# Controlled Generation of Stylized Text Using Semantic and Phonetic Representations

by

Egill Ian Gudmundsson

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, 2021

## Abstract

Neural networks are a popular choice of models for the purpose of text generation. Variational autoencoders have been shown to be good at reconstructing text and generating novel text. However, controlling certain aspects of the generated text (e.g., length, semantics, cadence) has proven a more difficult task. The objectives of disentanglement and controlled text generation have thus become areas of interest, with various approaches depending on the aspects we desire to control.

In this work we study controllable generation of lyric text based on semantic and phonetic criteria. The phonetic information takes the form of generalized phonetic patterns. A Bag-of-Words Variational Autoencoder (VAE) extracts and models the semantic information, while a phonetic pattern VAE handles the phonetic information. Each uses several regularization techniques for its respective latent space and the information from each is fed to a lyrics decoder to generate novel lyric lines that would satisfy both the Bag-of-Words and phonetic constraints.

The experiments show that our model can learn to reconstruct phonetic patterns extracted from text and use them with the Bag-of-Words representations to reconstruct the original lyric lines. Together, the learned representations of phonetic patterns and Bag-of-Words constraints can be used to generate new lyrics.

## Acknowledgements

I would like to thank Dr. Olga Vechtomova for her supervision and guidance during my studies at the University of Waterloo. Her help was paramount in the implementation of this thesis, as was her patience with me during these very trying times. I have benefited greatly from her assistance, advice and support throughout the entirety of my graduate studies.

Thank you Dr. Hoey and Dr. Harris for reviewing the thesis and providing valuable feedback.

Many thanks to lab members and friends Utsav Das, Gaurav Sahu, Drhuv Kumar, Vikash Balasubramanian, Kashif Khan, Brian Zimmerman, and Olivier Poulin for interesting discussion and instruction with various topics.

Thanks to the lab members and other friends for making the stay in Waterloo memorable, despite the pandemic's best efforts.

Last, but in no way least, I would like to thank my family. Without them, none of this would have been possible. For your unwavering support and love, I am eternally grateful.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background and Motivation

It has been a long-sought goal for many decades now to achieve Artificial Intelligence (AI), computer models capable of emulating human thought and reasoning. The first steps towards realizing this goal were taken as far back as 1943 [McCulloch and Pitts, 1943] with the design of artificial neurons. Today, certain models have already surpassed human ability in certain restricted domains, like game playing. A famous example is the AlphaGo model [Silver et al., 2017] capable of beating world-class players at the game of Go.

With the use of Machine Learning (ML), algorithms designed to leverage large amounts of data to learn optimal strategies, and Neural Networks, models that emulate the architecture of biological brain cells with code, these models' abilities have improved by leaps and bounds over the last several years. While greatly impressive on its own, there are still many challenges to be overcome to achieve the general AI standards of simulating a human mind.

One of the challenges that remains unsolved is that of getting AI models to understand the intricacies of human languages and making them capable of using language to convey information. This challenge as a whole is referred to as Natural Language Processing (NLP). NLP has been been broken down into various sub-fields such as question answering, text simplification, translation, text generation and more.

Text generation can be defined as the task of taking examples of text, extrapolating from them and generating novel text that shares some characteristics with the examples. Thus it becomes a balance between keeping some syntactic characteristics (i.e. the structure and grammar) and/or semantic characteristics (i.e. the meaning behind the words) and generating sequences that have enough diversity in both syntax and semantics to constitute new and novel texts.

Various text generation methods exist utilizing machine learning for a diverse set of purposes. One of these purposes or sub-tasks is lyric generation. Many methods utilizing language models, transformers and recurrent neural networks have been engineered for tasks such as text simplification, dialogue generation and auto-completion. Most of these tasks have clear goals and measurable results, often times with automatic metrics to measure performance. When generating lyrics, evaluating the quality can be a fraught task since there are few metrics to differentiate good lyrics from bad lyrics. Measuring other characteristics such as musicality, novelty and creativity can be both difficult and largely dependant on opinion. There is still much research to do in this field and some things that have been explored very little, if at all.

One of these is the effect of phonetic text structure on the quality of lyrics. Many lyrical and poetic characteristics can be defined with relation to phonemes. Rhyme and alliteration can both be defined as phonetic patterns within lyrics and prose. When singing, certain types of phonemes are easier to sing than others and put less strain on the voice. This may make it desirable to use certain phonemes at the end of lyric lines, when notes are often left to ring out for a while. The accompanying music may put constraints on the meter or rhythm of the lyrics, meaning we would want to use patterns that conform to the cadence of the music or at least use patterns that do not conflict with the music. Certain phonemes are harsher and others softer, meaning that an emphasis on some might serve better or worse depending on aggression and energy in the genre being played. Phonetic patterns may also contain information about meter and rhythm, both important in keeping a certain pace when singing. These patterns may even have a distinct distribution between artists and contain other lyrical information.

We believe that this makes it abundantly clear that phonemes and their structures play a big role in how to write lyrics and thus paying special attention to these may lead to text generation models with a better grasp on how to generate lyrics. Our hopes are to present a novel way of incorporating this information into the text generation task to produce better lyrics.

