their receptive field. This projection is multiplied to attention weights.

2.3.3 Learning and Decoding

Learning in our model is accomplished by stochastic gradient ascent on the marginal likelihood of each description with alignment variables marginalized out. Since alignment variables are independent of one another, we can marginalize over each z_i separately. This means running backpropagation through the decoder K times for each sentence, where K is the number of clusters. In practice K is relatively small and this direct approach to training is feasible. Following equation 2.1, we train both the generation and prior in an end-to-end fashion.

For decoding, we consider the following two problem settings. In the first setting, we consider the task of producing a single sentence in isolation. We evaluate in this setting by treating the sentences in the ground truth description as multiple reference captions. This setting is similar to the typical image captioning setting. In the second setting, we consider the full multi-sentence generation task where the system is required to produce a full description consisting of multiple sentences describing all differences in the input. Here, the generated multi-sentence text is directly evaluated against the multi-sentence annotation in the crowd-soured data.

Single-sentence decoding: For single sentence generation, we first select the value of z_i which maximizes the prior $P(z_i|X;w)$. Thereafter, we simply use greedy decoding to generate a sentence conditioned on the chosen z_i and the input image pair.

Multi-sentence decoding: Here, we first select a set of clusters to include in the output description, and then generate a single sentence for each cluster using greedy decoding. Since typically there are more clusters than sentences, we condition on the ground truth number of sentences and choose the corresponding number of clusters. We rank clusters by decreasing likelihood under the alignment prior and then choose the top T.

2.4 Experiments

Model	Bleu 1/2/3/4	Meteor	Cider	Rouge-L	Perplexity
NN	0.226 0.111 0.057 0.026	0.102	0.120	0.201	-
CAPT	0.304 0.194 0.126 0.073	0.105	0.263	0.256	16.78
CAPT-MASKED	0.301 0.200 0.131 0.078	0.108	0.285	0.271	15.12
DDLA-UNIFORM	0.285 0.175 0.108 0.064	0.106	0.250	0.247	9.96
DDLA	0.343 0.221 0.140 0.085	0.120	**0.328**	0.286	9.73

Table 2.4: **Single sentence decoding:** We report automatic evaluation scores for various models under single sentence generation setting. DDLA model fares better scores than various baseline methods for all the considered measures. Both the DDLA models get much better perplexities than baseline methods.

We split videos used to create the dataset into train, test, and validation in the ratio 80:10:10. This is done to ensure that all data points using images from the same video are entirely in one split. We report quantitative metrics like CIDEr (Vedantam et al., 2015a), BLEU (Papineni et al., 2002a), METEOR (Denkowski and Lavie, 2014), and ROUGE-L, as is often reported by works in image captioning. We report these measures for both sentence level setting and multi-sentence generation settings. Thereafter, we also discuss some qualitative examples. We implement our models in PyTorch (Paszke et al., 2017). We use mini-batches of size 8 and use Adam optimizer[1]. We use CIDEr scores on validation set as a criteria for early stopping.

Baseline models: We consider following baseline models: CAPT model considers soft attention over the input pair of images (This attention mechanism is similar to that used in prior image captioning works (Xu et al., 2015), except that we have two images instead of a single image input). We do not perform any masking in case of CAPT model, and simply ignore the cluster information. The model is trained to generate a single sentence. Thus, this model is similar to a typical captioning model but with soft attention over two images. CAPT-MASK model is similar to CAPT model except that it incorporates the masking mechanism defined earlier using the union of all the cluster masks in the corresponding image. We also consider a version of the CAPT model wherein the target prediction is the whole multi-sentence description – CAPT-MULTI – for this setting, we simply concatenate the sentences in any arbitrary order [2]. Additionally, we consider a nearest neighbor baseline (NN-MULTI), wherein we simply use the annotation of the closest matching training data point. We compute the closeness based on the extracted features of the image pair, and leverage sklearn's (Pedregosa et al., 2011b) Nearest-Neighbor module. For single sentence setting (NN), we randomly pick one of the sentences in the annotation.

We also consider a version of DDLA model with fixed uniform prior, and refer to this model as DDLA-UNIFORM . For single sentence generation, we sample z_j randomly from the uniform distribution and then perform decoding. For the multi-sentence generation setting, we employ simple heuristics

[1] Our data set can be obtained through https://github.com/harsh19/spot-the-diff

[2] Note that we do not provide CAPT-MULTI with ground truth number of sentences

Model	Bleu 1/2/3/4	Meteor	Cider	Rouge-L	LenRatio
NN-MULTI	0.223 0.109 0.056 0.026	0.087	0.105	0.181	1.035
CAPT-MULTI	0.262 0.146 0.081 0.045	0.094	0.235	0.174	1.042
DDLA-UNIFORM	0.243 0.143 0.085 0.051	0.094	0.217	0.213	0.778
DDLA	0.289 0.173 0.103 0.062	0.108	**0.297**	0.260	0.811

Table 2.5: **Multi-sentence decoding** We report automatic evaluation scores for various models under multi-sentence generation setting. DDLA model achieves better scores compared to the baseline methods. Note that these scores are not directly comparable with single sentence generation setting. **LenRatio** is the ratio of the average number of tokens in the prediction to the average number of tokens in the ground truth for the test set.

HUMAN: A white truck has appeared in the after image. A person is now walking on the footpath.
DDLA (multi-sentence): A white truck appeared on the road. There is a person walking in the after image
CAPT-multi (multi-sentence): There is a car. There is a person walking in the parking lot.

HUMAN: There are more people in the group.
DDLA (multi-sentence): There are more people in the after image
CAPT-multi (multi-sentence): The people in the right image.

Figure 2.6: Predictions from various methods for two input image pairs.

to order the clusters at test time. One such heuristic we consider is to order the clusters as per the decreasing area of the bounding box (smallest rectangular area enclosing the cluster).

Results: We report various automated metrics for the different methods under single sentence generation and multi-sentence generation in Tables 2.4 and 2.5 respectively. For the single sentence generation setting, we observe that the DDLA model outperforms various baselines as per most of the scores on the test data split. DDLA-UNIFORM method performs similar to the CAPT baseline methods. For the multi-sentence generation, the DDLA model again outperforms other methods. This means that having a learned prior is useful in our proposed method. Figure 2.6 shows an example data point with predicted outputs by different methods.

2.4.1 Discussion and Analysis

Qualitative Analysis of Outputs We perform a qualitative analysis on the outputs to understand the drawbacks in the current methods. One apparent limitation of the current methods is the failure to explicitly model the movement of same object in the two images (Figure 2.7) – prior works on object tracking can be useful here. Sometimes the models get certain attributes of the objects wrong. e.g. 'blue car' instead of 'red car'. Some output predictions state an object to have 'appeared' instead of 'disappeared' and vice versa.

Do models learn alignment between sentence and difference clusters? We performed a study on 50 image pairs by having two humans manually annotate gold alignments between sentences and difference clusters. We then computed alignment precision for the model's predicted alignments. To obtain model's predicted alignment for a given sentence S_i, we compute $argmax_k P(z_i = k|X)P(S_i|z_i = k, X)$. Our proposed model achieved a precision of 54.6%, an improvement over random chance at 27.4%.

Clustering for pre-processing Our generation algorithm assumed one sentence uses only one cluster and as such we tune the hyper-parameters of clustering method to get large clusters so that typically

DDLA: The blue truck is gone.

Figure 2.7: Some drawbacks with the current models: One apparent drawback with the single cluster selection is that it misses opportunity to identify an object which has moved significantly- considering it as appeared or disappeared as the case may be. In this example, the blue truck moved, but the DDLA model predicts that the truck is no longer there.

a cluster will entirely contain a reported difference. On inspecting randomly selected data points, we observe that in some cases too large clusters are marked by the clustering procedure. One way to mitigate this is to tune clustering parameters to get smaller clusters and update the generation part to use a subset of clusters. As mentioned earlier, we consider clustering as a means to achieve object level pre-processing. One possible future direction is to leverage pre-trained object detection models to detect cars, trucks, people, etc. and make these predictions readily available to the generation model.

Multi-sentence Training and Decoding As mentioned previously, we query the models for a desired number of 'sentences'. In future works we would like to relax this assumption and design models which can predict the number of sentences as well. Additionally, our proposed model doesn't not explicitly ensure consistency in the latent variables for different sentences of a given data point i.e the model does not make explicit use of the fact that sentences report non-overlapping visual differences. Enforcing this knowledge while retaining the feasibility of training is a potential future direction of work.

2.5 Related Work

Modeling pragmatics: The dataset presents an opportunity to test methods which can model pragmatics and reason about semantic, spatial and visual similarity to generate a textual description of what has changed from one image to another. Some prior work in this direction (Andreas and Klein, 2016; Vedantam et al., 2017) contrastively describe a target scene in presence of a distractor. In another related task – referring expression comprehension (Kazemzadeh et al., 2014; Mao et al., 2016; Hu et al., 2017) – the model has to identify which object in the image is being referred to by the given sentence. However, our proposed task comes with a pragmatic goal related to summarization: the goal is to identify and describe *all* the differences. Since the goal is well defined, it may be used to constrain models that attempt to learn how humans describe visual difference.

Natural language generation: Natural language generation (NLG) has a rich history of previous work, including, for example, recent works on biography generation (Lebret et al., 2016), weather report generation (Mei et al., 2016), and recipe generation (Kiddon et al., 2016a). Our task can viewed as a potential benchmark for coherent multi-sentence text generation since it involves assembling multiple sentences to succinctly cover a set of differences.

Visual grounding: Our dataset may also provide a useful benchmark for training unsupervised and semi-supervised models that learn to align vision and language. Plummer et al. (2015) collected annotation for phrase-region alignment in an image captioning dataset, and follow up work has attempted to predict these alignments (Wang et al., 2016; Plummer et al., 2017; Rohrbach et al., 2016). Our proposed dataset poses a related alignment problem: attempting to align sentences or phrases to visual differences. However, since differences are contextual and depend on visual comparison, our new task may represent a more challenging scenario as modeling techniques advance.

Image change detection: There are some works on land use pattern change detection ((Radke et al., 2005)). These works are related since they try to screen out noise and mark the regions of change between two images of same area at different time stamps. (Bruzzone and Prieto, 2000) propose an unsupervised change detection algorithms aim to discriminate between changed and unchanged pixels for multi-temporal remote sensing images. (Zanetti and Bruzzone, 2016) propose a method that allows unchanged class to be more complex rather than having a single unchanged class. Though image diff detection is part of our pipeline, our end task is to generate natural language descriptors. Moreover, we observe that simple clustering seems to work well for our dataset.

Other relevant works: (Maji, 2012) aim to construct a lexicon of parts and attributes by formulating an annotation task where annotators are asked to describe differences between two images. Some other related works model phrases describing change in color (Winn and Muresan, 2018), and code commit message summarizing changes in code-base from one commit to another (Jiang et al., 2017). There exist some prior works on fine grained image classification and captioning (Wah et al., 2014; Nilsback and Zisserman, 2006; Khosla et al., 2011). The premise of such works is that it is difficult for machine to find discriminative features between similar objects e.g. birds of different species. Such works are relevant for us as the type of data we deal with are usually of same object or scene taken at a different time or conditions.

2.6 Application to Chess Commentary Generation

We discuss application of the proposed technique in chess game commentary generation, where we model a game move as being represented by a pair of successive game states. Our approach utilizes a set of fixed programs on the game states to extract successively higher levels of abstractions : (1) which pieces has moved, (2) which other pieces are under threat, and (3) overall game is in which player's favor. The proposed approach leads to better overall performance compared to straightforward sequence to sequence baselines.

Automated game commentary generation can be a useful learning aid. Novices and experts alike can learn more about the game by hearing explanations of the motivations behind moves, or their quality. In fact, on sites for game aficionados, these commentaries are standard features, speaking to their interestingness and utility as complements to concrete descriptions of the game boards themselves. Game commentary generation poses a number of interesting challenges for existing approaches to language generation. First, modeling human commentary is challenging because human commentators rely both on their prior knowledge of game rules as well as their knowledge of effective strategy when interpreting and referring to the game state. Secondly, there are multiple aspects of the game state that can be talked about for a given move — the commentator's choice depends on the pragmatic context of the game. For example, for the move shown in Figure 6.1, one can comment simply that the pawn was moved, or one may comment on how the check was blocked by that move. Both descriptions are true, but the latter is most salient given the player's goal. However, sometimes, none of the aspects may stand out as being most salient, and the most salient aspect may even change from commentator to commentator. Moreover, a human commentator may introduce variations in the aspects he or she chooses to talk about, in order to reduce monotony in the commentary.

We introduce our new large-scale *Chess Commentary* dataset, share some statistics about the data, and discuss the variety in type of commentaries. The data is collected from the online chess discussion forum `gameknot.com`, which features multiple games self-annotated with move-by-move commentary. Prior work has explored game commentary generation. (Liao and Chang, 1990; Sadikov et al., 2006) have explored chess commentary generation, but for lack of large-scale training data their methods have been mainly rule-based. (Kameko et al., 2015) have explored commentary generation for the game of Shogi, proposing a two-step process where salient terms are generated from the game state and then composed in a language model. In contrast, given the larger amount of training data available to us, our proposed model uses an end-to-end trainable neural architecture to predict commentaries given the game state. Our model conditions on semantic and pragmatic information about the current state

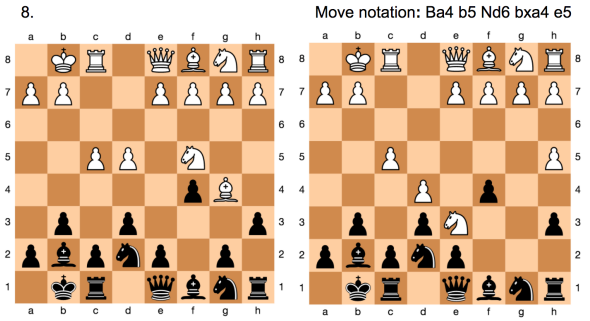


Figure 2.8: A multi-move, single commentary example from our data. Here, the sequence of moves Ba4 → b5 → Nd6 → bxa4 → e5 is commented upon.

Statistic	Value
Total Games	11,578
Total Moves	298,008
Average no. of recorded steps in a game	25.73
Frequent Word Types[3]	39,424
Rare Word Types	167,321
Word Tokens	6,125,921
Unigram Entropy	6.88
Average Comment Length (in #words)	20.55
Long Comments (#words > 5)	230745 (77%)

Table 2.6: Dataset and Vocabulary Statistics

and explicitly learns to compose, conjoin, and select these features in a recurrent decoder module. We perform an experimental evaluation comparing against baselines and variants of our model that ablate various aspects of our proposed architecture. Outputs on the 'Move Description' subset of data from our final model were judged by humans to be as good as human written ground truth commentaries on measures of fluency and correctness. The dataset consists of 298K aligned game move/commentary pairs. Some commentaries are written for a sequence of few moves (Figure 2.8) while others correspond to a single move. For the purpose of initial analysis and modeling, we limit ourselves to only those data points where commentary text corresponds to a single move. Additionally, we split the multi-sentence commentary texts to create multiple data points with the same chess board and move inputs.

What are commentaries about?: Commentary-type Ontology We observe that there is a large variety in the commentary texts. To analyze this variety, we consider labelling the commentary texts in the data with a predefined set of categories. The choice of these categories is made based on a manual inspection of a sub-sample of data. We consider the following set of commentary categories (Also shown in Table 2.7):

- **Direct move description (MoveDesc[4]):** Explicitly or implicitly describe the current move.
- **Quality of move (Quality[5]):** Describe the quality of the current move.
- **Comparative:** Compare multiple possible moves.

[4] MoveDesc & 'Move Description' used interchangeably

[5] Quality and 'Move Quality' used interchangeably

Category	Example	% in data	Val acc.
Direct Move Description	An attack on the queen	31.4%	71%
Move Quality	A rook blunder.	8.0%	90%
Comparative	At this stage I figured I better move my knight.	3.7%	77.7%
Planning / Rationale	Trying to force a way to eliminate d5 and prevent Bb5.	31.2%	65%
Contextual Game Info	Somehow, the game I should have lost turned around in my favor .	12.6%	87%
General Comment	Protect Calvin , Hobbs	29.9%	78%

Table 2.7: Commentary texts have a large variety making the problem of content selection an important challenge in our dataset. We classify the commentaries into 6 different categories using a classifier trained on some hand-labelled data, a fraction of which is kept for validation. % data refers to the percentage of commentary sentences in the tagged data belonging to the respective category.

- **Move Rationale or Planning (Planning):** Describe the rationale for the current move, in terms of the future gameplay, advantage over other potential moves etc.
- **Contextual game information:** Describe not the current move alone, but the overall game state – such as possibility of win/loss, overall aggression/defence, etc.
- **General information:** General idioms & advice about chess, information about players/tournament, emotional remarks, retorts, etc.

The examples in Table 2.7 illustrate these classes. Note that the commentary texts are not necessarily limited to one tag, though that is true for most of the data. A total of 1K comments are annotated by two annotators. A SVM classifier (Pedregosa et al., 2011a) is trained for each comment class, considering the annotation as ground truth and using word unigrams as features. This classifier is then used to predict tags for the train, validation and test sets. For "Comparative" category, we found that a classifier with manually defined rules such as presence of word "better" performs better than the classifier, perhaps due to the paucity of data, and thus we use this instead . As can be observed in Table 2.7, the classifiers used are able to generalize well on the held out dataset

The key contributions of this study can be summarized as follows:

(1) We propose and collect a dataset of more than 298K chess move/commentary pairs across $\approx$ 11K chess games. To the best of our knowledge, this is the first such dataset of this scale for a game commentary generation task. We provide an analysis of the dataset and highlight the large variety in commentary texts by categorizing them into six different aspects of the game that they discuss.

(2) We propose a chess commentary generation model to effectively use semantic and pragmatic information about chess games and explicitly model conjunctions of features.

(3) We perform an experimental evaluation comparing with baselines and variants of our model that ablate our proposed semantic and pragmatic features. Outputs from our final model were judged as good as human written ground truth commentaries on measures of fluency and correctness.

(4) Our results demonstrate the effectiveness of our proposed method to deal with content selection issue.

3 Fine-grained Reasoning for Knowledge-Base Grounded Text Generation

In this chapter, I discuss applying the notion of exposing intermediate operations to discover latent reasoning patterns in knowledge base grounded text generation. Consider a typical grounded NLG setup wherein we have to generate text output for some context. Prior work has demonstrated how one can leverage external knowledge bases to enrich models to leverage external information to generate more favorable outputs. A popular contemporary approach is to use large dump of relevant information from the knowledge base, instead of focusing on discovering and exposing latent reasoning patterns which may be present. Often such reasoning involves fine grained selection of sentences from knowledge base, as well as carefully combining the relevant pieces.

In rest of the chapter, I will describe and discuss models which uncover latent reasoning structures via knowledge bases to compose desired final outputs. First I will discuss a model for persona-grounded dialog (Majumder and Jhamtani et al. (2020)) which (1) performs an expansion operation to discover common-sense implications of given persona (2) uses a discrete random variable to select which particular aspect of the expanded persona entails a given dialog response. This is in contrast to prior work which encodes the entire persona instead of fine-grained selection. We observe that the proposed models lead to improved generalization compared to baselines, and utilizes the giver persona in a more interpretable and controllable manner. I will conclude the chapter with a brief discussion on applying related ideas on uncovering abstract reasoning chains as explanations for open domain multi-hop reasoning questions (Jhamtani and Clark, 2020). Compared to prior work which uses a dump of relevant sentences, we focus on selection and combinations of sentences which can as reasoning chains.

3.1 Introduction

Persona-grounded dialog generation is a 'chit-chat' dialog setup where a dialog agent is expected to communicate based on a given profile (Zhang et al., 2018a). Many recent works have focused on a popular benchmark dataset for this task: PERSONA-CHAT' (Zhang et al., 2018a) that provides personas as a set of sentences along with each dialog (example in 3.1). However, a careful analysis of state-of-the-art (SOTA) models reveals that they often struggle to respond to contexts that do not closely match given persona sentences, even when the implications might be obvious to a human.

For example, in 3.1, the user asks an *indirect* question to the bot related to one of its persona sentences: *I am an animal activist*. SOTA1, which concatenates all persona sentences with dialog history and finetunes a pre-trained generative model (e.g. GPT2) (Wolf et al., 2019), fails to infer implied commonsense from the dialog context and conditions on an incorrect persona. SOTA2, which separately selects a persona sentence given the dialog history (Lian et al., 2019) manages to choose the correct persona but merely copies it as the final response. Neither approach is in general capable of responding to context that goes beyond what is explicitly mentioned in the available persona sentences, which limits consistent and interesting conversation. The goal of our model is to understand that being 'an animal activist' may imply that the person wants 'to make a difference' via their activity towards animals and synthesizes a context-consistent and engaging response.

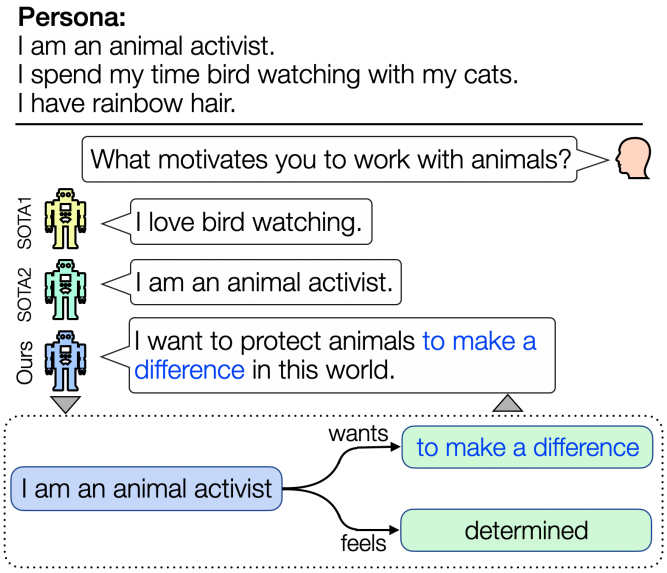


Figure 3.1: State-of-the-art models struggle to respond a user's query, where generating an engaging response depends on commonsense reasoning.

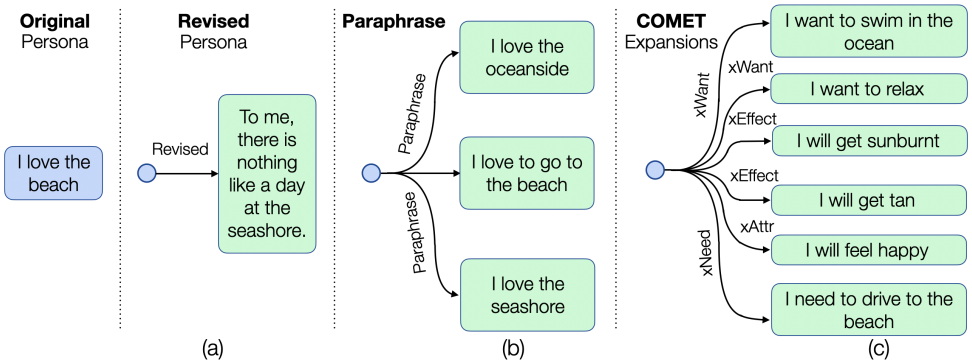


Figure 3.2: Expansions of an original persona via (a) human rewrite (Zhang et al., 2018a), (b) paraphrase, and (c) COMET.

In this paper, we focus on making persona-grounded chatbots more consistent with personas and implicit dialog context. We present a framework to expand available persona sentences to their commonsense implications by using an existing commonsense knowledge base or paraphrasing resources (see 3.2). We endow our dialog model with these expansions directly rather than requiring the model to learn them from scratch for being context-consistent. We find that expansions derived from a commonsense knowledge base are more useful to provide engaging contextual information compared to other expansion sources.

We further propose a **Com**mon Sense and **P**ersona **A**ligned **C**hatbot[1] (**COMPAC**) which models choices over the *expanded* persona set via a discrete latent random variable (See 3.3) as *fine-grained* persona grounding. Even though it is tractable to marginalize over all expansions, that would require a forward pass through the dialog generator for each outcome which is prohibitively slow during training. Instead, to accommodate hundreds of persona expansions, we train the model by optimizing a lower bound on the log-likelihood. We use amortized variational inference by approximating the true posterior using an inference network that eventually provides useful inductive bias. Particularly, we show that our Bayesian formulation for the fine-grained persona grounding was essential as simply providing expanded knowledge does not help the model generate better responses.

[1] Code is available at – https://github.com/majumderb/compac.

We also outperform competitive baselines in all dialog quality metrics as well as human evaluations which find COMPAC to be engaging and coherent. We demonstrate that COMPAC learns to be consistent with the dialog context with accurate persona grounding especially in the presence of commonsense expansions. Finally, we show that our model can reflect a change in response generation when a grounding persona is modified, indicating the possibility of controllable generation.

We use a popular benchmark dataset: PERSONA-CHAT‘ (Zhang et al., 2018a) for our persona-grounded dialog generation task. It contains 10,907 dialogs between pairs of speakers where each speaker follows their own persona; 968 dialogs are used for validation and 1,000 for testing. Each speaker is described by 3-5 persona sentences. (e.g. ‘I love the beach’, ‘My mother is a medical doctor’). Out of 1,155 total unique personas, 100 are used for validation and 100 for testing.

The task of persona-grounded dialog generation is: given a dialog history H and grounding persona sentences S, we must predict the next utterance x (Summary of notations in 3.1). Hence a dialog model should maximize the likelihood $p(x|H,S)$. From the PERSONA-CHAT‘ dataset, we use 131,438 utterances for training the dialog model, 15,602 for validation, and 15,024 for testing.

3.2 Persona Expansion

Persona sentences used in persona-grounded dialogs are instances of world events that often imply real-world consequences or richer information. For example, ‘I love surfing’ naturally implies that the person might be ‘adventurous’ or ‘loves the outdoors’. Similarly, it also means that the person wants ‘to go to the beach’ regularly. Inferring these *expansions* from the original fact is non-trivial without additional commonsense knowledge.

Zhang et al. (2018a) found evidence that having human written interpretations of a persona sentence via rephrasing often helps in providing novel information in persona grounding. While obtaining such expansions by manual rewriting is expensive, here we explore two automatic ways to generate them at scale and separately evaluate them on the downstream dialog modeling task.

3.2.1 COMET

COMET (Bosselut et al., 2019) is a framework that generates rich and diverse commonsense expansions of a given world event. It is a finetuned version of a pre-trained GPT2 (Radford, 2018) model on a pre-existing commonsense knowledge graph such as ATOMIC (Sap et al., 2019) that can generate novel nodes (events) and edges (relations), as seen in 3.2c. Specifically, ATOMIC provides tuples that belong to nine relation types spanning over cause-effect interrelations between events: `oEffect`, `oReact`, `oWant`, `xAttr`, `xEffect`, `xIntent`, `xNeed`, `xReact`, and `xWant`—where a prefix ‘`x`’ indicates an effect or cause on the person and ‘`o`’ denotes the same on others. While we tried COMET finetuned on an alternative commonsense knowledge base (e.g.) ConceptNet, not all of the expansions were appropriate to describe a persona, mainly because we observe that persona sentences are *event*-like (‘I love to go to the beach’) as opposed to *concepts* such as ‘beach’. For more details on COMET and ATOMIC we refer the reader to (Bosselut et al., 2019) and (Sap et al., 2019) respectively.

We use the COMET framework to generate expansions for each persona sentence along the nine relation types that ATOMIC provides. We obtain different samples while decoding via beam search from COMET for more diverse and unique expansions, as shown in 3.2c. We preprocess these expansions to add suitable prefixes to make them similar to the original persona. For example, expansions relating to `xWant` and `xAttr` are prefixed with ‘I want’ and ‘I am’ respectively. For each persona sentence, we generate 5 expansions per relation, i.e., in total we will obtain $5 \times 9 = 45$ expansions per persona sentence.

3.2.2 Paraphrasing

To explore alternative sources for generating commonsense expansions beyond COMET, we consider paraphrasing persona sentences. Paraphrases of a sentence convey almost the same meaning to a listener as the original. Often paraphrases use synonymous phrases or manipulate word-syntax of the original sentence, which implicitly involves both context comprehension and world knowledge (Zeng et al., 2019). We obtain these in two ways:

Paraphrase Network To generate paraphrases at scale, we use an off-the-shelf paraphrasing system based on back-translation (Xie et al., 2019; Federmann et al., 2019) with pre-trained language translation models. We make use of `En-Fr` and `Fr-En` pre-trained translation models as the components for back-translation.[2] While we tried other language pairs, the `En-Fr` pair proved the most satisfactory based on qualitative analysis on 500 samples. We generate 5 paraphrases per persona sentence, which readily provides more lexical and syntactic variants as shown in 3.2b.

