Dialog Response Generation Using Adversarially Learned Latent Bag-of-Words

by

Kashif Khan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see *Statement of Contributions* included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Chapter 3 is based on the following paper:

- Kashif Khan and Gaurav Sahu and Vikash Balasubramanian and Lili Mou and Olga Vechtomova. "Conditional Response Generation Using Variational Alignment." arXiv preprint arXiv:1911.03817 (2019).

I have contributed to implementation, experimentation, and preparation of the manuscript of the above mentioned paper.

Abstract

Dialog response generation is the task of generating response utterance given a query utterance. Apart from generating relevant and coherent responses, one would like the dialog generation model to generate diverse and informative sentences.

In this work, we propose and explore a novel multi-stage dialog response generation approach. In the first stage of our proposed multi-stage approach, we construct a variational latent space on the bag-of-words representation of the query and response utterances. In the second stage, transformation from query latent code to response latent code is learned using an adversarial process. The final stage involves fine-tuning a pretrained transformer based model called text-to-text transfer (T5) (Raffel et al., 2019) using a novel training regimen to generate the response utterances by conditioning on the query utterance and the response word learned in the previous stage.

We evaluate our proposed approach on two popular dialog datasets. Our proposed approach outperforms the baseline transformer model on multiple quantitative metrics including overlap metric (Bleu), diversity metrics (distinct-1 and distinct-2), and fluency metric (perplexity).

Acknowledgements

I would like to thank my supervisor Prof. Olga Vechtomova. Her continual guidance and feedback during my thesis project and other research projects has been extremely invaluable. She has been patient and accommodating while guiding me towards my research goals.

I am extremely grateful to Prof. Pascal Poupart and Prof. Mei Nagappan for agreeing to read my thesis and provide vital feedback.

I would also like to thank Vikash Balasubramanian and Gaurav Sahu for the work we have done jointly and for all the interesting discussions we have had which helped further my knowledge.

Finally, I would like to acknowledge and thank my family and friends who have supported me and motivated me to pursue my goals.

Dedication

I dedicate this thesis to my family who have believed in me and supported all my endeavours unconditionally.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Definition

The main goal in dialog response generation task is to generate response utterance given a query utterance and the dialog history in the form of all the previous utterances. The generated response utterance has to not only be coherent and fluent but relevant to the query uttterance as well. Furthermore, information from the dialog history has to be taken into account as well. Finally, an effective dialog response generation model should generate diverse and informative responses rather than safe and generic responses. In this work, we propose and explore a novel multi-stage dialog response generation approach.

1.2 Thesis Structure

We organize the rest of this work in the following way:

- We cover important background concepts related to our proposed approach in Chapter 2.
- In Chapter 3, we describe the related work done on the dialog generation task and describe the VAE-AM model which is an approach proposed in a previous joint work which provides the foundation for the approach proposed in this work.

- Chapter 4 contains details about the proposed dialog response generation approach. We also report details about the experiments performed to evaluate the proposed approach and results obtained.
- We conclude our work by summarizing it in chapter 5 and providing information about the future work to be performed on the proposed approach.

1.3 Contributions

We propose a novel multi-stage approach for dialog response generation. We evaluate our proposed approach on two dialog datasets and compare its performance against two state of the art models and a baseline transformer model. We show using quantitative metrics that our proposed approach generates fluent and diverse responses. Our proposed approach outperforms the state of the art models in quantitative diversity metrics and outperforms the baseline transformer model on all the quantitative evaluation metrics. As part of our multi-stage approach, we propose a novel fine-tuning regimen for a large transformer based pretrained model called text-to-text transfer (T5) (Raffel et al., 2019). The fine-tuning trains T5 to generate response utterance conditioned on the the query utterance and a set of words which are to be used in the response utterance generation. We show empirically that our fine-tuned T5 model generates response utterance using the provided response words.

Chapter 2

Background

2.1 Natural Language Generation

Natural Language Generation (NLG) is one of the major research area under the broader field of Natural language Processing (NLP). Natural Language Understanding (NLU) being the other major sub-field of NLP under active research. The main goal in natural language generation is to generate grammatically correct and coherent sequences of words in one of the many currently existing human languages. In the past, natural language generation systems have predominantly relied on template or rule based generation techniques (Reiter & Dale, 1997). Additionally, probabilistic models such n-gram or log-linear models have also been popular (S. F. Chen & Goodman, 1996; Koehn et al., 2003). In recent years, however, neural network based models have attained state-of-the-art results in various natural language generation tasks.

The field of natural language generation encompasses multiple applications including but not limited to the following applications:

- In **Neural Machine Translation** the main objective is to convert the text given in source language into semantically equivalent text of target language (Bahdanau et al., 2014; Luong et al., 2015).
- **Summarization** involves processing a given body of text and producing a smaller body of text containing only the most salient information contained in the original text (Kryściński et al., 2019; Nallapati et al., 2016).

- **Text Simplification** in which the objective is to simplify a given text such that the generated text is simpler in terms of prose structure and grammar used while still containing the same meaning and information (Nisioi et al., 2017; Surya et al., 2018).
- In **Dialog Generation** the main goal is to generate coherent and relevant response conditioned on the given utterance (Gu et al., 2018; J. Li et al., 2016; Serban, Sordoni, Bengio, et al., 2016).

2.2 Feedforward Neural Networks

Feedforward neural networks are one of the most commonly used neural network in deep learning models and form the basis for more advanced and specialized deep neural networks such as **recurrent neural networks (RNNs)** and **convolutional neural networks (CNNs)**. Feedforward neural networks are sometimes also called **multilayer perceptrons (MLP)** or **Deep feedforward networks**. As one of its alternative name suggests, feedforward neural networks were inspired by perceptrons, originally proposed by Rosenblatt in 1957 (Rosenblatt, 1957). Feedforward neural networks are used as function approximators. Let f^* be some function which maps input $\mathbf{x}$ to output y i.e. $f^*(\mathbf{x}) = y$, then a feedforward neural network can be thought of as a mapping $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$, where f is an approximator of f^* and $\boldsymbol{\theta}$ represents the learned neural network parameters which help achieve the best approximation performance.

A feedforward neural network typically consists an **input layer**, an **output layer** and one or more **hidden layers**. These networks can be represented as directed acyclic graphs. We provide an example visual representation of feedforward neural network in figure 2.1. During a forward pass of the network, the data passes from the input layer to the first hidden layer and then from first hidden layer to the second hidden layer and so on until the final hidden layer passes the data to the output layer. The data flow direction is always forward, hence the name feedforward neural network. The intermediate layers are called **hidden layers** because they are used to compute intermediate representations of the data and do not produce the desired output. Each layer in the feedforward neural network is made up of **units**. The input to these networks are typically vector-valued and input layer units compute identity function on a specific individual dimension of the input vector. Each unit in a layer is typically connected to every unit in the subsequent layer by an edge with a weight. Each unit in the hidden layers of the network receives the linear combination of the previous layer's units and their connecting edge weights to which it applies a linear or non-linear **activation function**. The output produced by the activation function serves as the value of that particular unit.

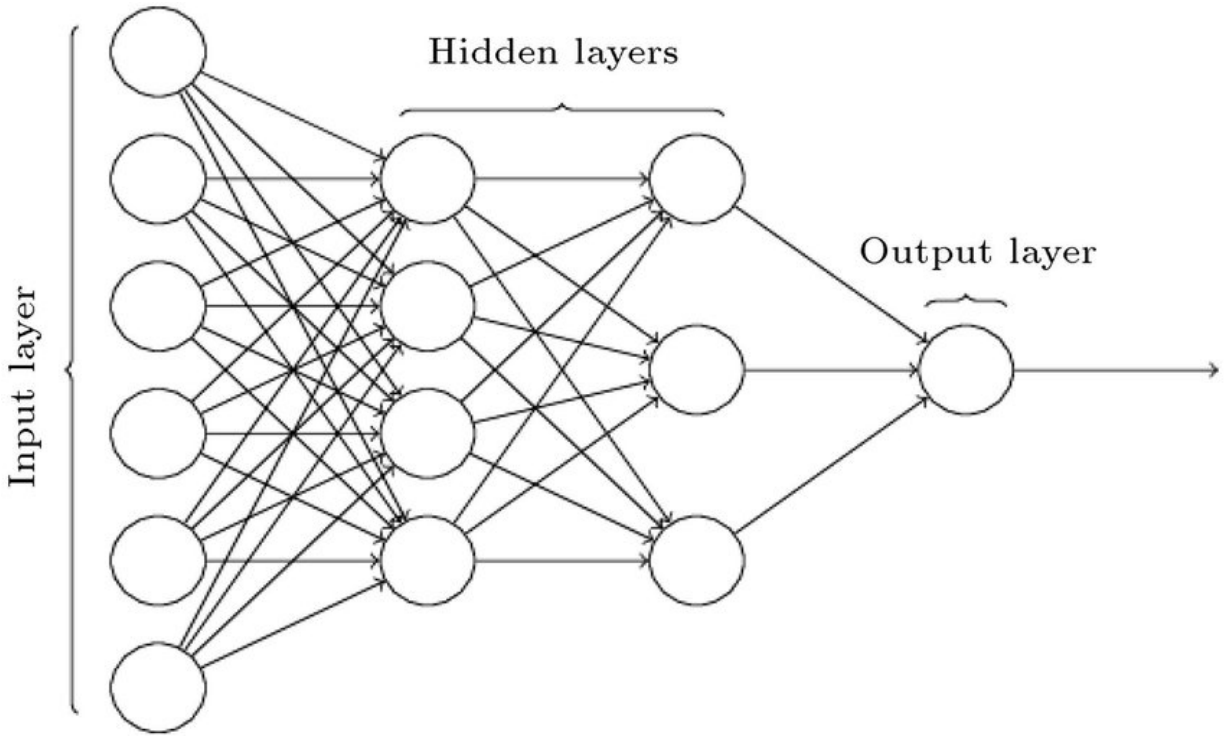


Figure 2.1: An Example of a feedforward neural network. [Source]

The number of hidden layers in the network determines its **depth** and the number of units in the hidden layers determines its **width**. Both the depth of the network and width of the network represent hyperparameters and their optimal values are determined through experimentation. The weights associated with the edges are called parameters of the neural network and are learned during the training of the neural network. An optimization algorithm such as **stochastic gradient descent (SGD)** along with an automatic differentiation technique called backpropagation is used to train the network.

The function being approximated by the neural network is called the **loss function**. Most commonly used loss functions are **mean squared error (MSE)** or **negative log likelihood (NLL)**.

Let $\mathcal{L}$ be the loss function, $\mathbf{x}$ be the input data, and $\mathbf{w}$ represent the randomly initialized parameters of a neural network. Then during the training, each iterative update of weights is performed as follows:

$$\mathbf{w_{i+1}} = \mathbf{w_i} - \eta \cdot \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{x}; \mathbf{w_i}) \tag{2.1}$$

where η is a hyperparameter called learning rate and $\nabla_{\mathbf{w}}$ is the gradient of the loss function with respect to the weights.

2.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are another type of artificial neural networks which evolved from feedforward neural networks and are particularly effective in problems involving variable length sequential data such as text, audio, etc. These networks are used for processing sequential data where the data element at **position** or **time step** t in the sequence is dependent on all the data elements upto time step $t-1$ in the given sequence. The goal of a recurrent neural network is to learn to predict the data element at time step t when it is provided the sequence of length $t-1$. For example, a recurrent neural network can be trained to predict the next word when given all the prior words in sentence e.g. when given the following sequence of words: "I will talk with you", the network predicts the next word in the sequence would be "tomorrow".

The two main features which allow recurrent neural networks to model variable length sequential data effectively are **parameter sharing** and **recurrent connections** between hidden units. Parameter sharing refers to the fact that recurrent neural network processes the input sequence one element at a time using the same neural network. Thus, same parameters are shared across time steps allowing recurrent neural networks to handle variable length input sequences with ease. Furthermore, each hidden unit in the network has a recurrent connection with itself allowing it to pass information about the previously observed sequence at each new time step. This allows the network learn the dependencies between the elements in sequence. We provide a graphical depiction of a basic one hidden unit unrolled recurrent neural network in figure 2.2.

We explain the notation used in the figure 2.2 below:

- x_t is the input at the time step t.
- o_t is the output at the time step t computed using the hidden state h_t at the time step t. The exact computation is dependent on the objective of the recurrent neural network. For example, in a language modeling task one would want the output to be a vector of probabilities across the vocabulary: $o_t = \text{softmax}(Wh_t)$.

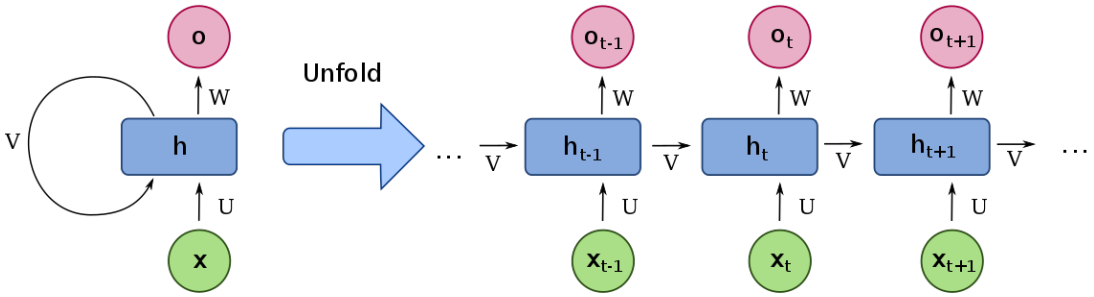


Figure 2.2: A basic one-unit recurrent neural network. [Source]

- U, V, and W are weights shared across all the time steps.
- h_t is the hidden state value at the time step t computed using the input x_t at the time step t and the hidden state h_{t-1} from the previous time state in the following manner:

$$h_t = f(Ux_t + Wh_{t-1}) \tag{2.2}$$

where f is typically an activation function such as tanh or ReLU.

As mentioned earlier, recurrent neural networks are effective at modeling sequential data and are frequently used in natural language processing problems. They are used both for natural language understanding and generation tasks (Bengio et al., 2003; Mikolov et al., 2010; Morin & Bengio, 2005). In natural language understanding problems, RNNs are used to obtain representation for textual input which can then be used perform tasks such as question-answering, sentiment classification, etc. They are also used as language models to perform natural language generation tasks.

2.3.1 Long Short Term Memory

One of the major shortcoming of recurrent neural networks is the problem of **vanishing or exploding gradients**. The problem of vanishing or exploding gradients arises in deep neural networks with deep computational graphs. Since recurrent neural networks repeatedly apply the same operation using the same parameters at each time step of typically

long temporal sequences, it constructs a very deep computational graph. For example, let's say that we apply the operation of matrix multiplication of parameters denoted by $\mathbf{W}$ at each time step. At time step t, it would be the same as multiplication by $\mathbf{W}^t$. Let the eigenvalue decomposition of $\mathbf{W} \coloneqq \mathbf{V}\text{diag}(\boldsymbol{\lambda})\mathbf{V}^{-1}$. Then,

$$\mathbf{W}^t = (\mathbf{V}\text{diag}(\boldsymbol{\lambda})\mathbf{V}^{-1})^t = \mathbf{V}(\text{diag}(\boldsymbol{\lambda}))^t\mathbf{V}^{-1} \tag{2.3}$$

This clearly illustrates the problem, any eigenvalue λ_i which is not close to 1 will either explode if it is > 1 in magnitude or vanish if it is < 1 in magnitude. Since the gradients in such computational graphs are also scaled according to $\text{diag}(\boldsymbol{\lambda})^t$, this also causes the gradients to explode or vanish as well.