## 1.2 Problem Definition

In this work, we will explore the express use of phonetic information in the generation of lyrics, as mentioned above. The evaluation of the "lyricality" of text can be very hard to estimate. Our hope is that by training a model to reconstruct a set of lyrics, it can pick up on what makes a piece of text lyrical. We will also keep track of how well it reconstructs phonetic structures, novelty and grammaticality of the lyrics generated. The end goal with this model is that it can then be used as an aid in writing novel lyrics that allows for fine-tuned control over semantics, cadence and other phonetic characteristics. When writing lyrics, we may hit upon a theme we are fond of but not manage to fit it into the cadence of a song. I.e. we have decided upon a melody or rhythm but can't get the lyrics to fit said melody and align it to the beats of the music. The other case is having a cadence, melody or a line of lyrics that one would like to change the theme of. This could be changing the mood, emotion or semantic aspects while retaining the rhythmic characteristics of the text, i.e. generating new text while preserving the stresses and pauses present in the flow of the accompanying music. In certain instances, we might only be able to sing a certain number of syllables before a line ends or need lyrics to line up with the music in specific ways. In the case of needing to fit a theme to a cadence, a user could supply a Bag-of-Words (BoW) with semantics and the model would produce lyrics with that theme. In the case of wanting to change the theme of the lyrics, one might have placeholder lyrics that can be input to guide the meter and the model would produce new lyrics with different themes and emotions. Another option would be to combine phonetic and semantic information from existing lines to generate new lyrics.

Given a corpus $\boldsymbol{X}$ of $l$ lyrical lines $\boldsymbol{x}$, we would like to use this data to make reconstructions $\hat{\boldsymbol{x}}$ of the original lyrics during training and produce new lyrics $\hat{\boldsymbol{s}}$ not found in the corpus. For each of the lines $\boldsymbol{x}$, we extract the semantic information and phonetic information. The semantic information will take the form of Bag-of-Words information projected into a latent space $Z_{BoW}$ with each line's information being encoded into a latent vector $\boldsymbol{z}_{BoW}$. Similarly, the line gets transformed into a sequence of phonemes and then those phonemes are projected into a latent space $Z_{phon}$ and the line's phonetic information is encoded into a latent vector $\boldsymbol{z}_{phon}$.

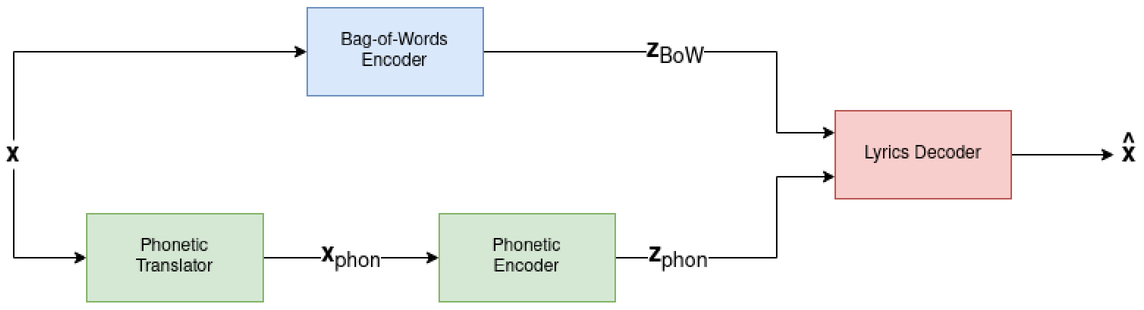


Figure 1.1: Overview of system components used for lyric reconstruction

These two latent variables are then fed into a decoder to produce the reconstructed lyrics $\hat{\boldsymbol{x}}$, which should ideally be the same as the initial $\boldsymbol{x}$. Using the Bag-of-Words and Phonetic models, we can also generate lyrics from scratch without any input. The training of the three models is more complicated and further details are provided in the Methodology chapter.

## 1.3 Contributions

Several contributions were made in the pursuit of implementing the system described above:

- A model that expressly extracts phonetic patterns from lyrics and is capable of generating new phonetic patterns to guide the lyric generation.
- A method for the controllable generation of lyrics based on both semantic and phonetic constraints provided, in that same model.
- Results from experimenting with different phonetic representations and vocabularies to determine which ones best suit a generative and reconstructive model.
- A method for receiving semantics from a separate model and the phonetic information, from the model mentioned above. It then uses the information from both to generate lyrics.
- A comparison of different training methods, i.e. pre-training the first two models versus training them alongside the final decoder, to see which methods produce the best results.
- Suggestions and discussion on how to evaluate results of lyric generation in concrete terms.

## 1.4 Chapter Outline

In chapter 2, we introduce the necessary background information to understand the workings of the models presented. In chapter 3, we talk further about the implementation and architecture of the models used. In chapter 4, we talk about the experiments performed and the quantitative results gained from experimentation, as well as the different metrics used to evaluate our models. Finally, chapter 5 talks about the conclusions we can draw from the results and what work could be expanded upon to perform additional experiments.

# Chapter 2

# Background and Related Work

## 2.1 Neural Network Models

Virtually all of the current ML models being used rely on variants of an architecture called a Neural Network (NN). The idea behind a NN is to model the structure found in biological neural cells, such as those found in brains, using aggregates of cells or neurons. The simplest architecture for a neuron is the classic Perceptron [Rosenblatt, 1958]. At its core, a Perceptron is a neuron that takes in a series of inputs $x_n$, multiplies them with learned weights $w_n$, adds a learned bias $b$ and then uses an activation function $f$ to determine its outputs $y_n$.