Manual Paraphrasing To compare with other expansions, we reuse manually written revised versions of persona sentences provided with PERSONA-CHAT‘ (Zhang et al., 2018a) though these are limited to only one paraphrase per sentence. We call them **revised** for short (see 3.2a).

3.3 Common sense and Persona Aligned Chatbot (COMPAC)

To infuse commonsense context in persona-grounded dialog generation, we imbue our dialog model with the expanded persona set instead of only original personas S. But these persona expansions lead to hundreds of new sentences as opposed to only a few given persona sentences which makes it infeasible to encode using a single transformer, as was done in prior works (Wolf et al., 2019). Additionally, encoding all persona sentences as a single text input leads to a lack of interpretability i.e., it is not clear which persona sentence was used by the model in generating a particular response.

Instead, we propose **COMPAC**: **Com**mon Sense and **P**ersona **A**ligned **C**hatbot that allows us to make a *fine-grained* choice of a persona sentence to generate the target response. Let C denote a list of expended personas, derived from S (including S itself). We further add a null persona $\varnothing$ in C considering that some utterances can purely condition on the dialog context. We are interested in modeling the conditional $p(x|H,C) = p(z|H,C)p(x|z,H,C)$ where $z \in \{1,2,\ldots,|C|\}$ is a latent discrete random variable, unobserved in the data. Given the dialog history H, first we sample a particular persona sentence C_z from a *prior network* $p_\theta(z|H)$ (see 3.3). Next, as depicted in 3.3, the dialog response x is sampled from a *generator network* $p_\phi(x|H,C_z)$ by conditioning on the history H and chosen persona sentence C_z.

In the generative model described above, the latent variable z is a discrete random variable which points to a single persona sentence. This decision (of conditioning on a single persona sentence) was based on the observation that most dialog responses in the datasets under consideration are relevant to only one persona sentence. It is possible to allow for multiple persona sentences by defining z to pick a subset of $|C|$ persona sentences instead of picking a single sentence. We leave this as a possible future extension.

3.3.1 Persona Choice Prior

The dialog history H can hold cues regarding which persona sentence might be applicable given the context. For example, in 3.3 the historical context suggests that ‘following fashion trends’ can be a consequence of ‘being fashionable’.

We encode both the dialog history H and persona sentence C_k by averaging RoBERTa subword embeddings (Liu et al., 2019a) as $e(H)$ and $e(C_k)$. We use an implementation from HuggingFace for

[2] https://github.com/google-research/uda

S	Set of original persona sentences
C	Set of expanded persona sentences (includes S and a null persona $\varnothing$)
H	Dialog history with alternative turns from each speaker
x	Target utterance
z	Discrete latent random variable $\in \{1, 2, \ldots, \lvert C \rvert\}$
e	Mean of RoBERTa subword embeddings as an encoder
t_k	Expansion type for k-th expansion
f_i	i-th feature function for prior network; $i \in \{1, 2, 3\}$
θ	Parameters for prior network $p_\theta(z \lvert H, C)$
ϕ	Parameters for generator network $p_\phi(x \lvert H, C_z)$
α	Parameters for inference network $p_\alpha(z \lvert x, H, C)$

Table 3.1: Summary of notation used in the paper

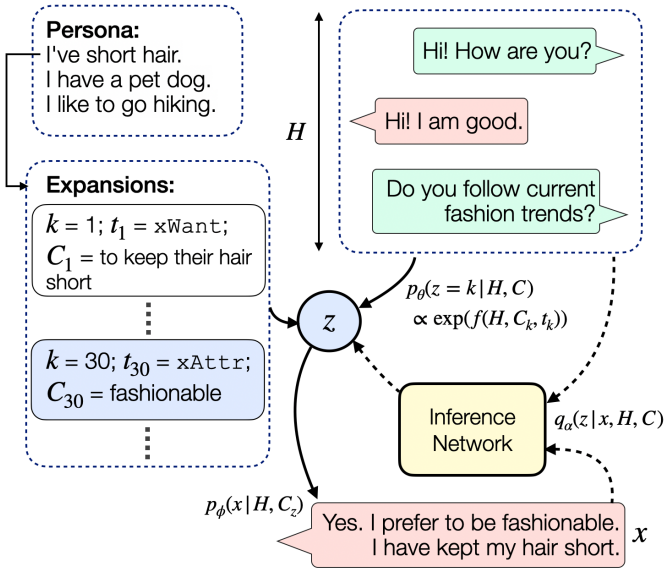


Figure 3.3: COMPAC samples a persona sentence from the prior and generates the response conditioned on the dialog context and sampled persona. The inference network is used only during training.

RoBERTa[3] with `roberta-base` as the pretrained model. Then we parameterize the prior $p_\theta(z|H,C)$ as a log-linear model with the following features:

Dialog historyWe obtain $f_1(H,C_k)$: a scalar feature using a bilinear product $\langle e(H), e(C_k)\rangle$ to align the persona sentences with the dialog history.

Expansion typesEach k-th persona expansion corresponds to an expansion type t_k. In the case of COMET, these types are the nine commonsense relations provided by ATOMIC (see 3.2.1). For paraphrased expansions, we annotate each as type `paraphrase` and the original persona sentences as `original`. We consider two additional features with expansion types: (a) $f_2(t_k)$ that represents a global preference over the relation type embedded via a type embedding layer; and (b) $f_3(t_k,H)$ that appends the expansion type embedding with dialog history encoding $e(H)$, followed by a linear layer to obtain a real-valued score for history-specific preference over the expansion type.

The dimension of the expansion type embedding was set to 5. Finally, the prior model can be represented concisely as $p_\theta(z=k|H,C) \propto \exp(f(H,C_k,t_k))$, where $f(H,C_k,t_k)$ is the sum $\lambda_1 * f_1(H,C_k) + \lambda_2 * f_2(t_k) + \lambda_3 * f_3(t_k,H)$ with λ_i's are trainable parameters.

[3] https://huggingface.co/transformers/model_doc/roberta.html

3.3.2 Generator Network

Following prior work (Wolf et al., 2019), we use pre-trained GPT2 (Radford, 2018) (Transformer with 12 layers, 768 hidden size, 12 heads— `gpt2-small`[4]) to generate dialog responses given the dialog history H, with the selected persona sentence C_z prepended to it. In the case of C_z being the null persona, an empty string is prepended. We further append the target response x to the combined context $(C_z; H)$, and feed the sequence to GPT2, after tokenization. To distinguish between persona tokens, history tokens, and target response tokens, we use segment indicators—{`Persona`, `Speaker1`, `Speaker2`}—for which corresponding embeddings are learned via a separate segment embedding layer in the model. We add the segment embedding to the corresponding token embedding in the model input layer. To obtain the conditional likelihood $p_\phi(x|H, C_z)$, we only consider the target tokens for cross-entropy loss calculation.

Wolf et al. (2019) also leveraged incorrect responses given a dialog history from PERSONA-CHAT' as negative samples in an auxiliary loss to encourage the correct candidate to obtain the highest likelihood compared to the incorrect ones. However, we did not find any improvement using this loss in COMPAC.

3.3.3 Learning and Inference

Our training data $\mathscr{D}$ consists of instances of dialog history H and ground truth dialog responses x. We train our model parameters θ and ϕ to maximize the likelihood of the target dialog response x given the dialog history: $\log p(x|H, C; \theta, \phi)$ totalled over $\mathscr{D}$. Since the discrete random variable z is unobserved in the training data, we must marginalize over z to compute the desired likelihood $p(x|H; \theta, \phi)$:

$$\log p(x|H; \theta, \phi) = \log \mathbb{E}_{z \sim p_\theta(z|H)}[p_\phi(x|z, H)];$$

where we drop C from the conditionals for simplicity.

Inference Network Note that the number of persona expansions is typically in the range 150-250, and thus it is computationally expensive to marginalize over the entire selection space of z during training. We instead optimize a variational lower bound (ELBO) of $\log p(x|H; \theta, \phi)$ given as

$$\begin{aligned} &\mathbb{E}_{z \sim q_\alpha(z|H)}[\log p_\phi(x|z, H)] \\ &- KL(q_\alpha(z|x, H)||p_\theta(z|H)), \end{aligned}$$

where we use the inference network $q_\alpha(z|x, H)$ to compute the approximate posterior (Kingma and Welling, 2014a). In our initial experiments, we observe that using an inference network leads to better perplexity values than using samples from the prior.

The architecture of the inference network is similar to that of the prior network, a log-linear model. Along with the features related to dialog history and expansion types, we additionally include another scalar feature: a bilinear product $\langle x, C_k \rangle$ between the encoded persona and ground truth response x encoded with RoBERTa embeddings to align the persona choice according to the target utterance.

Optimization The parameters of the generator network (ϕ) and prior network (θ) can be trained directly via back-propagation. Since z is a discrete latent variable, we use REINFORCE (Williams, 1992) to train the inference network parameters α. However, the REINFORCE estimator often suffers from high variance. To reduce the variance, we found it useful to (1) use a moving average baseline (Zhao et al., 2011); and (2) regularize the prior network by penalizing the entropy of the output categorical distribution. To avoid KL mode collapse, we use KL-annealing (Bowman et al., 2016) where we linearly increase the weight of the KL term beginning from 0 to 1 as training progresses.

[4] https://github.com/huggingface/transfer-learning-conv-ai

System	PPL	BLEU-1	BLEU-2	D-1	D-2
Original					
Per-CVAE (Song et al., 2019b)	48.37	0.19	0.11	0.03	0.21
LIC + KS (Lian et al., 2019)	30.50	0.18	0.07	0.07	0.24
GPT2 (Wolf et al., 2019)	21.46	1.42	0.78	0.05	0.11
COMPAC-original	19.56	3.24	1.31	0.15	0.25
Paraphrased					
GPT2-revised	21.01	1.54	0.97	0.13	0.25
GPT2-paraphrase	21.57	1.61	0.86	0.16	0.35
COMPAC-revised	18.12	3.52	0.99	0.48	0.65
COMPAC-paraphrase	17.09	3.83	**1.87**	0.56	0.85
COMET					
GPT2-COMET	21.12	1.62	0.81	0.21	0.39
COMPAC	**16.21**	**4.12**	1.82	**0.87**	**1.07**

Table 3.2: Dialog quality metrics on the PERSONA-CHAT' test set. PPL=Perplexity, D-1/2=% of distinct uni- and bi-grams.

Persona: I enjoy listening to classical music. I'm a Hindu. My favorite color is red.
User: Hi, recently I have got interests in religion.
GPT2 (Wolf et al., 2019)**:** Hi! How are you?
COMPAC-original: I'm a Hindu.
COMPAC-revised: Hi! I am a Hindu too.
COMPAC-paraphrase: That's great. I am religious.
COMPAC: That's great. I go to temple regularly and learn about Hinduism.

Table 3.3: Sample generations by different models. More examples are in Appendix §C.

Decoding At decoding time, we first sample k from the prior $p_\theta(z|H,C)$, and then C_k is fed to the generator network. Following previous work (Wolf et al., 2019), we use nucleus sampling (Holtzman et al., 2020a) (with $p = 0.95$) to decode the final response from the probabilities produced by the generator. We also found that high-temperature sampling from the prior often leads to more diverse generation.

3.4 Experiments

We conduct our experiments based on the following desiderata: (1) Do persona expansions help to generate high quality and diverse responses? (2) Does COMPAC achieve accurate persona grounding given a dialog context? (3) Does COMPAC enable persona-consistent and controllable generation? Hyperparameter details are in Appendix §A.

COMPAC vs.	GPT2		LIC + KS		GPT2-COMET		COMPAC-og		COMPAC-par		Gold	
Metric ↓	win	loss	win	loss	win	loss	win	loss	win	loss	win	loss
Fluency	**81.2***	5.1	**83.2***	6.7	**90.5***	2.3	**68.0**	26.0	**65.0**	19.4	40.1	**42.5**
Engagement	**90.5***	3.3	**87.4**	5.9	**97.6***	0.5	**86.5***	10.5	**81.5***	10.5	**62.1***	30.5
Relevance	**78.2***	4.8	**78.0***	7.7	**93.2***	1.8	**65.5***	18.5	**62.1**	15.6	32.8	**54.6***

Table 3.4: Pairwise comparison between responses generated by COMPAC vs. responses generated by other baselines (og: original, par: paraphrase) as well as the Gold response. All numbers are in percentages with **bold** indicates the highest. Ties are not shown. Entries with * denote significance with $p < 0.05$ from bootstrap tests on 1000 subsets of size 50.

3.4.1 Baselines

To demonstrate the efficacy of COMPAC, we compare it with three competitive baselines on the PERSONA-CHAT‘ dataset:

1. **Per-CVAE:** A CVAE model that exploits persona sentences for diverse generation with an external memory (Song et al., 2019b)

2. **LIC + KS:** The best performing transformer model (Lost in Conversation i.e., LIC) in terms of human evaluation in the ConvAI2 NeurIPS competition (Dinan et al., 2019a) combined with a knowledge-selection (KS) mechanism Lian et al. (2019) to achieve state-of-the-art results on PERSONA-CHAT‘;

3. **GPT2:** Finetuned GPT2 on PERSONA-CHAT‘ just by concatenating all persona sentences along with dialog history (Wolf et al., 2019) to obtain the best automatic metric in the ConvAI2 competition.

A minimal version of COMPAC is also considered, **COMPAC-original**, which only uses the original persona, for a direct comparison with other model architectures that only use the original persona. Furthermore, to justify the choice of fine-grained persona grounding for an effective utilization of persona expansions, we also consider baseline versions of GPT2 trained with each of the expansion strategies: **GPT2-revised**, **GPT2-paraphase**, and **GPT2-COMET**. To show that COMPAC can work with persona expansions derived from various sources, we compare with versions of COMPAC trained with paraphrase-based expansions: **COMPAC-revised** and **COMPAC-paraphrase**. By default, COMPAC indicates it is trained with COMET expansions.

3.4.2 Comparison of Dialog Quality

We measure perplexity for language modeling performance, and BLEU-1 (Papineni et al., 2002b) and BLEU-2 (Vedantam et al., 2015b) scores between generated and gold utterances to measure the fidelity of the generated responses. Given our goal of generating engaging responses with novel information, we deem it important to consider the diversity in the generated responses which we measure using D-1 and D-2 (percentage of distinct uni- and bi-grams respectively) (Li et al., 2016a).

3.2 shows that COMPAC outperforms three competitive baselines when trained on the original persona in all quality metrics indicating the efficacy of our architecture. Moreover, when combined with persona expansions, we observe a modest 3-8 point decrease in perplexity and a large improvement in both BLEU and diversity scores which confirms that COMPAC successfully leverages the persona expansions to improve dialog quality. COMPAC trained with COMET expansions achieves the best performance both in terms of fidelity and diversity which shows that COMET expansions help the model to respond to implicit context with commonsense and to explore novel information. But with revised personas, we

System	Persona Entailment		Human
	Prior	Inference Network	eval.
Original			
COMPAC-original	25.5	79.3	–
Paraphrased			
COMPAC-revised	20.6	78.9	40.6
COMPAC-paraphrase	27.8	87.3	67.8
COMET			
COMPAC	**37.9**	**96.4**	**87.3**

Table 3.5: Assessment of persona grounding with and without inference network using the DNLI entailment set. Human evaluation (eval.) was conducted to measure the relevance when an expanded persona is chosen–all entries are statistically significant.

find that both COMPAC and GPT2 provide marginal performance gains, mirroring the observation from (Zhang et al., 2018a). Finally we observe gradual degradation in performance when we trivially finetune GPT2 with paraphrase and COMET expansions. Note that GPT-2 could have implicitly learned to focus on a single persona attribute. However, the proposed COMPAC model performs better suggesting that fine-grained persona grounding acts as a useful inductive bias in effectively utilizing larger expansion sets.

3.4.3 Human Evaluation for Dialog Generation

Automatic evaluation of dialog systems is still notoriously unreliable (Liu et al., 2016; Novikova et al., 2017a) and such systems should be evaluated by human users. Hence, we perform pairwise comparisons between responses generated our best system, COMPAC trained on COMET expansions, and responses generated by four strong baselines: GPT2, GPT2-COMET, COMPAC-original, COMPAC-paraphrase (the best COMPAC model with paraphrase expansions). We also consider the gold responses for comparison. We conduct a human evaluation with 100 test examples on three aspects critical for practical use: (1) **Fluency** measures whether the generated output is fluent (in English); (2) **Engagement** measures whether the generated response is engaging or interesting; and (3) **Relevance** measures whether the generated output is relevant with respect to the dialog history. More details of the evaluation are in Appendix §B.

3.4 shows that human annotators found responses generated by COMPAC trained with COMET expansions more engaging as compared to responses from all the baselines as well as the gold responses by statistically significant margins. This confirms our hypothesis that COMET expansions were helpful in adding novel content. Human judges also found that despite a significant drop in perplexity, COMPAC was not more fluent than COMPAC-original and COMPAC-paraphrase with statistical significance, indicating similar language modeling performance. We find the inter-annotator agreement, as measured by Cohen's kappa (Cohen, 1960), for fluency, engagement, and relevance were 0.62, 0.71, and 0.73 respectively.

3.4.4 Fine-grained Persona Grounding

Next we want to investigate the extent of COMPAC's ability to ground the response generation with a fine-grained persona choice as a probing experiment. Specifically, we want to measure whether our model can choose a coherent persona from the available persona sentences given the dialog context. Note that in persona-grounded chitchat, not all utterances are tied to a personas and could be purely based on dialog context. We find that 44% of the time the model selects the null persona ($\varnothing$) and conditions only on the dialog history. To assess the persona grounding for the remaining (56%) utterances, we perform (a) a persona entailment experiment, and (b) human evaluation.

System	Unigram Overlap			BERT
	Recall	Precision	F1	Score
Original				
LIC + KS (Lian et al., 2019)	10.4	34.2	15.3	–
COMPAC-original	14.9	39.1	21.6	57.2
Paraphrased				
COMPAC-revised	15.2	40.3	22.1	58.1
COMPAC-paraphrase	17.8	42.2	25.1	72.9
COMET				
COMPAC	**21.4**	**48.9**	**29.8**	**78.8**

Table 3.6: Conditional generation performance on the PERSONA-CHAT‘ test set to show the similarity between generated responses and grounding persona sentences. We omit GPT2-based models since they do not select a particular persona sentence for grounding.

Persona Entailment We adapt the Dialogue Natural Language Inference (DNLI) dataset (Welleck et al., 2019b) and collect persona-utterance pairs that belong to an *entailment* relation. This results in a subset of 4,613 utterances with associated ground truth persona sentences in our test set. Next, we obtain a persona sentence by performing argmax over the prior $p_\theta(z|H,C)$ as well as the inference network $q_\alpha(z|x,H,C)$ from our COMPAC models and calculate accuracy with the ground truth persona. For models that use expanded personas, we track the original persona from the retrieved expansion for accuracy calculation. 3.5 shows that COMPAC with COMET achieves the most accurate persona grounding suggesting that inference networks can approximate the true posterior better when a commonsense persona is available for grounding. In the case of the prior, a better entailment accuracy than random chance ($1/5$) confirms our choice of the history-conditioned prior network rather than a uniform prior.

Human Evaluation Since DNLI does not entail expanded personas, we conduct a human evaluation to judge the relevance of a chosen persona *expansion* sampled from the inference network. Specifically, we ask: *Is this knowledge relevant to the given dialog history?*—with options as ‘Yes’, ‘No’, and ‘Uncertain’—and with 100 examples (more in Appendix §B) for each COMPAC variant that uses expanded personas. The inter-annotator agreement, as measured by Cohen’s kappa was 0.76. Again, 3.5 shows that models with COMET expansions can choose the most relevant persona sentence which corroborates our claim in persona entailment experiments. On average, we noticed that COMPAC with COMET expansions prefers to choose expanded personas 87% of the time out of all non-null persona choices. This reduces to 62% in the case COMPAC-paraphrase. In contrast, COMPAC-revised tends to select an original persona over an expansion more often.

3.4.5 Controllable Generation

Controllable generation of persona-grounded dialog can help to generalize the dialog agent to newer persona details just by changing the grounding in the conditional generator. While controllable text generation with a desired attribute has gained interest recently (Dathathri et al., 2020; Kong et al., 2019), we investigate the possibility of controlling generation with a desired persona and measure the performance of the conditional generator. For this, we observe a set of knowledge overlap metrics—the unigram recall/precision/F1 scores–from Dinan et al. (2019b) and BERT score (Zhang et al., 2020) for semantic similarity between the generated responses and the persona retrieved. 3.6 shows that conditional generation is strongest when COMPAC is trained with COMET suggesting commonsense expansions are more appropriate to the dialog context in influencing the response generation.

Next, we create a diagnostic dataset of 100 examples where we manually edit the persona by changing an entity in a persona sentence or swapping the selected persona expansion with another relevant one (See examples in 3.7) to directly measure controllability in response generation. We observe that COMPAC can successfully reflect the entity-change in the generated response based on the change in the

Performance	Example
Presence of changed entity **86%**	**Changing the key entity** Before: My favorite color is red After: My favorite color is **green** Conversation: User: What is your favorite color? Bot: My favorite color is **green**
BERT score with unedited persona: 46.2 edited persona: **74.6**	**Swapping with another expansion** Before: I want to swim in the ocean After: I want to buy a beach **umbrella** Conversation: User: What do you do at beaches? Bot: I will buy an **umbrella** at the beach

Table 3.7: Controlled generation with edited persona

Verdict	Persona	Dialog History	Persona Choice (z)	Response
Uses Original Persona	1. I'm the youngest of five. 2. I work at the hospital as a nurse. 3. I've pink hair.	What do you do for work?	I work at the hospital as a nurse.	I am a nurse.
Uses Expanded Persona	1. I just want to have fun with my friends. 2. I don't drink or do drugs or anything. 3. I am 19 and I cannot wait to move out of my parents home.	Are you enjoying life?	I just want to have fun with my friends → wants to have a party	Not really. I want to have a party.
Uses Improper Persona	1. I make a million dollars a year. 2. I'm married and have three kids. 3. I'm a baseball player.	I find it hard to support my family working at a bar. What about you?	Null persona (∅)	I enjoy my life.

Table 3.8: Examples showing correct and incorrect persona choices in various dialog contexts by COMPAC model. It shows that COMPAC is capable of choosing a correct persona sentence (original or expanded) but sometimes the prior network fails to sample an appropriate one (third case).

persona grounding 86% of the time. For a swapped persona expansion, we also see a higher BERT score (74.6) between the edited persona and newly generated response as opposed to a lower score (46.2) with the unedited persona. Together with the qualitative examples in 3.7 this suggests that COMPAC supports controllable generation with contextually modified personas.

3.4.6 Qualitative Analysis

3.3 shows responses from different models for a sample dialog context. Qualitatively, we find that COMPAC with COMET expansions responds to the context with commonsense using novel content from a commonsense expansion (being Hindu → to learn about Hinduism), where other responses remain generic or incoherent. In 3.8, we illustrate responses generated by the COMPAC model along with the underlying persona choice sampled from the prior network. Cases show that COMPAC successfully chooses an original or an expanded persona sentence, as appropriate, but also defaults to the null persona (∅) that leads to a bland response.

Q: What can cause a forest fire?
(1) rain (2) static electricity (3) microbes (4) ...
A: static electricity
Q+A (declarative): Static electricity can cause a forest fire.

Explanation (reasoning chain): [positive (valid)]

Static electricity can cause sparks // (from corpus)
AND *Sparks can start a forest fire* // (from corpus)
→ *Static electricity can cause a forest fire* // (Q+A)

Explanation (Generalized reasoning chain, GRC):

X *can cause* **Y AND Y** *can start* **Z** → **X** *can cause* **Z**

Figure 3.4: Our datasets contain annotated (valid and invalid) *reasoning chains* in support of an answer, allowing explanation classifier models to be trained and applied. We also find that using a variabilized version of the chains improves the models' robustness.

3.5 Related Work

Building personalized dialog agents has been a popular task recently, thanks to Zhang et al. (2018a) who extensively studied the task with a new dataset PERSONA-CHAT‘, later as a form of a challenge (Dinan et al., 2019a), where the dialog agent is seeded with a predefined persona in the form of multiple sentences of textual description, mirroring a casual human conversation which many times draws snippets from individual personal experiences and facts. Recent works focus on improving persona-grounded dialog generation performance (Wolf et al., 2019; Mazaré et al., 2018; Bao et al., 2019) as well as persona consistency in generated dialog (Welleck et al., 2019b; Li et al., 2019b; Song et al., 2019a). Bao et al. (2019) proposed a reinforcement-learning-based framework that promoting informativeness and persona-consistency via personal knowledge exchange. Xu et al. (2020b) focused on using plausible topical keywords related to the available persona facts using a neural topic model to explore beyond the given knowledge, possibly closest to our work. We rather focus on obtaining commonsense implications of the given persona in the form of text snippets that are more expressive than topical keywords.

Persona-grounded dialog generation is a special case of knowledge-grounded dialog generation. Knowledge grounding in dialog has many real-world applications that are well-studied in recent literature (Zhou et al., 2018b; Ghazvininejad et al., 2018a; Dinan et al., 2019b; Lewis et al., 2019). In this work we use fine-grained grounding/selection on persona which performed better than encoding the entire persona for each response. Such fine-grained selection has been found useful in prior works on text generation such as dialog (Lian et al., 2019) and image captioning (Jhamtani and Berg-Kirkpatrick, 2018). For dialog generation, a contextual knowledge selection has been successfully applied in prior works (Parthasarathi and Pineau, 2018). Specifically, Zhao et al. (2017) and later Song et al. (2019b) proposed a conditional-VAE framework to learn latent context given the dialog history to guide knowledge selection.

Finally, few recent works focused on augmenting grounding with commonsense knowledge with successful applications in open-domain topical dialog generation (Ghazvininejad et al., 2018a; Moon et al., 2019), story generation (Mao et al., 2019), etc.. In this work, we extend this effort into persona-grounded dialog generation via augmenting grounding persona with commonsense knowledge.

3.6 Application to Constructing Reasoning Chain Explanations for Multi-hop QA

In this section I discuss applying similar notions of knowledge base grounded generation discussed above to reasoning chain construction to act as explanations. More specifically, I propose methods for constructing reasoning chain explanations for multihop question answering where we 1) generates candidate reasoning chains through a retrieval procedure, (2) carry out a delexicalization operation by identifying

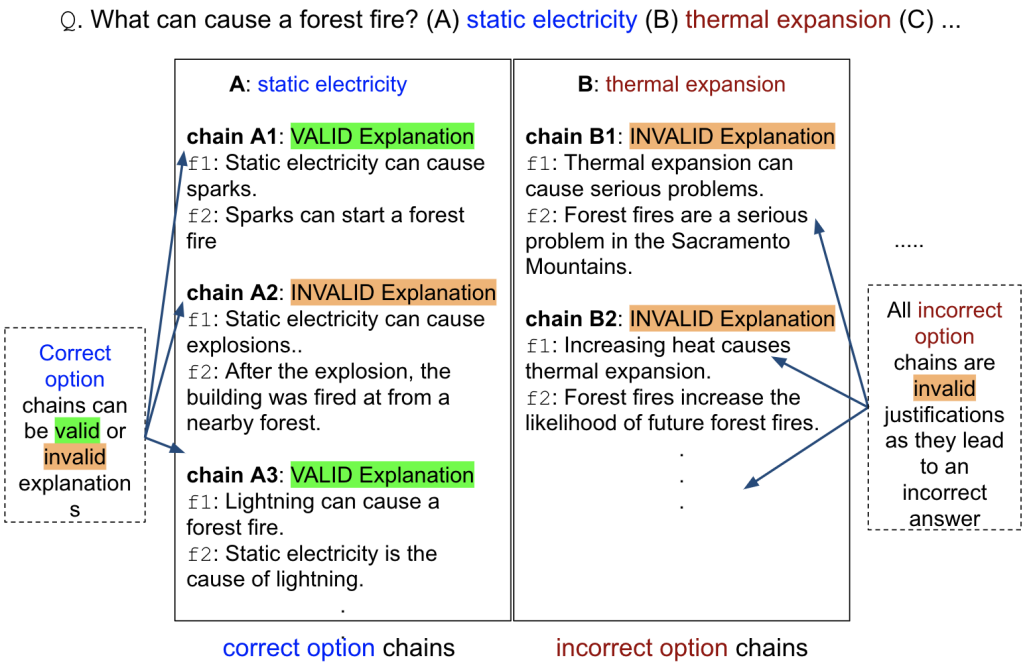


Figure 3.5: QASC contains multiple-choice questions, plus one gold (valid) reasoning chain for the correct answer. To find valid reasoning chains, we first generate candidates for each answer option using a 2-step retrieval process. We then collect annotations for the correct answer option chains to train and evaluate models to detect valid reasoning chains. (Above, chains A1 and A3 are valid, while A2, B1, and B2 are invalid).

overlapping entities in chains (3) operate in the resulting abstract pattern space to identify valid reasoning patterns. The proposed technique performs similar or better than several baselines, is more robust under certain perturbations, and leads to discovers of interesting reasoning patterns.