Apart from vanishing or exploding gradients, recurrent neural networks also have difficulty learning long-term dependencies due to assigning exponentially smaller weights to long-term interactions as opposed to short-term interactions (I. Goodfellow et al., 2016).

Various solutions have been proposed with varying degree of success to address these shortcomings of recurrent neural networks such as leaky units, adding skip connections through time, removing connections, etc. But the most effective and frequently used techniques have been the so called **gated RNNs** such as **long short term memory** and **gated recurrent unit**. The main idea behind gated RNNs is to create paths through time with gradients which neither vanish nor explode. This is accomplished by using connection weights which may change at each time step. Furthermore, gated RNNs have mechanism to reset the old state allowing them retain only the most useful information.

Long short term memory (LSTM) were proposed by Hochreiter et al. in 1997 (Hochreiter & Schmidhuber, 1997). The main idea which allowed LSTMs to alleviate the vanishing or exploding gradients problem was to introduce self-loops within the unit to create paths where gradients could flow for longer duration. Figure 2.3 shows a block diagram of LSTM unit.

The LSTM unit consists of the following gates: the input gate ($\mathbf{g}_t$), the forget gate ($\mathbf{f}_t$), and the output gate ($\mathbf{q}_t$). Each of these gates are sigmoid activation functions which produce a value between 0 and 1. The output of these adaptive gates basically produce a weight which regulates how much of the incoming information and the previously stored information is retained. We provide the equations for these gates below:

$$\mathbf{g}_t = \sigma(W_\mathbf{g} \cdot x_t + U_\mathbf{g} \cdot h_{t-1} + b_g) \tag{2.4}$$

$$\mathbf{f}_t = \sigma(W_\mathbf{f} \cdot x_t + U_\mathbf{f} \cdot h_{t-1} + b_f) \tag{2.5}$$

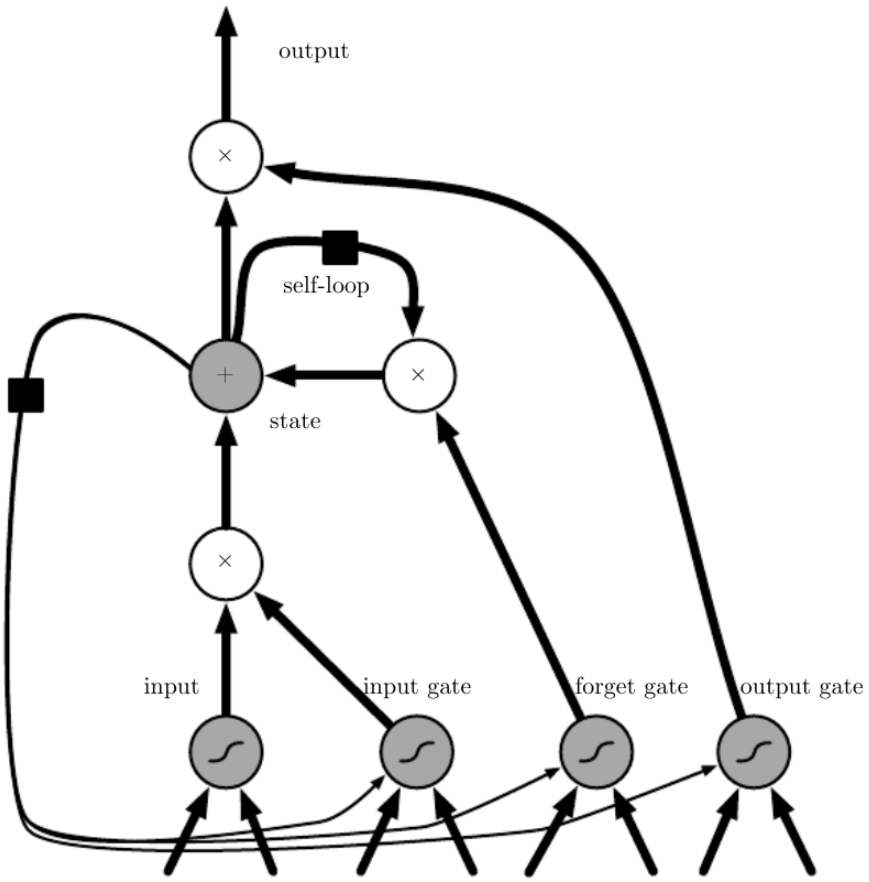


Figure 2.3: Block diagram of a LSTM Unit (I. Goodfellow et al., 2016)

$$\mathbf{q}_t = \sigma(W_{\mathbf{q}} \cdot x_t + U_{\mathbf{q}} \cdot h_{t-1} + b_q) \tag{2.6}$$

where σ denotes the sigmoid activation function, x_t the current input, h_{t-1} the hidden state for previous time step. Each gate has their own weights and bias parameters which we denote using U, W, b (the subscripts identify the specific gate).

The cell state is updated using:

$$\mathbf{s}_t = \mathbf{f}_t \otimes \mathbf{s}_{t-1} + g_t \otimes \sigma\Big(b + U \cdot x_t + W \cdot h_{t-1}\Big) \tag{2.7}$$

where $\mathbf{s}_{t-1}$ is the internal cell state of the previous time step and b, U, and W respectively denote the biases, input weights, and recurrent weights into the LSTM cell. $\otimes$ refers to the elmentwise multiplication.

Finally, the hidden state for the current timestamp is computed:

$$\mathbf{h}_t = \tanh(\mathbf{s}_t) \otimes \mathbf{q}_t \tag{2.8}$$

LSTMs outperform simple recurrent neural networks in learning long term dependencies which make them suitable for a variety of natural language processing tasks such as neural machine translation (Sutskever et al., 2014), coreference resolution (Lee et al., 2017), reading comprehension (Seo et al., 2016), named entity recognition (Chiu & Nichols, 2016), language modeling (Sundermeyer et al., 2012), etc.

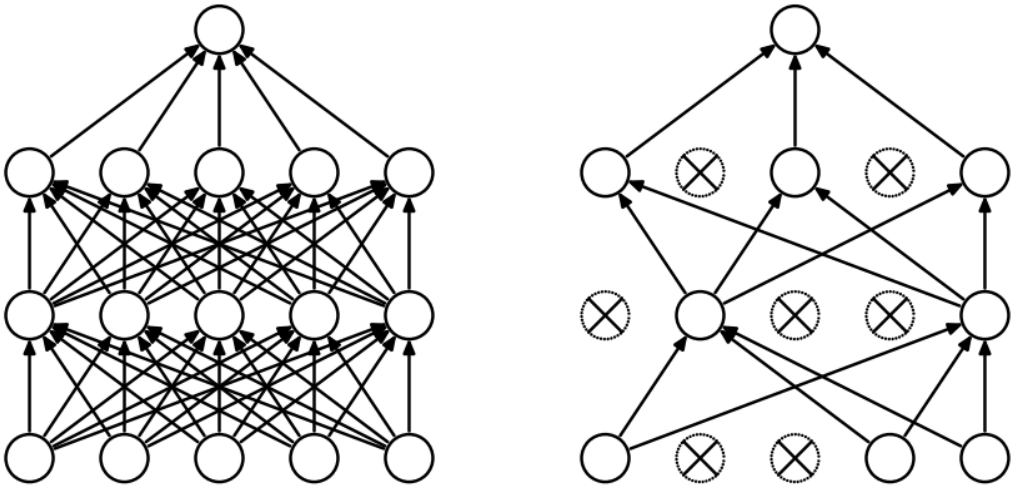
2.4 Regularization

Regularization refers to a variety of techniques used in deep learning models to improve the model's generalization capabilities. Examples of such techniques include parameter norm penalties, dataset augmentation through noise injection, early stopping, dropout, etc.

2.4.1 Dropout

Dropout is a regularization technique proposed by Srivastava et al. (Srivastava et al., 2014) to prevent overfitting on the training data by deep learning model and consequently improving its generalization capabilities.

Dropout can be thought of as a computationally practical way to apply bagging to an ensemble of an exponentially large number of neural networks. Dropout works by dropping non-output units within a neural network during the training with a predefined probability. During test time, the output of the units are scaled by the same predefined probability value to obtain an approximately averaged output from the ensemble of the thinned neural network models. Dropout have been shown to improve the generalization capabilities of neural network model on many computer vision tasks (Srivastava et al., 2014).



(a) Standard Neural Net (b) After applying dropout.

Figure 2.4: A visual representation of the dropout technique (Srivastava et al., 2014)

2.4.2 Batch Normalization

Batch normalization was proposed by Ioffe et al. to improve the training speed and stability of deep neural networks (Ioffe & Szegedy, 2015). Batch normalization works by reducing the so called *internal covariate shift*. Internal covariate shift is defined as the change in the distribution of the network unit outputs due to the change in the network parameter values (Ioffe & Szegedy, 2015). The reduction in internal covariate shift is obtained through batch normalization transformation of a layer's input defined as:

$$\mathbf{y} = \frac{\mathbf{x} - \mathrm{E}[\mathbf{x}]}{\sqrt{\mathrm{Var}[\mathbf{x}]}} * \boldsymbol{\gamma} + \boldsymbol{\beta} \tag{2.9}$$

where $\mathbf{x}$ is the layer inputs and $\boldsymbol{\gamma}$, $\boldsymbol{\beta}$ are learnable parameters.

2.5 Text Representations

For a neural network based model to work with textual data, we need to convert the textual data into a suitable numerical representation before using it as the model's input

and/or output. There are many ways to accomplish this including using **bag-of-words** representations and **Term frequency-inverse document frequency (tf-idf) weighting** statistics representations.

Typically in **bag-of-words** representations, an input text document is represented using an n-dimensional vector, where n is the size of the chosen vocabulary of the whole text corpus. Each index in the n-dimensional vector is used to represent a specific word in the vocabulary and contains either 1 indicating that the word is present in the input document or 0 indicating the absence of the word. Sometimes the word count is used to indicate the presence of a word in the vector representation of the input text document.

Tf-idf weighting statistics representation works similarly to bag-of-words representation with one major difference: for each word in the input document, we replace 1 or the word count in the input document's n-dimensional vector representation with that word's tf-idf weighting.

Term frequency is usually denoted as $\text{tf}(t, d)$ where t refers to the term or word and d is the document under consideration. As the name suggests, most simplest choice is to use the raw word count in the document. Sometimes boolean word frequencies are used instead of the word count: 1 if the word is present in the document, 0 otherwise.

Inverse document frequency (idf) measures the amount of information the word provides. For example, words which are commonly used and are present in high number of documents in the corpus are not likely to be informative at all whereas rare words have high informational value. Idf is usually denoted using $\text{idf}(t, D)$ where D refers to the set of all documents in the corpus. Idf is computed in the following way:

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2.10}$$

where N is the total number of documents in the corpus.

Now, we can compute the tf-idf weight:

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \tag{2.11}$$

2.5.1 Word Embeddings

Although text representation methods such as Bag-of-words representations and tf-idf weighting statistics representations are useful, they do have certain shortcomings. One

such shortcoming is the drastic increase in computational costs when working with large scale diverse text corpus with large vocabulary. Another major shortcoming is the inability to model the semantic and syntactic relationships between different words.

Word embeddings refer to the fixed size continuous vector representations of the words i.e. word embeddings map a word w in the vocabulary V to a vector $v \in \mathbb{R}^n$ where n is called the dimension of the word embedding. Since the dimension of the vector is fixed, the computational costs do not increase with the vocabulary size. Furthermore, the word embeddings usually map the semantically related words in the vocabulary to vectors in the nearby spaces.

Most commonly used word embeddings are Word2Vec proposed by Mikolov et al. (Mikolov et al., 2013) and GloVe by Pennington et al. (Pennington et al., 2014). Word2vec uses a shallow neural network to learn the distributed representations of the words using the following two tasks: (i) **continuous bag-of-word (CBOW)**: the shallow neural network is trained to predict the current word using the words from the context window, and (ii) **continuous skip-gram model (Skip-gram)**: the current word is used to predict the words present in the context window (Mikolov et al., 2013). GloVe method uses a specific weighted least squares model trained on global word co-occurrence statistics to obtain the word representations (Pennington et al., 2014). Pennigton et al. argue that the proposed global log-bilinear regression model has the necessary properties to produce linear directions of meaning and it combines global matrix factorization & local context window methods effectively (Pennington et al., 2014).

2.6 Autoencoders

Autoencoders are neural networks which learn to reconstruct its input. An autoencoder has two main components: an **encoder** and a **decoder**. The input is fed to the encoder to obtain a **latent code**. The decoder takes the latent code as input and tries to reconstruct the corresponding input. Figure 2.5 shows the basic autoencoder architecture.

Formally, an autoencoder can be thought of as a combination of two functions: an encoder function f and a decoder function g. Let $\boldsymbol{x}$ denote the input, $\boldsymbol{h}$ the corresponding latent code, and $\boldsymbol{x}^{'}$ the reconstruction. Then, $\boldsymbol{h} = f(\boldsymbol{x})$ and $\boldsymbol{x}^{'} = g(\boldsymbol{h}) = g(f(\boldsymbol{x}))$. Typically, the dimension of $\boldsymbol{h}$ is much lower than the dimension of the input $\boldsymbol{x}$ to encourage the model to prioritize learning the most salient features of the input.

Since the encoder maps the high dimensional input to a low dimensional latent code, autoencoders can be used for **dimensionality reduction** as well as **feature learning**.

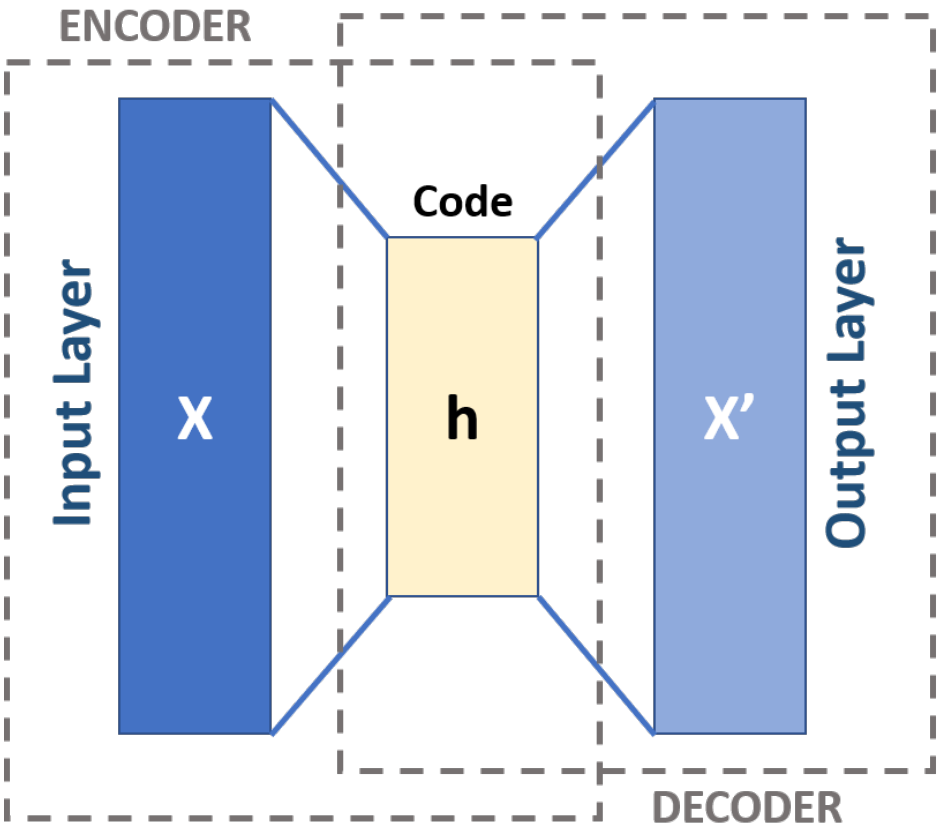


Figure 2.5: Basic Autoencoder Architecture [Source]

As the name suggests, **dimensionality reduction** refers to techniques which can be used to obtain a meaningful low dimensional representation of a high dimensional data element. **Feature learning** is learning to extract specific features from input data which can be used to perform downstream tasks such as classification more effectively.