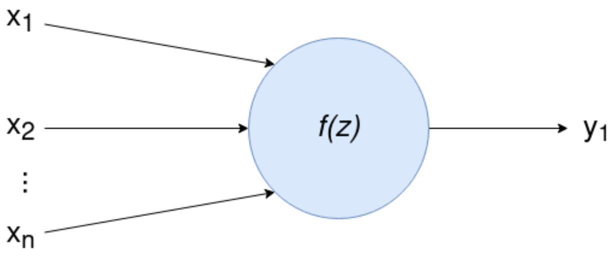


Figure 2.1: Simplified Perceptron representation

The variable $z$ is calculated using the method above and then the activation function uses that variable to determine output. One of the simplest activation functions we can use is a threshold function with a threshold of 0.5. We can define $z$ and the threshold activation function with the formulas:

$$z = \sum_{i=1}^{n} x_1 w_1 + b_i$$

$$y = \begin{cases} 1, & \text{if } z > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

The selection of $f$ will determine various effects of how effective the neuron is and how easy it is to use them in conjunction with other neurons to form a trainable model. A threshold function is simple, but not ideal. One of the most common activation functions is the sigmoid function $\sigma$ given with the formula $\sigma(x) = \frac{1}{1+e^{-x}}$. Another popular activation function is tanh, given as $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Generally speaking, a differentiable activation function is preferable in order to be able to perform backpropagation (which we will cover later) to adjust a model's weights.

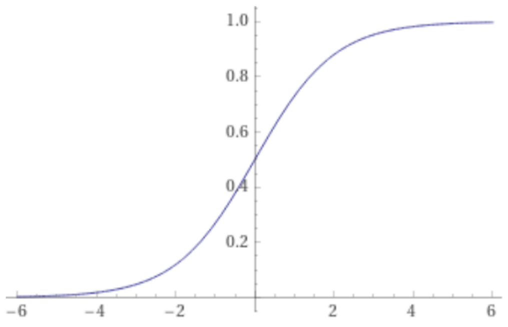


Figure 2.2: The sigmoid function

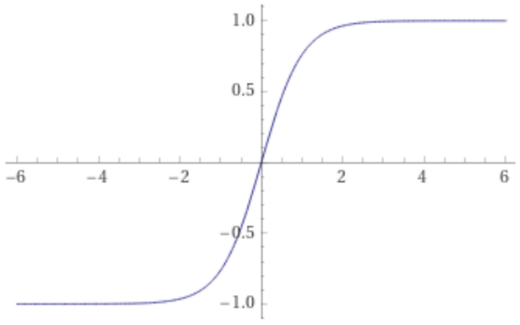


Figure 2.3: The tanh function

Once we have multiple neurons at our disposal, we can start linking them together to form Neural Networks. Different sets of of neurons can be grouped into layers, where there are no connections between the neurons within a layer. A NN must have at least two layers, an input and an output layer. In between these two layers, we can add additional layers called hidden layers. If a NN has at least one hidden layer, it is considered a Deep Neural Network.

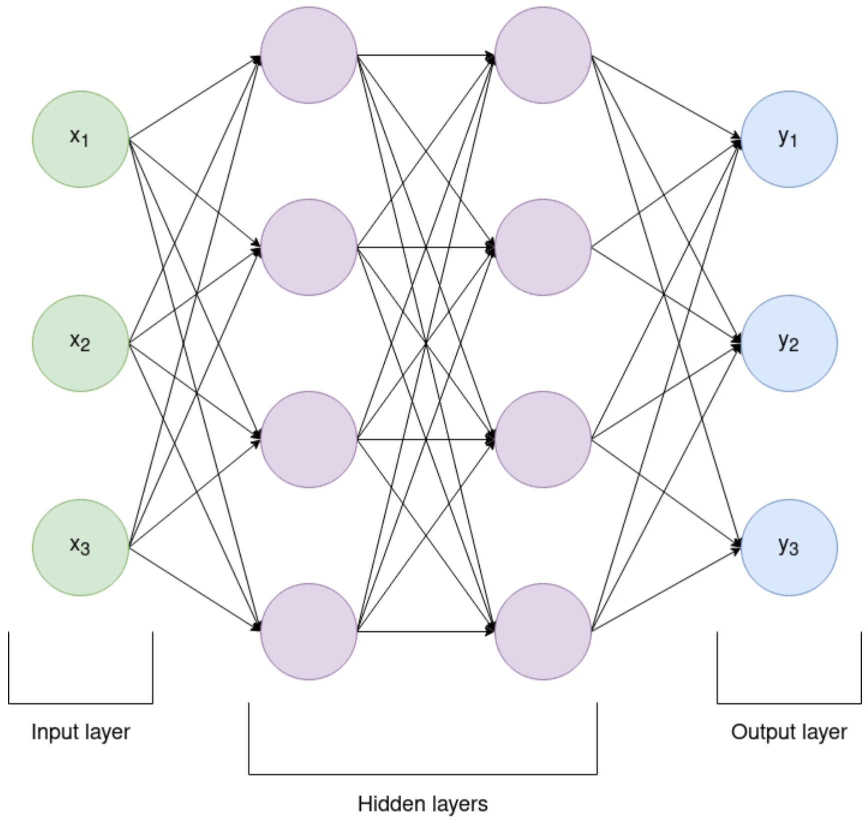


Figure 2.4: Example of Deep Neural Network with two hidden layers of the same size

Figure 2.4 shows an example of a fully connected NN, meaning that each of the neurons in one layer is connected to every neuron in the next layer. The input sequence is fed into the input layer. Then the output of each layer is fed to the next layer until it reaches the final layer, the output layer. The output from the output layer is then used as the final output of the entire model. When input is passed along like this all through the model, it is called a forward pass. A NN capable of doing this is called a feed-forward network and is capable of extracting knowledge from the inputs passed to it.