While neural systems have become remarkably adept at question answering (QA), e.g., (Clark and Gardner, 2018), their ability to *explain* those answers remains limited. This creates a barrier for deploying QA systems in practical settings, and limits their utility for other tasks such as education and tutoring, where explanation plays a key role. This need has become particularly important with *multihop question-answering*, where multiple facts are needed to derive an answer. In this context, seeing a *chain of reasoning* leading to an answer, can help a user assess an answer's validity. Our research here contributes to this goal.

We are interested in questions where the decomposition into subquestions - hence the explanation structure - is *not evident from the question*, but has to be found. For example, "Does a suit of armor conduct electricity?" might be answered (hence explained) by first identifying what material armor is made of, even though the question itself does not mention materials. (This contrasts with earlier multihop QA datasets, e.g., HotpotQA (Yang et al., 2018a), where the explanation structure is evident in the question itself. For example, "What nationality was James Miller's wife?" implies a chain of reasoning to first finds Miller's wife, then her nationality. Such cases are easier but less representative of natural questions.) Multihop datasets of this kind include OpenBookQA (Mihaylov et al., 2018) and more recently QASC (Khot et al., 2020). However, although providing QA pairs, these datasets provide limited explanation information. OpenBookQA does not come with any explanation data, and QASC only provides a single gold explanation for each answer, while in practice there may be multiple valid explanations.

To alleviate this lack of data, we contribute three new datasets: The first (and largest) is `eQASC`, containing annotations on over 98K candidate explanations for the QASC dataset, including on *multiple* (typically 10) possible explanations for each answer, including both valid and invalid explanations. The second, `eQASC-perturbed`, contains semantically invariant perturbations of a subset of QASC explanations, for better measuring the generality of explanation prediction models. Finally `eOBQA` adds

X can cause Y AND Y can start Z → X can cause Z
X is used for Y AND Z are X → Z are used for Y
X are formed by Y AND Y are made of Z
→ X are formed by Z
X are Y AND Y are Z → X are Z
X produce Y AND Y is a Z → X produce Z
X increases Y AND X occurs as Z → Z increases Y
X changes Y AND Y is Z → X changes Z
X is Y AND X carries Z → Y carries Z
X changes an Y AND Z are examples of X → Z change an Y
X are formed by Y AND X are formed through Z
→ Y can cause Z
X changes a Y AND Z start most X → Z can change Y

Figure 3.6: Examples of the highest scoring generalized reasoning chains (GRCs) found in eQASC.

adding explanation annotations to the OBQA test set, to further test generality of models trained on eQASC. In addition, we use these datasets to build models for detecting valid explanations, to establish baseline scores. Finally, we explore a delexicalized chain representation in which repeated noun phrases are replaced by variables, thus turning them into *generalized reasoning chains*, as illustrated in Figure 3.4. We find that generalized chains maintain performance while also being more robust to perturbations, suggesting a promising avenue for further research. Finally, we observe that top scoring GRCs identify interesting deductive and abductive reasoning patterns (Some of the top scoring chains are shown in Figure Figure 3.6)

4 Neuro-Symbolic Rules for Numerical Data

Much recent progress in various data-to-text tasks has relied on deep learning, often using neural networks with soft attention mechanisms to select salient aspects from data, and constructing fluent natural language text. However, in naturally occurring descriptions of data, humans often refer to higher-level patterns which may require complex computations on data. Consider a numerical table consisting of stock prices of a company over a period of time. A simple description would require simply selecting some value and state it in natural language. In contrast to it, a more complex pattern, such as 'Stock for DJI rose more steeply over last week compared to stock GSPC', will require much higher level reasoning over multiple cells/values. But how do we effectively learn valid and interesting aggregations of values. Can we do so in a manner which also exposes some abstractions of the underlying computations and reasoning leading to such descriptions? Turns out that such higher level patterns often cannot be extracted using neural models with attention alone. Moreover, even for cases where it can model some of the complex patterns, such models tend to imprecise (e.g. generate hallucinated descriptions), and offer little insights to a user about its working.

In this chapter, I will discuss methods for time series captioning (Jhamtani and Berg-Kirkpatrick, 2021) with truth conditional semantics. We propose methods to induce modules which detect useful trends such as peak or dip in time series numerical data, being guided only by accompanying natural language descriptions. The modules are combined via a latent computation graph to form programs, which outputs truth value of whether the feature represented by the composed computation graph holds true for a given data point or not. The resulting model gives highly precise outputs based on learned salient patterns, and exposes the modules needed for the computation.

4.1 Introduction

There has been large interest in generating automatic text description (McKeown, 1992) of tabular data – for example, prior work has sought to generate biographies from tables of biographical information (Lebret et al., 2016), and generating descriptions from structured meaning representations (Gardent et al., 2017). However, in many of these tasks the main focus is on designing systems that are able to *select entries* from tabular or equivalent data during generation by using neural attention mechanisms. In many naturally occurring descriptions of tabular data, humans often refer to higher-level patterns, for example in the description of stock index pricing over the week in Fig. 4.1, the speaker refers to how the stock price peaks towards the ending. Some recent work has looked into setups which require non-trivial inference (Wiseman et al., 2017; Chen et al., 2020). However, they typically don't involve inference about numerical patterns in time series data. Moreover, much recent prior work on identifying more complex patterns in data for captioning has relied on deep neural networks, often employing neural encoders and attention mechanisms. However, such approaches often fail to generate faithful responses and lack interpretability (Tian et al., 2019; Dhingra et al., 2019; Parikh et al., 2020).

We present a novel neural truth-conditional model for time series captioning, which learns to identify patterns which hold true for the input time series (Figure 4.2). We first sample a latent program from the space of learned neural operators. Each program produces a soft truth-value. Then, with probability proportional to each program's truth-value, a language decoder generates a caption. Thus, programs that yield low truth values, do not produce captions. Critically, the decoder takes *an encoding of the program*

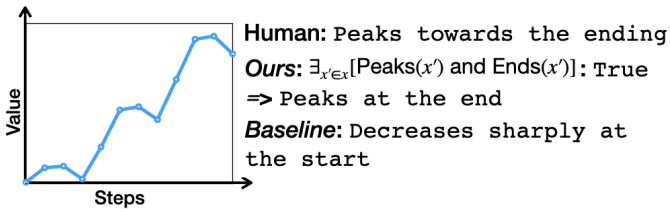


Figure 4.1: We propose a neural truth conditional model for high precision and diverse time series caption generation.

itself, rather than the time series, in order to determine output text. Overall, this approach allows for both: (a) precision in generated output through explicit truth conditioning, and explicit program structure as a representation of time series trends, and (b) diversity in caption generation through the sampling process.

While some of the patterns in data are complex, they can be considered to have been constructed by composing simpler concepts such as slope (rate of change of value) or comparisons (between values at give points). As such, our programs are constructed by composing simpler operations/modules. Such a modular design enables sharing of modules across multiple programs, leading to more data efficient learning of module parameters, and also providing better generalization to unseen compositions of modules. We consider a relatively simple space of three module types, using which our model is able to capture a significant fraction of the patterns present in data. The module types could be expanded in future to capture more complex patterns. Our model treats the choice of composed computation graph of programs as a latent variable, learned using natural language descriptions as the only supervision. In this respect, our approach is related to neural module networks used in Andreas et al. (2016a,b), which condition on a question to generate a program, which then operates on an image or other data to predict an answer. In our case, the constructed computation graph operates and identifies salient patterns in the source data directly, without being guided by an input question.

Our main contributions are as follows: We propose a novel method for time series captioning which first induces useful patterns via composing simpler modules, identifies the programs which hold true, and finally generates text describing the selected program. Towards this end, we collect and release two datasets consisting of time series data with accompanying English language description of salient patterns. We observe that the proposed method is able to learn useful patterns, exhibits compositionality and interpretability, and generates outputs that are much more faithful to the input compared to strong traditional neural baselines. [1]

4.2 Truth-Conditional Natural Language Description

Our goal is to learn models for describing salient patterns in time series data. The main research challenge involved is to learn the types of patterns that humans find salient in time series data, using natural language descriptions as the only source of supervision during training. Based on the novel dataset we collect (described in Section 4.3 , we find that the patterns humans identify tend to describe increasing or decreasing trends, volatility, comparisons of start and end values, presence of peaks and dips. They also mention temporal location of patterns, such as 'at the beginning' of the time series. Thus, our model should be able to learn patterns such as 'increase' or 'ends with higher value compared to start', and temporal aspects such as 'begin' or 'end'.

One way to operationalize this process is through the lens of formal logic: e.g. an increasing trend at the beginning of a time series x can be represented trough the logic z: $\left[\exists_i \text{ s.t. } \textsc{increase}(x_i) \text{ AND } \textsc{begin}(i)\right]$ Thereafter, if the program returns `true` on the input, one can condition on only the logical program z to generate output text that describes this pattern via a decoder, $p(y|z)$. However, this still requires learning or defining modules for patterns and temporal location. Inspired by neural module networks (Andreas et al., 2016a,b), we propose to use functions parameterized by neural networks (Figure

[1] Data and code can be found at https://github.com/harsh19/TRUCE.

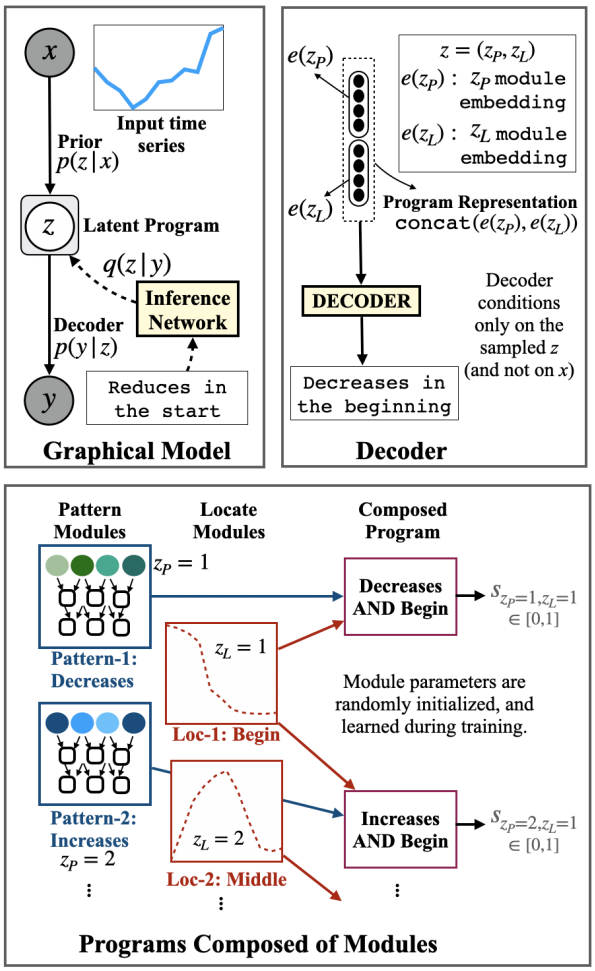


Figure 4.2: Method Overview: We present a truth conditional model for time series captioning, which first identifies patterns (composed of simpler modules) which hold true for a given data point. Decoder conditions only on a sampled program z (and not on input x), generating high precision outputs.

4.2) as modules, incorporating inductive bias through architecture design. However, unlike past work, we condition only on an encoding of sampled programs that return `true` to generate output text.

4.3 Datasets

We are interested in modeling numerical patterns and trends in time series data. However, there is a lack of existing data sources with time series data paired with natural language descriptions. Some prior work on weather forecasting data (such as Sumtime-Mausam (Sripada et al., 2003)) are typically small (only 1045 data instances), and are limited in the scope of patterns they encompass. ToTTo dataset (Parikh et al., 2020) contains a small fraction of descriptions based on numerical reasoning and patterns - however, the main challenge is to find the correct value(s) by identifying the relevant row and column in a table. LOGIC-NLG (Chen et al., 2020) consists of 37K tables and corresponding natural language descriptions, some of which require comparisons of cells in a table. In contrast, we focus on trends and patterns in time series data. Thus, we construct a new dataset where natural language descriptions are collected for naturally occurring stock price time series data (Section 4.3.1). Additionally, we collect natural language descriptions for a synthetically constructed set of time series to evaluate and analyse our models in a more controlled setup (Section 4.3.2).

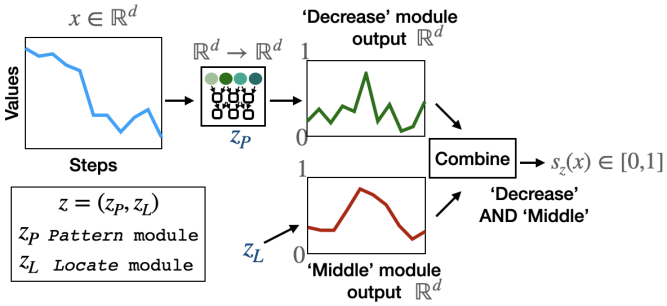


Figure 4.3: A program $z = (z_P, z_L)$ operates on an input time series x to given final output score $s_z(x)$. The module instances are learned from scratch during training.

4.3.1 STOCK Dataset

We collect naturally occurring time series data in the form of stock prices. We utilize the Google Finance API to collect stock prices of 7 randomly chosen technology companies over a period of 20 years. We collect weekly (beginning of week) as well as and daily stock price values. We sub-select a total of 1900 instances, each of consists of sequence of T(=12) values. Each instance is sampled from the stock data as follows: (1) we pick one of the companies uniformly at random (2) we randomly pick weekly or daily series with equal probability, (3) we pick a sequence of values of given length T, ensuring no overlap with any previously selected time series. (4) Additionally, since different company stocks can be in very different range of values, we normalize such that all the values are between 0 and 100: $v' = 100 * (v - min)/(max - min)$. However, normalizing this way directly would create undesirable biases in the dataset since each time series would necessarily cover entire range 0-100. Instead, to compute *max* and *min*, we additionally consider 10 values (chosen based on manual inspection) just before and just after the currently selected range.

Annotation collection: We collect 3 natural language annotations for each of the 1900 data points, leading to a total of 5700 paired time-series with natural language descriptions. We split the 1900 unique time series and associated captions into train, dev, and test splits with ratio 8:1:1.

Annotator description: We use Amazon Mechanical Turk as a crowd-sourcing platform. We limit to annotators from Anglophone countries, with HIT (Human Intelligence Task) acceptance rates of more than 90%, and minimum number of accepted HITs as 100. Annotators were paid 25 cents for each annotation (which comes to average hourly rate of over USD 23).

Quality Control: Based on initial pilot studies, we found it useful to show annotators plots instead of tables of values, as we are interested in high level patterns rather than specific values. We do not label the plot lines with actual stock names to remove any potential biases one may have about specific company stocks. Finally, we restrict annotations to a maximum of 9 words, so that one annotation reflects only one pattern. Each HIT is labelled by 3 different annotators. We manually inspected at least one annotation from each unique annotator, and ruled out (but still paid) annotations for about 7% annotators for being poor quality.

Encouraging Lexical Diversity: We encouraged annotators (through instructions) to not limit themselves to words shown in examples. Additionally, we limit each annotator to a maximum of 10 HITs to increase diversity in annotations.

Dataset Statistics: There are a total of 861 unique words across the 5700 captions. Most annotation sentences follow a simple syntactic structure. Additionally, we picked a random subset of 100 data points, and manually classified most of them into following major buckets: trend (increase/decrease trends: 48%) superlative(max/min values; peaks and troughs: 20%); comparisons(comparison of start and end values: 10%); volatility (flat/smooth; irregular: 12%).

4.3.2 Synthetic Time Series (SYNTH)

To develop and test models in a more controlled setup, we synthetically construct time series data. Our synthetic time series data is constructed such that each time series has exactly one of the following 6 patterns: increases-in-beginning, increases-in-middle, increases-in-end, decreases-in-beginning, decreases-in-middle, decreases-in-end. The resulting dataset consists of a total of paired 720 time series - natural language annotations.

Each synthetic time series is generated as follows: First, the trend is chosen: increase or decrease. A trend is realized through a straight line of length $L <= T/3$, with randomly chosen intercept and slope within a range based on the trend selected. Next, we randomly select one of the 3 temporal locations : begin, middle, end – and based on the choice, the pattern is placed in first 40 percentile, 30-70 percentile, or 60-100 percentile respectively, of the entire length T. The region outside the trend is flat. Finally, small noise is added to each point. The setup is such that the resulting values are always in (0,100) range. Examples and more specific details can be found in Appendix.

4.3.3 Model

Our goal is to generate a text caption y describing a salient pattern in an input time series x. Our model's generative process is depicted in Figure 4.2 and operates as follows: Conditioned on an input time series x, we first sample a program z from a learned prior, $p(z|x)$. The latent program z is composed of several operations/modules composed together, and outputs a truth value score. The prior is governed by the truth-values of corresponding programs, so that we are likely to sample programs with high truth values. Next, we sample caption y conditioning *only* on the encoding of sampled program z to generate the final text – i.e. y is independent of x given z. Intuitively, if the latent program encodes sufficient information to describe the pattern it detects, captioning need only depend on the program itself.

The set of latent 'programs' in our model are learned from data. On executing a program z on the input time series data x, we obtain output score $s_z(x)$ (between 0 and 1, both inclusive). Score $s_z(x)$ represents the model's confidence about whether the pattern corresponding to the program holds true for the given input time series. Note that $s_z(x)$ does *not* represent the prior probability of program z – since multiple programs can be true for a given time series, and $\sum_z s_z(x) \neq 1$. We provide our model with a set of building blocks / modules, which combine to form programs. The composition of modules into programs as well as the module parameters are unobserved in data, and are learned during model training. The compositionality in the program space enables modules being shared across programs, leading to more efficient learning. The programs we consider will prove quite effective in experiments, but are actually relatively simple, being composed of only three module types. Our framework is extensible, however, and future work might consider larger program spaces. We refer to our proposed method as TRUCE (**TRU**th **C**onditional g**E**neration).

4.3.4 Programs and Modules

As previously mentioned, each program z in our model is composed of several learnable operations/modules. Following prior work on neural modular networks (Andreas et al., 2016b), we consider multiple module types, and incorporate inductive biases in their architecture to learn useful numerical patterns. In the current study, however, we limit to three simple types of patterns: *pattern*, *locate*, and *combine*, leaving extensions to the module space as a future direction. These modules are composed together into programs that operate on the input time series (Figure 4.2)

The module types *pattern* and *locate*, output a vector of the same length as the input vector. Both of them output a temporally localized vector, with each value between 0 and 1 (achieved by applying a sigmoid activation function), representing the degree of confidence that the pattern it represents is present at the corresponding position on the temporal axis. For example, as shown in Figure 4.3, the output of a learned *locate* module is a vector with high values in the middle part, and the output of the *pattern* module is high on those positions where there is a decrease in the value in the input time series.

Method	COR	PPL	Bleu-3/4	Cider	Rouge	BERT
TRUCE	**92**%	13.9	0.61/0.46	1.40	0.74	0.77
FCENC	39%	16.7	0.45/0.28	0.81	0.61	0.65
LSTMENC	45%	11.2	0.43/0.28	0.87	0.62	0.63
CONVENC	53%	11.0	0.47/0.32	1.00	0.66	0.67
FFTENC	39%	22.7	0.38/0.22	0.67	0.58	0.54
NEARNBR	71%	NA	0.28/0.14	0.60	0.40	0.48

Table 4.1: Results on test split of SYNTH dataset: Human evaluation for correctness (COR) and various automated metrics. TRUCE performs much better than baselines as per correctness evaluation.

For the current study, we restrict the space of programs to consist of one *pattern* (z_P) module instance, and one *locate* (z_L) module instance. Outputs from the two modules are combined together using a *combine* module, which carries out position-wise multiplication of outputs from z_P and z_L, followed by a feed-forward layer and a sigmoid non-linearity.

Pattern modules are aimed at learning patterns such as peaks, dips, increasing trend, and so on. We realize *pattern* modules through multi layer 1-D convolutions. We argue that 1D convolutions provide appropriate architecture to induce aspects such as slopes, and compose them to identify patterns such as peaks. The *locate* module types are realized though a mixture model of K fixed Gaussians placed at equal intervals on the temporal axis of given length T. The weights of the components represent learnable parameters for such types of modules. The *combine* module type learns to transform the position-wise multiplied outputs to a real valued score, which is then passed through a sigmoid function.

4.3.5 Prior

As discussed above, the output of each program z is a real valued score between 0 and 1. We define prior over the set of programs Z as $p(z) \propto e^{\lambda s(z)}$, where λ is a hyperparameter. This formulation makes an implicit assumption that a program z being true for an input time series will make other programs less probable through conservation of probability mass. Such an assumption is necessary, as otherwise directly trying to optimize the likelihood without normalizing across programs will lead to trivial solutions, wherein each program will output high score for every input. Note that an alternative formulation could directly use softmax on an unrestricted real-value output from modules – such a formulation loses out on the semantics of soft truth output from the programs, and also fared worse in our preliminary experimental evaluations in comparison with the proposed formulation.

4.3.6 Decoder

As mentioned previously, our decoder conditions only on the program z sampled from the prior $p(z|x)$ to generate final text. To achieve this, we need to pass a program representation to the decoder. We consider an auto-regressive neural decoder such as LSTM or Transformer. At every step, the decoder considers embedding of previous token as well as the input program representation.

A straightforward approach to obtain program representation is to associate each unique program with a low dimension embedding vector. However, such an approach will not fully exploit the program structures and shared modules. Instead, we first associate each module with an embedding. Next, the representation of a program is constructed by appending the embeddings of the corresponding modules (using a fixed pre-determined order of module types). Such a representation achieves sharing of module embeddings across programs. Moreover, it enables obtaining representation of a new (unseen) program composed using the same set of modules.

Method	COR
TRUCE	**97**%
FCENC	38%
LSTMENC	50%
CONVENC	59%
FFTENC	39%
NEARNBR	72%

Table 4.2: Models trained on SYNTH data (where each time series has T=12 values) are tested on another synthetic data with T=24 without any fine-tuning.

4.4 Experiments with Synthetic Data

4.4.1 Methods

For SYNTH data, we consider several baselines listed below (More detailed descriptions are provided in the Appendix). Note that all non-retrieval baselines use the same LSTM decoder architecture as our model. (1) **NEARNBR:** The ground-truth caption of the closest matching training data instance is used as the prediction. The closest matching instance is identified via L2 distance between input time series. (2) **FCENC**: Encodes the input time series sequence using a multi-layer feed-forward encoder. (3) **LSTMENC**: Encodes the input time series sequence using a LSTM recurrent neural network. (4) **CONVENC**: Encodes time series using a multi layer convolutional neural network. (5) **FFTENC**: Encodes time series using Fourier transform features of the input.

4.4.2 Results

For TRUCE, we pick the highest scoring program, according to the prior, for description generation. We generate captions (using greedy decoding) from each of the methods for the test split.
Automated metrics measure overlap between model generated caption and the reference ground truth captions. We report Perplexity (**PPL**), BLEU-3/4 (Papineni et al., 2002a), METEOR (Banerjee and Lavie, 2005), ROUGE-L (**Rouge**) (Lin, 2004), and BertScore-Precision (**BERT**) (Zhang et al., 2020). The proposed TRUCE method gets favorable scores as per various automated metrics on the test split of SYNTH (Table 4.1).

Human Evaluations for Correctness: Automated metrics may not correlate well with actual quality of the generated output in text generation tasks (Celikyilmaz et al., 2020). As such, we report human evaluation results as well. We recruit human annotators who are requested to provide a binary label on factual correctness **(COR)** of the captions for the test split. Each caption is annotated by three annotators, and the majority label is used. The proposed method is able to achieve a high correctness score of 92%, which is much better than the baselines. This demonstrates the usefulness of the proposed truth-conditional model in generating highly faithful captions. Output samples are provided in the Appendix.

4.4.3 Analysis

Generalization to different time series duration: SYNTH data consists of time series instances with T=12 sequence of values. We experiment the extent to which models trained on SYNTH can accurately detect patterns in time series data of different lengths without any fine-tuning. For this, we evaluate results on a separate synthetic data consisting of 100 time series with T'=24 values per time series (dataset created in the same manner as SYNTH and consists of the same set of 6 classes as in SYNTH).

We observe that TRUCE retains high correctness of the output captions (Table 4.2), whereas some of the high performing baseline show significant reduction in correctness. Note that some of the employed methods like NEARNBR and FCENC cannot work directly on inputs of length different than present in

Module id	Most freq. words associated with learned modules
pattern-1	increases, rises
pattern-2	decreases, decline, dips
locate-1	end, late
locate-2	beginning , start, initial
locate-3	middle, halfway

Table 4.3: Some of the most frequent words associated with some of the learned module instances for SYNTH data.

the training data. For such models, we first adjust length of series. For example, for length 24 input, we consider alternate values only, thereby reducing the series to length 12 (same as in the training data).

Analyzing Learned Modules: We analyze the characteristics of the learned modules by identifying the top words (excluding stop words) associated with each learned module. To do so, for a given series, we find program with highest score, and associate the annotations for that series to corresponding modules in that program. Finally, we collect the most frequent words in annotations associated with each module. We show a summary in the Table 4.3. The two trend modules seem to be getting activated for increase and decrease patterns respectively.

Compositionality of Learned Modules We analyze if the proposed model uses its compositional parameterization effectively. To do so, we conduct a simple analysis as follows: We train TRUCE on a subset of synthetic data consisting of only the following 4 patterns: increase-beginning, decreases-end, increase-middle, decreases-middle. We examine this trained model's behavior on test data points consisting of the two unseen patterns: increase-end and decrease-beginning. More specifically, we analyze the argmax program prediction as per the conditional prior. Based on manual inspection of modules (similar to what we discussed for analysis in Table 4.3), we know before hand the program which should be selected for these patterns. Model's prediction is considered to be correct if, for example, for an input with 'decrease-beginning' pattern, model assigns highest score to the program composed using modules corresponding to 'decrease' and 'beginning'. We observe that the highest scoring program is the correct/expected program for 92% of the cases in the test split.

4.5 Experiments with STOCK Dataset

4.5.1 Posterior Regularization:

In the initial experiments with STOCK dataset, we observe that our model suffers from model collapse, and degenerates into learning a single program only. This is perhaps because randomly initialized modules do not have much guidance to begin with. To mitigate such mode collapse issues, prior work has used mutual posterior divergence (MPD) regularization (Ma et al., 2019) $-E_{y_i,y_j}KL(q(z|y_i)||q(z|y_j))$, where y_i and y_j captions for two randomly chosen data points.

However, we note that MPD term enforces the divergence in an indiscriminate manner – divergence is encouraged even if captions are paraphrases of each other. An alternate way to encourage divergence in the inference network prediction is to encourage divergence only when two captions y_i and y_j represent different programs or patterns. However, such information is not available in the training data. Instead, we use an approximation as follows: We identify the M most frequently occurring words excluding stopwords (list available in Appendix) in the captions and are manually labelled to to represent pattern or locate or neither. Each of the words labelled to be of type pattern or locate is assigned a unique *pattern* or *locate* module id respectively. The corresponding captions thus get tagged with some heuristic (but potentially noisy) labels for module ids. Only those captions are tagged which have exactly one 'locate' word and one 'pattern' word. This leads to about 31% of the captions being assigned such heuristic labels, while the remaining data stays unlabelled.