Variational autoencoders (VAE) (Kingma & Welling, 2013) and **Wasserstein autoencoders (WAE)** (Tolstikhin et al., 2017) are extensions to the basic autoencoders where deterministic encoder and decoder functions are replaced by stochastic mappings $p_{\text{encoder}}(\boldsymbol{h}|\boldsymbol{x})$ and $p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{h})$.

2.6.1 Variational Autoencoders

Variational autoencoder (VAE) (Kingma & Welling, 2013) is a generalization of basic autoencoder where the deterministic encoder-decoder functions are replaced by stochastic mappings. Figure 2.6 shows the block diagram of a variational autoencoder.

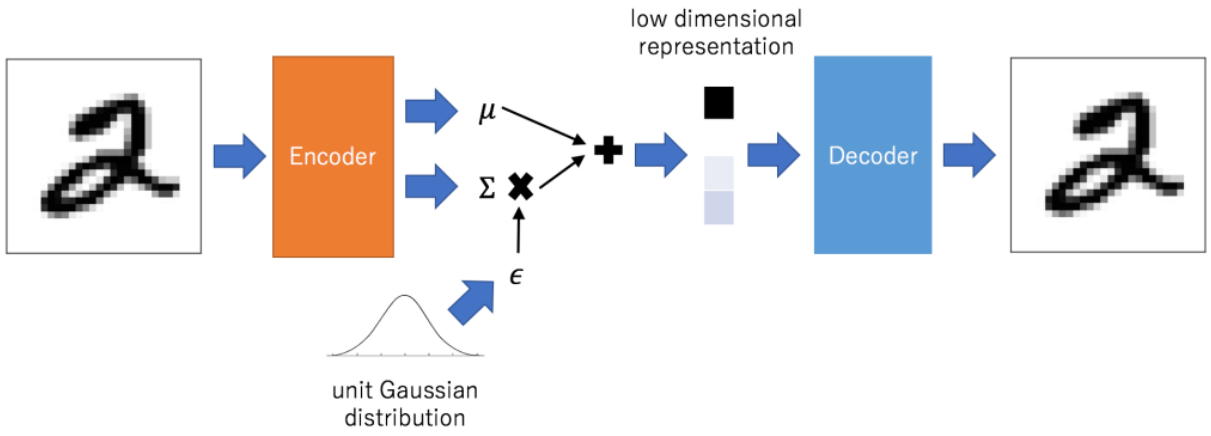


Figure 2.6: Variational Autoencoder Block Diagram [Source]

Similar to an autoencoder, a variational autoencoder also has two components: a **recognition model** (probabilistic encoder) denoted by $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and a **generative model** (probabilistic decoder) denoted by $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ where $\boldsymbol{x}$ is the input variable, $\boldsymbol{z}$ is the corresponding latent variable, and ϕ & θ are the parameters are of the encoder and decoder respectively. The recognition model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ serves as the approximation to the intractable true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$.

During training, the encoder is trained to map the input $\boldsymbol{x}$ to the parameters of approximate posterior distribution $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. To ease the computational requirements, $\boldsymbol{\Sigma}$ is constrained to be diagonal. Since sampling $\boldsymbol{z}$ directly from the distribution parameters obtained from the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ removes the option of using backpropagation for training, **reparameterization trick** is employed to sample $\boldsymbol{z}$:

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\Sigma} * \epsilon \tag{2.12}$$

where $\epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

The decoder is trained to reconstruct the original input $\boldsymbol{x}$ using the sampled latent code $\boldsymbol{z}$. In addition to the reconstruction loss, the variational autoencoder has an additional regularization term in its overall objective. The regularization term computes the KL divergence between the approximate posterior of the latent variable and its prior. Typically, an isotropic gaussian distribution with $\mathbf{0}$ mean and unit variance i.e. $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is used as the prior for the latent variable. Therefore, the overall loss function used for training a variational autoencoder is given by:

$$\mathcal{L}_{\text{VAE}} = \mathrm{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log(p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - \mathbb{KL}(q_\phi(\boldsymbol{z})||p(\boldsymbol{z})) \tag{2.13}$$

where $p(\boldsymbol{z})$ is the prior distribution of the latent variable.

2.6.2 Bag-of-Words Variational Autoencoders

Miao et al. proposed an unsupervised generative model of text based on the variational autoencoders called **Neural Variational Document Model** (NVDM) (Miao et al., 2015) (Figure 2.7).

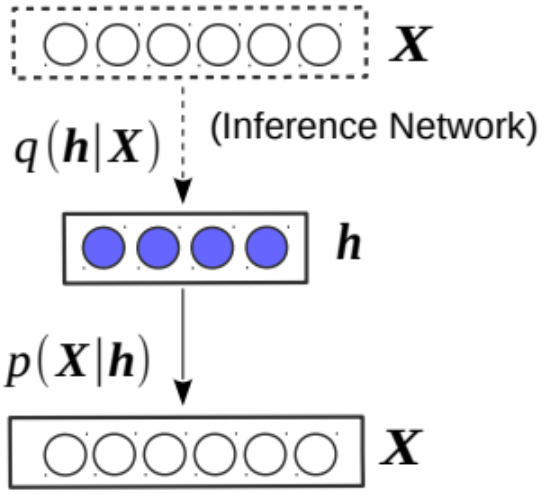


Figure 2.7: Neural Variational Document Model (Miao et al., 2015)

The main objective of the proposed NVDM is to obtain a continuous latent representation for each document's semantic content. The model consists of an encoder (inference

network) parameterized by a multilayer perceptron (MLP) which operates upon the bag-of-words document representation and returns its continuous latent representation, and a softmax decoder (generative model) which reconstructs the document by generating the words independently. The NVDM achieved state of the art perplexities when applied to document topic modeling tasks using multiple datasets such as *20NewsGroups* and *RCV1-v2*.

2.7 Sequence-to-Sequence Models

Sequence to sequence models were proposed by Sutskever et al. as way to model the transformation of variable length source sequences to variable length target sequences (Sutskever et al., 2014). The transformation learned by the model depends on the task that the model is trained to perform. For example, Sutskever et al. applied the proposed sequence to sequence model to perform the WMT'14 English to French translation task (Sutskever et al., 2014). Figure 2.8 shows the components of a typical sequence to sequence model.

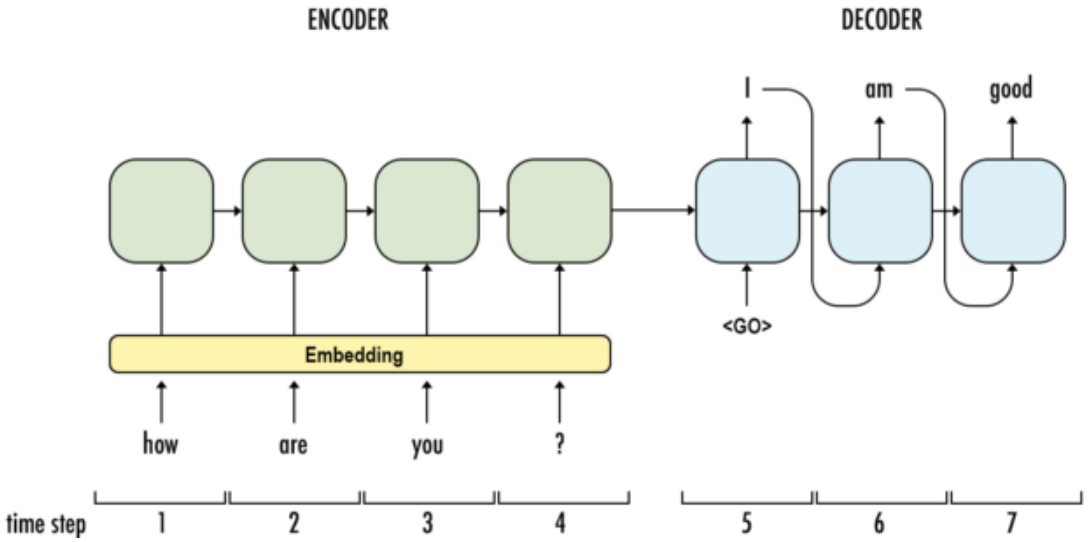


Figure 2.8: Sequence-to-sequence Model [Source]

Similar to an autoencoder, a sequence to sequence model is made up of an encoder and a decoder. The encoder takes the source sequence as input and produces a fixed size vector representation of the whole source sequence. The decoder uses the vector representation of

the source sequence to produce the target sequence in a sequential manner. Typically, both the encoder and decoder are implemented using recurrent neural networks such as Long Short Term Memory or Gated Recurrent Unit. The encoder RNN's hidden state corresponding to the last time step of the source sequence is usually used as the fixed size vector representation of the source sequence. This vector representation of the source sequence is used to initialize the hidden state of the decoder when generating the corresponding target sequence.

Since their introduction, sequence to sequence models have been successfully applied to a variety of natural language generation tasks including, but not limited to, dialog generation (Shang et al., 2015; Vinyals & Le, 2015) , question answering (Yin et al., 2015), text summarization (See et al., 2017; Zhou et al., 2018), etc.

2.8 Generative Adversarial Networks

Generative adversarial network (GAN) (I. J. Goodfellow et al., 2014) is a framework for learning generative models through the use of an adversarial process. The GAN typically contains two neural networks called the **generator** denoted by G and the **discriminator** denoted by D. The adversarial process contains training both the generator G and the discriminator D simultaneously. The generator G is to trained to learn the true data distribution. The discriminator D, on the other hand, acts as the adversary of the generator and is trained to distinguish between the real data samples and the samples generated by the generator G. Figure 2.9 provides a high level overview the GAN architecture.

Let p_g denote the true distribution over the data $\boldsymbol{x} \in \mathcal{X}$ and $p_{\boldsymbol{z}}(\boldsymbol{z})$ denote the input noise's prior. Then, the generator $G(\boldsymbol{z}; \theta_g)$ represents a mapping from the noise space to the data space and is paramterized by a multilayer perceptron with parameters θ_g. The discriminator $D(\boldsymbol{x}; \theta_d)$ is also parameterized by a separate multilayer perceptron with parameters θ_d and produces a singular probability value. The higher the output probability the more likely it is that the input to the discriminator was sampled from the true data distribution. The generator is trained to output samples which increasingly mimic the true data distribution and fool the discriminator. The discriminator is simultaneously trained to output probabilities which can be used to correctly identify whether input sample was sampled from the true data distribution or generated by generator. More formally, the discriminator D and the generator G play the following minimax game with the value function $V(G, D)$:

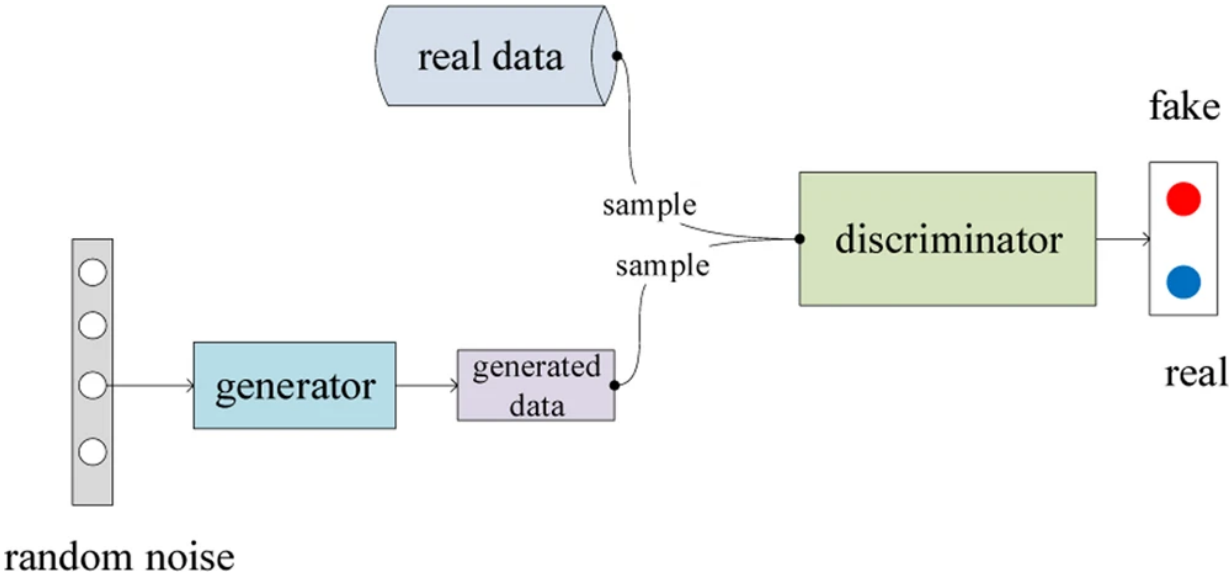


Figure 2.9: GAN architecture [Source]

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}))] \tag{2.14}$$

In practice, during initial phase of the GAN training the generator distribution differs from true data distribution significantly which makes it very easy for the discriminator correctly tell apart the real samples from the samples produced by the generator. This, in turn, makes the signals generated by the discriminator gradients weak and not very useful for training the generator. Therefore, it is suggested that instead of training G to minimize $\log(1 - D(G(\boldsymbol{z})))$, it is more useful to train G to maximize $\log(D(G(\boldsymbol{z})))$ (I. J. Goodfellow et al., 2014).

Since their introduction GANs have become widely used for generative modeling tasks in both vision (Bao et al., 2017; Brock et al., 2018; Dai et al., 2017) and NLP (Xu et al., 2018; Zhang et al., 2017) domains with great success.

2.8.1 Conditional GAN

One of the shortcoming of the GANs is that there is no way to control the data samples generated by the generator. Mirza et al. proposed **conditional generative adversarial**

nets (CGANs) to mitigate this problem (Mirza & Osindero, 2014). The proposed CGANs allowed the generation of data samples by the generator which were conditioned on some auxiliary information about the input data items. Let $\boldsymbol{y}$ denote some auxiliary information about the data denoted by $\boldsymbol{x}$. Then, the GAN can be extended to CGAN by simply feeding the $\boldsymbol{y}$ to both the generator and the discriminator (Mirza & Osindero, 2014) using additional input layers. The new objective function is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y}))] \tag{2.15}$$

2.8.2 Auxiliary Classifier GAN

Despite the high popularity and widespread usage, GANs have certain limitations including, but not limited to, the mode collapse problem and difficulty of training (Arjovsky & Bottou, 2017). The mode collapse problem results in a lack of diversity in the samples generated by the generator. Various techniques have been proposed to mitigate these problems without fully resolving these issues. A variant of GAN called an **auxiliary classifier GAN (ACGAN)** was proposed by Odena et al. and was shown to generate high quality and diverse images while exhibiting higher training stability (Odena et al., 2016). The structure of an AC-GAN is similar to a CGAN where the generated samples are conditioned on some class label. The main difference being that the discriminator is trained to produce a probability distribution over the class labels in addition to the probability distribution over the sources (Odena et al., 2016). Figure 2.10 shows the architectures of GAN, CGAN, and ACGAN.

2.9 Transformers

Recurrent neural networks such as long short term memory and gated recurrent unit have achieved tremendous amount of success on various NLP tasks but due to their recurrent nature do not easily lend themselves to parallelization. This puts an upper bound on the size of the training set which can be used train recurrent neural network based models. To alleviate this problem Vaswani et al. proposed a novel encoder-decoder model architecture based entirely on attention mechanism and without any use of recurrent neural networks called the **Transformer** (Vaswani et al., 2017). The transformer model uses the attention mechanism extensively to model global dependencies between the input and output.