These outputs of the model are its prediction for a given input. For some input sequence of words, these outputs might correspond to a prediction of sentiment, relevancy, translation or something else. Ideally, we would want to be able to instruct the model now about how correct or incorrect it was in its predictions and the model could then update its weights to make better predictions in the future. One way of achieving this is performing backpropagation [David E. Rumelhart, 1986].

Roughly speaking, backpropagation is the way in which we update the weights of a model to reduce error in subsequent forward passes. The output of the model is from the last layer and with it we can calculate how far off the model's guess is, i.e. the error. Updating the weights in the last layer that gives us the output is easy enough. But updating the intermediate layers is harder since it is not immediately clear how much each neuron in the intermediary layers contribute to the result and thus even harder to say how to update their weights. A simple way of calculating the error $E$ of a model's predictions with parameters or weights $\theta$ for a set $\boldsymbol{X}$ of $n$ number of samples (each sample $x$ having some associated true prediction $y$ and prediction made by the model $\hat{y}$) is using the mean squared error:

$$E(\boldsymbol{X}, \theta) = \frac{1}{n} \sum_{k=1}^{n} (\hat{y}_k - y_k)^2$$

For a given weight $w$ in layer $l$ to node $j$, from node $i$ in the previous layer, we would like to be able to calculate how much that given weight contributed towards the total error. We can derive this amount by using partial derivatives and the chain rule. The error with regard to a certain weight can be written as:

$$\frac{\delta E}{\delta w_{ij}^{l}} = \frac{1}{n} \sum_{k=1}^{n} \frac{\delta E_d}{\delta w_{ij}^{l}}$$

This gives us the amount by which we need to adjust a certain weight. Given a hyper-paramter called the learning rate, symbolized with $\eta$, that dictates how much we want to adjust the weights with each backpropagation, we can then adjust the weights with:

$$\Delta w_{ij}^{l} = \eta \frac{\delta E}{\delta w_{ij}^{l}}$$

## 2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [Jordan, 1986] [Rumelhart and McClelland, 1987] are different from NNs in the way that they process input. Not only do they use the input sequence, they also use hidden variables to keep track of sequence information and memorize dependencies of the sequences in the input and the generated output. There are several different types of RNNs, such as Gated Recurrent Units [Chung et al., 2014] and Bidirectional Recurrent Neural Networks [Schuster and Paliwal, 1997].

### 2.2.1 Long Short-Term Memory

A popular architecture choice for RNNs are so-called Long Short-Term Memory networks (LSTM) [Hochreiter and Schmidhuber, 1997]. Traditional NN models tend to have problems with either exploding or vanishing gradients. Gradient clipping can be used to somewhat mitigate gradients that have grown exponentially and keeping models smaller can reduce vanishing gradients as we backpropagate further, but these problems still persist. Thanks to the LSTM architecture, this problem is further reduced. It is also better than many NN architectures at modelling long-range dependencies in sequences, i.e. focusing on the entire sequence instead of just the current word.

RNNs are recurrent, meaning that their output gets passed back in for the next step, along with the input for that step. The inner workings of the network mean that it can decide to encode pertinent information into a hidden variable that will carry over to subsequent passes through the network. Let's start by looking at the equations that define the model's behavior [Sak et al., 2014].

$$f_t = \sigma(W_f x_t + b_f + W_{h_f} h_{t-1} + b_{h_f}) \tag{2.2}$$

$$i_t = \sigma(W_i x_t + b_i + W_{h_i} h_{t-1} + b_{h_i}) \tag{2.3}$$

$$g_t = \tanh(W_g x_t + b_g + W_{h_g} h_{t-1} + b_{h_g}) \tag{2.4}$$

$$o_t = \sigma(W_o x_t + b_o + W_{h_o} h_{t-1} + b_{h_o}) \tag{2.5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2.6}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.7}$$

Each of $i_t$, $f_t$, $g_t$ and $o_t$ have Neural Network layers and trainable weights of their own and $\odot$ is point-wise matrix multiplication. The model takes input $x_t$ and the states $h_{t-1}$ and $c_{t-1}$ from the previous time step for every word $x$ in the input sequence. The states $h$ and $c$ are often referred to as the hidden state and cell state respectively. The hidden state contains information pertinent for the next item in the sequence whereas the cell state contains information about long-range dependencies. Let's display the interaction of these variables in the model before going into further detail.