Method	COR	Bleu-3/4	Cider	Rouge	BERT
TRUCE(Ours)	**88.4%**	0.35 / 0.19	0.36	0.50	0.57
FCENC	64.2%	0.32 / 0.19	0.43	0.47	0.56
LSTMENC	65.5%	0.35 / 0.21	0.41	0.50	0.61
CONVENC	65.9%	0.33 / 0.18	0.41	0.49	0.59
FFTENC	61.8%	0.34 / 0.19	0.39	0.49	0.58
NEARNBR	47.2%	0.12 / 0.06	0.14	0.28	0.35

Table 4.4: Results with STOCK data: Proposed method TRUCE scores the best on correctness evaluation. The best performing baseline scores 20% less on correctness evaluation. Greedy decoding was used for all the methods.

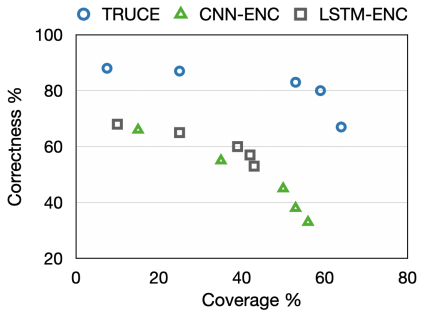


Figure 4.4: Coverage and Correctness of model outputs at different sampling settings. In general, settings with higher coverage of human written captions have lower precision of generated captions. TRUCE achieves much higher correctness scores compared to baselines for similar coverage values.

The above procedure does involve a small human-in-the-loop component. However, we note that it is a pretty light-weight involvement. For example, the system presents M(=10) most frequent pairs of words (excluding stopwords) in captions, and a person spends a couple of minutes labeling their type (locate or pattern).

4.5.2 Results

We now report results with STOCK dataset. As mentioned above, we utilize heuristic labels as an auxiliary loss when training the proposed method. Thus, for a fair comparison, the baselines **LSTMENC**, **CONVENC** and **FCENC** also use the same set of heuristic labels via a classification loss on the encoded representation in a multi-task learning setup.

The proposed method TRUCE produces high precision captions as judged by human annotators (Table 4.4). We additionally report automated text overlap scores against reference captions, though the automated metrics seem only mildly correlated with human judgement ratings. Interestingly, some of the baselines show large differences in performance in STOCK vs SYNTH datasets. For example, NEARNBR performs well on SYNTH but rather poorly on STOCK dataset, perhaps because of variety in time series instances in SYNTH being small, while the same being large in STOCK.

Diversity and Coverage: Ideally, we want models which can identify all the interesting patterns present in an input time series. Correctness results discussed earlier are indicative of faithful generation but do not necessarily capture coverage of patterns. We compute coverage of various models via the following procedure. First, we collect L(=12) samples per data point from the model. Next, we recruit human annotators to rate whether a human written reference annotations for that data point is covered by the set of L generated captions or not. For TRUCE, we perform sampling at the program selection stage, while baselines admit sampling only at the token generation stage.

Note that this makes the coverage score depend on the settings used in the sampling process (e.g. top-p value in nucleus sampling), which will also affect the correctness of the generated captions. In Figure 4.4, we demonstrate coverage and correctness values of TRUCE and two of the baseline models

Module id	Most freq associated words
pattern-1	increases, rises, gains
pattern-3	stays, remains, flat
pattern-4	bottoms, out, decline, dips
loc-1	start, beginning, initially

Table 4.5: Inference Network Analysis: Analyzing words frequently present in captions when the argmax program prediction from inference network comprises of a give module-id.

under different sampling conditions. In general, restricting samples to a low value of top-p leads to lower coverage but higher correctness. Overall, TRUCE behaves in a more favorable manner. For example, comparing TRUCE against CONVENC, for roughly same level of coverage (e.g. 50%), correctness is much higher for TRUCE (83% against 45% for CONVENC). However, there still seems to be a gap in the coverage of patterns, and can perhaps be addressed by incorporating more module types.

4.5.3 Analysis

Direct conditioning on the input: Our decoder conditions only an encoding of a sampled program. We hypothesize that such an approach creates a bottleneck discouraging the decoder from learning spurious correlations between the input time series and the output text. To inspect the usefulness of the proposed abstraction, we consider an alternative model wherein the decoder conditions on the input time series as well – by providing output of a convolutional encoder (same as in CONVENC) to the decoder. More specifically, the program representation and the encoder representation are concatenated before being fed to the decoder. Lets refer to such a model with decoder having direct access to the input as TRUCE-D. For STOCK data, TRUCE-D gets correctness of 69% compared to 88% for TRUCE.

Analysis of Inference Network: We analyze the predictions of the inference network at the end of model training. Particularly, we associate the set of ground truth annotations in validation split to module-ids present in the argmax program prediction from the inference network. Next, we identify the most frequently occurring tokens present for each module-id/module-instance. We observe that the inference network seems to be associating semantically similar words to the same module instance (Table 4.5).

4.6 Related Work

Time-Series Numerical Data and Natural Language (Andreas and Klein, 2014) worked on grounding news headlines to stock time series data by aligning sub-trees in sentence parses to segments of time series. (Murakami et al., 2017) generate stock data commentary using encoders such as convolutional and recurrent neural networks, similar to the baselines used in our experiments. (Sowdaboina et al., 2014) focus on the task of describing wind speed and direction. Time series data in the form of charts has been utilized in some prior work in figure question answering (Kahou et al., 2018; Chen et al., 2019a).

Past work has explored ways to handle numerical data in a variety of input data domains using neural networks. (Trask et al., 2018) propose neural logic unit for tasks such as counting objects in images. Prior work has investigated handling of numeracy in question answering datasets (Dua et al., 2019; Andor et al., 2019; Gupta et al., 2020), typically using a predefined set of executable operations or using specific distributions for number prediction (Spokoyny and Berg-Kirkpatrick, 2020; Thawani et al., 2021).

Neuro-Symbolic Methods: Andreas et al. (2016b) proposed to use neural modular networks for visual question answering. Since then, similar approaches have been used for several other tasks such as referring expression comprehension (Cirik et al., 2018), image captioning (Yang et al., 2019), and text question answering (Andreas et al., 2016a; Khot et al., 2021). Compared to such past efforts, we induce

the latent numerical and temporal detection operations, pick a high scoring program, and condition only on a program encoding to generate the output description. In this respect, our work is also related to prior work on neural discrete representation learning (van den Oord et al., 2017; Zhao et al., 2018), though none of these past works explore utilizing such techniques for data to text problems. Our proposed model abstracts the numerical pattern detection from text generation. Related ideas have been explored in the past in other domains and tasks (Gehrmann et al., 2018; Jhamtani and Berg-Kirkpatrick, 2018; Amizadeh et al., 2020).

Data to Text: Tabular or structured data to text generation has been explored in prior work (Lebret et al., 2016; Novikova et al., 2017b; Wiseman et al., 2017; Jhamtani et al., 2018; Gehrmann et al., 2021). The Rotowire dataset (Wiseman et al., 2017) is comprised of sports summaries for tabular game data which may require modeling of numerical operations and trends. However, much of the past work has relied on neural models with attention mechanisms, without explicit and interpretable notions of numerical operations. Fidelity to the input in the context of neural text generation has received a lot of attention lately (Cao et al., 2018). Prior work has approached the aspect of fidelity to input through changes in model training and/or decoding methods (Tian et al., 2019; Kang and Hashimoto, 2020; Majumder et al., 2021; Goyal and Durrett, 2021; Liu et al., 2021). We explore a different approach that increases fidelity through conditional independence structure and model parameterization.

5 Structured Discriminators for Modeling Long-Range Latent Patterns

Existing neural language models often fail to capture higher-level structure present in text such as rhyming constraints in poetry. Much prior work on poetry generation uses manually defined constraints which are satisfied during decoding using either specialized decoding procedures or rejection sampling. The rhyming constraints themselves are typically not learned by the generator. In this chapter, we propose an alternate approach that uses a structured discriminator to learn a poetry generator that directly captures rhyming constraints in a generative adversarial setup (Jhamtani et al., 2019). By causing the discriminator to compare poems based only on a learned similarity matrix of pairs of line ending words, the proposed approach is able to successfully learn rhyming patterns in two different English poetry datasets (Sonnet and Limerick) without explicitly being provided with any phonetic information. Finally, I will conclude the chapter with a brief discussion on application of the proposed approach on modeling self repetition in music generation (Jhamtani and Berg-Kirkpatrick, 2019).

5.1 Introduction

Many existing approaches to text generation rely on recurrent neural networks trained using likelihood on sequences of words or characters. However, such models often fail to capture overall structure and coherency in multi-sentence or long-form text (Bosselut et al., 2018; Holtzman et al., 2018). To rectify this, prior work has proposed losses which encourage overall coherency or other desired behavior (Li et al., 2016b; Zhang and Lapata, 2017; Bosselut et al., 2018). However, most of these approaches rely on manually provided definitions of what constitutes a good or suitable structure, thereby limiting their applicability. In this paper we propose a method for English poetry generation that directly learns higher-level rhyming constraints as part of a generator without requiring strong manual intervention. Prior works on poetry generation have focused mostly on ad-hoc decoding procedures to generate reasonable poetry, often relying on pruning from a set of candidate outputs to encourage desired behavior such as presence of explicitly-defined rhyming patterns (Oliveira, 2017; Ghazvininejad et al., 2018b).

We propose an adversarial approach to poetry generation that, by adding structure and inductive bias into the discriminator, is able to learn rhyming constraints directly from data without prior knowledge. The role of the discriminator is to try to distinguish between generated and real poems during training. We propose to add inductive bias via the choice of discriminator architecture: We require the discriminator to reason about poems through pairwise comparisons between line ending words. These learned word comparisons form a similarity matrix for the poem within the discriminator's architecture. Finally, the discriminator evaluates the poem through a 2D convolutional classifier applied directly to this matrix. This final convolution is naturally biased to identify spatial patterns across word comparisons, which, in turn, biases learned word comparisons to pick up rhyming since rhymes are typically the most salient spatial patterns.

Recent work by (Lau et al., 2018) proposes a quatrain generation method that relies on specific domain knowledge about the dataset to train a classifier for learning the notion of rhyming: that a line ending word always rhymes with exactly one more ending word in the poem. This limits the applicability of their method to other forms of poetry with different rhyming patterns. They train the classifier

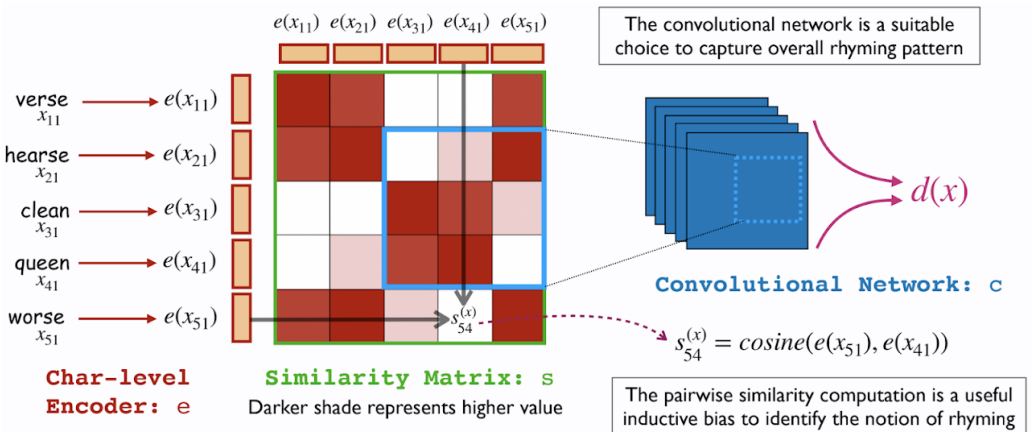


Figure 5.1: Model Overview: We propose a structured discriminator to learn a poetry generator in a generative adversarial setup. Similarities between pairs of end-of-line words are obtained by computing cosine similarity between their corresponding representations, produced by a learned character-level LSTM encoder. The discriminator operates on the resulting matrix S representing pair-wise similarities of end words. The proposed discriminator learns to identify rhyming word pairs as well as rhyming constraints present in the dataset without being provided phonetic information in advance.

along with a language model in a multi-task setup. Further, at generation time, they heavily rely on rejection sampling to produce quatrains which satisfy any valid rhyming pattern. In contrast, we find that generators trained using our structured adversary produce samples that satisfy rhyming constraints with much higher frequency.

Our main contributions are as follows: We introduce a novel structured discriminator to learn a poetry generation model in a generative adversarial setup. We show that the discriminator induces an accurate rhyming metric and the generator learns natural rhyming patterns without being provided with phonetic information. We successfully demonstrate the applicability of our proposed approach on two datasets with different structural rhyming constraints. Our poem generation model learned with the structured discriminator is more sampling efficient compared to prior work – many fewer generation attempts are required in order to obtain an valid sample which obeys the rhyming constraints of the corresponding poetry dataset.

5.2 Method

Many forms of poetry make use of rhyming patterns on line-ending words (Oliveira, 2017). Therefore, to characterize a rhyming poem, a model needs (1) to know what it means to rhyme (2) to identify the specific permissible rhyming patterns for a particular poem type. For example, a limerick is a 5 line poem with a rhyming constraint of the type AABBA, i.e. the ends of the first, second, and fifth lines rhyme. We consider an adversarial learning setup with a hierarchical language model and a structured discriminator, such that the discriminator is trained to distinguish between generated examples and training examples, and the generator is trained to *fool* the discriminator. Our novel structured discriminator operates on a matrix which encodes a *learned* pair-wise similarity function of the line ending words. We refer to our model as **RHYME-GAN**.

5.2.1 Neural Generation Model

Our generator is a hierarchical neural language model (Figure 5.1) that first generates a sequence of line-ending words, and thereafter generates the poem's lines conditioned on the ending words. We use recurrent neural networks for ending word generation as well line generation conditioned on ending words. Following prior work (Lau et al., 2018), we generate words in each line in reverse order (i.e.

right to left), and begin generation with the last line first. Let $\hat{x}$ represent a sample from the current generator distribution, denoted by p_θ, where θ represents the generator parameters. We initialize the word embeddings in the generator with pre-trained word embeddings (Lau et al., 2018) trained on a separate non-sonnet corpus.

5.2.2 Structured Discriminator

We introduce a structured discriminator, denoted by function $f_\phi(x)$, which outputs the probability that x is a sample from the dataset as opposed to generated. Our architecture defines an intermediate matrix $S \in R^{T \times T}$, where T denotes the number of lines in the poem, which encodes pair-wise similarities between line ending words in order to capture rhyming structure. The discriminator's output is determined by a two layer 2D convolutional neural network applied to S. Convolutional neural networks have been shown to capture local as well as global patterns in 2D data – for example, images. Thus, our discriminator is composed of two main components: computation of a matrix S, and a convolutional neural network to classify the computed matrix S. The pair-wise computation provides a useful inductive bias to identify the notion of rhyming, whereas the convolutional network is a suitable choice to capture overall rhyming patterns.

More specifically, let the words at the ends of lines in x be denoted by e. The number of ending words will be same as the number of lines in x, which we denote as T. We encode each ending word using a character-level LSTM (Hochreiter and Schmidhuber, 1997) denoted by g_{ϕ_g}, and use the last hidden state of the LSTM as a vector representation of the word. We let S_{ij} be the cosine similarity between the representations of ending words e_i, e_j, given by following equation:

$$S_{ij} = \frac{g(e_i)g(e_j)}{|g(e_i)||g(e_j)|} \tag{5.1}$$

The matrix S is passed through a convolutional neural network composed with a linear layer, together denoted by c_{ϕ_c}. The final output is passed through a sigmoid non-linearity, so that $f_\phi(x) \in [0,1]$. The value of $f_\phi(x)$ represents the discriminator's assessment of the probability that datum x belongs to the *real* dataset, rather than being a generated sample. The discriminator's objective will train it to distinguish between a sample x from training data $\mathscr{X}$, and a generated sample $\hat{x}$, in a binary classification setup. Specifically, we define the discriminator loss for $x, \hat{x}$ as follows:

$$d(x, \hat{x}; \phi) = -\log(f_\phi(x)) - \log(1 - f_\phi(\hat{x})) \tag{5.2}$$

5.2.3 Learning

Generator parameters θ and discriminator parameters ϕ are trained together under following objective:

$$\min_\theta \left[\mathbb{E}_{x \in \mathscr{X}} \left[-\log p_\theta(x) + \lambda \max_\phi \mathbb{E}_{\hat{x} \sim p_\theta}[-d(x, \hat{x})] \right] \right] \tag{5.3}$$

Note, in addition to using a traditional adversarial objective, we also include a likelihood term to help stabilize the generator. λ is a hyperparameter which controls the relative weight of the two terms. Since sampling of $\hat{x}$ from generator involves discrete choices, we use the REINFORCE (Williams, 1992) algorithm to train the generator using learning signal from the adversarial loss term. The generator simultaneously gets an exact gradient from the likelihood portion of the objective. We observe training is more stable when we pretrain the LSTM word encoder $g_{\phi_g}(.)$ part of the discriminator, along with a separate LSTM decoder, using an auto-encoding objective on words in the vocabulary.

Model	Expected #Samples	
	SONNET	LIMERICK
DEEP-SPEARE	153.8	N/A
RHYME-LM	169.5	500
RHYME-GAN-NS	4.8	26.6
RHYME-GAN	**3.7**	**4.7**

Table 5.1: Sampling efficiency: We obtain 10K samples of poetry without additional intervention during decoding, and report the expected samples as inverse of the fraction of samples satisfying valid rhyming patterns for the corresponding dataset. Lower values are better.

5.3 Experiments

5.3.1 Datasets

We work with the Shakespeare SONNET dataset (Lau et al., 2018) and a new LIMERICK corpus. Each sonnet in the Sonnet dataset is made up of 3 quatrains of 4 lines each, and a couplet. The dataset consists of 2685 sonnets in train, and 335 each in validation and test splits. The quatrains typically have one of the following rhyming structures: AABB, ABAB, ABBA, though some deviations are observed in the dataset. This may be because rhyming patterns are not always strictly followed in writing quatrains, and there are possible inaccuracies in the word pronunciation dictionaries used (e.g. some words can have multiple different pronunciations based on context).

A limerick is a form of verse with five lines. Limericks typically follow a rhyming pattern of AABBA. We collect limericks from an online collection[1]. Due to a large vocabulary in the full collection, we filter the dataset to retain only those limericks whose all the words are in a subset of 9K most frequent words. Our final dataset consists of $10,400$ limericks in train and 1300 each in validation and test splits. We train and evaluate the models separately on each corpus.

5.3.2 Poem Generator

Sampling efficiency We compute the expected number of samples needed before we sample a quatrain which satisfies one of the hand-defined rhyming patterns. Towards this end, we get 10K samples from each model without any constraints (except avoiding *UNK* - unknown tokens). Following prior work (Lau et al., 2018), words are sampled with a temperature value between 0.6 and 0.8. We use CMU dictionary (Weide, 1998) to look up the phonetic representation of a word, and extract the sequence of phonemes from the last stressed syllable onward. Two words are considered to be rhyming if their extracted sequences match (Parrish, 2015). We consider a generated quatrain to have an acceptable pattern if the four ending words follow one of the three rhyming patterns: AABB, ABBA, ABAB. Similarly for LIMERICK, we consider only those samples to be acceptable which have line endings of the rhyming form AABBA.

We consider a baseline **RHYME-LM**, which has the same generator architecture as RHYME-GAN but is trained without the discriminator. We also compare with **RHYME-GAN-NS** which uses a simpler non-structured discriminator. Specifically, it uses a discriminator which first runs a character-level encoder for each ending word - similar to RHYME-GAN - but then instead of computing pair-wise similarity matrix, it utilizes a LSTM on the sequence of the computed representations.

As can be observed from Table 5.1, DDLA needs fewer samples than other methods to produce an acceptable quatrain or a limerick, indicating that it has learned natural rhyming structures more effectively from data. Note we do not report DEEP-SPEARE on Limerick due to their SONNET specific assumption that for a given end-of-line word there is exactly one more rhyming word among other end-of-line words. Additionally, RHYME-GAN-NS performs worse than RHYME-GAN, and the difference in

[1] http://hardsoft.us. Accessed May 2019.

Model	NLL	
	SONNET	LIMERICK
DEEP-SPEARE	4.38	N/A
RHYME-LM	**3.97**	**3.48**
RHYME-GAN	3.98	3.49

Table 5.2: Held out negative log likelihood per token for poems in test split.

performance is more prominent in LIMERICK – demonstrating that the proposed structure in the discriminator provided useful inductive bias. Note that compared to 4 line quatrains in SONNET, LIMERICK has 5 line poems and has arguably more complex rhyming pattern constraints.

Likelihood on held out data We report negative log likelihood (NLL) on test splits (Table 5.2). For SONNET, RHYME-GAN achieves a test set NLL of 3.98. Our model without adversarial learning i.e. RHYME-LM, achieves a test set NLL of 3.97. DEEP-SPEARE reports a test set NLL of 4.38. Note that our language model is hierarchical while DEEP-SPEARE has a linear model. The NLL for RHYME-LM and RHYME-GAN are very similar, though RHYME-GAN gets much better sampling efficiency scores than RHYME-LM.

Our generator implementation is largely based on that of Lau et al. (2018). The main difference is that we first generate all the line-ending words and then condition on them to generate the remaining words. The change was made to make it more amenable to our proposed discriminator. However, our hierarchical language model (RHYME-LM) performs worse than DEEP-SPEARE as per sampling efficiency. Therefore, structured discriminator is the driving factor behind the observed improvement with RHYME-GAN. However, committing to the ending words of all lines before completing preceding lines can be a limitation, and addressing it is a possible future direction.

5.3.3 Analyzing Learned Discriminator

We probe the the word representations $g(.)$ to check if rhyming words are close-by in the learned manifold. We consider all pairs of words among the ending words in a quatrain/limerick, and label each pair to be rhyming or non-rhyming based on previously stated definition of rhyming. If the cosine similarity score between the representations of pairs of words is above a certain threshold, we predict that word pair as rhyming, else it is predicted as non-rhyming. We report F1 scores for the binary classification setup of predicting word-pairs to be rhyming or not. We consider some additional baselines: **RHYM-EM** (Reddy and Knight, 2011) uses latent variables to model rhyming schemes, and train parameters using EM. **GRAPHEME-K** baselines predict a word pair as rhyming only if the last $K = \{1,2,3\}$ characters of the two words are same.

For SONNET data, we observe that RHYME-GAN obtains a F1 score of 0.90 (Table 5.3) on the test split (threshold chosen to maximize f1 on dev split). We repeat the above analysis on the LIMERICK dataset and observe an F1 of 0.92 for RHYME-GAN. DEEP-SPEARE model reports F1 of 0.91 on SONNET. As stated earlier, DEEP-SPEARE'S model is not amenable to LIMERICK - we do compare though with the max-margin classifier in DEEP-SPEARE model trained on LIMERICK which gets F1 score of 0.81. The scores are understandably lower since the AABBA pattern in limericks is not amenable to SONNET specific assumptions made in DEEP-SPEARE model. On the other hand, RHYME-GAN achieves high F1 scores for both the datasets without incorporating any domain specific rhyming pattern information.

RHYME-GAN performs much better than RHYM-EM and GRAPHEME-K baselines. RHYM-EM does not perform well - probably because it operates at word-level and fails to generalize. Note that **RHYME-GAN-NS** gets F1 score of 0.85 in case of SONNET dataset and 0.87 for LIMERICK. These values are lower than corresponding scores for RHYME-GAN, demonstrating that the proposed structure in the discriminator was useful in learning the notion of rhyming.

Model	SONNET	LIMERICK
GRAPHEME-1	0.71	0.79
GRAPHEME-2	0.78	0.79
GRAPHEME-3	0.69	0.67
RHYM-EM	0.71	0.73
DEEP-SPEARE/MAX-MARGIN	0.91	0.81
RHYME-GAN-NS	0.85	0.87
RHYME-GAN	**0.90**	**0.92**

Table 5.3: Rhyming probe: We use the cosine similarity score of the learned representations to predict a word pair as rhyming or not, and report F1 score for this classification task. RHYM-EM is an unsupervised rhyming pattern discovery method. GRAPHEME-K baselines predict based on exact match of last k characters.

5.3.4 Human Evaluations

Following prior work (Lau et al., 2018), we requested human annotators to identify the human-written poem when presented with two samples at a time - a quatrain from the Sonnet corpus and a machine-generated quatrain, and report the annotator accuracy on this task. Note that a lower accuracy value is favorable as it signifies higher quality of machine-generated samples. Using 150 valid samples (i.e. samples belonging to one of the allowed rhyming patterns), we observe 56.0% annotator accuracy for RHYME-GAN, and 53.3% for DEEP-SPEARE – indicating that the post-rejection sampling outputs from the two methods are of comparable quality (the difference in annotator accuracy is not statistically significant as per McNemar's test under $p < 0.05$). If we use pre-rejection samples, we observe 60.0% annotator accuracy for RHYME-GAN, and 81.3% for DEEP-SPEARE (the difference being statistically significant as per McNemar's test under $p < 0.05$) – indicating that unfiltered samples from RHYME-GAN are of higher quality compared to DEEP-SPEARE.

5.4 Related Work

Early works on poetry generation mostly used rule based methods (Gervás, 2000; Wu et al., 2009; Oliveira, 2017). More recently, neural models for poetry generation have been proposed (Zhang and Lapata, 2014; Ghazvininejad et al., 2016, 2017; Hopkins and Kiela, 2017; Lau et al., 2018; Liu et al., 2019b). (Yan et al., 2013) retrieve high ranking sentences for a given user query, and repeatedly swap words to satisfy poetry constraints. (Ghazvininejad et al., 2018b) worked on poetry translation using an unconstrained machine translation model and separately learned Finite State Automata for enforcing rhythm and rhyme. Similar to rhyming and rhythm patterns in poetry, certain types of musical compositions showcase rhythm and repetition patterns, and some prior works model such patterns in music generation (Walder and Kim, 2018). Generative adversarial learning (Goodfellow et al., 2014) for text generation has been used in prior works (Fedus et al., 2018; Wang et al., 2018, 2019b; Rao and Daumé III, 2019), though has not been explored with regard to the similarity structure proposed in this paper.

5.5 Application to Modeling Repetition in Music Generation

Musical compositions often demonstrate repetitions (Pareyon, 2011; Walder and Kim, 2018), in terms of patterns related to rhythm, pitch, and other musical properties. Some prior works focus on modeling long term structures in music generation (Eck and Schmidhuber, 2002; Huang et al., 2018; Roberts et al., 2018). However, there are only few works on explicitly representing self-repetition (Walder and Kim, 2018). In this paper, we propose **SSMGAN** (Figure 6.2) - a generative adversarial network (Goodfellow et al., 2014) for learning a neural model to generate monophonic compositions with rich self-repetition structures by feeding a measure-level **s**elf-**s**imilarity **m**atrix representation to a convolutional discriminator, which can be more informative than taking localized decisions with self-attention. Instead of

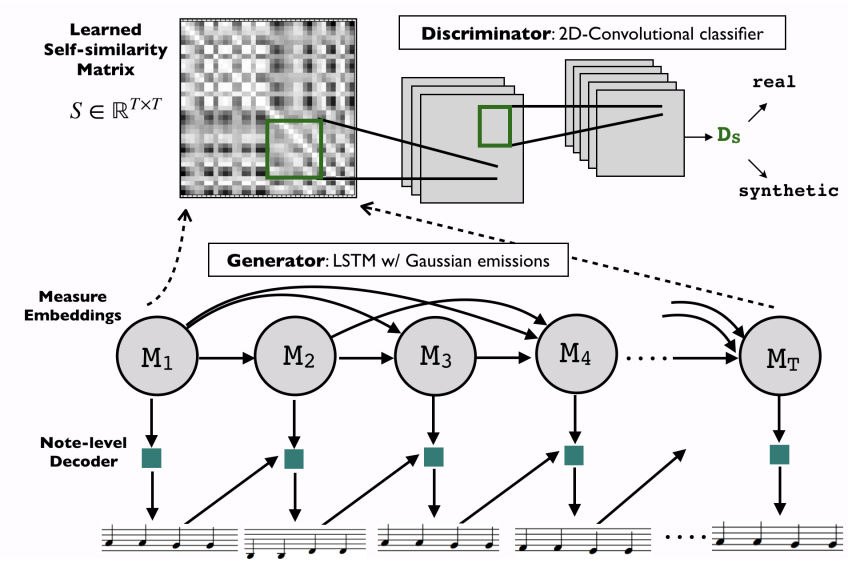


Figure 5.2: SSMGAN model: We sample a sequence of measure embeddings from the generator, which are decoded to obtain sequences of notes. One of the discriminators operates on a self-similarity matrix obtained via cosine similarity between pairs of measure embeddings. The proposed architecture encourages the model to identify useful notions of similarity between measures, and identify overall self-repetition patterns from data.

explicitly defining the notion of similarity between two measures, we propose to encode a measure - a sequence of notes - into a continuous representation, and compute a self-similarity matrix using pair-wise cosine similarity of measure representations/embeddings in the composition.