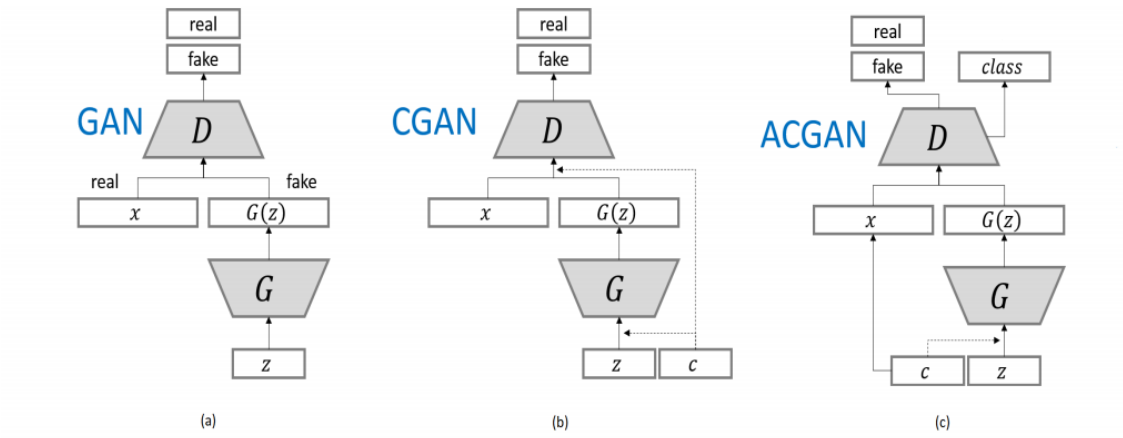


Figure 2.10: GAN, CGAN, and ACGAN architectures (Mino & Spanakis, 2018)

The transformer model, due to its architectural choices, also allows for significantly more parallelization which enables training on significantly larges training sets using distributed computational resources (Vaswani et al., 2017). Figure 2.11 shows the architecture of the transformer model.

As mentioned earlier, the transformer model uses the encoder-decoder structure. The left half of the figure 2.11 shows the encoder architecture and the right half shows the decoder architecture. The encoder is made up of $N = 6$ identical layers. Each layer itself is made up of two sub-layers. The first sub-layer is a multi-head attention layer and the second sub-layer is a position-wise, fully connected feedforward network. Residual connections are used around each of the sub-layer. The output of the sub-layer and the residual connection value are added and layer normalization is performed. The decoder is also made up of $N = 6$ identical layers. Each layer contains three sub-layers. The first sub-layer is masked multi-head attention which uses masking to prevent positions from attending to subsequent positions. The second sub-layer is another multi-head attention layer which performs attention over the output of the encoder stack. The third sub-layer is a position-wise, fully connected feedforward network. The decoder also uses residual connections around each sub-layer in the same way as the encoder. The outputs fed to the decoder are shifted to the right by one position. This along with the fact that position can not attend to the subsequent position due to masking ensures that the predictions for the position i can only make use of the known outputs at positions less than i (Vaswani et al., 2017).

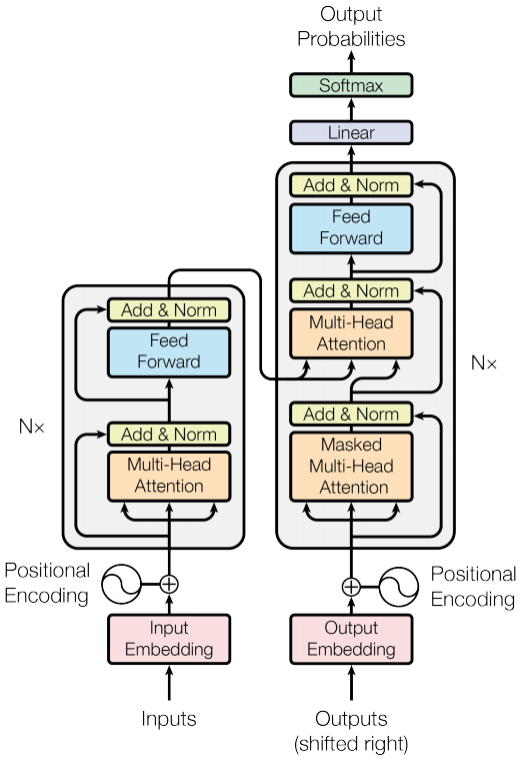


Figure 2.11: Transformer Architecture (Vaswani et al., 2017)

Transformer relies entirely on **self-attention** mechanism to compute the representations of its input and outputs without making use of any recurrent neural networks or convolutional neural networks. **Self-attention**, sometimes also called **intra-attention**, refers to applying attention mechanism to capture dependencies between different positions within a single sequence. Vaswani et al. refer to the attention used in the proposed transformer model as **scaled dot-product attention**. The input to the attention mechanism is made up of queries, keys, and values. The queries and keys are of the same dimension d_k and values are of the dimension d_v. For each query, dot products are computed with all the keys and the results are divided by $\sqrt{d_k}$. Finally, a softmax function is applied to obtain the weights which are used to scale the values. Practically, this is all computed using matrix operations. Let Q, K, V be the matrices denoting queries, keys and values respectively. Then, the attention function is computed in the following way:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.16}$$

The transformer performs this attention function $h = 8$ times, calling it a **multi-head attention**, by linearly projecting the queries, keys, and values h times with different, learned linear projections (Vaswani et al., 2017). The output values of each of the attention function is concatenated and once again projected to obtain the final output values. Formally:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.17}$$

Where the $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ are the projection parameter matrices.

The transformer model was evaluated on the language translation task and achieved state of the art results (Vaswani et al., 2017).

2.9.1 T5

As mentioned earlier, the transformer model lends itself to effective parallelization of the model training process. This opened up the possibility of training very large transformers based models on massive NLP corpus using distributed computational resources. Researchers discovered that training very large transformer based models on massive NLP corpus using various unsupervised objectives such as language modeling, next sentence prediction, etc. can be used to train the model to learn general purpose abilities and knowledge which significantly improve the model performance on various downstream tasks. Multiple such models have been proposed including BERT (Devlin et al., 2018), GPT2 (Radford et al., 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), etc.

Text-to-Text Transfer Transformer (T5) (Raffel et al., 2019) is a transformer based model which uses the encoder-decoder structure. It uses the same basic architecture as proposed in the original transformer paper (Vaswani et al., 2017) with some minor variations. It is based on the basic idea that the most problems in NLP can be formulated as text to text transformation. In other words, given a sequence of words as input the model produces another sequences of words as output. Figure 2.12 shows how the input and output are formulated for performing a variety of NLP tasks using the T5 model.

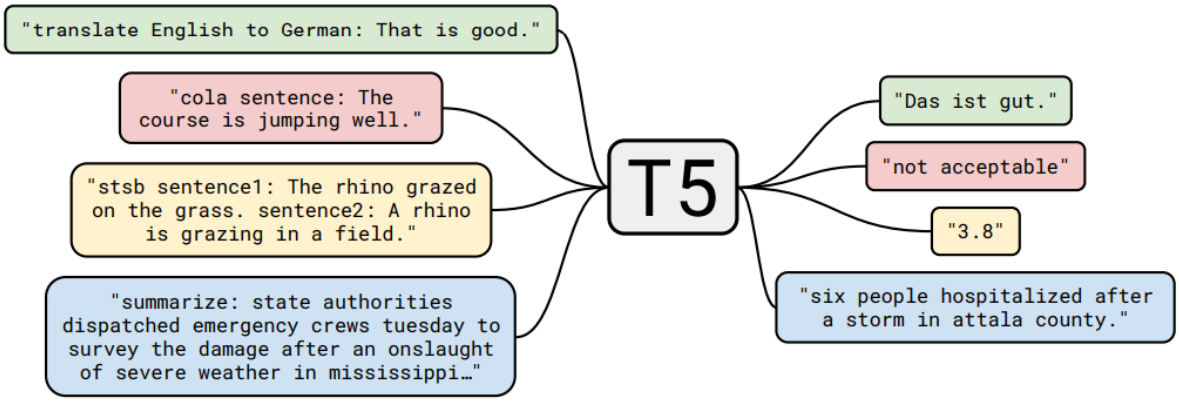


Figure 2.12: Text-to-Text Transfer Transformer (T5) Framework (Raffel et al., 2019)

Raffel et al. generate a new very large unlabeled text dataset called the **colossal clean crawled corpus (C4)** using the common crawl dataset. The C4 dataset is generated by applying some filtering rules to the common crawl dataset to only retain clean English language documents which contain natural language text rather than computer generated boilerplate text or computer code (Raffel et al., 2019). The newly generated C4 dataset dataset, which is about 750 GB in size and orders of magnitude larger than other datasets used for pretraining, is used for pretraining the T5 model using the encoder-decoder setup and a basic denoising objective (figure 2.13). The pretraining uses teacher forcing and cross entropy loss. After pretraining, the T5 model is fine-tuned on individual supervised tasks by adjusting the input and output sequences. T5 achieved state of the art results on multiple NLP tasks including reading comprehension, machine translation, entailment, etc (Raffel et al., 2019).

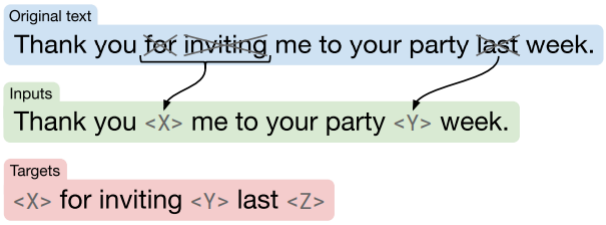


Figure 2.13: Inputs and outputs for the denoising objective (Raffel et al., 2019)

Chapter 3

Related Work and VAE-AM

3.1 Dialog Response Generation

The main goal of a dialog response generation model is to generate response utterance based on the query utterance while taking previous utterances in the same dialog into account. The generated response utterance should be coherent and relevant to the query utterance. Furthermore, an effective dialog response generation model should generate diverse and informative response utterances rather than generic response utterances.

Dialog response generation is an active area of research and rapid advances in deep learning methods have motivated researchers to explore generative deep learning models for dialog generation task. Various approaches have been explored to develop an effective dialog generation model. After Sutskever et al., 2014 introduced sequence-to-sequence (Seq2seq) models for machine translation task, they were successfully applied to the dialog generation tasks by multiple researchers (Shang et al., 2015; Sordoni et al., 2015). Soon after, it became evident that the Seq2seq based models tend to generate trivial and safe response responses such as "I don't know" or "I'm ok" (Vinyals & Le, 2015). To improve the diversity of the responses generated by Seq2seq model, various variants have been proposed which include using objective which promotes diversity (J. Li et al., 2016), incorporating situational information for response selection (Sato et al., 2017), etc. Various dialog generation models which are based on the variational framework (Kingma & Welling, 2013) have also been proposed including, but not limited to, variable hierarchical recurrent encoder decoder (VHRED) (Serban, Sordoni, Lowe, et al., 2016) which models the encoded utterances in a dialog using latent variables at various levels, Collaborative VAE (Shen et al., 2018) which trains a pair of collaborating autoencoder and conditional

VAE (Sohn et al., 2015) to improve the overall performance, etc. Generative adversarial networks (GANs) (I. J. Goodfellow et al., 2014) have shown impressive generational capabilities when applied to image generation tasks. Application of GANs to text generation is difficult due to its inherent discrete nature but various workarounds have been proposed. J. Li et al., 2017 combine adversarial learning with reinforcement learning for dialog generation. They propose adversarial REINFORCE algorithm where the generator is rewarded for generating utterances which resemble human generated utterance as determined by the discriminator. Another approach is to apply GANs to the latent space representations of the utterances. An example of this approach is DialogWAE (Gu et al., 2018) where an adversarial process is used to match the generator's output latent code with the autoencoding latent code. Our proposed approach is similar in the sense that we apply a GAN to the latent space representation of the utterances.

3.2 VAE-AM

In the next few sections we describe the joint work done previously on dialog response generation. The multi-stage approach proposed in this previous work called VAE-AM and described in the next two subsections provides the foundation and motivation for the approach proposed and explored in this thesis. We first provide a high level overview of the proposed approach in the next paragraph and then provide more details in the next subsections.

The proposed multi-stage approach uses adversarial learning on the latent space of the dialog utterances for dialog response generation. In the first stage, a variational autoencoder (VAE) (Kingma & Welling, 2013) is trained to reconstruct the utterances. This allows us to obtain a semantic continuous vector respresentation for all the query and response utterances in the dialog dataset. In the second and final stage, a generative aderversarial network (GAN) is trained on the latent space of the VAE. The second stage GAN is trained to transform the latent code of a query to the latent code of the corresponding response. At inference time, we use the encoder to obtain the latent code of the response and then use the generator to transform it into the corresponding response latent code. Finally, the decoder is used to generate the response utterance text using the previously obtained response uttarance latent code. Since the GAN is applied to the continuous latent space, no reinforcement learning is needed, and we can benefit from the GANs mode-capturing property. This simplifies the training procedure, and our model generates informative sentences. Further, a mean squared error (MSE) auxiliary loss is used on the generator during the adversarial learning process, which mitigates the mode-missing problem in GANs (Che

et al., 2017), resulting in more relevant and diverse responses.

3.2.1 Approach

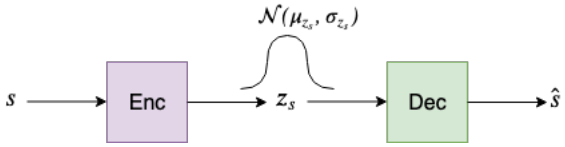


Figure 3.1: Step 1: Variational Autoencoder. During the training, the encoder takes an utterance s as input and produces its semantic continuous vector representation. This representation is then projected to obtain the parameters of the approximate posterior distribution. Using the reparameterization trick, a latent code z_s obtain. The decoder is trained to use this latent code to reconstruct the original utterance s.

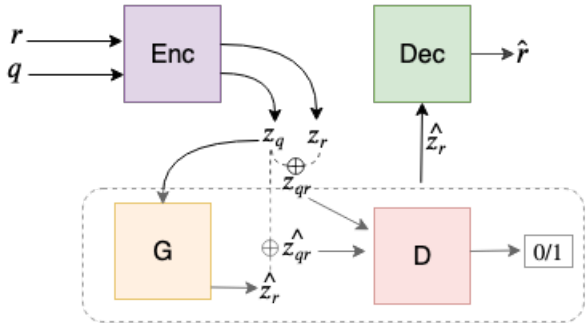


Figure 3.2: Step 2: Pretrained encoder from step 1 is utilized to get the latent codes z_q and z_r of the query (q) and response (r) utterances, respectively. After that, query latent variable (z_q) is fed to the generator (G) which maps it to the corresponding response latent variable $\hat{z}_r$. When training the generator, we aim to match z_r and $\hat{z}_r$ through the generator loss combined with a mean-squared error loss. When training the discriminator, we pass z_{qr} and $\hat{z}_{qr}$ (obtained by concatenating z_q with z_r and z_q with $\hat{z}_r$, respectively) through a classification layer that tries to guess its source of input. **Note:** $\bigoplus$ denotes concatenation..

Figure 3.1 and 3.2 provide a high level overview of the proposed approach.