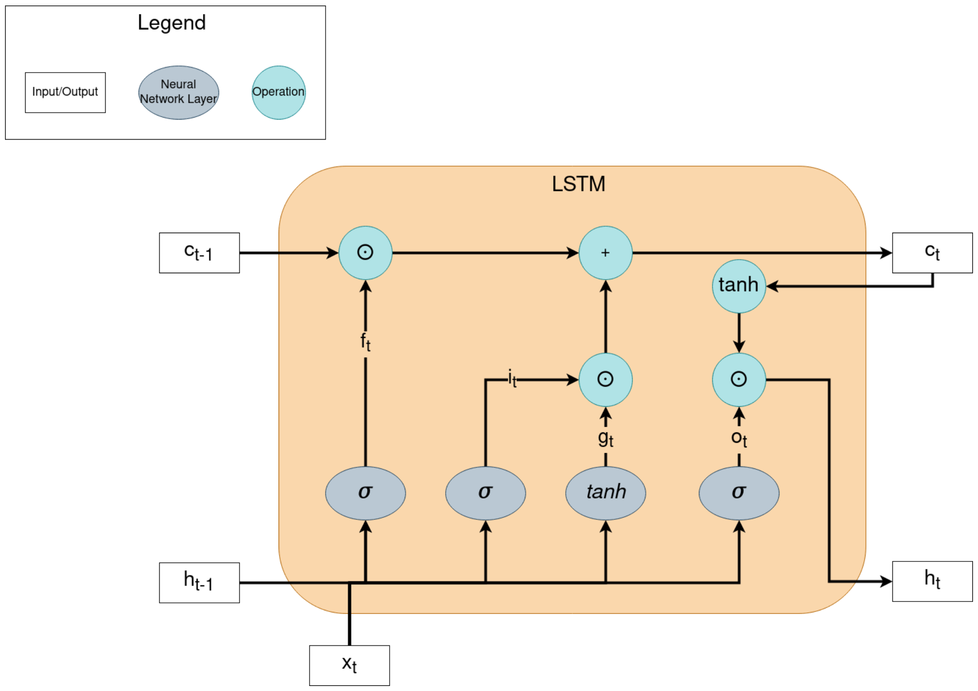


Figure 2.5: Inner workings of an LSTM model

For our very first step where $t = 0$, the states $c_0$ and $h_0$ are zero vectors. Our first variable to be calculated is $f_t$, often called the forget gate output, given by equation 2.2. Its output can be thought of as a sort of mask that reduces or nullifies cell state information from the previous steps. For instance, if we have a long, run-on sentence and the subject of the sentence has now changed, the output of the forget gate would erase features of the old subject in the cell state.

The two variables $i_t$ and $g_t$, given with equations 2.3 and 2.4 respectively, decide which information from the new word $x_t$ should be encoded into our new states. Once we have forgotten the unnecessary information from $C_{t-1}$ using $f_t$ (2.2), we can add in our new information using $i_t$ (2.3), $g_t$ (2.4) and then use equation 2.6 to calculate the next cell state $c_t$.

Now that we have gathered everything we need, we can calculate the next hidden state $h_t$ using equation 2.7. We use our newly calculated $c_t$ (2.6) and the output $o_t$ (2.5) of one of the sigmoid layers, thus infusing the information we got out of the current word input with any relevant long-term information and outputting $h_t$. For the next word $x_{t+1}$ our output states $c_t$ and $h_t$ get fed back into the model for the next time step $t + 1$ as inputs.

This is what makes the model recursive, the fact that it uses its previous output as input for the next steps, and the reason why it is called a RNN.

Most other RNN architectures generally only have the hidden state $h$ and no cell state $c$. They are good at building sequences where each word depends entirely on the word immediately preceding it. Once they need more information than just the previous word, that's when they start to falter. Adding the cell state $c$ mitigates this and gives us a more robust model capable of modelling long-term dependencies. Thus it has both long-term and short-term memory, giving it the slightly confusing name Long Short-Term Memory.

## 2.3 Tokenization and Word Embeddings

We have spoken about inputting sequences and words into models, but not provided details on how to make them understandable to models. In order to perform calculations with words, they need to be translated into mathematical concepts that can be fed into a statistical model. An ideal format that possesses enough information, but doesn't take up too much space, is a vector. One of the simplest ways of translating a word $x$ into a vector is using a one-hot vector. Given a vocabulary $V$ of size $v$, each index represents a word. Thus for a vocabulary of size $v = 5$, if the index for the word "computer" is 2 and 4 for "sky" (starting from 0), the one-hot vector for "computer" would be $[0, 0, 1, 0, 0]$ and for "sky" it would be $[0, 0, 0, 0, 1]$.

The obvious disadvantage here is that with larger vocabularies, one-hot vectors become increasingly sparse and wasteful with regards to space. This also means that computation would become increasingly complex, even though each vector would have very little information. A method that makes good use of space while providing a high level of granularity is Word2Vec [Mikolov et al., 2013a]. Another popular framework that similarly compresses words into more efficient vector form is GloVe [Pennington et al., 2014], but we will focus on Word2Vec here.

In this method, we have a corpus of sentences and a NN that uses another two NN models in order to divine the meaning of words. A dimensionality (the standard being 300) is chosen for the word-vector representations. For each new word encountered in the corpus, a 300-dimensional vector is uniformly randomly initialized and then passed to the two models to be fine-tuned. The first model is the Continuous Bag-of-Words (BoW) model. In this architecture, we iterate through each word in each sentence. The word is masked so the model doesn't know what it is and then uses the other words in the sentence to make a prediction and classify the masked word. The model not only uses the previous words, but also the future words to make its prediction.

The second model is a Continuous Skip-gram model, which is almost the opposite of the BoW model. The model is given one word from a sequence and then tries to predict a set amount of words before and after that word, with less weight given to words further away from our masked word. Thus it becomes a classification problem where a window of possible words is selected around a selected word and the model guesses which words occur there.