Prior works have often chosen to characterize repetition in terms of rhythmic or other manually defined musical properties, and edit distances at note level (Walder and Kim, 2018). However, musical similarity might go beyond such simple formulations (Flexer et al., 2006; Prockup et al., 2015). By representing measures in a continuous space, our model can learn more complex notions of similarities between measure. Additionally, we feed a self-similarity matrix to a discriminator D_S, which learns to identify repetition structures in existing compositions using multiple layers of 2D convolution neural networks. Moreover, since we represent measures in a continuous space, loss from D_S is fully differentiable with respect to the measure representations.

5.5.1 Methodology

Measure and Self-repetition representations: We propose to encode a measure N_i consisting of a sequence of notes $N_i^1, N_i^2, .., N_i^{|N_i|}$ to a low dimensional embedding M_i. We train a variational auto-encoder (Kingma and Welling, 2013) at measure level using a LSTM encoder, denoting the last hidden state of encoder as corresponding measure embedding M_i. However, instead of having a decoder operate on each measure embedding individually, we propose a decoder which has access to the last hidden state of previous measure's decoder (Figure 5.2). The proposed decoder can lead to smoother transition across measure boundaries. We define **self-similarity matrix** $\mathbf{S_M} \in R^{T*T}$ such that S_{ij} is the cosine similarity score between pair of measures N_i and N_j, while T is the number of measures in the composition.

GAN formulation Generative Adversarial Networks (GANs) employ two types of networks - *generator* and *discriminator*, such that discriminator is trained to identify generated examples from training examples, and the generator is trained to *fool* the discriminator.

Neural Generation Model We first sample a sequence of measure embeddings $M_1, .., M_T$ such that M_i depends on all $M_{<i}$ (Figure 5.2). We sample M_i from a Gaussian distribution whose mean and variance are computed by feeding the output h_i of the LSTM to two feed-forward networks. We use re-sampling trick to train the model end-to-end. Thereafter, the decoder (same as the one used during training measure representations) decodes the sampled sequence of measure embeddings into the final sequences of notes.

Discriminators: We employ two discriminators operating on 1) Self-similarity matrix 2) Sequence of measures.

$\mathbf{D_S}$: The discriminator $\mathbf{D_S}(S)$ uses a multi-layer convolutional encoder to encode a self similarity matrix S and is trained to distinguish S of a generated composition from that of a training data composition.

$\mathbf{D_L}$: We consider a convolutional neural network discriminator $\mathbf{D_L}$ which looks at windows of K measure embeddings at a time. We encode the sequence using a LSTM, followed by a linear layer and a sigmoid to model this binary classification discriminator. We experiment with multiple values of K.

Training: The generator and the discriminators are trained simultaneously. We run the encoder on training data compositions to obtain a sequence of *real* measure embeddings, while we sample from the generator to get *synthetic* compositions. We note that training the full model end-to-end can become problematic as the measure encoder can work with the generator to *fool* the structure discriminator at the cost of generating good measure representations. So instead we pre-train and then freeze the parameters of the measure encoder-decoder. Thereafter, we use GAN framework to train the LSTM Gaussian model while keeping the measure encoder decoder fixed.

5.5.2 Experiments with Music Generation

We work with Nottingham dataset (**GOLD**) (Shlien; Boulanger-Lewandowski et al., 2012) which is a collection of 1200 British and American folk tunes, with over 7 hours of music with a total of over 176K notes. We identify the measure boundaries in every composition. Following prior works, we perform listening tests with human annotators on Amazon Mechanical Turk. Annotators rate the generated samples on overall quality **O** on a 1-5 Likert scale, (with 5 being the most favorable score), and a binary yes/no question about presence of repetition **R** (Table 5.4).

Model	R (% yes)	Overall (O)
SSMGAN	63%	4.28
SSMGAN (-D_S)	26%	3.84
NOTE	27%	4.02
GOLD	64%	4.64

Table 5.4: Human evaluation results with 108 samples of each method on 1) self-repetition (R) 2) overall musical quality for our method SSMGAN, a note level LSTM baseline (NOTE), and samples from Nottingham data (GOLD). SSMGAN (-D_S) denotes our method without D_S discriminator.

Measure Embeddings and Self-Similarity: We use LSTM cells with hidden size of 128 as measure encoder and decoder, with note embedding size of 128. As discussed, during pre-training phase we only train measure encoder-decoder, and observe 1.137 note-level perplexity on test split of the Nottingham data. Additionally, we observe similar measures are close-by in embedding space. (Some relevant visualizations in Appendix B). We observe that our model learns to generate sequences with rich repetition structures (Some relevant visualizations in Appendix A). Moreover, we observe that cosine similarity between pairs of measure embeddings is correlated with a note level edit distance of pitch as well as rhythm between measures (Pearson correlation coefficients of 0.40 and 0.35 respectively), demonstrating that proposed self-similarity matrices encode repetition in terms of meaningful musical properties.

Other Related Works: Dong et al. (Dong et al., 2018) and Yang et al. (Yang et al., 2017) propose GAN based methods for music generation, with latter using a 2D convolutional discriminator on sequence of generated bars. Widmer et al (Widmer et al., 2018) use a convolutional restricted boltzmann machine to generate music while imposing a given repetition structure of a piece. In contrast to such earlier works, we have proposed to encode measures in a low dimensional embeddings space, and use a discriminator on self-similarity matrix - enabling our model to automatically learn useful notions of similarity between measures, and identify meaningful self-repetition patterns from data.

6 Latent Discrete Generation Plans for Controllable and Coherent Generation

Maintaining long-term narrative flow and consistency are important concerns when aiming to generate a plausible story (Porteous and Cavazza, 2009; Hou et al., 2019). Prior work on long form text generation has focused on generating consistent narratives via outlines using keywords or key phrases (Xu et al., 2018; Yao et al., 2019), event-based representations (Riedl and Young, 2010; Martin et al., 2018; Fan et al., 2019), plot graphs (Li et al., 2013) or a sentence representing theme (Chen et al., 2019b). However, many these approaches have used heuristics or off-the-shelf models to first tag data with the desired type of plan, and then train generation models in a supervised fashion. In (Jhamtani and Berg-Kirkpatrick, 2020), we propose a deep latent variable model that treats the generation plan as discrete latent variables, and learns them from data.

6.1 Introduction

Maintaining long-term narrative flow and consistency are important concerns when aiming to generate a plausible story (Porteous and Cavazza, 2009; Hou et al., 2019). Prior work on narrative text generation has focused on generating consistent stories via story outlines using keywords or key phrases (Xu et al., 2018; Yao et al., 2019), event-based representations (Riedl and Young, 2010; Martin et al., 2018; Fan et al., 2019), plot graphs (Li et al., 2013) or a sentence representing theme (Chen et al., 2019b).

(Yao et al., 2019) note that compared to specific event based representations, using keywords to form the outline is more generalizable and widely applicable. In this work, we consider a sequence of anchor words as a means to model story outlines. For example, in Figure 6.1, given a story title 'Winning the Race', our model first predicts a sequence of anchor words which represents a high level story plan. Thereafter, a decoder conditions on the title and generated sequence of anchor words to generate the final story. We assume an alignment between the anchor words and the story sentences – the i^{th} anchor word corresponds to the i^{th} sentence in the story.

However, stories do not naturally occur with a tagged set of such anchor words or keywords. Many prior works use off the shelf tools to first label stories with plan outlines, thus using external supervision for learning plot structures. For example, (Yao et al., 2019) use the RAKE heuristic (Rose et al., 2010) to first identify the most important keyword in each sentence, and then use this to train a model in a supervised fashion. This approach leads to improved coherency and control, but creates a reliance on such heuristics and does not jointly learn anchor words along with the generator.

Inspired by prior work indicating that anchor words can effectively capture and control high-level generation structure, we investigate to what extent high-level control can be learned in a fully unsupervised fashion, directly from natural story data. We design a hierarchical latent variable model (Figure 6.2) that induces sequences of anchor words that explain observed stories, while at the same time learning to generate entire stories by first generating anchor sequences. For training, we use amortized variational learning (Kingma and Welling, 2014b), where an inference network is used to approximate the posterior on anchor sequences.

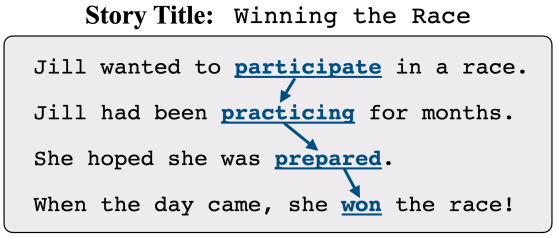


Figure 6.1: Our aim is to generate a story given a title. We propose models which first generate a high level story plan realized via a sequence of anchor words.

At test time, given a title, we first sample a sequence of anchor words using the prior model conditioned on only the title, and then generate the actual story using the decoder conditioning only on the title and the sampled anchor words.

To induce a useful latent generation plan and to effectively condition on a sampled plan, we propose a constrained story decoder and constrained inference network. Specifically, our constrained decoder begins a story sentence by deterministic *copying* the corresponding anchor word, and then generates words to the left and then to the right (Figure 6.3). For this decoder, the corresponding true posterior on anchor words is sparse: the anchor word must be chosen from the observed sentence. Thus, we constrain the output vocabulary of the corresponding inference network to the words of the input sentence. We observe that the proposed constrained inference network does not suffer from mode collapse, leading to models which can effectively learn useful anchor words. Further, we also contrast this approach with a model whose decoder is not constrained to use each anchor word in each sentence. The true posterior in this case is over the full vocabulary. We conduct experiments with both constrained and unconstrained decoders and inference networks, and find that the best results are achieved through the combination of an unconstrained decoder with a constrained inference network – indicating, perhaps, that while it is more effective to use flexible models, using a constrained inference network can add a useful inductive bias, leading the model to mimic the constraint of the inference network.

We experiment with two English story datasets, and observe that our best models achieve favorable scores relative to several baselines when evaluated on perplexity, diversity, coherency, and controllable story generation as per various automatic and human evaluations.

Finally, we note that our modelling approach for story generation has an interesting connection with work that treats text as a latent variable in deep generative models (Miao and Blunsom, 2016; Wen et al., 2017). We treat a latent sequence of anchor words as a form of hierarchical control over generated outputs, while related work treats the latent sequence itself as sequential text that is the output of the model.

6.2 Model

Our goal is to generate a story x, consisting of multiple sentences $x_1, x_2, ..x_K$, given a title t. Our model's generative process is depicted in Figure 6.2 and operates as follows: First, a sequence of anchor words representing a generation plan is sampled from an auto-regressive prior conditioned on the title. Next, for each anchor word, a sentence is generated conditioned on the anchor words and previously generated sentences using a decoder. During training, the sequence of anchor words is unobserved and treated as a latent variable. As described in more detail later, we will explore several choices of decoder – those that treat anchor words as an explicit token in the sentence to be generated, generating surrounding context to the left and right, and those that simply treat the anchor words as conditioning information. In the former case, the posterior must be sparse. In the latter case, our choice of variational learning scheme will bias (but not force) the model to use anchor words in output story sentences. We shall refer to our proposed model as Latent Anchor Plan model (**LAP**).

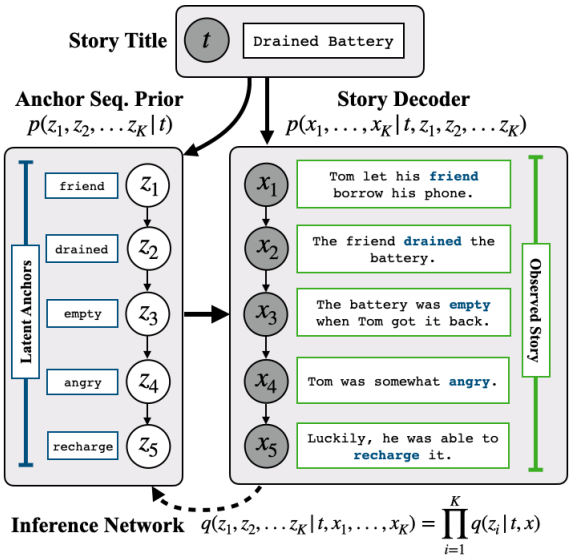


Figure 6.2: Model Overview: We consider multi-sentence text generation via a latent generation plan realized through a sequence of anchor words with one word per sentence. [We show sequence models with first-order Markov assumption for simplicity, even though all sequence models in our approach are auto-regressive with full context.]

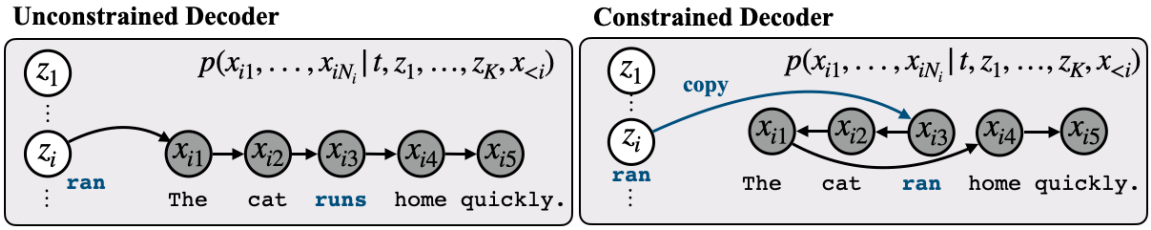


Figure 6.3: Simplified demonstration of generation of a sentence conditioned on anchor words and preceding sentences for the two types of decoders: (1) Unconstrained decoder is based on the story generation model of (Yao et al., 2019), which may or may not use the corresponding anchor word. (2) Constrained decoder is forced to use anchoring words in corresponding sentences, generating words to the left and then to the right of an anchor word. [Again, we show sequence models with a first-order Markov assumption for simplicity, even though all sequence models are auto-regressive with full context.]

6.2.1 Anchor Sequence Prior

We model the sequence of anchor words representing the generation plan via a sequence of discrete random variables $z_1, z_2, .., z_K$. Since our aim is to induce latent plans, we assume z are unobserved. We consider an auto-regressive prior model $p_\phi(z|t) = \prod_i p_\phi(z_i|z_{<i}, t)$ where each anchor word is conditioned on preceding anchor words and the title t.

6.2.2 Story Decoder

Our decoder $p_\theta(x|t,z)$ generates a story given the title t and anchor words z. As mentioned earlier, z_i is aligned to the sentence x_i. We consider two decoders: (1) an unconstrained decoder which is not bound to use z_i in x_i, and (2) a constrained decoder which assumes z_i is present in x_i, and constructs words to the left and then to the right of the anchor word z_i.

Unconstrained Decoder: Our unconstrained decoder is based on (Yao et al., 2019)'s decoder which does

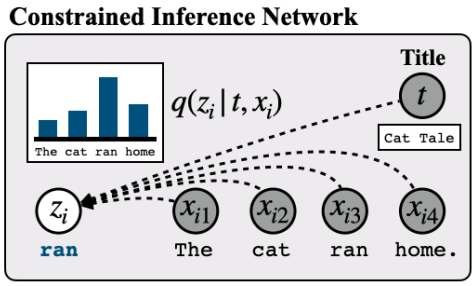


Figure 6.4: Constrained Inference Network: Proposed model is trained through amortized variational learning using an inference network. One of the proposed models is trained using a constrained inference network which assigns non-zero probability to only the words present in corresponding sentences.

not use any explicit alignment of anchor words to corresponding sentences (Figure 6.3). The decoder is fed the title and the anchor words appended together, and is trained to generate the multi-sentence text. The decoder is not bound to use the anchor word z_i for x_i, but may have incentive to do so depending on the training objective, as discussed later. At the same time, the unconstrained decoder has higher flexibility and can skip using an anchor word if it doesn't fit with the preceding context.

Constrained Decoder: We consider a constrained decoder that always uses z_i while generating x_i. This is achieved by first copying z_i, then generating to the left until the sentence start, and then to the right. Such a decoder is bound to use the corresponding anchor word by design, and will potentially demonstrate higher control of the anchor words on the story.

Our decoder architecture follows from (Yao et al., 2019), who use a 3-layer LSTM recurrent model. Our final reported model uses 1000 dimensional hidden layer, with tied input and output word embeddings. Moreover, the prior model shares the underlying LSTM modules with the decoder. Since our goal is to induce a latent discrete plan and compare with keyword tagging based methods, we stick to the same choice of decoder as in prior work.

6.3 Learning and Inference

Our goal is to maximize the log likelihood of the stories conditioned on the corresponding titles. Since z is unobserved at training, we must marginalize over all possible values of z.

$$\sum_{t,x \in \mathscr{D}} \log p(x|t) = \sum_{t,x \in \mathscr{D}} \log \mathbb{E}_{z \sim p_\phi(z|t)}[p_\theta(x|t,z)]$$

, $\mathscr{D}$ represents the dataset of titles and corresponding stories. Since it is infeasible to compute the exact marginal stated above, we use amortized variational learning by introducing an inference network q_γ, and train the model to maximize the following evidence lower-bound (ELBO):

$$\underbrace{\mathbb{E}_{z \sim q_\gamma(z|x,t)}[\log p_\theta(x|z,t)]}_{\text{Reconstruction}} - \underbrace{\text{KL}(q_\gamma(z|x,t)||p_\phi(z|t))}_{\text{KL-term}}$$

We shall refer to the first term as the reconstruction term and the second term as the KL-term.

We make a mean-field assumption in the posterior approximation on z as follows: $q(z|x,t) = \prod_{i=1}^{K} q(z_i|x_i,t)$. Note that $p(z|t)$ is auto-regressive, and thus it is intractable to exactly compute the KL term. We resort to Monte Carlo sampling to approximate the ELBO by drawing samples from inference network; though we will perform this differently for the KL term and the reconstruction term (more details in Section 6.3.2).

6.3.1 Inference Network and Posterior Sparsity

Constrained Inference Network With the constrained decoder discussed earlier, the true posterior is sparse – so making the inference net also sparse would help the learning procedure better approximate the true posterior (Figure 6.4). To leverage this observation, we constrain the inference network's output distribution to have non-zero probabilities only on the tokens present in the corresponding sentence:

$$\begin{aligned} q(z_i = v|x_i,t) &= 0 \text{ if } v \notin x_i \\ &\propto \exp(s_v) \text{ otherwise} \end{aligned}$$

Here, s_v is the logit output for the token v produced by the inference network. Our constrained inference network is a BiLSTM model which generates an encoding h_j for j^{th} token in a story sentence. A linear layer transforms h_j to a score s_j. Finally, for sentence x_i, we compute a softmax over the scores of words in x_i to obtain $q(z_i|x)$.

Unconstrained Inference Network We also consider an unconstrained inference network which does not constrain the inference network's output – i.e. the output distribution is over the entire vocabulary. We use a LSTM model to encode each sentence, obtain the last word hidden state, and then finally employ a linear layer to transform it to the vocabulary size.

When the decoder is not constrained, it may be interesting to compare the choice of inference network. Using the constrained inference net with the unconstrained decoder will bias the decoder to use the anchor words in the aligned sentences – the model is not required to do this, but variational learning will pull the inference network and true model posterior towards each other (i.e. the ELBO objective pressures them to agree). Thus, if the inference net is constrained, but the decoder is not, learning will try to find a weakly constrained decoder to match the approximate posterior.

6.3.2 Optimization

Reconstruction term: As mentioned earlier, we draw samples from the inference network to approximate the reconstruction term. The decoder parameters θ can be trained directly through back-propagation to minimize the approximate reconstruction loss. However, since z is discrete, we use the REINFORCE (Williams, 1992) algorithm to train the parameters γ of the inference network $q(z|x,t)$. Following prior work (Xu et al., 2015), we use an entropy regularizer term and a moving average baseline to reduce the variance of the resulting gradient estimator for inference network parameters γ.

KL term: Note that the KL term can be simplified as follows:

$$\begin{aligned} \mathrm{KL}(q_\gamma(z)||p_\phi(z)) = \mathrm{KL}(q_\gamma(z_1)||p_\phi(z_1))+ \\ \mathbb{E}_{z_1\sim q_\gamma(z_1)}[\mathrm{KL}(q_\gamma(z_2)||p_\phi(z_2|z_1))+ \\ \mathbb{E}_{z_2\sim q_\gamma(z_2)}[\mathrm{KL}(q_\gamma(z_3)||p_\phi(z_3|z_{<3})] + \ldots]]] \end{aligned}$$

We draw samples of z from $q(z)$ to approximate the KL term.

KL term for the constrained inference network: For the constrained inference network, we have a sparse approximate posterior. Given the fact that typical sentences in our dataset are 5-20 words in length, it is computationally easy to exactly compute individual $\mathrm{KL}(q(z_i)||p(z_i|z_{<i}))$ terms by summing over the tokens in x_i instead of the whole vocabulary. This is still an approximation to the full KL term

Method	Inference N/W	Decoder	PPL↓	NLL↓		DIV↑		DIV-B↓
			test	test	dev	plan	story	story
No Plan								
ROC-DATA	NA	NA	NA	NA	NA	NA	9.01	0.23
NOPLAN-LM	NA	Unconstrained	**17.3**	154.0	160.7	NA	7.70	0.50
With Plan								
SUPERVPLAN	NA [1]	Unconstrained	≤28.3	≤180.3	≤187.6	8.71	7.74	0.49
LAP-CINF-UDEV	Constrained	Unconstrained	≤**21.3**	≤168.9	≤176.5	9.24	**7.93**	**0.45**
LAP other variants:								
LAP-CINF-CDEC	Constrained	Constrained	≤**20.9**	≤166.9	≤174.1	9.24	**7.98**	**0.44**
LAP-UINF-UDEC	Unconstrained	Unconstrained	≤17.5	≤154.2	≤160.9	0.01	7.67	0.52

Table 6.1: Automated metrics: We report Negative Log Likelihood (NLL), perplexity (PPL) (computed using importance weighted samples for models with latent variables), and diversity (DIV and DIV-B). LAP-CINF-UDEV performs better than SUPERVPLAN on perplexity as well as diversity. We also experiment with two other variants for LAP. LAP-UINF-UDEC, which does not constrain the inference network, suffers from posterior collapse. LAP-CINF-CDEC, which uses the constrained decoder, achieves perplexity and diversity results that are comparable to LAP-CINF-UDEV.

since we cannot feasibly sum over the context.

$$\begin{aligned} \mathrm{KL}(q(z_i)||p(z_i|z_{<i}) &= \sum_{z_i \in V} q(z_i) \log q(z_i)/p(z_i) \\ &= \sum_{z_i \in \mathbf{x_i}} q(z_i) \log q(z_i)/p(z_i) \end{aligned}$$

Thus, for the constrained inference network, KL computation now proceeds as follows: we first compute $\mathrm{KL}(q(z_1)||p(z_1))$ as described above. Then we sample $z1 \sim q(z_1)$, and compute $\mathrm{KL}(q(z_2)||p(z_2|z_{<1}))$, and so on – we still need to use samples, but can exactly compute each of the K individual KL terms, one at each of the K steps in the plan, similar to the approach of (Yang et al., 2018b). We observe that the constrained inference network leads to lower variance in the KL term approximation, thereby leading to more stable gradients.

Pretraining: Pretraining the inference network in an autoencoder setup has been found useful for VAE training (Li et al., 2019a). We pretrain the inference network in an autoencoder setup where the decoder reconstructs the corresponding sentences (rather than whole story). Thereafter, we train the decoder and prior keeping the inference network fixed. Finally we perform the full training with all parameters being updated. We observe that pretraining through this procedure leads to more stable training.

6.4 Experiments

We evaluate and report generation quality of various models using automatic metrics for fluency and diversity, as well as human evaluations for coherence of story and relevance to title. We also analyze the ability of anchor words to control the generated story, and highlight comparisons between various choices of inference networks and decoders.

6.4.1 Dataset

We use a subset of the ROC-stories corpus (ROC-DATA) (Mostafazadeh et al., 2016a) used earlier by (Yao et al., 2019). (Yao et al., 2019) had chosen a subset of the original ROC corpus in order to select

only those stories which are accompanied by a title. The train, validation and test splits consist of 78529, 9816, and 9816 stories respectively. Most of the data consist of five sentence stories. Additionally, we experiment with the visual story dataset (only the text portion), which we discuss in more detail in Section 6.4.8.

6.4.2 Methods

NOPLAN-LM: This baseline does not consider any story generation plan and conditions only on the title. We use the same 3-layer LSTM as in the proposed model.

SUPERVPLAN: This baseline is based on the work of (Yao et al., 2019) which utilizes RAKE-tagged keywords as observed anchor words. The model is trained to predict the the observed anchor words and the story given the title. We can view this baseline as a latent variable model that was trained using RAKE keywords as the output of a deterministic inference network.

LAP: (1) We will refer to our model with the constrained inference network and unconstrained decoder as **LAP-CINF-UDEV**. (2) **LAP-UINF-UDEC** uses the unconstrained inference network and unconstrained decoder. (3) **LAP-CINF-CDEC** uses the constrained inference network with the constrained decoder. We found that the model with constrained decoder and unconstrained encoder performed poorly during training, and so we do not include it in experiments.

Decoding procedure: For all the methods, we generate samples with top-p sampling (Holtzman et al., 2020b) with $p = 0.6$ at the time of story generation. Unless otherwise stated, the same decoding procedure is followed for the evaluations of diversity, story quality, and controllable generation discussed below. Later in the analysis we discuss the effect of changing the parameter p on some of the evaluation metrics.

6.4.3 Perplexity

For the models with latent generation plans, we use importance weighting (IW) (Burda et al., 2016) (with 20 samples) to estimate perplexity scores since (IW) has been shown to provide a tighter bound than ELBO for evaluation purposes (Li et al., 2019a). For the baseline, SUPERVPLAN, we also evaluate its marginal likelihood for comparison with our model. To do this, we separately train an inference network (with the same architecture as that used by the LAP-CINF-UDEV model) to approximate the posterior on anchor words for the trained SUPERVPLAN (by keeping the trained model parameters fixed). This approximate posterior is then used to compute an upper bound on NLL and perplexity.

The proposed model LAP-CINF-UDEV performs better than the baseline SUPERVPLAN, which uses separately tagged generation plans (Table 6.1). However, the proposed method's perplexity is close to that of NOPLAN-LM, which does not consider any generation plan. This is not uncommon for deep latent variable models – since their held-out likelihood is intractable, and most approximations yield upper bounds on perplexity, their reported perplexity is always pessimistic. Among LAP variants, we observe that LAP-UINF-UDEC suffers from posterior collapses, and behaves similarly to NOPLAN-LM since the latent variables z are not informative or useful. Finally, LAP-CINF-CDEC performs similar on likelihood evaluations compared to the LAP-CINF-UDEV model with an unconstrained decoder .

6.4.4 Diversity

We generate story samples for all the titles in the test split. We employ two evaluations to report diversity in the generated outputs:

DIV We compute the geometric mean of empirical unigram, bigram, and trigram distribution entropy

LAP-CINF-UDEV vs Method M	Coherence win-tie-loss	Title-Fidelity win-tie-loss
M=SUPERVPLAN	0.31 0.37 0.32	**0.39** 0.27 0.34
M=NOPLAN-LM	**0.36** 0.35 0.29 †	0.33 0.37 0.30
M=ROC-DATA	0.12 0.08 0.80 †	0.08 0.15 0.77 †

Table 6.2: Human preference evaluations when pitting various methods against LAP-CINF-UDEV (i.e. preference for LAP-CINF-UDEV is reported under *win*). Compared to SUPERVPLAN, LAP-CINF-UDEV performs better on fidelity to title and similar on coherence. Loss vs win judgements marked with † are statistically significant under bootstrap test ($p < 0.05$) considering 1000 subsets each of size 400.

from the generated set of stories (Jhamtani et al., 2018). For methods which use generation plans, we also compute this diversity metric on anchor word sequences. Table 6.1 shows the results for various models. LAP-CINF-UDEV performs better than SUPERVPLAN, achieving higher diversity for both story and plans. Among the LAP variants, using the non-constrained inference network (LAP-UINF-UDEC) leads to worse results on story diversity, and fares poorly in plan diversity (due to posterior collapse). LAP-CINF-CDEC again performs similarly to LAP-CINF-UDEV.