We start by formulating our task formally. We use **Q** & **R** to denote the set of all the query utterances and response utterances respectively in the dialog dataset under

consideration. $\mathbf{q} \in \mathbf{Q}$ and $\mathbf{r} \in \mathbf{R}$ denote the individual query and the corresponding response utterance. Let $\mathbf{S}$ denote the set of all the utterances made up of query and response utterances i.e. $\mathbf{S} = \mathbf{Q} \cup \mathbf{R}$. An utterance $\mathbf{s} \in \mathbf{S}$ containing n words is denoted by $\mathbf{s} = (x_1, x_2, x_3, ..., x_n)$. We use $\mathbf{z_q}$ & $\mathbf{z_r}$ to denote the latent codes corresponding to a query and response utterance respectively. Finally, $p(\mathbf{z_q}) = p(\mathbf{z}|\mathbf{q})$ and $p(\mathbf{z_r}) = p(\mathbf{z}|\mathbf{r})$ refer to the approximate posteriors of the query and response utterance learned by the VAE.

The problem statement is: given a query utterance $\mathbf{q} \in \mathbf{Q}$, generate the corresponding response utterance $\mathbf{r} \in \mathbf{R}$.

It is easy to see that generating the response utterance for a given query utterance can be viewed as learning the conditional distribution $p(\mathbf{z_r}|\mathbf{z_q})$. We propose a two-step method to learn the conditional distribution.

Step 1. Our main objective in step 1 is to effectively learn the posterior distribution of the continuous latent variables of the utterances in the dialog corpus. We need to be able to obtain the posterior distribution of the continuous latent distribution of a given utterance and conversely, be able to reconstruct the utterance given a latent code sample from a posterior distribution.

Towards this end, we adopt a variational autoencoder (VAE) (Kingma & Welling, 2013) for our first step. A VAE encodes an input utterance $\mathbf{s}$ to a probabilistic, latent continuous representation $\mathbf{z}$, from which VAE further decodes the input utterance $\mathbf{s}$.

Given an utterance $\mathbf{s}$, the encoder produces the parameters of the posterior distribution $\text{Enc}(\mathbf{s}) = p(\mathbf{z}|\mathbf{s}) = \mathcal{N}(\boldsymbol{\mu}, \text{diag}\, \boldsymbol{\sigma}^2)$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and standard deviation of the distribution respectively. We use the standard training objective of the VAE which minimizes the expected reconstruction loss and a regularization term which penalizes the KL divergence between the posterior and the prior. We use the typical standard normal $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ as the prior distribution of $\mathbf{z}$. This is given by

$$J_{\text{AE}}(\theta_{\text{Enc}}, \theta_{\text{Dec}}) = -\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{s})}(\log\, p(\mathbf{s}|\mathbf{z})) + \lambda_{\text{KL}} \text{KL}(p(\mathbf{z}|\mathbf{s})||p(\mathbf{z})) \tag{3.1}$$

where θ_{Enc}, θ_{Dec} are encoder and decoder parameters respectively, λ_{KL} balances the two terms.

When compared to a deterministic autoencoder, VAE learns a smoother latent space due to the KL regularization. This is especially useful due to the fact that the second step might not learn the conditional distribution perfectly and during inference, the generator might predict response latent codes which are not perfectly aligned with the ground truth response latent codes. But due to the smooth latent space, the non-perfectly aligned response latent code still produce reasonable response reconstruction.

Step 2. The main goal of the second step is to effectively predict the latent code of the response utterance given the latent code of the corresponding query utterance. During inference time, the query utterance latent code can then be used to predict the corresponding response utterance latent code and the decoder trained in the first step can be used to reconstruct the response utterance text.

For this purpose, we use a conditional GAN (CGAN) (Mirza & Osindero, 2014) which is trained to learn the conditional distribution $p(\mathbf{z_r}|\mathbf{z_q})$. During training, we use the VAE encoder trained in step 1 to obtain the query utterance latent code $\mathbf{z_q}$ and the response utterance latent code $\mathbf{z_r}$. The generator G is trained to transform the query utterance latent code $\mathbf{z_q}$ to response utterance latent code $\hat{\mathbf{z}}_\mathbf{r}$ i.e. $G(\mathbf{z_q}) = \hat{\mathbf{z}}_\mathbf{r}$. The discriminator D is trained to distinguish between the samples: $\mathbf{z_q} \oplus \mathbf{z_r}$ and $\mathbf{z_q} \oplus \hat{\mathbf{z}}_\mathbf{r}$. We use $\mathbf{z_q} \oplus \mathbf{z_r}$ denotes the concatenation of the ground truth query utterance latent code and the corresponding response utterance latent code which is treated as the real sample. Similarly, $\mathbf{z_q} \oplus \hat{\mathbf{z}}_\mathbf{r}$ is used to denote the concatenation of the ground truth query utterance latent code and the corresponding response utterance latent code generated using the generator and treated as the fake sample. The adversarial loss for the CGAN is given by:

$$J_{\text{CGAN}} = \min_G \max_D V(D, G) \tag{3.2}$$

$$\text{where } V(D, G) = \mathbb{E}_{\mathbf{z_r} \sim p(\mathbf{z_r})}[\log D(\mathbf{z_r}|\mathbf{z_q})] + \mathbb{E}_{\mathbf{z_q} \sim p(\mathbf{z_q})}[\log(1 - D(G(\mathbf{z_q})))] \tag{3.3}$$

In addition, we use an auxiliary mean square error (MSE) loss on the generator G:

$$J_{\text{MSE}} = ||\mathbf{z_r} - G(\mathbf{z_q})||^2 = ||\mathbf{z_r} - \hat{\mathbf{z}}_\mathbf{r}||^2 \tag{3.4}$$

The MSE loss on the generator helps stabilize the GAN training and reduce the severity of the mode collapse problem associated with the GANs (Che et al., 2017). Thus, the overall training objective is given by

$$J = J_{\text{CGAN}} + \alpha J_{\text{MSE}} \tag{3.5}$$

where α is a hyperparameter that moderates the effect of the MSE loss.

The proposed approach is trained and evaluated in two different settings: single-turn and multi-turn. In the single-turn setting, every possible pair of consecutive utterances belonging to the same dialog is extracted to form the query-response instance. In the multi-turn setting, every response utterance is paired with every preceding utterances in the same dialog i.e. response latent code is predicted using the latent codes for all the preceding utterances using a recurrent neural network as the generator in the CGAN.

3.2.2 Experiments and Results

In this section we provide the details about the evaluation results of the approach described in the preceding section.

Datasets. We evaluate the approach using DailyDialog (Y. Li et al., 2017) and Switch-Board (Godfrey et al., 1992) dialog datasets. DailyDialog (Y. Li et al., 2017) is a manually labeled, human written, multi-turn dialog dataset. The dialogs included in the dataset are day-to-day conversations which occur in life on a variety of topics. We use the original splits provided by the dataset after deduplicating the validation and test sets (Bahuleyan et al., 2018). Switchboard (Godfrey et al., 1992) dataset contains transcriptions of telephone conversations on various topics.

Baselines. We compare the proposed approach's performance against the following baseline models. Seq2Seq: The standard sequence to sequence model based on LSTMs; WED-S: a stochastic wasserstein encoder decoder model (Bahuleyan et al., 2019); and DialogWAE: a recent model based on adversarial regularization of autoencoders (Gu et al., 2018) which reports state of the art results on the selected datasets.

Evaluation Metrics. We use the following evaluation metrics to compare the performance of the proposed approach against the baselines models quantitatively. For each of the following metric, we compute the metric for each individual query and then report the average over the entire test set.

- **BLEU** Bleu metric measures the overlap of n-grams between the reference response and the generated hypotheses responses (Papineni et al., 2002). Higher bleu score indicates better performance. For each query, 10 responses are generated. We compute smoothed 3-gram bleu scores for each of the responses using smoothing 7 (B. Chen & Cherry, 2014). Following Zhao et al., 2017, we compute recall (R) bleu and precision (P) bleu by taking the maximum and average of the bleu scores for the 10 responses generated for a query. We also report the harmonic mean (F) of the precision and recall values.

- **Diversity** We measure the diversity of the generated responses by computing the ratio of the unique unigrams and bigrams with the total number of unigrams & bigrams in the generated responses. Intra distinct 1 and intra distinct 2 denote the ratio of the unique unigram & bigrams to total number of unigrams & bigrams for each response. On the other hand, Inter distinct 1 and inter distinct 2 denote the ratio of the unique unigrams & bigrams to total number of unigrams & bigrams over the 10 responses generated for a query. Higher scores indicate higher diversity in the generated responses.

- **Token Type Ratio (TTR)** TTR measures the lexical diversity of the generated responses. We compute it by taking the dividing the number of unique words by the total number of words in the generated responses. We only use one response per query for this purpose. Higher TTR scores indicate better overall lexical diversity in the generated responses.

- **Perplexity (PPL)** The PPL metric measures the fluency of the generated responses and how similarly structured the generated responses are to utterances found in the dataset. For computing perplexity, we train a trigram Kneser-Ney trigram language model (Kneser & Ney, 1995) on the entire dataset. Lower PPL scores are better and indicate that model generates fluent responses which strucutred similar to the ground truth responses. The PPL is computed in the following way:

$$\text{PPL} = e^{-\frac{1}{N}\Sigma_{i=1}^{N}\log(p(w_i))} \tag{3.6}$$

  where N is the total number of words in the sequence, w_i is the i-th word in the sequence and $p(w_i)$ is the probability of the assigned to w_i by the language model.

Implementation Details. We implement the VAE encoder and decoder with single layer Bidirectional LSTM (Hochreiter & Schmidhuber, 1997) and unidirectional LSTM with a hidden size of 512 respectively. We use word2vec word embeddings of size 300 for the input and set the latent code dimension to 128. Furthermore, we adopt KL-annealing and word dropout from (Bowman et al., 2016) to stabilize VAE's training. For the GAN, both the generator and the discriminator are implemented using two layer feedforward network using LeakyReLU activations. In multiturn setting, we replace the feedforward network with a bidirectional LSTM to generate the response latent code using the ground truth latent codes of all the previous utterances in the same dialog.

We use the training set for training the model, bleu metric performance on the validation set is used for selecting the best performing model and the final evaluation is performed on the test set. We also compute the average length of the responses generated by the models and compare it with the average lengths of the responses in the test set. Ideally, a model should generate responses with lengths similar to the average length of responses in the test set.

Results and Analysis. We provide the results for the single-turn and multi-turn settings for the DailyDialog test set in table 3.1. Results for Switchboard test set for both the single-turn and multi-turn settings are provided in table 3.2. Since in single-turn settings, DialogWAE and our approach are the best performing models on most of

the metrics, we compare only these 2 models for the multi-turn setting. Apart from the baselines models and our full model, we also provide results for our model which have some its components removed. VAE-AM refers to our full model, VAE-A refers to the model where we only use the adversarial loss to train the CGAN and do not apply MSE loss to the generator, and VAE-M refers to the model where we train the generator only using the MSE loss. We also provide some select responses generated by our model and the DialogWAE model for comparison.

We also performed a small scale human evaluation on 50 randomly selected queries from the DailyDialog test set. We ask 4 human evaluators to assign a rating on a scale of 1-5 (1 being the worst and 5 being the best) to each response based on the fluency of the generated response and the relevance to the query. We report the result in table 3.3. We compare samples generated using our model with the samples generated using the DialogWAE model. Our model outperforms the DialogWAE model on both criteria.

For the DailyDialog dataset, we can see from the quantitative results that our model outperforms the baseline models on most of the metrics in both the single-turn and multi-turn settings. We see that our model generates diverse and fluent responses in addition to being more related to the queries. We get strong results on the switchboard dataset as well. In the reported results, we note that the seq2seq baseline models performs really well on diversity and PPL metric but this is mainly due to the fact seq2seq model tend to generate really short and generic responses for all the queries as evidenced by the much lower bleu scores and average sentence length.

We hypothesize that our model learns to generate diverse and fluent responses due to the fact that we operate on the smooth latent space of the variational autoencoder. We somewhat mitigate the mode collapse problem of the GAN by using MSE loss on the generator. This allows the GAN to capture multiple modes present in the training set while simultaneously learning to transform query latent code to response latent code more effectively.

Model	BLEU			Diversity						Fluency
	P	**R**	**F**	**Intra-1**	**Intra-2**	**Inter-1**	**Inter-2**	**ASL (14.43)**	**TTR**	**PPL**
Seq2Seq	0.143	0.217	0.172	**0.99**	**0.99**	0.46	0.49	4.63	0.019	18.45
WED-S	0.215	0.357	0.268	0.94	**0.99**	0.48	0.74	10.42	0.034	33.91
DialogWAE	0.296	0.356	0.323	0.85	0.97	0.42	0.74	19.34	0.005	20
VAE-M (ours)	0.191	0.293	0.231	0.98	**0.99**	**0.5**	0.79	9.36	0.029	19.7
VAE-A (ours)	0.295	0.359	0.323	0.93	**0.99**	0.46	0.76	13.64	0.035	21.38
VAE-AM (ours)	**0.306**	**0.367**	**0.334**	0.91	**0.99**	0.46	**0.82**	16.90	**0.034**	**17.01**
Multi-turn results										
DialogWAE	0.279	0.365	0.316	0.79	0.92	0.35	0.68	19.84	0.007	161.86
VAE-AM (ours)	**0.314**	**0.371**	**0.34**	**0.847**	**0.98**	**0.41**	**0.73**	15.3	**0.036**	**119.39**

Table 3.1: DailyDialog Dataset results, suffix A- adversarial loss, suffix M - MSE loss.

Model	BLEU			Diversity						Fluency
	P	**R**	**F**	**Intra-1**	**Intra-2**	**Inter-1**	**Inter-2**	**ASL (8.49)**	**TTR**	**PPL**
Seq2Seq	0.088	0.176	0.118	**0.989**	0.956	**0.816**	**0.927**	2.66	0.026	23.62
WED-S	0.193	**0.395**	0.259	0.941	0.989	0.404	0.525	10.41	0.032	35.63
DialogWAE	0.235	0.375	0.289	0.739	0.712	0.354	0.571	10.32	0.017	25.36
VAE-M (ours)	0.231	0.3	0.261	0.954	0.998	0.322	0.479	8.84	0.045	24.58
VAE-A (ours)	0.229	0.376	0.285	0.725	0.751	0.218	0.354	11.73	0.053	27.55
VAE-AM (ours)	**0.259**	0.364	**0.303**	**0.989**	**0.999**	0.436	0.569	7.08	**0.062**	**21.87**
Multi-turn results										
DialogWAE	0.267	**0.394**	**0.318**	0.779	0.844	0.325	0.513	14.7	0.019	183.82
VAE-AM (ours)	**0.271**	0.372	0.313	**0.954**	**0.966**	**0.412**	**0.559**	8.72	**0.065**	**136.81**

Table 3.2: Switchboard Dataset results, suffix A- adversarial loss, suffix M - MSE loss.

Model	Relevance	Fluency
VAE-AM	2.3625	3.2225
DialogWAE	2.3125	3.1775

Table 3.3: Human evaluation results on the subset of samples selected from DailyDialog test set

Generated Responses
Query: o . k . well , go to google.com , and type in battle of gettysburg .
VAE-AM: oh , there are a lot of things that i need to do .
DialogWAE: then ill have to check the details . what can i do for you today ?
Query: yes . i m going to the untied states for travel . what places do you recommend ?
VAE-AM: here are some leaflets detailing everything . i have nt seen you in ages .
DialogWAE: it s very nice to see you again , but i think it s a good idea to do .
Query: medium one will be perfect , thank you
VAE-AM: is that all ?
DialogWAE: in that case , i have to be able to get some of them .
Query: the annual charge rate is NUM .
VAE-AM: that s a lot of money . can you make it a little cheaper ?
DialogWAE: then what about the <UNK> ? i mean , i am not sure .
Query: but it is too loose.do you have the smaller size ?
VAE-AM: ok , we will . how about this one ?
DialogWAE: in that case , i have to be able to get some of them . i think i can get a new one .