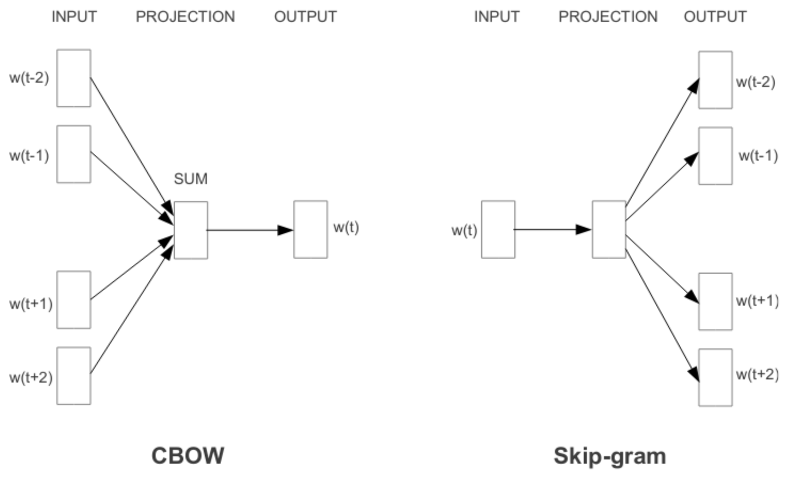


Figure 2.6: BoW and Skip-gram overview from [Mikolov et al., 2013a]

These two tasks of BoW and Skip-gram classification complement each other nicely and give the model the ability not only to understand each word on its own, but to also give it a better understanding of how words relate to each other. In fact, the words end up being arranged in the multi-dimensional vector space in a way where similar words are closer together.

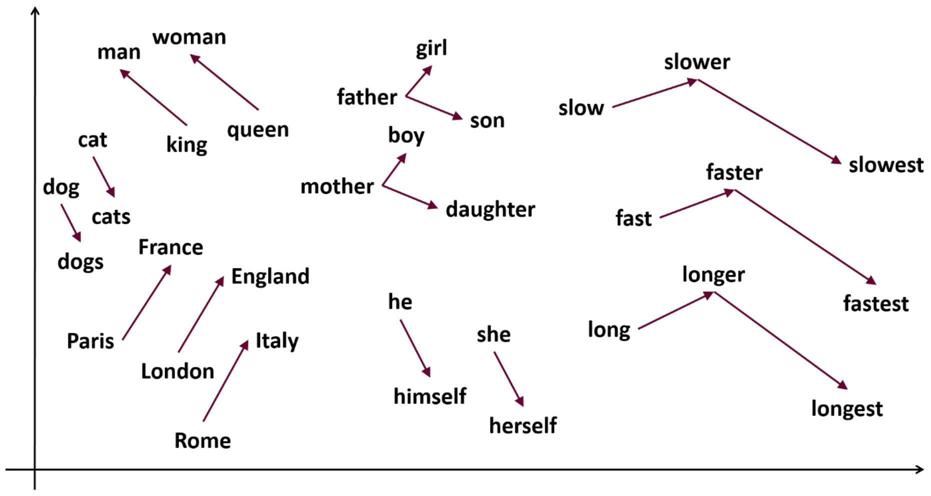


Figure 2.7: Stylized and simplified representation of Word2Vec word space from [Wor, 2019]

Not only that, but words that have similar types of relations will have similar relative distances between them, with an example shown in **??**. This shows that not only do the co-ordinates within this space correlate with a specific meaning, trajectories within the space also preserve relations between words. This means we can use measures such as L2 distances between word-vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ to determine the similarity between words like:

$$\text{similarity}(\mathbf{a}, \mathbf{b}) = ||\boldsymbol{a} - \boldsymbol{b}||^2$$

A slightly more reliable and traditional measure for word similarity, that uses the same idea, is cosine similarity. The cosine similarity of the same two vectors is defined as:

$$cos(\theta) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{||\boldsymbol{a}||^2||\boldsymbol{b}||^2}$$

The higher the value of the cosine similarity, the more similar the two words are. Conversely, the lower the cosine similarity, the less related the two are. this can be used to determine how far off a model's guess at a word is and perform backpropagation of error to train it. Now that we have mathematical word representations, we can start using them as inputs for models.

## 2.4 Sequence-to-Sequence Models

For the purposes of generating lyrics, we would like to be able to feed a model example lines of lyrics so that it can learn from them and then get it to output lyrics of its own. In other words, we want to feed the model sequences of words and get it to output sequences of words as well. This is a model that belongs to the class of models called Sequence-to-Sequence models [Sutskever et al., 2014], sometimes abbreviated Seq2Seq. In our case of reconstruction, the model should receive a line of lyrics as input and then output that same line. With regards to generation, we will go into further detail in subsection 2.4.3 about Variational Autoencoders.

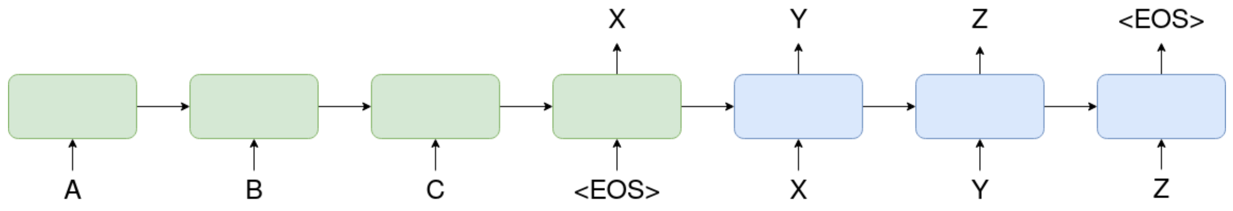


Figure 2.8: Example of an input sequence ABC and corresponding output sequence XYZ, where each token could be a word. E.g. ABC could correspond to an input line "and your love" and XYZ could be the model's prediction "also thine affection". The parts in green show the encoding steps and the parts in green show the decoding steps.