DIV-B We also report inter-story BLEU4 scores (Zhu et al., 2018). We compute samples from various methods for 1000 titles in the test split. For each generated story, the remaining 999 are treated as references. Thus, lower values indicate higher diversity in the generated stories. Table 6.1 shows the results. LAP-CINF-UDEV performs better than SUPERVPLAN, though is still far from the values for human written stories in the ROC dataset itself.

6.4.5 Human Evaluations

We conduct human evaluations on Amazon Mechanical Turk to evaluate the quality of generated stories given the title. We evaluate the story samples with respect to: (1) coherence, which measures the logical and coherent narrative flow in a story, and (2) fidelity to title, which measures the degree to which the story is relevant to the given title. Given two stories from two different methods, we request human annotators to provide their preference (or mark as tie).

In order to ensure the quality of human evaluations, we restrict the annotation task to annotators from Anglophone countries, and limited to workers with more than 90% HIT (Human Intelligence Task) acceptance rates. We randomize the order of presented stories to avoid positional bias effects. Additionally, we added two 'check' data points with each HIT. More specifically, to construct a 'check', we pick a random story from the development set, and then prepare a 'decoy' story by replacing three lines of the story with that of another randomly chosen story. The HITs where annotators marked the 'decoy' as the preferred story relative to the unaltered story with respect to either coherence or fidelity for either of the two check data points are skipped. These skipped HITs are then re-annotated.

Based on the automated metrics and manual qualitative inspection, we pick LAP-CINF-UDEV as the best configuration among all the variants of our model for human evaluation. We randomly selected 200 titles from the test split, generate samples from all the methods under consideration, and evaluate each method against LAP-CINF-UDEV. Each comparison is rated by three different annotators leading to a total of 600 judgements per pair. Table 6.2 shows the results. We observe that on average, annotators found LAP-CINF-UDEV outputs similar or better on coherence and fidelity compared to the baselines. LAP-CINF-UDEV is judged better than NOPLAN-LM on coherence, perhaps because having a plan provides a rough sketch of the story leading to more coherent outputs. Compared to SUPERVPLAN, outputs from the proposed method LAP-CINF-UDEV are judged similar in quality in terms of coherence but better in terms of fidelity to title, perhaps because the ELBO objective encourages the inference

[1] We retrofit an inference network to a trained SUPERVPLAN to approximate PPL and NLL for evaluation purposes only. Training the SUPERVPLAN model does not involve any inference network.

Method	CTRL
SUPERVPLAN	38.8%
LAP-CINF-UDEV	**72.9**%
LAP variants:	
LAP-CINF-CDEC	100.0%
LAP-UINF-UDEC	0.0%

Table 6.3: We evaluate models for the extent to which the story follows the generation plan by evaluating the fraction of anchor words used in corresponding sentences (CTRL). LAP-CINF-UDEV demonstrates better control compared to SUPERVPLAN. Model with LAP-UINF-UDEC inference network collapses, while LAP-CINF-CDEC demonstrates perfect control due to the nature of the decoder.

TITLE:	the exam
ANCHOR WORDS:	midterm knew nervous performed passed
STORY:	I had a big geometry exam today. I knew that i would have to do it. I was nervous. I had not performed since i was a little girl. I passed out.
TITLE:	the new bed
ANCHOR WORDS:	alex new store amazing glad
STORY:	Alex was trying to find a new bed. She needed a new one. She went to the store to get one. She found a amazing one. She was glad she bought it.
TITLE:	picnic
ANCHOR WORDS:	goes fancy least eating leave
STORY:	Last week i visited my friends to the park. It was at the fancy park. They had to eat the food and water. I had a great time eating. I had to leave.

Table 6.4: Generated samples from the proposed method LAP-CINF-UDEV. We observe that samples from the proposed method demonstrate fidelity to the title, better follow the sampled plan of anchor word sequences, and are in aggregate more coherent than baselines which do not consider a generation plan.

network to pick anchor words which can be more easily predicted from the title by the prior model, leading to better title fidelity. We show example generated samples from LAP-CINF-UDEV in Table 6.4. More examples and qualitative analysis can be found in the Appendix.

We found LAP-CINF-CDEC outputs to be slightly worse than LAP-CINF-UDEV and SUPERVPLAN outputs on coherency. Compared to LAP-CINF-UDEV, the constrained decoder achieves slightly better scores for perplexity and diversity (Table 6.1) and control (next subsection), but suffers on overall coherency. This behavior is likely due to the reduced flexibility of the model architecture (an example output is provided in Table 6.5). In contrast, the non-constrained decoder achieves a favorable balance between control and coherency. This highlights an interesting balance between the generation plan and the degree to which the decoder must follow the plan.

6.4.6 Controllable Generation

We evaluate models for the extent to which the story follows the generation plan. To evaluate this, we draw one story sample per title in the test split, and report the fraction of anchor words which were used in corresponding sentences (**CTRL**). LAP-CINF-UDEV (73%) fares much better than SUPERVPLAN (39%) (Table 6.3). We note that in some outputs from LAP-CINF-UDEV, even though the exact anchor word was not used, we observe semantically equivalent concepts being used – for example, for the sampled anchor word 'dismay', the generated story sentence was: 'She then realized she wasn't able to attempt it'.

We also analyze CTRL and DIV-B values when sampling with different values of parameter p in top-p sampling. As we increase p, we observe higher diversity in samples, along with lower scores for

LAP-CINF-CDEC	**TITLE:**	the exam
	ANCHOR WORDS:	failing nervous tried test shocked
	STORY:	Jessica was failing her math class. She was nervous to try to take the test. She tried to help. She took the test. She was shocked and confident
LAP-UINF-UDEC	**TITLE:**	the new bed
	ANCHOR WORDS:	forms forms forms forms forms
	STORY:	Jane was about to get a new bed. She had been trying to catch a few new sheets. She decided to get a new bed. She looked at the new sheets. It was the right choice.

Table 6.5: Generated samples from LAP-CINF-CDEC and LAP-UINF-UDEC variants of the proposed model class. We observe that when using the constrained decoder variant, story outputs lack coherence more often than when using the unconstrained decoder, though they demonstrate better control by design. The LAP-UINF-UDEC variant suffers from posterior collapse, leading to a generic anchor word sequence, and often produces stories that lack overall structure.

p	LAP-CINF-UDEV		LAP-CINF-CDEC		SUPERVPLAN	
	CTRL	DIV-B	CTRL	DIV-B	CTRL	DIV-B
0.5	80%	0.48	100%	0.48	43%	0.54
0.6	73%	0.45	100%	0.44	39%	0.48
0.7	67%	0.41	100%	0.40	34%	0.43
0.8	59%	0.35	100%	0.34	29%	0.38

Table 6.6: Using higher p in top-p sampling leads to lower control of story via plan and higher diversity.

CTRL for LAP-CINF-UDEV as well as SUPERVPLAN (Table 6.6). This further shows an interesting trade-off between control and diversity.

6.4.7 Inference Network

The latent plan model with no constraint on the inference network, LAP-UINF-UDEC, suffers from severe mode collapse and essentially ignores the plan. This demonstrates that constraining the inference network was useful in mitigating the posterior collapse issue. In preliminary experiments, we also observed that using a bag-of-words inference network instead of the BiLSTM leads to worse performance on perplexity, diversity and control, which indicates that the learned posteriors for the BiLSTM network are in fact considering words in context rather than just identifying topical words in the vocabulary.

On analyzing the argmax outputs from the inference network of the trained LAP-CINF-UDEV model, we find that 42% of the predicted anchor words are nouns, 39% of them are verbs, and 11% are adjectives, compared to 58%, 33% and 6% respectively for the RAKE extracted keywords for the ROC data. Thus, the inference network learned along-with the LAP-CINF-UDEV model has higher preference for verbs and adjectives compared to the RAKE algorithm.

6.4.8 Visual Storytelling Dataset

We also conduct experiments with the text portion of a visual story dataset (Huang et al., 2016). The dataset consists of 40155, 4990, and 5055 stories in train, dev, and test splits. Compared to the ROC data, there are no titles associated with stories, and we learn unconditional anchor word sequence $p(z)$. We train the best model configuration LAP-CINF-UDEV (with constrained inference network and unconstrained decoder). To train the baseline SUPERVPLAN, we run the RAKE algorithm to tag the data with the anchor words. We observe that LAP-CINF-UDEV performs better in terms of diversity of generated stories and plans, as well as perplexity relative to SUPERVPLAN (Table 6.7). Diversity computations are

Model	PPL↓		DIV↑	
	dev	test	plan	story
No Plan				
VIZSTORYDATA	NA	NA	NA	8.9
NOPLAN-LM	**38.5**	40.0	NA	6.3
With Plan				
SUPERVPLAN	≤41.5	≤42.2	6.5	6.5
LAP-CINF-UDEV	≤**39.9**	≤**40.8**	8.0	6.6

Table 6.7: Experiments with a second story dataset. We experiment with the text portion of the Visual Story Dataset. We observe that LAP-CINF-UDEV is able to perform better than SUPERVPLAN on perplexity and diversity.

performed with 200 generated samples. We provide further example generations from various methods in the Appendix.

6.5 Related Work

Prior work on story generation has largely focused on plot outline via keywords or key phrases (Yao et al., 2019; Xu et al., 2018), event-based representations (Martin et al., 2018; Fan et al., 2019), or a sentence theme (Chen et al., 2019b). Liu et al. (2020) propose a method to generate a story conditioned on a character description. Prior work on narrative text generation with plans has mostly relied on external resources or tools to extract outlines (Zhou et al., 2018a; Fan et al., 2019), and then training in a supervised manner. For example, using VADER (Hutto and Gilbert, 2014) to tag sentiment polarity (Luo et al., 2019).

Much prior work has used manually defined objectives to encourage coherence in generated text. In this context, reinforcement learning has been used to encourage stories to follow certain manually defined goals such as being locally coherent (Tambwekar et al., 2018; Xu et al., 2018). Prior work on visual story generation aim to learn topically coherent visual story generation (Huang et al., 2019; Wang et al., 2019a). Compared to topics, keywords provide more fine-grained plan, and thus are more likely to provide fine-grained control over generated outputs.

In this work we have proposed a constrained inference network and a constrained decoder for story generation. Pointer networks (Vinyals et al., 2015a) have been used for amortized inference in prior work on summarization (Miao and Blunsom, 2016), though in a semi-supervised context. Non-monotonic sequence generation has been explored in past for tasks such as machine translation (Welleck et al., 2019a). One of the goals of this study is to generate coherent stories relevant to a given title. Fidelity and coverage of text input has been explored in prior work (Tu et al., 2016; Gangal et al., 2017), though not in context of the proposed constrained decoder. In the proposed model, the generation plan can be used to control the story via the anchor words. Hard and soft constraints for incorporating keywords into generation have been explored in (Kiddon et al., 2016b; Miao et al., 2019; Anderson et al., 2016). Controllable text generation has been explored in other tasks as well (Fan et al., 2018; Keskar et al., 2019).

7 Retrieved Snippets as Discrete Plans for Guided Generation

Outputs of many existing dialog models are limited by the 'knowledge' available to the models at training time. In this chapter, I discuss methods to inject relevant knowledge at decoding time. In Majumder et al. (2021), we equip persona grounded dialog models with 'background stories' related to a persona by leveraging fictional narratives from existing story datasets (e.g. ROCStories). Our method works with a model trained just with dialog data i.e. without access to story corpus at training time. In Jhamtani et al. (2021), we equip dialog models with dictionary *translations* of figurative English expressions to their literal counterparts to improve handling of figurative language in dialog systems. A major advantage of such class of approaches is flexibility in adding new or updated knowledge sources without the need for re-training the models from scratch.

7.1 Introduction

People often rely on specific incidents and experiences while conversing in social contexts (Dunbar et al., 1997). Responses from existing chitchat dialog agents often lack such specific details. To mitigate this, some prior work has looked into assigning personas to dialog agents (Zhang et al., 2018a; Majumder et al., 2020b). However, persona descriptions are often shallow and limited in scope, and while they lead to improvements response specificity, they still lack the level of detail with which humans share experiences.

In this work, we propose methods to enrich dialog personas with relevant background events using fictional narratives from existing story datasets such as ROCStories (Mostafazadeh et al., 2016b). For example, for a persona attribute 'I have two children and a dog,' we are able to identify a relevant narrative from a story corpus (Figure 7.1). However, such stories may not directly fit fluently in the dialog context. Thus, retrieved stories should be adapted to construct a response that is fluent and relevant to the context. Since existing datasets (such as PersonaChat (Zhang et al., 2018a)) do not contain responses with such background stories, such adaptation has to be done in an unsupervised fashion with decoders trained to generate responses conditioned only on a dialog history and persona.

To adapt a retrieved narrative incident as a relevant background story, we use a decoding procedure which encourages the generated response to (1) be fluent with the dialog history, (2) be consistent with the original persona, and (3) be minimally different from the retrieved story. While fluency with dialog context is encouraged directly by the likelihood as per the underlying language model the remaining two constraints are incorporated via iterative updates to the decoder output distributions at inference time. Our inference-time decoding method is different from the only recent effort by Su et al. (2020) that leverages non-dialog data (forum comments, book snippets) as distant labels to train dialog systems with supervision. Our contributions can be summarized as follows:

- We propose a novel approach to enrich dialog agent personas with relevant backstories, relying only on existing story datasets.

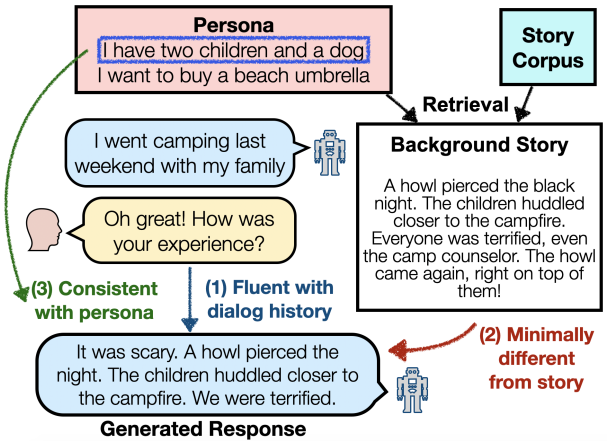


Figure 7.1: We enrich agent personas with 'background stories' from an existing corpus. We propose a gradient-based technique which encourages the generated response to be fluent with the dialog history, minimally different from the retrieved story, and consistent with the persona. The proposed approach leads to more specific and interesting responses.

- We propose to use an unsupervised back-propagation based decoding procedure[1] to adapt the relevant stories such that the resulting response is fluent with the dialog history and consistent with the dialog agent persona. Our method works with a model trained just with dialog data i.e. without access to story corpus at training time.

- Our experiments demonstrate that the proposed approach results in much more engaging and specific dialog outputs in a persona-grounded dialog setup. This fills a gap in existing dialog models which often lack the capability to generate responses about specific events and experiences relevant to persona attributes.

7.2 Method

Given dialog history h and persona C consisting of several (typically 3-5, example shown in Figure 7.1) attributes, our goal is to construct a dialog response x. Our underlying model is based on the discrete persona attribute choice model from Majumder et al. (2020b). To generate a dialog utterance x, we first sample a persona attribute $c \sim p(c|h)$ conditioned on the dialog history h. x is then generated conditioned on the dialog history and the chosen persona attribute. The underlying dialog model's decoder is initialized with a pretrained GPT-2 model, and is fine-tuned on the PersonaChat dataset (Zhang et al., 2018a). However, in our current setup, we also have to identify relevant background stories and use them to construct fluent responses at decoding time. Therefore, we propose a different decoding procedure.

To generate a response, we first sample a persona attribute $c \sim p(c|h)$. Next we retrieve stories corresponding to the persona attribute c (Section 7.2.1). However, the underlying dialog model is trained to generate responses conditioned only on the dialog history and persona. To incorporate the retrieved story in the response, we perform gradient-based inference (Section 7.2.2), that only assumes a left-to-right language model trained on dialog context and responses, and the story is handled at decoding time in an unsupervised fashion. We refer to the proposed method as **PABST** (Unsupervised **P**erson**A** enrichment with **B**ackground **ST**ories).

7.2.1 Retrieving Relevant Stories

For a persona attribute c, we aim to identify relevant stories from a story corpus. Toward this goal, we rank the stories using the F1 component of BERT-score (Zhang et al., 2020) based retrieval using the

[1] Code can be found at
https://github.com/majumderb/pabst

persona attribute c as the query and the highest scoring story is chosen. Note that many of the stories are written in the third person. For use as background stories, we must first transform them to first–person. Following prior work (Brahman and Chaturvedi, 2020), we identify the protagonist of such stories as the most frequently occurring character. Thereafter, we use co-reference resolution (Lee et al., 2017) to identify all words or phrases that refer to the protagonist. Finally, all words or phrases so identified are replaced with suitable first person pronouns (e.g. 'his books' to 'my books').

7.2.2 Gradient-based Inference

Our underlying dialog model is not trained to condition on a retrieved story, and cannot be directly used to construct a desirable response using s. To tackle this, we consider a decoding strategy which, in addition to fluency with history h, encourages response x to follow two soft constraints: (1) be minimally different from story s, and (2) be consistent with persona c.

First, we generate an initial response based only on the dialog history. Then we perform an iterative procedure which alternates between performing a forward pass on the language model to encourage fluency, and a backward pass which updates the response via back-propagation to respect the two soft constraints. However, x is discrete, and cannot be directly updated using gradients from back-propagation. Instead, we maintain and update a soft representation o of x, where o_i corresponds to the last hidden state representation for the i^{th} token position, i.e., $p(x_i) \sim \text{softmax}(Wo_i/\tau)$, where τ is the temperature parameter, W is the embedding matrix, and $Wo_i \in \mathscr{R}^V$ (V is the vocabulary size). Our approach is inspired by recent works that use gradient-based decoding for text generation with soft constraints (Dathathri et al., 2020; Qin et al., 2020). Next we describe the backward and forward passes of the iterative procedure.

Backward Pass with Soft Constraints We define the following soft constraints on response x:
(1) **Divergence from story:** We want to encourage x to be *minimally different* from the story s. Following prior work (Qin et al., 2020), we compute a cross entropy loss (denoted by cross-entr henceforth) with story $s = \{s_1, \ldots, s_T\}$ tokens as labels and $Wo_1, \ldots, Wo_T$ as the logits.

(2) **Consistency to persona:** We want x to be *consistent with persona attribute* c. Consider a classifier $q_\phi(o, c)$ which predicts the probability of x (or rather the soft representation o of x) entailing c. The classifier $q_\phi(o, c)$ is a bag-of-words classification head on decoder hidden states o, fine-tuned on the Dialogue-NLI dataset (Welleck et al., 2019b) to predict whether pairs of persona attributes and responses are entailed or not. The objective to maximize can be written as:

$$\mathscr{L}(c, s; o) = \lambda_c \log q_\phi(o, c) - \lambda_d \,\text{cross-entr}(s, Wo)$$

where λ_c and λ_d are hyper-parameters. We update o through back-propagation by computing the gradient $\nabla_o \mathscr{L}(c, s; o)$, while keeping the model parameters constant. Let the resulting o after the gradient-based updates be denoted by o^b.

Forward Pass to Encourage Fluency Next we perform a forward pass of the underlying dialog model, with the goal of regularizing the hidden states towards the unmodified language model values. On computing the forward pass at the j^{th} token, we mix the final hidden states o_j^f from the forward pass with o_j^b computed in the backward pass, via weighted addition to get the resulting $o_j = \gamma \times o_j^f + (1 - \gamma) \times o_j^b$, where $\gamma \in (0, 1)$ is a hyperparameter. The resulting o_j is used for computing the logits at the next time step $j + 1$.

We initialize the output response by performing greedy decoding from the underlying dialog model, conditioned on the dialog history and persona attribute. Then we iteratively update o by alternate backward and forward passes. We sample the final response $x \sim \text{softmax}(Wo/\tau)$. In practice, we found that 5 iterations are sufficient to generate good quality outputs.

Method	Training	Decoding	D-1	D-2	ENTR
W/o Story Data					
TRANSFERO	PERSONA-CHAT‘	Nucleus	0.05	0.11	1.21
COMPAC	PERSONA-CHAT‘	Nucleus	0.15	0.25	1.25
COMPAC	CS-KB	Nucleus	0.87	1.07	2.04
With Story Data					
COMPAC	PSEUDO	Nucleus	0.91	2.45	2.89
COMPAC	MULTITASK	Nucleus	0.99	2.54	2.71
COMPAC	PERSONA-CHAT‘	NEARNBR	2.56	9.67	3.86
PABST (Ours)	PERSONA-CHAT‘	Grad. Inf.	1.56	3.57	3.21

Table 7.1: Diversity metrics on the PersonaChat test set. D-1/2 is the % of distinct uni- and bi-grams. ENTR is the geometric mean of n-gram entropy. Grad. Inf. is the unsupervised gradient-based decoding as opposed to Nucleus sampling (Holtzman et al., 2020a).

PABST vs.	TRANSFERO		COMPAC		NEARNBR		PSEUDO		MULTITASK		w/o DNLI		Gold	
Aspect	win	loss	win	loss	win	loss	win	loss	win	loss	win	loss	win	loss
Sensible	**83.8**	5.6	**71.4**	10.1	**72.7**	15.7	**94.3**	3.1	**92.8**	1.5	**81.7**	6.8	29.1	**33.6**
Engaging	**96.7**	1.8	**82.6**	6.7	**87.2**	5.3	**94.7**	2.1	**94.1**	2.6	**70.9**	14.8	**70.2**	15.6

Table 7.2: Pairwise comparison (% win/loss cases, ties not reported) between responses from PABST and from other baselines as well as gold. All differences are significant with $p < 0.05$. Cohen's Kappa (Cohen, 1960) for sensibility and engagement were 0.79 and 0.82 respectively. Complete details of the human evaluations are in Appendix §B.

Dialog History Agent: How was your last trip? User: It was great. I went to Utah and saw the Canyonlands.
Retrieved Story: I went on a trip. I went to Florida. When we got there, there was no air conditioning. I was irritated. It was too hot.
COMPAC: That's great. You must be adventurous.
NEARNBR: I went on a trip. I went to Florida. When we got there, there was no air conditioning. I was irritated. It was too hot.
PSEUDO: I have a house in Florida.
MULTITASK: I like Florida. I went to the beaches.
PABST ($\lambda_d = 5$): I went on a trip. I went to Florida. I have a house there. We do not have air condition. It was hot.
PABST ($\lambda_d = 1$): That's great. I frequently go to Florida. I have a house there but it gets hot in summer.

Dialog History Agent: I work as stunt double. What about you? User: I'm an engineer. I just got back from work. How's yours?
Retrieved Story: I was a working actor. For my new role I was a stunt double. I started working on new stunts. When I jumped off a building yesterday I got hurt. I had to be rushed to the hospital.
COMPAC: That's great. I am looking for a job.
NEARNBR: I was a working actor. For my new role I was a stunt double. I started working on new stunts. When I jumped off a building yesterday I got hurt. I had to be rushed to the hospital.
PSEUDO: I got a new job. I am an actor.
MULTITASK: I started a new role. I got hurt.
PABST ($\lambda_d = 5$): I was working as a stunt double. I jumped off a building. I got hurt. I had to be rushed to the hospital.
PABST ($\lambda_d = 1$): Yesterday while I was working, I jumped off a building and I got hurt. I had to be taken to the hospital.

Table 7.3: Generations from different models. More examples are in Appendix §C.

7.3 Experiments

We evaluate methods in terms of their capability to generate diverse, fluent and engaging responses. Hyperparameters are noted in Appendix §A.

Datasets We experiment with the PersonaChat dialog dataset (Zhang et al., 2018a) consisting of 131,438 utterances for training, 15,602 for validation, and 15,024 for testing. For stories, we use the training split of the ROCStories dataset (Mostafazadeh et al., 2016b), that consists of 78,529 stories, each typically of 4 to 5 sentences.

Baselines We consider two broad groups of models as baselines: **(1)** ***Without access to story corpus***: We use finetuned GPT2 (**TRANSFERO**) on PersonaChat, and the discrete persona attribute choice model (**COMPAC**) from Majumder et al. (2020b). We also consider a version of COMPAC which enriches personas with inferences from a commonsense knowledge base (**CS-KB**). **(2)** ***Baselines using story corpus*:** To allow COMPAC models to generate story-like responses, we adapt an alternative training regime (**PSEUDO**) from (Su et al., 2020), where we randomly replace some of the target dialog responses with retrieved stories—treating them as pseudo labels. Finally, we also consider a **MULTITASK** training setup from (Su et al., 2020), wherein the decoder is trained on PersonaChat as well as with a language modeling objective on ROCStories. We additionally consider a **NEARNBR** baseline that uses the retrieved story verbatim as the dialog response.

7.3.1 Automatic Evaluation

We hypothesize that that the proposed approach to leverage external non-dialog data can increase the diversity of the generated responses. Following prior work (Li et al., 2016a), we report the percentage of distinct uni-grams and bi-grams (**D-1** and **D-2** respectively). Note that these values do not capture the actual frequency distribution of different word types. Therefore, we also report the geometric mean of entropy values of empirical frequency distributions of n-grams of words (n $\in \{1,2,3\}$) (Jhamtani et al., 2018), denoted by **ENTR**.

We observe that methods that use story data show much higher diversity compared to methods that do not (Table 7.1). Among methods using story data, gradient-based decoding (PABST) performs better than COMPAC trained with PSEUDO or MULTITASK. Note that just using NEARNBR outputs as-is leads to even more diverse outputs than PABST. However, they are much less sensible with the context, as shown in human evaluations.

7.3.2 Human Evaluation

Since we do not have ground truth story-like responses in the dialog dataset, we perform human evaluation with 150 test examples to investigate if PABST generates responses that are 1) **sensible** with the dialog history and 2) **engaging**. We hired two Anglophone (Lifetime HIT acceptance % > 85) annotators for every test sample. The order of the systems present in the interface is randomized. A snapshot of the human evaluation interface is provided in Appendix §C. All differences in values from human evaluations are significant with $p < 0.05$ from bootstrap tests on 1000 subsets of size 50. Cohen's Kappa (Cohen, 1960) to measure inter-annotator agreement for sensibility and engagement were 0.79 and 0.82 respectively.

From the results (shown in Table 7.3), we note that in comparison to responses from baselines, responses from PABST are more engaging and more sensible with respect to the dialog history. We further make following observations. Firstly, using the gradient-based decoding approach with retrieved stories (PABST) works significantly better than using distant supervision with stories data (PSEUDO and MULTITASK). Secondly, background stories provide sufficient detail for an engaging conversation compared to COMPAC which expands persona attributes using commonsense knowledge (Majumder et al., 2020b). Finally, we also observe that PABST performs worse when we do not use the consistency constraint (w/o DNLI).

Choice of λ_d We also experiment with different values of the weight for the divergence term (λ_d) in $\mathscr{L}$: High ($\lambda_d = 5$), Moderate ($\lambda_d = 1$), and Low ($\lambda_d = 0.05$). We consider 100 samples for this experiment. We attribute a high λ_d to responses strictly copying the story. We find that PABST (moderate λ_d) wins wins 81.2% and 69.1% cases against PABST (high λ_d) on 'sensible' and 'engaging' response criteria respectively. Similarly, PABST (moderate λ_d) wins 93.2% and 84.7% cases against PABST (low λ_d) in terms of sensibility and engagement respectively.

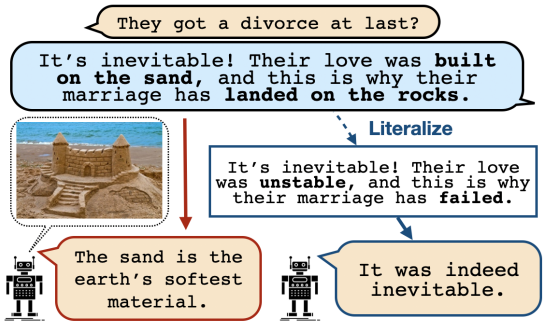


Figure 7.2: An example illustrating how model responses are affected by figurative constructs in dialog context. Here, the model conflates the metaphorical use of *build on the sand* with its literal meaning, leading to an inappropriate, atopical response.