Table 3.4: Generated responses in single-turn setting for DailyDialog dataset

Generated Responses
Query: allan, i am really disappointed by your lack of effort on this project .
VAE-AM: it seems you are not aware of the actual work split . i have done more than anyone else.
DialogWAE: ok , it's not that bad . but it's not that bad . i don't want to have any other questions about it .
Query: when is he coming ?
VAE-AM: next week . he is enjoying his month long vacation in miami .
DialogWAE: sorry , he's not very serious . he's got a lot of money to do .
Query: i ll try my best to make a good impression on the interviewer .
VAE-AM: do nt worry pal. only weak candidates should. just dress properly and try to make it alive there haha ...
DialogWAE: that is ok , we ' ll be able to get a raise in the next few days . we will have to pay you a lot of money .
Query: can you tell me where the pots and pans are ?
VAE-AM: absolutely not . i m not letting you cook today . you messed it real bad yesterday .
DialogWAE: yes , of course .
Query: what kind of personal information ?
VAE-AM: your name , height , sex , and contact . also tick this check box if you have faced this issue before .
DialogWAE: not at the moment . but there is a lot of money in the city , but we have to pay a lot of money for the money .

Table 3.5: Generated responses in multi-turn setting for DailyDialog dataset

Chapter 4

Approach

In this chapter, we briefly describe the motivation behind our proposed approach. We describe our proposed approach in a detailed manner. Furthermore, we discuss the experiments performed to evaluate the proposed approach and analyse the reported results.

4.1 Approach

We provide a high level overview of the proposed approach in Figure 4.1, 4.2, and 4.3. Our proposed approach is an extension of the approach described in the previous chapter. We start by training a bag-of-words variational autoencoder and a language model which is used to encode the dialog history. Next, we train an auxiliary classifier GAN which learns to transform the concatenation of the query latent code and the encoded context into the response latent code. Finally, we finetune a pretrained T5 model (Raffel et al., 2019) to generate dialog responses conditoned on the query and the predicted response bag-of-words.

We observe that the variational autoencoder used in the approach described in the previous chapter tries to encodes the whole utterance into the latent code. An utterance contains both the semantic and syntactic component. A reasonably successful reconstruction requires access to both kind of information. It is possible that the VAE assigns same significance to both kind of information while encoding. While this does not negatively affect the autoencoding performance, it will have a negative effect on the second step where we are trying to learn to transform the query latent code to response latent code. Intuitively, it is reasonable to assume that having more semantic information retained in the latent codes will make it easier to transform the query latent code to response latent code.

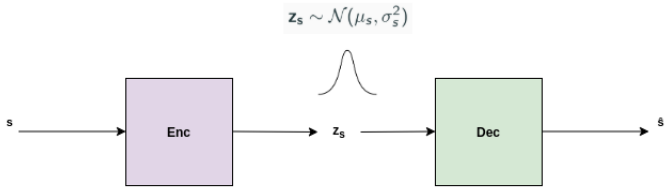


Figure 4.1: Step 1: Bag-of-words Variational Autoencoder (BOW-VAE). During the training, the encoder takes one hot encoded vector $\mathbf{s}$ containing upto n words from an utterance as input and produces its semantic continuous vector representation. This representation is then projected to obtain the parameters of the approximate posterior distribution. Using the reparameterization trick, a latent code $\mathbf{z_s}$ is obtained. The decoder is trained to use this latent code to independently predict the words contained in the input bag-of-words $\mathbf{s}$. We additionally, train a Bidirectional LSTM based language model LM to be used later for encoding the context.

We once again start by formulating our task formally. We use $\mathbf{U}$ & $\mathbf{V}$ to denote the set of all the query utterances and response utterances respectively in the dialog dataset under consideration. $\mathbf{m} \in \mathbf{M}$ and $\mathbf{v} \in \mathbf{V}$ denote the individual query and the corresponding response utterance. Similarly, we use $\mathbf{Q}$ & $\mathbf{R}$ to denote the set of all the query and response utterance bag-of-words (BOW) one hot encoded vectors respectively in the dialog dataset under consideration. Each of the one hot encoded vectors encode upto n words extracted from the corresponding utterance. $\mathbf{q} \in \mathbf{Q}$ and $\mathbf{r} \in \mathbf{R}$ denote the individual query and the corresponding response utterance bag-of-words vectors. Let $\mathbf{S}$ denote the set of all the utterance bag-of-words vectors i.e. $\mathbf{S} = \mathbf{Q} \cup \mathbf{R}$ and $\mathbf{s} \in \mathbf{S}$ denote a single utterance bag-of-words vector. We use $\mathbf{z_q}$ & $\mathbf{z_r}$ to denote the latent codes of a query $\mathbf{q}$ and the corresponding response $\mathbf{r}$ utterance BOW respectively. Similarly, $\mathbf{z_s}$ is used denote the latent code for a given utterance BOW $\mathbf{s}$. Finally, $p(\mathbf{z_q}) = p(\mathbf{z}|\mathbf{q})$, $p(\mathbf{z_r}) = p(\mathbf{z}|\mathbf{r})$ and $p(\mathbf{z_s}) = p(\mathbf{z}|\mathbf{s})$ refer to the approximate posteriors of the query $\mathbf{q}$, response $\mathbf{r}$, and utterance $\mathbf{s}$ learned by the VAE.

The problem statement is: given a query utterance $\mathbf{u} \in \mathbf{U}$, generate the corresponding response utterance $\mathbf{v} \in \mathbf{V}$.

We again formulate it as a distribution estimation problem. We learn the conditional distribution $p(\mathbf{z_r}|\mathbf{z_q})$ and use it to generate the response BOW. We use the predicted response words along with the query utterance to generate the full response utterance.

4.1.1 First Stage - Bag-of-Words Variational Autoencoder

The main objective of the first step is to learn the latent variable posterior of the query and response utterance bag-of-words. The model chosen for this purpose should be able to effectively encode the utterance bag-of-words one hot encoded vectors into meaningful latent codes and conversely, be able to reconstruct the original bag-of-words given a latent code corresponding to an utterance.

Most of the details in step 1 remain the same as in the step 1 of the approach described in the previous chapter, please refer to section 3.2.1 for details. We highlight the differences below and give the objective function.

One major difference is that we train a bag-of-words (BOW) VAE instead of a sentence VAE. We adopt the neural variation document model (NVDM) (Miao et al., 2015) based on the variational autoencoder (VAE) (Kingma & Welling, 2013) for our first step. We use one hot encoded vector representations for each utterance. We select upto n words using tfidf weighting statistics for encoder input vectors and we select words corresponding to top n probabilities while decoding.

The objective function for the BOW VAE is given by:

$$J_{\mathrm{AE}}(\theta_{\mathrm{Enc}}, \theta_{\mathrm{Dec}}) = -\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{s})}(\log\, p(\mathbf{s}|\mathbf{z})) + \lambda_{\mathrm{KL}}\mathrm{KL}(p(\mathbf{z}|\mathbf{s})||p(\mathbf{z})) \tag{4.1}$$

where θ_{Enc}, θ_{Dec} are encoder and decoder parameters respectively, λ_{KL} balances the two terms, $\mathbf{s}$ is the one hot encoded vector of an utterance, and $\mathbf{z}$ is the latent code.

Additionally, we train a language model using the next word prediction objective and cross entropy loss. We use this language model to encode the context of query-response pair. This encoded context is used in the next step to train the auxiliary discriminator.

4.1.2 Second Stage - Adversarially Learning the Latent Bag-of-Words

The main goal of the second step is to effectively predict the latent code of the response utterance $\mathbf{z_r}$ given the latent code of the corresponding query utterance $\mathbf{z_q}$. During inference time, the query utterance latent code can then be used to predict the corresponding response utterance latent code and the decoder trained in the first step can be used to obtain the subset of words in the response utterance.

To fulfil the goal of learning the transformation of the query latent code $\mathbf{z_q}$ to $\mathbf{z_r}$, we adopt an auxiliary classifier GAN (AC-GAN) (Odena et al., 2016). AC-GAN is a simple extension of the conditional GAN which yields multiple benefits in the form of improved generation and stabilized training (Odena et al., 2016). AC-GAN extends the CGAN by not only conditioning the generator on the additional class labels but also adding additional loss function to the discriminator to produce probability distribution over the class labels. Hence, the name auxiliary classifier GAN.

In addition to using the query latent code $\mathbf{z_q}$, we also use the dialog history in the form of a fixed size continuous vector representation, denoted as $\mathbf{c}$, obtained using the language model trained in first step. We use $\mathbf{c}$ as auxiliary class labels to setup the auxiliary classifier loss. The main intuition underlying the usage of the language model encoded context $\mathbf{c}$ is the fact that due to semantic relationship between the context $\mathbf{c}$ and the response latent code $\mathbf{z_r}$, it should be possible to distinguish between correctly and incorrectly paired $\mathbf{c}$ and $\mathbf{z_r}$. This allows us to use the source of this pair as the class label for the auxiliary classifier. Furthermore, instead of using the same discriminator D to generate the probability distributions over the sources and class labels we use an auxiliary discriminator D_{aux} which is trained to generate the probability distribution over the class labels.

We train the generator G to transform the concatenation of the query utterance latent code $\mathbf{z_q}$ and the encoded dialog history $\mathbf{c}$ to the response utterance latent code $\hat{\mathbf{z}}_\mathbf{r}$ i.e. $G(\mathbf{z_q}, \mathbf{c}) = \hat{\mathbf{z}}_\mathbf{r}$. The discriminator D is trained to distinguish between the samples: $\mathbf{z_q} \oplus \mathbf{z_r}$ and $\mathbf{z_q} \oplus \hat{\mathbf{z}}_\mathbf{r}$. We use $\oplus$ to denote the concatenation. Similarly, the auxiliary discriminator D_{aux} is trained to distinguished between the following pairs: $\mathbf{c} \oplus \mathbf{z_r}$ and $\mathbf{c} \oplus \hat{\mathbf{z}}_\mathbf{r}$.

The adversarial loss for our adaption of the AC-GAN is given by:

$$J_{\text{AC-GAN}} = \min_G \max_{D, D_{\text{aux}}} V(G, D, D_{\text{aux}}) \tag{4.2}$$

$$\begin{aligned} \text{where } V(G, D, D_{\text{aux}}) = {} & \mathbb{E}_{\mathbf{z_r} \sim p(\mathbf{z_r})}[\log D(\mathbf{z_r}|\mathbf{z_q})] + \mathbb{E}_{\mathbf{z_r} \sim p(\mathbf{z_r})}[\log D_{\text{aux}}(\mathbf{z_r}|\mathbf{c})] + \\ & \mathbb{E}_{\mathbf{z_q} \sim p(\mathbf{z_q})}[\log(1 - D(G(\mathbf{z_q})))] + \\ & \mathbb{E}_{(\mathbf{z_q}, \mathbf{c}) \sim p(\mathbf{z_q}, \mathbf{c})}[\log(1 - D_{\text{aux}}(G(\mathbf{z_q}, \mathbf{c})))] \end{aligned} \tag{4.3}$$

We use an auxiliary mean square error (MSE) loss on the generator G:

$$J_{\text{MSE}} = ||\mathbf{z_r} - G(\mathbf{z_q})||^2 = ||\mathbf{z_r} - \hat{\mathbf{z}}_\mathbf{r}||^2 \tag{4.4}$$

To reiterate, we use MSE loss on the generator to stabilize the GAN training and reduce the severity of the mode collapse problem associated with the GANs (Che et al., 2017).

Furthermore, we use the pretrained encoder and decoder from step 1 to help train the generator using a reconstruction objective. This also results in further finetuning of the encoder and decoder. We use the same reconstruction loss as used in the VAE objective:

$$J_{\text{REC}} = -\mathbb{E}_{\hat{\mathbf{z}}_\mathbf{r} \sim G(\mathbf{z_q})}(\log\, p(\mathbf{r}|\hat{\mathbf{z}}_\mathbf{r})) \tag{4.5}$$

Thus the overall objective is given by:

$$J = J_{\text{AC-GAN}} + \alpha J_{\text{MSE}} + \gamma J_{\text{REC}} \tag{4.6}$$

where α and γ are hyperparameters that moderate the effect of the MSE and reconstruction loss.

4.1.3 Third Stage - Text Generation Using the Learned Bag-of-Words

In the third step, we aim to train a model which can be used to reconstruct the response utterance by conditioning on the predicted response words and the query utterance. Towards this end, we finetune a pretrained text-to-text transfer (T5) model (Raffel et al., 2019) to perform this task.

Text-to-text transfer (T5) model is a transformer based model which uses encoder-decoder structure (Raffel et al., 2019). T5 is trained on a large text corpus called C4 corpus in an unsupervised manner using a denoising objective. After training on the C4 dataset, the T5 model was fine-tuned for several NLP tasks such as translation, entailment, summarization by casting them as text to text transformation. For each NLP task, the T5 model was fine-tuned using the task specific dataset in a supervised manner. For each specific task, the input sequences are prefixed with a task specific phrase enabling T5 to recognize the transformation to be performed during the inference. For example, while fine-tuning for English-to-German translation tasks, the prefix "translate English to German" was used. T5 was shown to achieve state-of-arts performances on various NLP tasks using this fine-tuning mechanism.

We fine-tune T5 using a custom prefix which enables it generate query responses based on both the query utterance and predicted response keywords. Please refer to figure 4.3 to

see how the input sequences were formed during the training and inferences stages. When fine-tuning T5, we use the teacher forcing and a standard cross-entropy loss.

4.2 Experiments

We evaluate our proposed approach on two dialog datasets. We provide details about the experiments and report results in the following sections.

4.2.1 Datasets

DailyDialog (Y. Li et al., 2017) dataset is popular multi-turn English language dialog dataset. The dataset is human written and contains dialogs from everyday human life covering a variety of topics. We use the original train, valid, and test split provided by the dataset after deduplicating the valid and test set following (Bahuleyan et al., 2018).

Switchboard (Godfrey et al., 1992) dataset contains transcriptions of telephone conversations. We use the train, valid, and test split provided by (Gu et al., 2018).

4.2.2 Baseline Models

We saw from the results in the previous chapter that on both the DailyDialog and the Switchboard dataset, the best performing models on various quantitative evaluation metrics were our proposed approach (VAE-AM) and DialogWAE (Gu et al., 2018) model. Therefore, we use those two models as our baseline model. Since our approach uses the whole dialog history to predict the keywords, we compare our results against the multi-turn results for these two models. Additionally, we use the pretrained T5 model fine-tuned on dialog generation tasks on each of the datasets as an additional baseline model.

4.2.3 Evaluation Metrics

We use the same evaluation metrics as were used in the experiments detailed in the previous chapter. We cover them here briefly, please refer to section 3.2.2 for details.

We use n-gram smoothed bleu score ($n < 4$) (Papineni et al., 2002) to measure the n-gram overlap between the generated responses and the ground truth response. We use

smoothing 7 proposed in B. Chen and Cherry, 2014. We generate 10 responses for each query and compute the recall and precision bleu (Zhao et al., 2017). Higher bleu score indicates more overlap. Harmonic mean (F) of the recall and precision bleu are also reported. We measure diversity of the generated response by computing the ratio of the unique unigrams and bigrams with total number of unigrams and bigrams. We compute this for each of the generated responses (Intra Distinct 1 & Intra Distinct 2) and across all the 10 generated responses for a given query (Inter Distinct 1 & Inter Distinct 2). We also measure the lexical diversity of all the generated responses by computing ratio of unique tokens generated for one response per query over the total tokens generated. Higher diversity metrics indicate higher diversity in the generated responses. We use the perplexity metric to measure the fluency of the generated responses. For the purpose of measuring the perplexity, we train a trigram Kneser-Ney trigram language model (Kneser & Ney, 1995) on the respective dataset. We also report the average sentence length (ASL) of all the generated responses.