### 2.4.1 Language Models

Language Models (LMs) [Bengio et al., 2001] are somewhat similar to word embedding models, but have a different end goal. While word embedding models like Word2Vec produce vector representations of words and organize them in their vector space, LMs produce a probability function that informs us how likely a sequence of words is to occur. At its simplest, an LM can give the probability of a word being the next word in a sequence or sentence. We can denote this as:

$$P(x_n|x_1, x_2, ..., x_{n-1})$$

If we chain these predictions together for a sequence of words $X = \{x_1, x_2, ..., x_n\}$ that form a sequence, we can get a measure of how likely a sequence of words is to occur. We can denote this as:

$$P(X) = \prod_{i=1}^{n} P(x_i | x_1, x_2, ..., x_{i-1})$$

We can then use this type of prediction, along with some sort of smoothing function to make up for the gap between what the model sees during training and possible novel words it will encounter later on, to train an LM. Assuming that an LM is trained on a good set of data and similar to the one we are working with, we now have a metric of how good a fit a sequence of words is in our corpus. A higher probability means that the sequence $X$ shares characteristics with the sequences in our corpus, while a low probability signifies that the sequence does not conform to the corpus and is likely a bad fit. We will talk more about this relation in the chapter on Perplexity (subsection 4.3.2).

LMs have improved dramatically in the twenty years since they were conceptualized and have a plethora of different architectures these days. Older models only use Markov assumptions, the assumption that only a certain number of words in proximity to the word being evaluated need to be taken into account, making the problem tractable and yielding usable models. In effect, the model is trained on a corpus of sequences with a fixed word-window to ascertain which words fit best and then goes through training. But newer models like BERT [Devlin et al., 2019] and GPT-2 [Radford et al., 2019] make use of masking, attention, massive datasets and more to yield extremely good pre-trained LMs.

When such models are overkill, other methods can be used. Using a counting method with several n-grams instead of just a fixed word window, in conjunction with Kneser-Ney smoothing will yield a smaller and faster Language Model [Heafield et al., 2013] that should still work well enough for our purposes.

### 2.4.2 Autoencoders

Seq2Seq models usually consist of two different parts. The first is an encoder, which handles the input sequence $[x_1, x_2, ..., x_n]$ and compresses the input into a latent variable $\boldsymbol{z}$. The second part is a decoder, which receives $\boldsymbol{z}$ as input and then generates an output sequence $[y_1, y_2, ..., y_m]$. A Seq2Seq model can be made to generate a different type of output from the input, e.g. translate an input sequence language into an output sequence in another language or take a question as input and generate an answer. A Seq2Seq model can also be made to try and make its output the same as the input, in which case it becomes a reconstructive model. When a Seq2Seq model becomes a reconstructive model like this, it is called an Autoencoder [Ballard, 1987] [Baldi, 2012]. The benefits of an Autoencoder are not immediately obvious, but it can be a powerful tool in text generation.

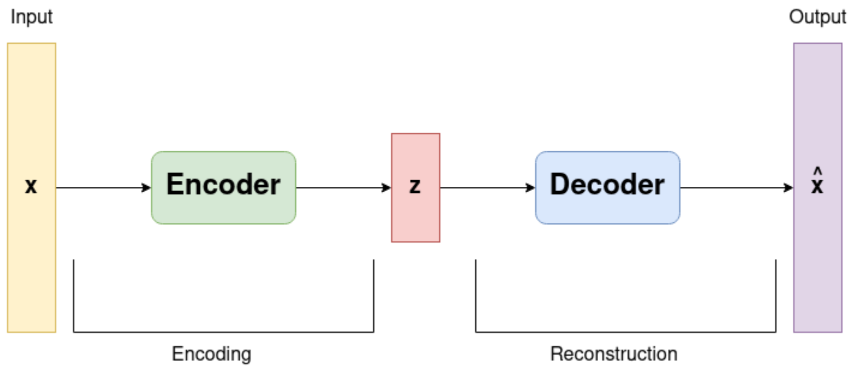


Figure 2.9: Example of Autoencoder reconstructing an input sequence. Ideally, $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ should be the same.

Autoencoders are RNNs that consist of an encoder linked to a decoder. Both the encoder and the decoder are separate RNNs. The output of the encoder, sequences encoded into hidden variables in a latent space, are fed into the decoder, whose responsibility is to decode the hidden variable back into a sequence. Let's say that the input is a sentence $\boldsymbol{x}$ which consists of $n$ number of tokens (e.g. words) $\boldsymbol{x} = x_1, x_2, ..., x_n$. The encoder translates the sentence into a hidden variable $\boldsymbol{z}$ via the use of a learned encoding function $e(\boldsymbol{x}) = \boldsymbol{z}$. The decoder then takes this variable $\boldsymbol{z}$ and decodes it back into a sentence with its learned decoding function $d(\boldsymbol{z}) = \hat{\boldsymbol{x}}$. Thus the goal is to create a reconstruction as close to the original sequence as possible. If each token has an embedding, we can use an L2 Loss to calculate the Reconstruction Loss for a sequence as:

$$L_{Reconstruction} = \sum_{k=1}^{n} ||\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k||_2^2$$

This Reconstruction Loss can then be used to backpropagate error in the model and used to train the Autoencoder to reconstruct sequences. This will give us an encoder that is capable of compressing a sequence into a smaller variable $\boldsymbol{z}$. More importantly, it will give us a decoder (which if given a variable $\boldsymbol{z}$) can generate a sequence. If we give it a $\boldsymbol{z}$ it has not encountered before, it might be able to generate a novel sequence. But there is a problem with this design.

If we generate a random $\boldsymbol{z}$ and decode it, we should be able to generate a sequence that is aligned with the sequences in the corpus we trained on. The problem is that the randomly sampled $\boldsymbol{z}$ has no guarantee that it will correspond to any coherent sequence in the latent space. We have not trained our Autoencoder to produce a continuous latent space, so there may be areas within it with no corresponding sequences. What we would want is to have a continuous latent space so we can sample from anywhere inside it and get a valid sequence.