Qualitative Analysis Table 7.3 shows responses generated by different baselines. We observe that PABST is able to follow the retrieved story (same as output from NEARNBR) while modifying the response to be conversation-like and sensible with dialog history. Responses from other baselines remain verbose or incoherent. Mirroring the human evaluation, we observe that choosing a higher λ_d makes the model to almost repeat the retrieved story but a lower value smooths the output to make it more sensible with the ongoing dialog.

7.4 Related Work

A desired impact of the proposed approach is increase in diversity of the generated responses. To tackle the issue of diversity in dialog model outputs, prior work has focused on decoding strategies such as diversity-promoting sampling (Holtzman et al., 2020a); training strategies such as discouraging undesirable responses via unlikelihood training (Li et al., 2020); model changes such as using stochastic variables (Serban et al., 2017); and using external data such as forum data (Su et al., 2020) or external knowledge bases (Majumder et al., 2020b). In contrast to these, our proposed method generates responses with background stories using a gradient-based decoding approach.

One of the steps in our proposed approach is to retrieve relevant stories from an external corpus. Prior work has explored using retrieval of similar dialog instances as an initial step in improving response diversity and other human-like desiderata in dialog (Roller et al., 2020; Weston et al., 2018). Distant supervision by using retrieved text snippets as pseudo responses has been explored in prior work (Su et al., 2020; Roller et al., 2020). We use an external data source to improve dialog responses, a theme shared with some efforts in other tasks such as machine translation (Khandelwal et al.). The use of narrative text in dialog has been explored in prior work, mostly as a 'script' or template for conversation (Xu et al., 2020a; Zhu et al., 2020). We adapted a BERT-based retrieval method (Zhang et al., 2020) in our case to retrieve relevant story given dialog context and use retrieved story in the decoding phase.

Gradient-based for text generation with soft constraints has been explored in prior work (Dathathri et al., 2020; Qin et al., 2020). Song et al. (2020) focused on generating response which are consistent to given persona. Differently, we use a gradient-based decoding to generate a dialog response while honoring constraints such as consistency to persona and similarity to retrieved story.

7.5 Application to Improved Handling of Figurative Language in Dialog Systems

People frequently employ figurative language such as metaphors (Carbonell, 1982) and idioms (Jackendoff, 1995) for effective and/or stylistic communication. Thus, dialog models interacting with humans should be equipped to handle these forms of communication. However, understanding figurative language might be challenging for machines since figurative constructions often exhibit non-compositional semantics and may rely on shared cultural and common-sense knowledge (Carbonell and Minton, 1983). For example, a powerful GPT2 model fine-tuned on DailyDialog dataset is unable to handle the metaphor 'built on the sand' (Figure 7.2), and the response seems to rely on the unintended literal sense of 'sand'.

In Jhamtani et al. (2021), we investigate the performance of existing dialog models when faced with inputs containing figurative language use. We identify the subsets in existing datasets (such as DailyDialog (Li et al., 2017) and PersonaChat (Zhang et al., 2018b)) which have figurative language use such as metaphors and similes. We propose a simple defense against occurrences of figurative language in dialog context. More specifically, we use existing classifiers to detect presence of certain types of figurative language in dialog contexts, and transform them to their literal counterparts before feeding them to the dialog models. We use detected figurative language phrases to query certain external dictionaries to identify their literal counterparts. For example, literal equivalent of 'on the sand' can be 'unstable' (Figure 7.2). We observe that performance of dialog models improves when using literal equivalents in place of figurative language. The proposed technique is lightweight, does not require any retraining of the models, and is effective – though gaps still remain, leaving scope for interesting future explorations.

8 Conclusion

8.1 Summary of Contributions

This thesis describes my work on grounded natural language generation through interpretable hierarchical operations. I have demonstrated how inducing relevant discrete structure in text generation models can expose intermediate underlying computations, enable controllable generations, and often improve results on automated metrics as well as human evaluations (Figure 8.1).

In **Part 1** of the thesis, I discuss models for grounded text generation via latent interpretable operations. Compared to just using neural soft attention, the proposed models using latent operations perform better than per various automated and human evaluation studies. Our analysis reveals that the proposed models often enable controllable generation, and are more interpretable compared to various baselines. Thus, the results establish the usefulness of inducing latent operations on data. In **Part 2**, I discuss how naively trained language models often struggle in capturing of complex patterns in natural language such as content plans in narratives and rhyming schemes in poetry. I propose new models and learning techniques to effectively learn such complex patterns present in natural language.

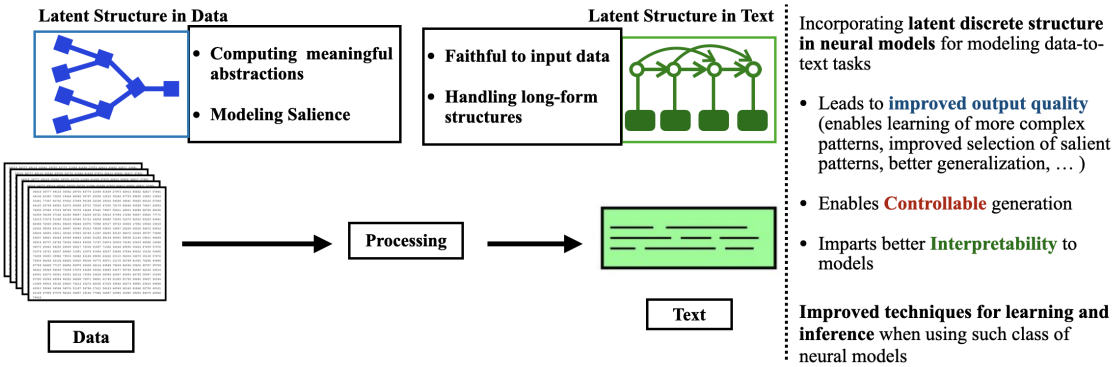


Figure 8.1: Summary of Contributions: In this thesis, I discuss models that induce meaningful latent discrete structure for grounded natural language generation. Compared to using soft attention alone, such structure can often better model complex patterns on data, enables controllable generation, and leads to models that are more interpretable and robust compared to many contemporary methods relying on neural soft-attention alone.

Additionally, I have put a major emphasis on inducing latent structure rather than assuming access to data tagged with the desired structure. Focus on learning *latent* structure is an important aspect if we want to effectively leverage large amounts of unlabelled data and reduce reliance of labelled data. I discuss and propose new machine learning techniques to enable learning such structures effectively with respect to grounded natural language generation.

8.2 Broader Impact

Business Intelligence: Grounded natural language generation has several applications in business intelligence to generate insightful reports of data such as sales numbers or profits. In such reports, raw

data such as numerical tabular data and images are often accompanied by narrative insights guiding a reader through the data. Learning to summarize interesting abstract patterns from numerical tabular data (Jhamtani and Berg-Kirkpatrick, 2021) can be extended for automated report generation. Our proposed approaches can leverage previously written summary reports of data to train models to identify salient abstract patterns from the corresponding tabular data. Work on difference description generation discussed in chapter 2 (Jhamtani and Berg-Kirkpatrick, 2018; Jhamtani et al., 2018) can be extended for tracking and summarizing interesting changes in data.

Health: There has been rapid digitization of medical data, including test reports and data from sensors on wearable devices such as digital watches. However, such data is of not much use if people can't understand it. An automated system can potentially unlock valuable insights by, for instance, detecting and giving an early warning for potential health disorders. Importantly, natural language is a key medium for such smart machines to effectively deliver useful insights since users might not have the expertise to interpret a graph or understand a spreadsheet. A lot of work discussed in this thesis has relevant applications in such use cases. For example, work on describing differences between two similar images (Jhamtani and Berg-Kirkpatrick, 2018) can be extended for comparing two body scan images taken at different points in time. Another example is the work on accurately describing salient patterns in numerical time series (Jhamtani and Berg-Kirkpatrick, 2021) which can be extended to summarize sensor data for use by a patient or a doctor.

Education and Tutoring: NLP technologies have made rapid inroads in the field of education via tools and techniques for tasks such as automated essay scoring. Grounded natural language generation has several applications in the field of education. For example, methods for explanation generation for multi-hop question answering (Jhamtani and Clark, 2020) discussed in the thesis have applications in building science tutoring systems for children. Similarly, automated game commentary generation (Jhamtani et al., 2018) can help in building systems for people to learn to play a game by consuming generated game commentary.

Creative Content Authoring: Models for music generation (Jhamtani and Berg-Kirkpatrick, 2019), and creative text generation such as poetry (Jhamtani et al., 2019) could be useful for creative professionals in the music and arts industry. Additionally, systems for generative narratives (Jhamtani and Berg-Kirkpatrick, 2020) have applications in children's storybook generation. Furthermore, work on poetry generation (Jhamtani et al., 2019) and portmanteau predictions (Gangal et al., 2017) has applications in advertisement generation. While advertisements can be a high-stakes use case and not amenable to complete automation, machine-generated outputs can still provide valuable suggestions and ideas for a person to build upon the suggestions through some post-editing process.

Trustworthy AI: This thesis describes my work on grounded natural language generation through interpretable hierarchical operations. I have demonstrated how inducing relevant discrete structures in text generation models can expose intermediate underlying computations, often providing a certain degree of interpretability and enabling controllable generations. The notions of interpretability and controllability are considered important factors in improving the usability of and building user trust in NLG systems. Additionally, I demonstrate that in many cases the proposed ideas of incorporating latent discrete structure in various grounded NLG tasks leads to significant improvements in the factuality of the output and reduces the issue of hallucination. For example, in chapter 4, I propose a truth-conditional modeling (Jhamtani and Berg-Kirkpatrick, 2021) approach that is suitable for generating text outputs that are faithful to the input data.

Green AI: Recent progress in NLP has been in part possible due to improved computational resources. Building large models, however, often requires substantial energy consumption.One way to mitigate the financial and environmental cost of such models is to invent new and more efficient training and inference techniques. In chapter 7, I discuss methods to introduce relevant additional 'knowledge' at decoding time without the need to re-train existing dialog models (Majumder et al., 2021; Jhamtani et al., 2021). Such post-hoc injection methods are much more efficient compared to fine-tuning or training the models to work with new data sources.

8.3 Discussions and Future Work

8.3.1 Machine-learning based NLG Systems in Real-world

An automated approach for text generation is fraught with several risks. Machine-generated natural language descriptions can hallucinate content and be factually incorrect. Therefore, such outputs can potentially be unreliable and may cause serious harm. For example, an inaccurate machine-generated report of some medical data can be dangerous and lead to injury. Another example of a potential issue is that machine-generated text can be offensive and can contain biases against a race or a gender type. However, as discussed in this thesis, machine-learning-driven NLG methods can provide certain desirable characteristics such as improved output diversity, interesting outputs, and generalizing to previously unseen scenarios. Additionally, compared to a person doing manual writing, machine learning-based approaches can easily scale and has applications in automated content generation in sectors such as health, finance, and education.

In the near term, a more feasible approach in terms of deploying NLG systems might be to aim for a mix of neural and rule-based systems. In such cases, the system under certain situations reverts to a 'backup' template or a rule-based NLG system, which is considered safer to use. Another useful paradigm is a human-in-the-loop setup. Under such a paradigm, we expect a person to edit and verify machine-generated outputs. Additionally, it might be possible to add some constraints to limit the space of outputs that the machine can produce. For example, it might be possible to block some hateful words from being generated. Finally, real-world users want to have control over the system and configure it according to their specific needs. In such a context, continued focus on interpretability and controllability aspects in neural NLG models seems to be important.

8.3.2 Broad Directions for Future Work

Extension to Dynamic Environments: Much of the work in this thesis deals with static datasets and static models. More challenging situations may encompass dynamic and evolving environments, with the need to update models based on new information, data, and even feedback from users. For example, chapter 4 of this thesis deals with generating descriptions for high-level interesting patterns in time-series data. However, some users might be interested in small sudden changes, while others might be more interested in long-term trends. To account for such feedback or preferences, we need to build systems that users can interact with to pick the desired set among the exposed programs/patterns.

NLG and Communication Goals: Many of the experiments discussed in the thesis involve human annotators to judge the quality of generated text outputs. However, additional work is needed to evaluate how the generated descriptions help achieve the communication goal in question. For example, does automatically generated commentary help people understand an ongoing chess game better? Such considerations will require carrying out user studies with deployed systems.

Emphasis on Human-in-the-Loop Setups: Much of the work discussed in this thesis deals with machines generating the final outputs with little or no interactions with a person. Though the ability to work with little human intervention is a desirable train in many situations, there are a plethora of cases where having a human involved in achieving the final output is preferred. For example, in chapter 6, I discuss narrative text generation given a title. An extension for the proposed model is to iterate on a generated story based on some user-suggested edits. Note that the involvement of a person does not mean that machine-generated output does not add value. Machine-generated outputs can, for example, be much easier and faster to select from as opposed to writing from scratch, and can also be used as ideation or collaborative tool by the users.

Compositionality of Models: Some of the work in this thesis discusses the compositionality of identified sub-patterns in data. A possible next step is to explore compositionality at a much higher level of abstractions – perhaps to the point of composing models trained on different datasets and tasks. For example, chapter 2 deals with the modeling of difference descriptions in images, and chapter 4 deals with

generating descriptions for high-level interesting patterns in time-series data. Now let's say we wish to describe differences between two-time series. Can we leverage the two models (for difference description and time series captioning) as high-level abstractions, and effectively combine the two systems without the need for thousands of data points for the new task?

Extensions to Multimodal Output Space: Much of the work in this thesis focuses on only text outputs. In many cases, there might be a choice to present insights in a visual format or mix of textual and visual format. Some information is more suited for visual format (e.g. visualizing geo-location on a map) while some other might be more suitable for textual format (e.g. a sentence summarizing interesting pattern in a time series). An important future direction is to include such decisions (what information to present in which format), and perhaps generate multi-modal making effective use of multiple modalities.

Bibliography

Saeed Amizadeh, Hamid Palangi, Alex Polozov, Yichen Huang, and Kazuhito Koishida. 2020. Neuro-symbolic visual reasoning: Disentangling "visual" from "reasoning". In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, Proceedings of Machine Learning Research PMLR.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Guided open vocabulary image captioning with constrained beam search. *CoRR*, abs/1612.00576.

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving bert a calculator: Finding operations and arguments with reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5949–5954.

Jacob Andreas and Dan Klein. 2014. Grounding language with points and paths in continuous spaces. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 58–67.

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4,2016*, pages 1173–1182.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Learning to compose neural networks for question answering. In *Proceedings of NAACL-HLT*, pages 1545–1554.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

Siqi Bao, Huang He, Fan Wang, Rongzhong Lian, and Hua Wu. 2019. Know more about each other: Evolving dialogue strategy via compound assessment. In *ACL*.

Antoine Bosselut, Asli Celikyilmaz, Xiaodong He, Jianfeng Gao, Po-Sen Huang, and Yejin Choi. 2018. Discourse-aware neural rewards for coherent text generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 173–184.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, and Yejin Choi. 2019. COMET: commonsense transformers for automatic knowledge graph construction. In *ACL*.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2012. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*.

Samuel Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *SIGNLL*.

Faeze Brahman and Snigdha Chaturvedi. 2020. Modeling protagonist emotions for emotion-aware storytelling. In *EMNLP*, pages 5277–5294.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Lorenzo Bruzzone and Diego F Prieto. 2000. Automatic analysis of the difference image for unsupervised change detection. *IEEE Transactions on Geoscience and Remote sensing*, 38(3):1171–1182.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. 2016. Importance weighted autoencoders.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*.

Jaime G Carbonell. 1982. Metaphor: an inescapable phenomenon in natural-language comprehension. *Strategies for natural language processing*, pages 415–434.

Jaime G Carbonell and Steven Minton. 1983. Metaphor and common-sense reasoning. Technical report, Carnegie Mellon University.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

Charles Chen, Ruiyi Zhang, Eunyee Koh, Sungchul Kim, Scott Cohen, Tong Yu, Ryan A. Rossi, and Razvan C. Bunescu. 2019a. Figure captioning with reasoning and sequence-level training. *CoRR*, abs/1906.02850.

Gang Chen, Yang Liu, Huanbo Luan, Meng Zhang, Qun Liu, and Maosong Sun. 2019b. Learning to predict explainable plots for neural story generation. *arXiv preprint arXiv:1912.02395*.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020. Logical natural language generation from open-domain tables. *arXiv preprint arXiv:2004.10404*.

Volkan Cirik, Taylor Berg-Kirkpatrick, and Louis-Philippe Morency. 2018. Using syntax to ground referring expressions in natural images. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*. AAAI Press.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pages 845–855. Association for Computational Linguistics.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *ICLR*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Bhuwan Dhingra, Manaal Faruqui, Ankur P. Parikh, Ming-Wei Chang, Dipanjan Das, and William W. Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander H. Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, Shrimai Prabhumoye, Alan W. Black, Alexander I. Rudnicky, Jason Williams, Joelle Pineau, Mikhail Burtsev, and Jason Weston. 2019a. The second conversational intelligence challenge (convai2). *CoRR*, abs/1902.00098.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019b. Wizard of wikipedia: Knowledge-powered conversational agents. In *ICLR*.

Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. 2018. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Robin IM Dunbar, Anna Marriott, and Neil DC Duncan. 1997. Human conversational behavior. *Human nature*, 8(3):231–246.

Douglas Eck and Juergen Schmidhuber. 2002. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 103.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, NMT@ACL 2018,*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. *arXiv preprint arXiv:1902.01109*.

Christian Federmann, Oussama Elachqar, and Chris Quirk. 2019. Multilingual whispers: Generating paraphrases with translation. In *W-NUT@EMNLP*.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*.

Arthur Flexer, Fabien Gouyon, Simon Dixon, and Gerhard Widmer. 2006. Probabilistic combination of features for music classification. In *ISMIR*, pages 111–114.

Varun Gangal, Harsh Jhamtani, Graham Neubig, Eduard Hovy, and Eric Nyberg. 2017. Charmanteau: Character embedding models for portmanteau creation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2907–2912.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

Sebastian Gehrmann, Yuntian Deng, and Alexander M. Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing EMNLP*, pages 4098–4109.

Sebastian Gehrmann et al. 2021. The GEM benchmark: Natural language generation, its evaluation and metrics. *CoRR*, abs/2102.01672.

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.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018a. A knowledge-grounded neural conversation model. In *AAAI*.

Marjan Ghazvininejad, Yejin Choi, and Kevin Knight. 2018b. 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)*, volume 2, 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.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. *Proceedings of ACL 2017, System Demonstrations*, pages 43–48.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Tanya Goyal and Greg Durrett. 2021. Annotating and modeling fine-grained factuality in summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT 2021*.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2020. Neural module networks for reasoning over text. In *8th International Conference on Learning Representations, ICLR 2020*.

Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*, 2(3):258–268.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020a. The curious case of neural text degeneration. In *ICLR*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020b. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649.

Jack Hopkins and Douwe Kiela. 2017. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 168–178.

Chenglong Hou, Chensong Zhou, Kun Zhou, Jinan Sun, and Sisi Xuanyuanj. 2019. A survey of deep learning applied to story generation. In *International Conference on Smart Computing and Communication*. Springer.

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2017. Modeling relationships in referential expressions with compositional modular networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4418–4427. IEEE.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, and Douglas Eck. 2018. An improved relative self-attention mechanism for transformer with application to music generation. *arXiv preprint arXiv:1809.04281*.

Qiuyuan Huang, Zhe Gan, Asli Celikyilmaz, Dapeng Wu, Jianfeng Wang, and Xiaodong He. 2019. Hierarchically structured reinforcement learning for topically coherent visual story generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33.

Ting-Hao (Kenneth) Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross B. Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. 2016. Visual storytelling. In *NAACL 2016*.

Clayton J. Hutto and Eric Gilbert. 2014. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014*.

Ray Jackendoff. 1995. The boundaries of the lexicon. *Idioms: Structural and psychological perspectives*, 133:165.

Sarthak Jain and Byron C Wallace. 2019. Attention is not explanation. 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 3543–3556.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2018. Learning to describe differences between pairs of similar images. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2018. Learning to describe differences between pairs of similar images. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing EMNLP 2018*.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2019. Modeling self-repetition in music generation using generative adversarial networks. In *Machine Learning for Music Discovery Workshop, ICML 2019*.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2020. Narrative text generation with a latent discrete plan. In *Findings of EMNLP 2020*.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2021. Truth-conditional captioning of time series data. *arXiv preprint arXiv:2110.01839*.

Harsh Jhamtani and Peter Clark. 2020. Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Taylor Berg-Kirkpatrick. 2021. Investigating robustness of dialog models to popular figurative language constructs. *arXiv preprint arXiv:2110.00687*.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Learning to generate move-by-move commentary for chess games from large-scale social forum data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1661–1671.

Harsh Jhamtani, Sanket Vaibhav Mehta, Jaime Carbonell, and Taylor Berg-Kirkpatrick. 2019. Learning rhyming constraints using structured adversaries. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically generating commit messages from diffs using neural machine translation. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 135–146. IEEE.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. 2018. Figureqa: An annotated figure dataset for visual reasoning. In *6th International Conference on Learning Representations, ICLR 2018, Workshop Track Proceedings*. OpenReview.net.

Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. 2015. Learning a game commentary generator with grounded move expressions. In *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*, pages 177–184. IEEE.

Daniel Kang and Tatsunori Hashimoto. 2020. Improved natural language generation via loss truncation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation. *CoRR*.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. 2011. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, volume 2, page 1.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *AAAI Conference on Artificial Intelligence 2020*.

Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2021. Text modular networks: Learning to decompose tasks in the language of existing models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016a. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016b. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Diederik P. Kingma and Max Welling. 2014a. Auto-encoding variational bayes. In *ICLR*.

Diederik P. Kingma and Max Welling. 2014b. Auto-encoding variational bayes.

Xiang Kong, Bohan Li, Graham Neubig, Eduard H. Hovy, and Yiming Yang. 2019. An adversarial approach to high-quality, sentiment-controlled neural dialogue generation. *CoRR*, abs/1901.07129.

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

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick, and Yiming Yang. 2019a. A surprisingly effective fix for deep latent variable modeling of text. In *EMNLP-IJCNLP*.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. 2013. Story generation with crowdsourced plot graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. AAAI Press.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL HLT*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Margaret Li, Stephen Roller, Ilia Kulikov, Sean Welleck, Y-Lan Boureau, Kyunghyun Cho, and Jason Weston. 2019b. Don't say that! making inconsistent dialogue unlikely with unlikelihood training. *CoRR*, abs/1911.03860.

Margaret Li, Stephen Roller, Ilia Kulikov, Sean Welleck, Y-Lan Boureau, Kyunghyun Cho, and Jason Weston. 2020. Don't say that! making inconsistent dialogue unlikely with unlikelihood training. In *ACL*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *IJCNLP*, pages 986–995.

Rongzhong Lian, Min Xie, Fan Wang, Jinhua Peng, and Hua Wu. 2019. Learning to select knowledge for response generation in dialog systems. In *IJCAI*.

Jen-Wen Liao and Jason S Chang. 1990. Computer Generation of Chinese Commentary on Othello Games. In *Proceedings of Rocling III Computational Linguistics Conference III*, pages 393–415.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

Danyang Liu, Juntao Li, Meng-Hsuan Yu, Ziming Huang, Gongshen Liu, Dongyan Zhao, and Rui Yan. 2020. A character-centric neural model for automated story generation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*. AAAI Press.

Tianyu Liu, Xin Zheng, Baobao Chang, and Zhifang Sui. 2021. Towards faithfulness in open domain table-to-text generation from an entity-centric view. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019a. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

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

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Fuli Luo, Damai Dai, Pengcheng Yang, Tianyu Liu, Baobao Chang, Zhifang Sui, and Xu Sun. 2019. Learning to control the fine-grained sentiment for story ending generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Xuezhe Ma, Chunting Zhou, and Eduard H. Hovy. 2019. MAE: mutual posterior-divergence regularization for variational autoencoders. In *7th International Conference on Learning Representations, ICLR 2019*.

Subhransu Maji. 2012. Discovering a lexicon of parts and attributes. In *Computer Vision–ECCV 2012. Workshops and Demonstrations*, pages 21–30. Springer.

Bodhisattwa Prasad Majumder, Taylor Berg-Kirkpatrick, Julian J. McAuley, and Harsh Jhamtani. 2021. Unsupervised enrichment of persona-grounded dialog with background stories. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, ACL 2021*.

Bodhisattwa Prasad Majumder, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Julian McAuley. 2020a. Like hiking? you probably enjoy nature: Persona-grounded dialog with commonsense expansions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bodhisattwa Prasad Majumder, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Julian J. McAuley. 2020b. Like hiking? you probably enjoy nature: Persona-grounded dialog with commonsense expansions. In *EMNLP*.

Inderjeet Mani and Mark T Maybury. 1999. *Advances in automatic text summarization*. MIT press.

Huanru Henry Mao, Bodhisattwa Prasad Majumder, Julian J. McAuley, and Garrison W. Cottrell. 2019. Improving neural story generation by targeted common sense grounding. In *EMNLP*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20.

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 *AAAI*.

Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018. Training millions of personalized dialogue agents. In *EMNLP*.

Kathleen McKeown. 1992. *Text generation*. Cambridge University Press.

Hongyuan Mei, TTI UChicago, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of NAACL-HLT*, pages 720–730.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *ACL*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016a. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016b. A corpus and evaluation framework for deeper understanding of commonsense stories. *CoRR*, abs/1604.01696.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*.

M-E. Nilsback and A. Zisserman. 2006. A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454.

Jekaterina Novikova, Ondrej Dusek, Amanda Cercas Curry, and Verena Rieser. 2017a. Why we need new evaluation metrics for NLG. In *EMNLP*.

Jekaterina Novikova, Ondrej Dusek, and Verena Rieser. 2017b. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue 2017*.

Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis, et al. 2011. A large-scale benchmark dataset for event recognition in surveillance video. In *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*, pages 3153–3160. IEEE.

Hugo Gonçalo Oliveira. 2017. 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.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems Neurips 2017*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002b. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Gabriel Pareyon. 2011. *On Musical Self-Similarity: Intersemiosis as Synecdoche and Analogy*. Gabriel Pareyon.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Allison Parrish. 2015. Pronouncing: Interface for the cmu pronouncing dictionary. [Online; accessed May 2019].

Prasanna Parthasarathi and Joelle Pineau. 2018. Extending neural generative conversational model using external knowledge sources. In *EMNLP*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011a. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011b. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Bryan A Plummer, Arun Mallya, Christopher M Cervantes, Julia Hockenmaier, and Svetlana Lazebnik. 2017. Phrase localization and visual relationship detection with comprehensive image-language cues. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1928–1937.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2641–2649. IEEE.

Julie Porteous and Marc Cavazza. 2009. Controlling narrative generation with planning trajectories: The role of constraints. In *Interactive Storytelling, Second Joint International Conference on Interactive Digital Storytelling, ICIDS*, Lecture Notes in Computer Science.

Matthew Prockup, Andreas F Ehmann, Fabien Gouyon, Erik M Schmidt, and Youngmoo E Kim. 2015. Modeling musical rhythmatscale with the music genome project. In *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 1–5. IEEE.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. In *EMNLP*.

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

Richard J Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam. 2005. Image change detection algorithms: a systematic survey. *IEEE transactions on image processing*, 14(3):294–307.

Sudha Rao and Hal Daumé III. 2019. Answer-based adversarial training for generating clarification questions. 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 143–155.

Sravana Reddy and Kevin Knight. 2011. Unsupervised discovery of rhyme schemes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 77–82.

Ehud Reiter. 2007. An architecture for data-to-text systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*, pages 97–104.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, volume 1, page 3.

Mark O. Riedl and Robert Michael Young. 2010. Narrative planning: Balancing plot and character. *J. Artif. Intell. Res.*, 39.

Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. 2018. A hierarchical latent vector model for learning long-term structure in music. In *International Conference on Machine Learning*, pages 4361–4370.

Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. 2016. Grounding of textual phrases in images by reconstruction. In *European Conference on Computer Vision*, pages 817–834. Springer.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. 2020. Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*.