We note that even though higher bleu scores of the generated responses is an indication that the generated responses are more related to the query, the converse is not always true. This is due to the fact that for each query, there are usually multiple valid responses which do not overlap at all with the ground truth response used to measure the blue score. Therefore, it is imperative to evaluate the performance of a dialog generation model using a combination of metrics.

Apart from 1-gram bleu, we also report the precision and recall metric for the first and second step models:

$$\text{Precision} = \frac{\text{tp}}{\text{tp} + \text{fp}} \tag{4.7}$$

$$\text{Recall} = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{4.8}$$

where tp, fp, and fn refer to true positive, false positive, and false negative respectively. In our second stage, we predict 10 response latent codes for a given query latent. These 10 predicted response latent codes are used to generate 10 corresponding bag-of-words using the decoder. Finally, we compute recall and precision 1-gram bleu for the generated bag-of-words following (Zhao et al., 2017).

4.2.4 Implementation and Training Details

We implement the language model trained in step 1 which is used to encode the context using three layer bidirectional LSTM (Hochreiter & Schmidhuber, 1997) with 64 hidden units. We use word2vec (Mikolov et al., 2013) word embeddings obtained using a word2vec model trained on the respective datasets.

The encoder and decoder in the bag-of-words variational autoencoder are implemented using three layers of fully connected feedforward network. We utilize batch normalization (Ioffe & Szegedy, 2015) between layers to prevent overfitting to the training set. We use LeakyReLU activation function. Input to the encoder is in the form of one-hot encoded vector with upto $n = 6$ words selected from each utterance. We use tfidf-weighting statistics to select the words from each utterance. Higher tfidf-weighting of a word indicates that the word is more informative as opposed to less informative words such as stopwords. While decoding, we select $n = 6$ words corresponding to the highest probabilities. We utilize KL-annealing to prevent posterior collapse using a tanh function based scheduling. We evaluate the performance of the BOW VAE model using the 1-gram, precision, and recall metric. We provide the results of the selected BOW VAE model on DailyDialog and Switchboard datset in table 4.1. We also provide an example of the utterance reconstructions for both DailyDialog and Switchboard dataset in tables 4.2 and 4.3 repectively.

Dataset	Bleu	Precision	Recall
DailyDialog	0.73	0.73	0.78
Switchboard	0.71	0.71	0.76

Table 4.1: BOW VAE evaluation results on the DailyDialog and Switchboard test set

Utterance	Input BOW	Reconstruction BOW
I just don't know how to bring it up. Well, all right. She has the right to know anyways.	anyways, bring, has, know, right, she	has, know, mean, now, right, she

Table 4.2: BOW VAE reconstruction example from DailyDialog test set

For our AC-GAN implementation, we use similar architectures for the generator, discriminator, and the auxiliary discriminator. We again use three layers of fully connected feedforward network with ReLU activation functions. We also use dropout (Srivastava et al., 2014) with $p = 0.33$ after each layer in the generator to prevent overfitting to the

Utterance	Input BOW	Reconstruction BOW
aspect to the human rights issues and the sort of things that democracy in this country has really stood for and you know and brought it back to the limelight compared to the sixties and seventies	aspect, democracy, human, limelight, sixties, stood	capital, certainly, democracy, hopefully, human, people

Table 4.3: BOW VAE reconstruction example from Switchboard test set

training set. During inference on the validation and test set, we sample 10 latent codes from the query utterance BOW latent posterior which are used to generate 10 corresponding response uterrance BOW latent codes. We report the results for the selected AC-GAN model on the test set of the DailyDialog dataset and Switchboard dataset in table 4.4. We compare the performance of the CGAN with AC-GAN on the DailyDialog test set in table 4.5. In CGAN model, we condition on just the query BOW latent whereas in AC-GAN we condition on the encoded dialog history in addition to the query BOW latent code. Finally, we present examples of the predicted response words for both the DailyDialog and Switchboard datasets in tables 4.6 and 4.7.

Dataset	Precision Bleu	Recall Bleu	Precision	Recall
DailyDialog	0.08	0.19	0.06	0.07
Switchboard	0.05	0.17	0.05	0.05

Table 4.4: AC-GAN evaluation results on the DailyDialog and Switchboard test set

Model	Precision Bleu	Recall Bleu	Precision	Recall
GAN	0.05	0.14	0.03	0.03
AC-GAN	0.08	0.19	0.06	0.07

Table 4.5: CGAN vs AC-GAN on the DailyDialog test set

For our final stage, we use the pretrained t5-base model provided by the huggingface's pytorch transformer module (Wolf et al., 2019). We finetune the pretrained T5 model on each of the dialog dataset's train set for generating the response utterance by conditioning on the query utterance and ground truth response utterance BOW. Refer to figure 4.3 to see how we format the input and target sequences. For inference on the test set, we use the response keywords predicted by our second stage AC-GAN generator. We verify that the

Query Utterance	sorry, i am not sure. can i take a message?
Query BOW	am, message, not, sorry, sure, take
Ground Truth Response Utterance	No, thanks. I will call back later.
Ground Truth Response BOW	back, call, later, no, thanks, will
Predicted Response BOW #1	call, help, ll, no, ok, thanks
Predicted Response BOW #2	can, he, him, know, may, take
Predicted Response BOW #3	emory, facial, is, skim, thanks, that
Predicted Response BOW #4	all, are, other, thank, that, you
Predicted Response BOW #5	for, it, john, player, re, you
Predicted Response BOW #6	me, much, owe, tell, thank, you
Predicted Response BOW #7	change, gift, on, the, was, why
Predicted Response BOW #8	call, right, single, that, was, when
Predicted Response BOW #9	are, feel, her, what, where, you
Predicted Response BOW #10	are, know, me, no, what, you

Table 4.6: AC-GAN predicted response words example from DailyDialog test set

finetuned T5 model does indeed generate responses using the response words predicted by our second stage model by computing the percentage of generated responses in the test set of the datasets which use at least one predicted in the generated response. Additionally, we compute average number of predicted words used in the generated responses. We report these numbers for both datasets in table 4.8. Note that we predict 6 words for each response.

4.2.5 Results and Analysis

We report the results of the experiments on the DailyDialog and Switchboard dataset test sets in tables 4.9 and 4.10. We also provides the some of the response samples generated for the DailyDialog and Switchboard datasets in tables 4.11 and 4.12. We provide the query utterance, the predicted response words, and the generated response utterance.

We see that the proposed approach outperforms the T5 baseline model on every reported quantitative metric for both the DailyDialog and Switchboard datasets. When comparing the results of both the T5 baseline model and our proposed approach with the DialogWAE and VAE-AM models, we notice that, on the whole, these models seem to generate more diverse outputs. Our proposed approach outperforms the DialogWAE model on the perplexity measure for both the datasets but lags behind the VAE-AM model.

Based on the bleu scores, the DialogWAE and the VAE-AM do generate responses which have much higher n-gram overlap with the ground truth response when compared to our proposed approach. We hypothesize that the one of the reason could be due to the low recall and precision scores achieved by our second step model on both the datasets. We believe that improvement in our second step model could lead to much higher bleu scores.

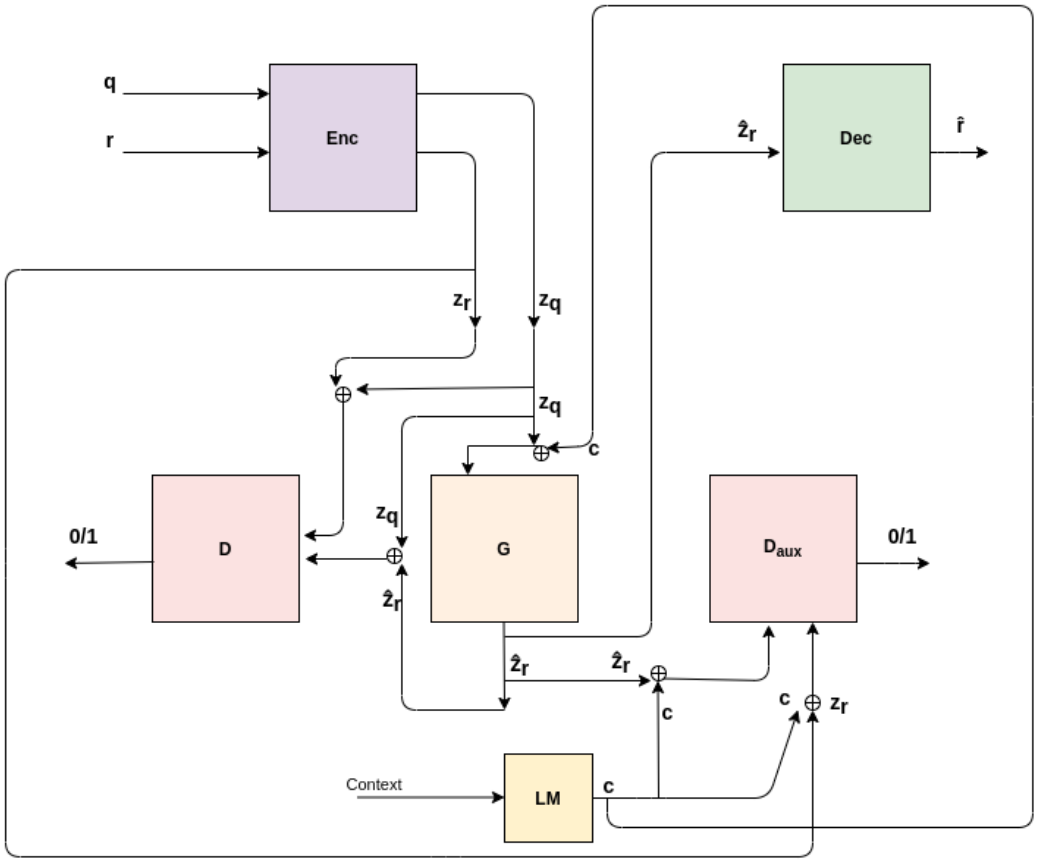


Figure 4.2: Step 2: Auxiliary Classifier GAN (AC-GAN). Pretrained encoder from step 1 is utilized to get the latent codes $\mathbf{z_q}$ and $\mathbf{z_r}$ of the query ($\mathbf{q}$) and response ($\mathbf{r}$) utterance bag-of-words, respectively. We use the pretrained language model from step 1 to encode the context as a fixed size continuous vector $\mathbf{c}$. The input to the generator is the concatenation of the query latent code $\mathbf{z_q}$ and the context vector $\mathbf{c}$ which maps it to the corresponding response latent code $\hat{\mathbf{z}}_\mathbf{r}$. When training the generator G, we aim to match $\mathbf{z_r}$ and $\hat{\mathbf{z}}_\mathbf{r}$ through the generator loss combined with a mean-squared error loss. When training the discriminator D, we pass $\mathbf{z_{qr}}$ and $\hat{\mathbf{z}}_\mathbf{qr}$ (obtained by concatenating $\mathbf{z_q}$ with $\mathbf{z_r}$ and $\mathbf{z_q}$ with $\hat{\mathbf{z}}_\mathbf{r}$, respectively) through a classification layer that tries to guess its source of input. We train the auxiliary discriminator to perform a similar task except we use the $\mathbf{c}$ for concatenation instead of $\mathbf{z_q}$. **Note:** $\oplus$ denotes concatenation.

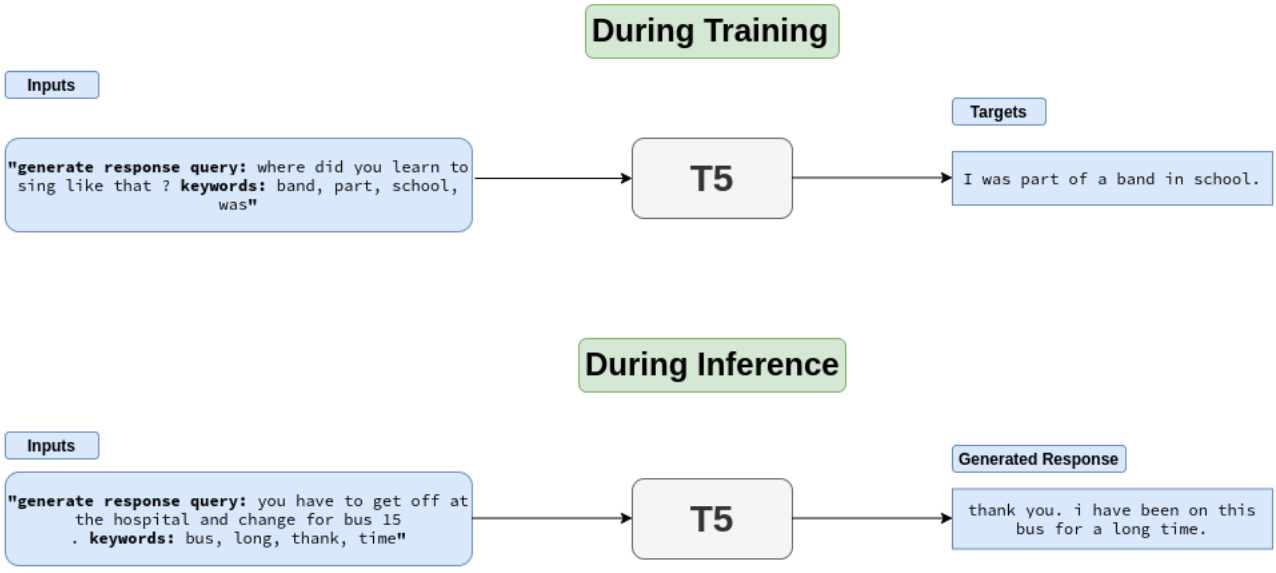


Figure 4.3: Step 3: T5 Finetuning. We finetune T5 to perform the task of dialog response generation using the keywords predicted by the AC-GAN in step 2. During training, we use the keywords extracted from the ground truth response. We limit the number of extracted keywords to be the same as the number of keywords used during the BOW-VAE training. During inference, we use the keywords predicted by the AC-GAN in step 2.