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

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Aleksander Sadikov, Martin Moina, Matej Guid, Jana Krivec, and Ivan Bratko. 2006. Automated chess tutor. In *International Conference on Computers and Games*, pages 13–25. Springer.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. ATOMIC: an atlas of machine commonsense for if-then reasoning. In *AAAI*.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D Manning. 2019. Do massively pretrained language models make better storytellers? In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 843–861.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Seymour Shlien. Nottingham dataset. `http://ifdo.ca/~seymour/nottingham/nottingham.html`.

Haoyu Song, Wei-Nan Zhang, Jingwen Hu, and Ting Liu. 2019a. Generating persona consistent dialogues by exploiting natural language inference. *CoRR*, abs/1911.05889.

Haoyu Song, Wei-Nan Zhang, Jingwen Hu, and Ting Liu. 2020. Generating persona consistent dialogues by exploiting natural language inference. In *AAAI*.

Haoyu Song, Weinan Zhang, Yiming Cui, Dong Wang, and Ting Liu. 2019b. Exploiting persona information for diverse generation of conversational responses. In *IJCAI*.

Pranay Kumar Venkata Sowdaboina, Sutanu Chakraborti, and Somayajulu Sripada. 2014. Learning to summarize time series data. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 515–528. Springer.

Daniel Spokoyny and Taylor Berg-Kirkpatrick. 2020. An empirical investigation of contextualized number prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*.

Somayajulu Sripada, Ehud Reiter, and Ian Davy. 2003. Sumtime-mousam: Configurable marine weather forecast generator. *Expert Update*, 6(3).

Hui Su, Xiaoyu Shen, Sanqiang Zhao, Xiao Zhou, Pengwei Hu, Randy Zhong, Cheng Niu, and Jie Zhou. 2020. Diversifying dialogue generation with non-conversational text. In *ACL*.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J Martin, Animesh Mehta, Brent Harrison, and Mark O Riedl. 2018. Controllable neural story plot generation via reinforcement learning. *arXiv preprint arXiv:1809.10736*.

Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro A. Szekely. 2021. Representing numbers in NLP: a survey and a vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P. Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *CoRR*, abs/1910.08684.

Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. 2018. Neural arithmetic logic units. In *Advances in Neural Information Processing Systems*, pages 8046–8055.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811*.

Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. 2017. Context-aware captions from context-agnostic supervision. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1070–1079. IEEE.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015a. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015b. Cider: Consensus-based image description evaluation. In *CVPR*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Catherine Wah, Grant Van Horn, Steve Branson, Subhransu Maji, Pietro Perona, and Serge Belongie. 2014. Similarity comparisons for interactive fine-grained categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–866.

Christian Walder and Dongwoo Kim. 2018. Neural dynamic programming for musical self similarity. In *International Conference on Machine Learning*, pages 5092–5100.

Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016. Learning deep structure-preserving image-text embeddings. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5005–5013.

Ruize Wang, Zhongyu Wei, Piji Li, Haijun Shan, Ji Zhang, Qi Zhang, and Xuanjing Huang. 2019a. Keep it consistent: Topic-aware storytelling from an image stream via iterative multi-agent communication. *arXiv preprint arXiv:1911.04192*.

William Yang Wang, Sameer Singh, and Jiwei Li. 2019b. Deep adversarial learning for nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 1–5.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 899–909.

R Weide. 1998. The CMU pronunciation dictionary, release 0.6. *Carnegie Mellon University*.

Sean Welleck, Kianté Brantley, Hal Daumé III, and Kyunghyun Cho. 2019a. Non-monotonic sequential text generation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*.

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. 2019b. Dialogue natural language inference. In *ACL*.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org.

Jason Weston, Emily Dinan, and Alexander H. Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. In *SCAI@EMNLP*.

Gerhard Widmer, Maarten Grachten, and Stefan Lattner. 2018. Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints. *Journal of Creative Music Systems*, 2(2).

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Olivia Winn and Smaranda Muresan. 2018. 'lighter'can still be dark: Modeling comparative color descriptions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 790–795.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in Data-to-Document Generation. *arXiv preprint arXiv:1707.08052*.

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

Xiaofeng Wu, Naoko Tosa, and Ryohei Nakatsu. 2009. New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. In *International Conference on Entertainment Computing*, pages 191–196. Springer.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation. *CoRR*, abs/1904.12848.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, pages 451–466. Springer.

Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *EMNLP*.

Jun Xu, Zeyang Lei, Haifeng Wang, Zheng-Yu Niu, Hua Wu, and Wanxiang Che. 2020a. Enhancing dialog coherence with event graph grounded content planning. In *IJCAI*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81.

Minghong Xu, Piji Li, Haoran Yang, Pengjie Ren, Zhaochun Ren, Zhumin Chen, and Jun Ma. 2020b. A neural topical expansion framework for unstructured persona-oriented dialogue generation. *CoRR*, abs/2002.02153.

Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. 2013. i, poet: Automatic chinese poetry composition through a generative summarization framework under constrained optimization. In *IJCAI*, pages 2197–2203.

Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. 2017. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *ISMIR*.

Xu Yang, Hanwang Zhang, and Jianfei Cai. 2019. Learning to collocate neural modules for image captioning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018a. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. 2018b. Unsupervised text style transfer using language models as discriminators. *arXiv preprint arXiv:1805.11749*.

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*.

Massimo Zanetti and Lorenzo Bruzzone. 2016. A generalized statistical model for binary change detection in multispectral images. In *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*, pages 3378–3381. IEEE.

Daojian Zeng, Haoran Zhang, Lingyun Xiang, Jin Wang, and Guoliang Ji. 2019. User-oriented paraphrase generation with keywords controlled network. *IEEE Access*, 7.

Li Zhang, Flood Sung, Feng Liu, Tao Xiang, Shaogang Gong, Yongxin Yang, and Timothy M Hospedales. 2017. Actor-critic sequence training for image captioning. *arXiv preprint arXiv:1706.09601*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018a. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018b. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*, pages 2204–2213.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *ICLR*.

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.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931*.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskénazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*.

Tingting Zhao, Hirotaka Hachiya, Gang Niu, and Masashi Sugiyama. 2011. Analysis and improvement of policy gradient estimation. In *NIPS*.

Deyu Zhou, Linsen Guo, and Yulan He. 2018a. Neural storyline extraction model for storyline generation from news articles. In *NAACL-HLT*.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018b. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. *CoRR*, abs/1802.01886.

Yutao Zhu, Ruihua Song, Zhicheng Dou, Jian-Yun Nie, and Jin Zhou. 2020. Scriptwriter: Narrative-guided script generation. In *ACL*.

A Data and Generated Samples

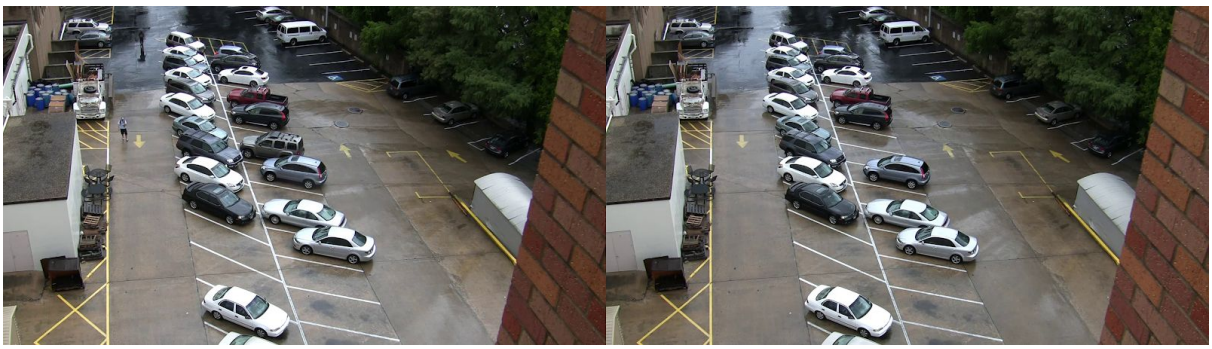
A.1 Spot the difference (Chapter 2): Data Samples

We provide some additional random data samples from our dataset in the following pages. For each sample, we show the the image pairs in the data point, and corresponding crowd-sourced annotation (often containing multiple sentences). We additionally provide corresponding *diff* images.

1.

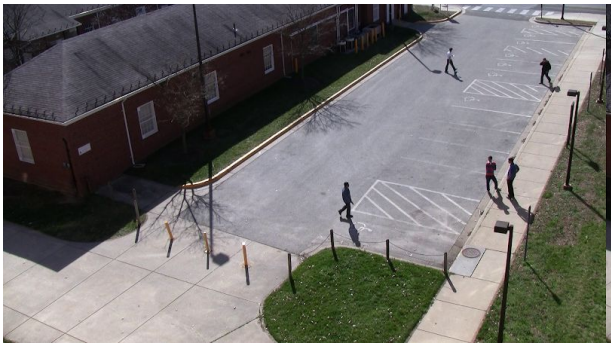
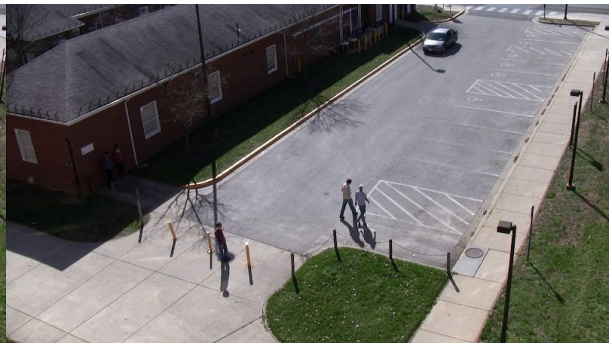

Man by yellow poles in after pic wasn't there before.
There are two people in middle of court that were not there earlier.
Person crossing crosswalk is no longer there

2.



The person in the blue shirt is no longer present.
The dark grey SUV is gone.

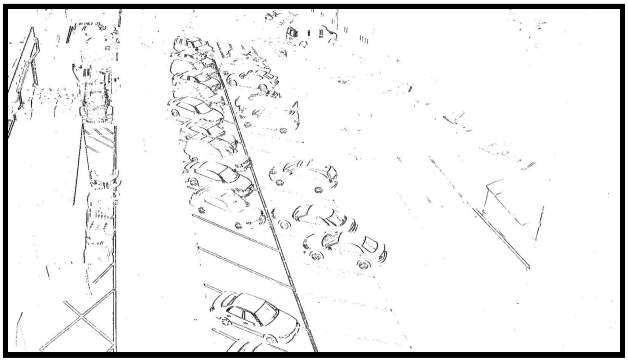
3.

The car in the upper left corner is gone.
The two people in the upper part of the picture are gone.
The person near the handicapped parking place is no longer there.
There appears to be a figure, maybe man that was near handicapped parking place, standing near a pole in the bottom of the frame.
The two people who were on the sidewalk are no longer present.
There are two people now present near the handicapped parking place.

4.

I do not see any noticeable differences in the above photos.



5.

On the right there is 6 people and in the left there are 5 people.

A.2 Persona Grounded Dialog (Chapter 3): Sample Generations

Tables A.1 A.2 A.3 A.4 present generations from COMPAC for sampled dialog histories with comparison across baselines.

Persona: I've short hair. I have a pet dog. I like to go hiking.
User: Do you follow current fashion trends?
GPT2 (Wolf et al., 2019)**:** Yes. What about you?
COMPAC-original: I have a pet dog.
COMPAC-revised: I think you really need to go shopping.
COMPAC-paraphrase: I like shopping.
COMPAC: Yes, I prefer to be fashionable.

Table A.1: Sample generations by different models.

Persona: I just want to have fun with my friends. I don't drink or do drugs or anything. I am 19 and I cannot wait to move out of my parents home.
User: Are you enjoying life ?
GPT2 (Wolf et al., 2019)**:** Yes! What about you?
COMPAC-original: Hey yes! I don't drink.
COMPAC-revised: Yes. I like to go out with friends.
COMPAC-paraphrase: I don't want to be alone.
COMPAC: Not really. I want to have a party.

Table A.2: Sample generations by different models.

Persona: I've rainbow hair. I am an animal activist. I spend my time bird watching with my cats.
User: What motivates you to work with animals?
GPT2 (Wolf et al., 2019)**:** I love bird-watching.
COMPAC-original: I love bird-watching. What about you?
COMPAC-revised: I love animals.
COMPAC-paraphrase: I am in love with animals.
COMPAC: I want to protect animals to make a difference in this world.

Table A.3: Sample generations by different models.

Persona: I'm always wearing a hat. I've a broken arm. My friends are all skateboarders.
User: Do you skate? I do.
GPT2 (Wolf et al., 2019)**:** Yes. How about you?
COMPAC-original: I wear a hat.
COMPAC-revised: Yes. My friends are skateboarders.
COMPAC-paraphrase: That's great. My friends are skateboarders.
COMPAC: My friends and I go to the park for skateboarding.

Table A.4: Sample generations by different models.

A.3 Neuro-Symbolic Structures for Time-Series Data (Chapter 4): Additional Details

A.3.1 Additional Details on Data Sets

A downloadable json file for each of the two datasets is provided in the github repository [1].

Synthetic Data

Our synthetic time series data is constructed such that each time series has exactly one of the following 6 patterns: increases-in-beginning, increases-in-middle, increases-in-end, decreases-in-beginning, decreases-in-middle, decreases-in-end. The position in which the pattern is placed is based on the temporal choice (begin/middle/end). i.e. L must lie withing first one-third of the time-series (0,T/3) in case of 'begin' pattern, should lie in middle one-third for 'middle', and last one third for 'end' respectively. We consider equation a*x+b of a line, where 'a' represents the slope and 'b' represents the y-axis intercept. We pick a random slope value between 0 and 2, and a random intercept value between 1 and 20. Finally, we pick $|L|$ random integral values for x such that ax+b point lies between 0 and 1. The points in the time series outside the pattern are fixed to be same as the nearest point in the patter. Finally, small noise is added to each point using U(-2,2).

Some random data samples are shown in Fig. A.1. The text corresponding to 'HUMAN' marker represents one of the collected annotations for the corresponding time series data.

STOCK data

Figures A.2 show data samples for STOCK dataset. The text corresponding to 'HUMAN' marker represents one of the collected annotations for the corresponding time series data. The total number of unique words (considering train and validation splits) are 861, out of which only 560 words occur more than once in the dataset.

A.3.2 Additional Results

SYNTH: Generated Samples

Additional examples are provided in Figure A.1.

STOCK: Generated Samples

Figure A.2 shows some generated samples on STOCK dataset.

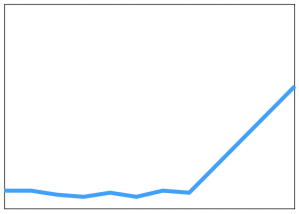
Validation Split Results

Tables A.5 and A.6 show automated metrics on the validation split.

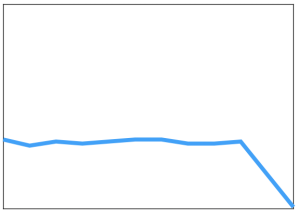
Analyzing Learned Modules

Figure A.3 shows visualization of a learned *locate* module when model is trained on SYNTH data.

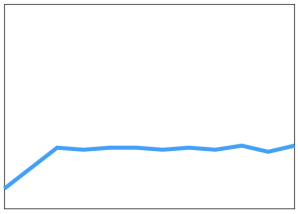
[1] https://github.com/harsh19/TRUCE



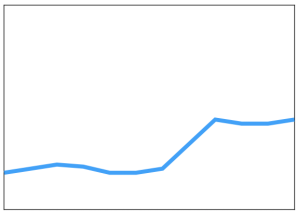
HUMAN: Increases dramatically in the second half
TRUCE: Increases at the end
LSTM-ENC: Increases in the middle



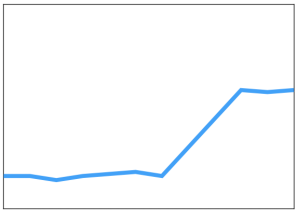
HUMAN: consistent for the first three quarters
TRUCE: Decreases at the end
FFT-ENC: Decreases in the middle



HUMAN: slightly climbs up at the start
TRUCE: Increases in the beginning
FF-ENC: Increases at the end



HUMAN: increase in the middle
TRUCE: Increases in middle
CONV-ENC: Increases at the end



HUMAN: Steadily increases in the middle
TRUCE: Increases at the end

Figure A.1: SYNTH: Data and Generated Samples. The captions marked in red were judged as incorrect by human annotators. TRUCE achieves very high precision of 95% on outputs for the test split of SYNTH dataset.

Method	PPL	Bleu-3/4	Cider	Rouge	BERT
TRUCE	9.02	0.61/0.50	1.92	0.74	0.76
FCENC	9.66	0.41/0.34	1.17	0.63	0.57
LSTMENC	7.5	0.43/0.35	1.39	0.63	0.63
CONVENC	7.6	0.63/0.53	1.99	0.73	0.71
FFTENC	15.7	0.39/0.29	1.26	0.61	0.62
NEARNBR	NA	0.32/0.19	0.68	0.50	0.48

Table A.5: Results on validation split for SYNTH dataset.

Additional Ablation Studies

We consider following ablations for the TRUCE: (1) TRUCE-NOINF: Train TRUCE without the use of inference network (2) TRUCE-NOHEUR: Train TRUCE without the use of heuristic labels

A.3.3 Additional Training Details

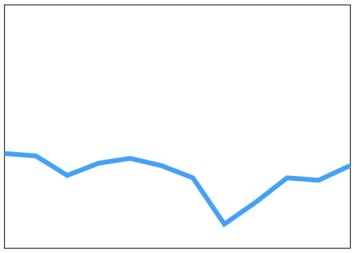
We code our models in Pytorch library.

Heuristic Labels

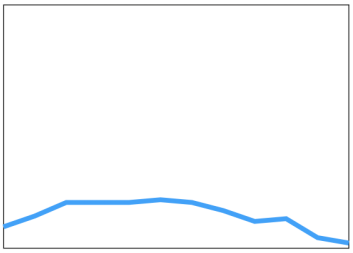
List of the keywords selected for use in constructing heuristic labels:
— 'locate':['beginning','middle','end','throughout'],
— 'pattern':['increase','decrease','peak','flat','dip']

Optimizer

We use Adam optimizer with initial learning rate of $1e-4$.



HUMAN: Increases towards the end
TRUCE: Decreases in the middle
RETRIEVAL: Steadily decreases in the first three quarters
FF-ENC: Stays flat at the end
LSTM-ENC: Increases at the end
CONV-ENC: Decreases at the end
FFT: Decreases at the end



HUMAN: consistent for the first three quarters
TRUCE: Increases at the beginning
RETRIEVAL: Fairly flat throughout
FF-ENC: Decreases at the end
LSTM-ENC: Stays flat at the end
CONV-ENC: Remains flat throughout
FFT: Decreases at the end

Figure A.2: STOCK: Data and Generated Samples. The captions marked in red were judged as incorrect by human annotators. (Best viewed in color)

Method	Bleu-3/4	Cider	Rouge	BERT
TRUCE(Ours)	0.36 / 0.22	0.40	0.50	0.58
FcEnc	0.32 / 0.20	0.38	0.47	0.56
LstmEnc	0.34 / 0.18	0.33	0.51	0.61
ConvEnc	0.34 / 0.17	0.35	0.50	0.60
FftEnc	0.32 / 0.18	0.36	0.48	0.56
NearNbr	0.11 / 0.05	0.11	0.27	0.37

Table A.6: Results on validation split of STOCK data.

Infrastructure

We use GeForce RTX 2080 GPUs for training models.

Additional method details

While the automated metrics are only moderately correlated with quality, we found it reasonable to select best model configurations based on the Bleu-4 scores on validation split. The model configurations, when using STOCK dataset, are as follows:

- LSTM Decoder: Token embedding size and hidden size are varied from the set {32,64,128,256}.
- Weight for the classification loss term (in case of multitask objective in baselines): Following three weights of classification loss (i.e. the weight of the classification term which is present in addition to the conditional language modeling objective) are tried: 0.3,1.0,3.0.
- TRUCE: Program embedding encoding size. Number of module instantiations are varied in following ranges:
  - LOCATE: 4-7 instantiations of each of locate
  - PATTERN: 6-10 instantiations of each of trend
  - COMBINE: 1 instantiation

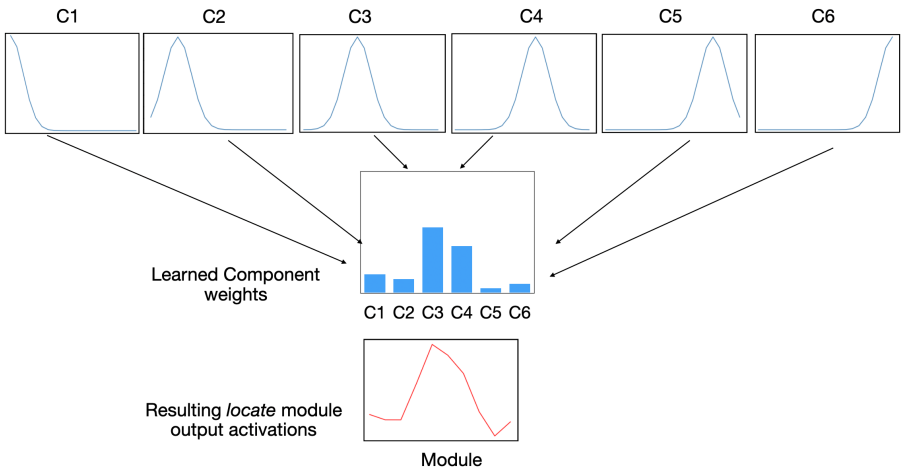


Figure A.3: Visualizing a learned 'locate' module. Our locate modules are weighted mixtures of equally spaced Gaussians. The module's weight on each of these components is shown, along with the resulting distribution – the module being visualized seems to have learned to focus on middle part of the time series.

- Module embedding is varied in the set $\{9,18,36,72\}$. Final module embedding size is 18.
- Number of trainable parameters: 466K (excluding inference network parameters since inference network is used only at training and not at prediction time)

- FFTENC: - Number of trainable parameters: 462K - Construct features based on numpy:fft:rfft functions, using real as well as imaginary components from the transformation.

- CONVENC: Number of trainable parameters: 463K

- LSTMENC: - Representation: A single LSTM step involves feeding an embedding of the input and using the previous step's hidden state. To construct an input embedding of size h for a given number x_t, we simply repeat the number x_t for h times.
  - Number of trainable parameters: 464K

- NEARNBR: We experiment with L2 distance and L1 distance, and observed former to perform better in terms of automated as well as human evaluations.

A.4 Discrete Latent Generation Plan (Chapter 6): Sample Generations

Some additional generated samples from various models are shown in Table A.7. We note that LAP-CINF-UDEV plans often exhibits good control over the generated story. For example, samples S3 and S4 samples in Table A.7 for the same title by-and-large follow the generated plan. We do observe a certain degree of repetition in some samples e.g in sample S2, the first and third sentences both discuss mowing the lawn.

Sample S6 further demonstrates the generation order for LAP-CINF-CDEC. Each sentence begins by copying the corresponding anchor word, generating words to the left and then to the right of it. Thus LAP-CINF-CDEC model commits to using the corresponding anchor words. In contrast LAP-CINF-UDEV has more flexibility in using the anchor words, and sometimes anchor words get dropped or get used in a paraphrased manner. e.g. for sample S2, the model did not use the word 'tired' in the last story sentence, though the second last sentence evokes similar understanding. However, this also means the control of the story via the plan is somewhat weaker. e.g. for S4, the story does not use the anchor word 'boiled'. Finally, compared to outputs from LAP-CINF-UDEV and LAP-CINF-CDEC, we observe that SUPERVPLAN outputs demonstrate much weaker control of the plan on the story, and generate plans less relevant to the title.

.

S.no.	Method	Title	Story
1	LAP-CINF-UDEV	**TITLE:** **ANCHOR WORDS:** **STORY:**	race participate high challenged sure won Jill wanted to participate in a race. She was in high school and had no idea how to race. Jill challenged her friend to a race. She wasn't sure if she would win. Ultimately she won the race.
2	LAP-CINF-UDEV	**TITLE:** **ANCHOR WORDS:** **STORY:**	lawn mowing lawn mowed every tired I was mowing my lawn one day. I was on the lawn by myself. I mowed the lawn. I worked on it for hours. Eventually I was able to go back to work.
3	LAP-CINF-UDEV	**TITLE:** **ANCHOR WORDS:** **STORY:**	the dryer load cold wash everything long A girl got a load of laundry. She put everything in the dryer. She wanted to wash it off. She did everything but the washing machine was not so good. She had to take a long time to clean it up.
4	LAP-CINF-UDEV	**TITLE:** **ANCHOR WORDS:** **STORY:**	the dryer cooking boiled dried put cooking There was a pot of beans in the microwave. The pot was piled on the table. They dried it and cooked it. They put the noodles in the oven. The kids were all cooking dinner.
5	LAP-CINF-UDEV	**TITLE:** **ANCHOR WORDS:** **STORY:**	lost phone bar opened someone left found Tim was at the bar. He opened the door to find a phone. It was of a person. Tim left for a couple of minutes. He took it home and it turned out that he lost it.
6	LAP-CINF-CDEC	**TITLE:** **ANCHOR WORDS:** **STORY:** **STORY (reordered):**	race entered win losing trained win entered tim <start> a local gym. win to wanted he <start>. losing up wound he <start> the race. trained tim <start> to get an extra. win to decided tim <start> the race. Tim entered a local gym. He wanted to win. He wound up losing the race. Tim trained to get an extra. Tim decided to win the race.
7	LAP-UINF-UDEC	**TITLE:** **ANCHOR WORDS:** **STORY:**	race forms forms forms forms forms Cathy and his friends decided to go to the fair. Cathy was worried about the show that she was having to be on the track team. Cathy was worried that the race would have to be over. Cathy was nervous to win by a point of winning the race. Cathy caught the race.
8	SUPERVPLAN	**TITLE:** **ANCHOR WORDS:** **STORY:**	race today good day race race I was the fastest runner on my high school track team. It was very hard to win, but my friend convinced me to do it. I practiced with the runners and practiced every day. I practiced every day after school. I can go to the race every day.
9	SUPERVPLAN	**TITLE:** **ANCHOR WORDS:** **STORY:**	lost phone package wanted found battery missed Tom was getting a new phone. He wanted to buy a new one. When he found it it was still a broken battery. Tom was sad because it was broken. Tom tried to save it but it wasn't enough.

Table A.7: Samples from different methods (with minor edits such as captalization)

A.5 Retrieved Snippets as Guiding Plans (Chapter 7): Sample Outputs

Tables 8 and 9 show generations from PABST and baselines for sampled text examples.

Dialog History Agent: I volunteer in shelters. User: It is a noble act.
COMPAC: Thank you. I like helping people.
NEARNBR: I was at my house bored. I couldn't figure out what to do. I looked on TV for answers. I saw a commercial about homeless people. So I decided to volunteer at a homeless shelter.
PSEUDO: Last week, I went to the local animal shelter on the pet adaptation day.
MULTITASK: Thank you! What about you?
PABST ($\lambda_d = 5.0$): I was bored. I wanted to help people. I looked on TV about homeless people. So I decided to volunteer at a homeless shelter.
PABST ($\lambda_d = 1.0$): Thank you. I was inspired by a TV commercial and I volunteered at the homeless shelter.

Table 8: Generations from different models

Dialog History Agent: I enjoy American sports. What sports do you like? User: I love playing volleyball. I was in the university team.
COMPAC: That's great. I like adventure sports.
NEARNBR: I always thought golf was for old people. One day, my dad invited me to play golf with him. I reluctantly accepted his invitation. On the golf range, I saw lots of people of my age. I realized golf was a sport for everyone to enjoy regardless of age.
PSEUDO: Last week, I went to play golf. My dad invited me. I was reluctant.
MULTITASK: That's great! I play volleyball during weekend.
PABST ($\lambda_d = 5.0$): I like playing golf. I always thought it is for old people. Recently I went to a golf range. But I saw lots of people of my age. I realized golf is a sports for everyone.
PABST ($\lambda_d = 1.0$): That's great. I like playing golf. I always thought it is for old people. Recently I went to a golf range. But I saw lots of people of my age. I realized golf is a sports for everyone.

Table 9: Generations from different models