Query Utterance	use their influence in not always the most positive ways
Query BOW	always, influence, most, positive, use, ways
Ground Truth Response Utterance	yes that's true i think they do things that make it easier for our elected representatives to live the good life
Ground Truth Response BOW	easier, elected, life, live, representatives, true
Predicted Response BOW #1	call, executive, hum, into, manager, um
Predicted Response BOW #2	are, live, not, re, still, think
Predicted Response BOW #3	much, politically, that, themselves, think, true
Predicted Response BOW #4	absolutely, do, expect, oh, we, yes
Predicted Response BOW #5	budget, do, okay, so, we, what
Predicted Response BOW #6	but, easier, know, mean, that, life
Predicted Response BOW #7	care, checks, deductible, purchase, them, without
Predicted Response BOW #8	help, little, lot, of, things, way
Predicted Response BOW #9	good, idea, see, that, true, very
Predicted Response BOW #10	can, more, of, on, or, that

Table 4.7: AC-GAN predicted response words example from Switchboard test set

Dataset	% of responses with at least one predicted word used	Avg. Words Used
DailyDialog	99.71	4.76
Switchboard	99.92	5.18

Table 4.8: Computing percentage of generated responses using at least one predicted response word and the average number of predicted response words used on the DailyDialog and Switchboard test set

Model	BLEU			Diversity						Fluency
	P	R	F	Intra-1	Intra-2	Inter-1	Inter-2	ASL (14.43)	TTR	PPL
T5	0.155	0.226	0.184	0.82	0.98	0.50	0.84	11.49	**0.076**	236.09
DialogWAE	0.279	0.365	0.316	0.79	0.92	0.35	0.68	19.84	0.007	161.86
VAE-AM	**0.314**	**0.371**	**0.34**	**0.847**	0.98	0.41	0.73	15.3	0.036	**119.39**
Ours	0.221	0.274	0.245	0.84	**0.99**	**0.52**	**0.89**	13.05	0.051	127.06

Table 4.9: DailyDialog Dataset results

Model	BLEU			Diversity						Fluency
	P	R	F	Intra-1	Intra-2	Inter-1	Inter-2	ASL (8.49)	TTR	PPL
T5	0.129	0.188	0.153	0.79	0.98	0.48	0.84	7.55	**0.073**	321.64
DialogWAE	0.267	**0.394**	**0.318**	0.779	0.844	0.325	0.513	14.7	0.019	183.82
VAE-AM	**0.271**	0.372	0.313	**0.954**	0.966	0.412	0.559	8.72	0.065	**136.81**
Ours	0.19	0.247	0.215	0.84	**0.99**	**0.52**	**0.89**	10.90	0.064	163.07

Table 4.10: Switchboard Dataset results

Query Utterance	**Predicted Response Words**	**Generated Response Utterance**
good morning, mr zhang, i wonder whether you have got an answer from your superior.	afraid, leave, someone, to, today, told	I'm afraid not. I told you to leave today.
do you eat a lot of vegetables?	all, lot, much, of, them, too	I eat a lot of them, too much of them. All of them
I'm sorry, hold on a moment. I'll check it out. But we have already shipped it to you last month. Would you like us to contact the express company to know what's going on?	going, hope, things, time, to, work	i hope things are going to work out in time.
i'll tell you what, i can get you some hamsters and we'll take it from there.	good, great, of, that, the, would	that would be great. That would be good of you.
which book? this one?	about, conditions, here, one, our, which	here. Which one is about our conditions?

Table 4.11: DailyDialog Generated Response Samples

Query Utterance	**Predicted Response Words**	**Generated Response Utterance**
blues are okay i tell you i do like i grew up on blue grass music my dad is a big blue grass fan and so	for, huh, kinds, like, music, uh	uh-huh yeah i like all kinds of music for that
so that is another way of budgeting	account, amounts, bills, financial, investments, money	you know you have to put money in a financial account and put money in investments and bills
yeah so what are your favorite tv shows	news, on, other, the, tv, watch	i watch the news on the other tv
i think twin peaks went off	great, have, oh, really, see, would	oh really oh that would have been great to see
do you eat cheese do you eat eggs an	don, kidding, no, not, now, right	no kidding right now i don't

Table 4.12: Switchboard Generated Response Samples

Chapter 5

Conclusion and Future Work

5.1 Summary

In this work, we proposed a novel multi-stage approach for generating dialog response. In the first stage of the proposed approach, we train a bag-of-words variational autoencoder (Miao et al., 2015) to encode bag-of-words representation of each utterance in the dialog corpus into a continuous fixed-size latent vector. We use tfidf-weighting statistics to extract upto n informative words from each utterance to build the one-hot encoded vector representation to be used as the input. While decoding, we pick n words corresponding to the top probabilities. Additionally, we train a language model to encode dialog history into a continuous fixed size vector. In the second stage, we train an axuiliary classfier GAN (Odena et al., 2016) to learn to transform the query utterance BOW latent code and the encoded dialog history into the response utterance latent code. This predicted response latent code is fed to the decoder from the first stage to obtain the predicted response utterance words. In the third and final stage, we fine-tune a pretrained transoformer based text-to-text transfer model (T5) (Raffel et al., 2019) to generate generate response utterance conditioned on the query response utterance and the predicted response words. While fine-tuning the T5 model on the training set of the dialog corpus, we use the ground truth response utterance words. During the inference, we use the response utterance words predicted by our second stage model and decoded using the decoder from the first stage.

We provide detailed description of a previous joint work on dialog response generation which provided the motivation for the proposed approach. We evaluate the proposed approach on two dialog datasets: DailyDialog (Y. Li et al., 2017) and Switchboard (Godfrey et al., 1992). Using quantitative evaluation metrics, we showed that the proposed approach

generates diverse and fluent sentences. Furthermore, as part of our multi-stage approach, we proposed a novel fine-tuning regimen for pretrained T5 model which can be used to generate response utterances conditioned on the query utterance and response words.

5.2 Future Work

There many directions in which we can extend this work. We showed using quantitative metrics that the proposed approach generates diverse and fluent response utterances when compared to the state of the art approaches but lags behind when using the n-gram bleu metric. We would like to apply this method to a text generation task where diversity and fluency are distinctly more important than overlap with the ground truth reference text. One such example is the story generation task.

We would also like to explore ideas to improve the performance of the second stage such as selecting word for bag-of-words using a different criteria such as part-of-speech tag, thereby generating more meaningful latent codes. Using an end-to-end auto-regressive model which combines the first and second stage allowing us to predict response words using the entire dialog history. We fine-tune the T5 model in the third stage by updating all the available parameters. Various strategies have been proposed for fine-tuning large pretrained transformer based models which drastically reduce the number of updatable parameters. One such strategy is using adaptable layers (Bapna et al., 2019). In the future, we would like to explore adapting this strategy to reduce the computational resource requirements which would allow us to use bigger T5 models for fine-tuning.

References

Arjovsky, M., & Bottou, L. (2017). Towards principled methods for training generative adversarial networks.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate.

Bahuleyan, H., Mou, L., Vechtomova, O., & Poupart, P. (2018). Variational attention for sequence-to-sequence models, In *Proceedings of the 27th international conference on computational linguistics*, Santa Fe, New Mexico, USA, Association for Computational Linguistics. https://www.aclweb.org/anthology/C18-1142

Bahuleyan, H., Mou, L., Zhou, H., & Vechtomova, O. (2019). Stochastic Wasserstein autoencoder for probabilistic sentence generation, 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)*, Minneapolis, Minnesota, Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1411

Bao, J., Chen, D., Wen, F., Li, H., & Hua, G. (2017). Cvae-gan: Fine-grained image generation through asymmetric training, In *Proceedings of the ieee international conference on computer vision (iccv)*.

Bapna, A., Arivazhagan, N., & Firat, O. (2019). Simple, scalable adaptation for neural machine translation.

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, *3*(null), 1137–1155.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A., Jozefowicz, R., & Bengio, S. (2016). Generating sentences from a continuous space, In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, Berlin, Germany, Association for Computational Linguistics. https://doi.org/10.18653/v1/K16-1002

Brock, A., Donahue, J., & Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis.

Che, T., Li, Y., Jacob, A. P., Bengio, Y., & Li, W. (2017). Mode regularized generative adversarial networks, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. https://openreview.net/forum?id=HJKkY35le

Chen, B., & Cherry, C. (2014). A systematic comparison of smoothing techniques for sentence-level BLEU, In *Proceedings of the ninth workshop on statistical machine translation*, Baltimore, Maryland, USA, Association for Computational Linguistics. https://doi.org/10.3115/v1/W14-3346

Chen, S. F., & Goodman, J. (1996). An empirical study of smoothing techniques for language modeling, In *Proceedings of the 34th annual meeting on association for computational linguistics*, Santa Cruz, California, Association for Computational Linguistics. https://doi.org/10.3115/981863.981904

Chiu, J. P., & Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, *4*, 357–370. https://doi.org/10.1162/tacl_a_00104

Dai, B., Fidler, S., Urtasun, R., & Lin, D. (2017). Towards diverse and natural image descriptions via a conditional gan, In *Proceedings of the ieee international conference on computer vision (iccv)*.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, *abs/1810.04805*arXiv 1810.04805. http://arxiv.org/abs/1810.04805

Godfrey, J. J., Holliman, E. C., & McDaniel, J. (1992). Switchboard: Telephone speech corpus for research and development, In *Proceedings of the 1992 ieee international conference on acoustics, speech and signal processing - volume 1*, San Francisco, California, IEEE Computer Society.

Gong, M., Xu, Y., Li, C., Zhang, K., & Batmanghelich, K. (2019). Twin auxiliary classifiers GAN. *CoRR*, *abs/1907.02690*arXiv 1907.02690. http://arxiv.org/abs/1907.02690

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., & Bengio, Y. (2014). Generative adversarial nets (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger, Eds.). In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 27: Annual conference on neural information processing systems 2014, december 8-13 2014, montreal, quebec, canada*. http://papers.nips.cc/paper/5423-generative-adversarial-nets

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

Gu, X., Cho, K., Ha, J.-W., & Kim, S. (2018). Dialogwae: Multimodal response generation with conditional wasserstein auto-encoder.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes.

Kneser, R., & Ney, H. (1995). Improved backing-off for m-gram language modeling, In *Icassp*. https://doi.org/10.1109/ICASSP.1995.479394

Koehn, P., Och, F. J., & Marcu, D. (2003). Statistical phrase-based translation, In *Proceedings of the 2003 human language technology conference of the north American chapter of the association for computational linguistics*. https://www.aclweb.org/anthology/N03-1017

Kryściński, W., Keskar, N. S., McCann, B., Xiong, C., & Socher, R. (2019). Neural text summarization: A critical evaluation.

Lee, K., He, L., Lewis, M., & Zettlemoyer, L. (2017). End-to-end neural coreference resolution. *CoRR*, *abs/1707.07045*arXiv 1707.07045. http://arxiv.org/abs/1707.07045

Li, J., Galley, M., Brockett, C., Gao, J., & Dolan, B. (2016). A diversity-promoting objective function for neural conversation models, In *Proceedings of the 2016 conference of the north American chapter of the association for computational linguistics: Human language technologies*, San Diego, California, Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-1014

Li, J., Monroe, W., Shi, T., Ritter, A., & Jurafsky, D. (2017). Adversarial learning for neural dialogue generation. *CoRR*, *abs/1701.06547*arXiv 1701.06547. http://arxiv.org/abs/1701.06547

Li, Y., Su, H., Shen, X., Li, W., Cao, Z., & Niu, S. (2017). DailyDialog: A manually labelled multi-turn dialogue dataset, In *Proceedings of the eighth international joint conference on natural language processing (volume 1: Long papers)*, Taipei, Taiwan, Asian Federation of Natural Language Processing. https://www.aclweb.org/anthology/I17-1099

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

Luong, T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation, In *Proceedings of the 2015 conference on empirical methods in natural language processing*, Lisbon, Portugal, Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1166

Miao, Y., Yu, L., & Blunsom, P. (2015). Neural variational inference for text processing. *CoRR*, *abs/1511.06038*arXiv 1511.06038. http://arxiv.org/abs/1511.06038

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space (Y. Bengio & Y. LeCun, Eds.). In Y. Bengio & Y. LeCun

(Eds.), *1st international conference on learning representations, ICLR 2013, scottsdale, arizona, usa, may 2-4, 2013, workshop track proceedings*. http://arxiv.org/abs/1301.3781

Mikolov, T., Karafiát, M., Burget, L., ernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model, In *Interspeech*.

Mino, A., & Spanakis, G. (2018). Logan: Generating logos with a generative adversarial neural network conditioned on color.

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets.

Morin, F., & Bengio, Y. (2005). Hierarchical probabilistic neural network language model, In *Aistats*.

Nallapati, R., Zhou, B., dos santos, C. N., Gulcehre, C., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond.

Nisioi, S., Štajner, S., Ponzetto, S. P., & Dinu, L. P. (2017). Exploring neural text simplification models, In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 2: Short papers)*, Vancouver, Canada, Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-2014

Odena, A., Olah, C., & Shlens, J. (2016). Conditional image synthesis with auxiliary classifier gans.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation, In *Proceedings of the 40th annual meeting on association for computational linguistics*, Philadelphia, Pennsylvania, Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073135

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation (A. Moschitti, B. Pang, & W. Daelemans, Eds.). In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 conference on empirical methods in natural language processing, EMNLP 2014, october 25-29, 2014, doha, qatar, A meeting of sigdat, a special interest group of the ACL*, ACL. https://doi.org/10.3115/v1/d14-1162

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer.

Reiter, E., & Dale, R. (1997). Building applied natural language generation systems. *Nat. Lang. Eng.*, *3*(1), 57–87. https://doi.org/10.1017/S1351324997001502

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton project para*. Cornell Aeronautical Laboratory. https://books.google.ca/books?id=P%5C_XGPgAACAAJ

Sato, S., Yoshinaga, N., Toyoda, M., & Kitsuregawa, M. (2017). Modeling situations in neural chat bots, In *Proceedings of ACL 2017, student research workshop*, Vancouver, Canada, Association for Computational Linguistics. https://www.aclweb.org/anthology/P17-3020

See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *CoRR*, *abs/1704.04368*arXiv 1704.04368. http://arxiv.org/abs/1704.04368

Seo, M. J., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. *CoRR*, *abs/1611.01603*arXiv 1611.01603. http://arxiv.org/abs/1611.01603

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A., & Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models, In *Proceedings of the thirtieth aaai conference on artificial intelligence*, Phoenix, Arizona, AAAI Press.

Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., & Bengio, Y. (2016). A hierarchical latent variable encoder-decoder model for generating dialogues.

Shang, L., Lu, Z., & Li, H. (2015). Neural responding machine for short-text conversation. *CoRR*, *abs/1503.02364*arXiv 1503.02364. http://arxiv.org/abs/1503.02364

Shen, X., Su, H., Niu, S., & Demberg, V. (2018). Improving variational encoder-decoders in dialogue generation. *CoRR*, *abs/1802.02032*arXiv 1802.02032. http://arxiv.org/abs/1802.02032

Sohn, K., Yan, X., & Lee, H. (2015). Learning structured output representation using deep conditional generative models, In *Proceedings of the 28th international conference on neural information processing systems - volume 2*, Montreal, Canada, MIT Press.

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., & Dolan, B. (2015). A neural network approach to context-sensitive generation of conversational responses.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, *15*(1), 1929–1958. http://dl.acm.org/citation.cfm?id=2670313

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). Lstm neural networks for language modeling, In *Interspeech*.

Surya, S., Mishra, A., Laha, A., Jain, P., & Sankaranarayanan, K. (2018). Unsupervised neural text simplification.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *CoRR*, *abs/1409.3215*arXiv 1409.3215. http://arxiv.org/abs/1409.3215

Tolstikhin, I., Bousquet, O., Gelly, S., & Schoelkopf, B. (2017). Wasserstein auto-encoders.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *CoRR*, *abs/1706.03762*arXiv 1706.03762. http://arxiv.org/abs/1706.03762

Vinyals, O., & Le, Q. (2015). A neural conversational model.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., & Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, *abs/1910.03771*.

Xu, J., Ren, X., Lin, J., & Sun, X. (2018). Diversity-promoting GAN: A cross-entropy based generative adversarial network for diversified text generation, In *Proceedings of the 2018 conference on empirical methods in natural language processing*, Brussels, Belgium, Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1428

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, *abs/1906.08237*arXiv 1906.08237. http://arxiv.org/abs/1906.08237

Yin, J., Jiang, X., Lu, Z., Shang, L., Li, H., & Li, X. (2015). Neural generative question answering.

Zhang, Y., Gan, Z., Fan, K., Chen, Z., Henao, R., Shen, D., & Carin, L. (2017). Adversarial feature matching for text generation.

Zhao, T., Zhao, R., & Eskenazi, M. (2017). Learning discourse-level diversity for neural dialog models using conditional variational autoencoders.

Zhou, Q., Yang, N., Wei, F., Huang, S., Zhou, M., & Zhao, T. (2018). Neural document summarization by jointly learning to score and select sentences, In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*, Melbourne, Australia, Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1061