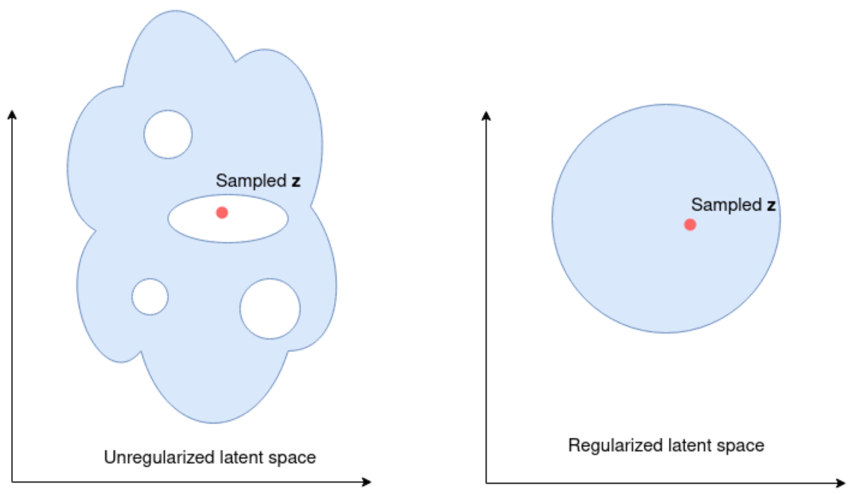


Figure 2.10: A point corresponding to a sampled $\boldsymbol{z}$ in an unregularized latent space with empty spaces in between and a regularized, smooth latent space with no empty spaces. The blue areas symbolize parts of the latent space that contain encoded information, whereas the blank areas contain no encoded information.

What we would like to do is introduce regularization to create a smooth latent space and make sure our generated $\boldsymbol{z}$ corresponds to an actual sequence. So long as we stay within the bounds of the regularized space, we can be certain that our sampled $\boldsymbol{z}$ is correlated to a meaningful sequence.

### 2.4.3 Variational Autoencoders

The Autoencoders discussed previously were deterministic, meaning there is no variability in the model. If you input a sequence $\boldsymbol{x}$ it will always return the same sequence $\hat{\boldsymbol{x}}$ and it has the problem mentioned above of a latent space with possible holes in it. By introducing variation, we can regularize the latent space and fix this issue [Kingma and Welling, 2014a].

For our task of constructing lyrics, we have a dataset of lyric lines $\boldsymbol{x}$ and would like to construct a function which utilizes the dataset to construct new and novel samples $\boldsymbol{z}$ that share similarities with the examples given. In our case, the example lyrics and samples should be liness with certain phonetic structures and semantics. A practical method of constructing this ideal function is to use Variational Inference [Blei et al., 2017] that uses Bayesian probability. We have a function $p(z)$ which is the distribution of the latent space and is called the prior. As mentioned previously, we would like the prior to take a shape where there are no holes in the latent space. A simple function that fits this description well is the Gaussian function $\mathcal{N}(0, 1)$. If our data conforms to this function, we can easily sample from the smooth latent space it provides. The prior is closely related to the distribution $p(x|z)$, that is the likelihood of the sequence $x$ given a latent variable $z$. Modelling the distribution of these two variables with Bayesian statistics, we get:

$$p(z \cap x) = p(z) \cdot p(x|z) \tag{2.8}$$

What we would like to do is calculate the posterior distribution $p(z|x)$, that is, the density of the variable $z$ given the sequence $x$. Re-writing the sequence above, we get:

$$p(z|x) = \frac{p(x \cap z)}{p(x)} \tag{2.9}$$

We would like to calculate the distribution $p(x)$, sometimes referred to as the marginal likelihood or evidence. The marginal likelihood can be re-written as:

$$p(x) = \int p(z)p(x|z)dz \tag{2.10}$$

The problem with this calculation is that it is intractible in all but the most simple of cases. Instead, we must resort to an approximation of the posterior to solve this problem. In order to do this, we can utilize Kullback-Liebler Divergence (KL-Divergence) [Kullback and Leibler, 1951]. KL-Divergence can be described as a score that measures the disparity between two distributions. The lower the score, the more similar the two distributions are. We can now take our selected posterior approximation function and the true posterior distribution and minimize the KL-Divergence between the two to get an approximation. Given two probability distributions $P$ and $Q$ in a shared space $X$, we can write KL-Divergence as:

$$KL(P||Q) = \sum_{x \in X} P(x) \log \left( \frac{P(x)}{Q(x)} \right) \tag{2.11}$$

Or in the case of a continuous random variable:

$$KL(P||Q) = \int_{-\infty}^{\infty} \log \left( \frac{P(x)}{Q(x)} \right) dx \tag{2.12}$$

What we would like to do then is find a function $Q(Z)$ that minimizes the KL-Divergence of the true posterior $P(Z|X)$. We can re-write the KL-Divergence term now as:

$$KL(Q||P) = \sum_{Z} Q(Z) \left[ \log(\frac{Q(Z)}{P(Z|X)}) \right] \tag{2.13}$$

$$KL(Q(Z)||P(Z|X)) = \mathbb{E}_{Q(Z)}[\log(Q(Z))] - \mathbb{E}_{Q(Z)}[\log(P(Z,X))] + \mathbb{E}_{Q(Z)}[\log(P(X))] \tag{2.14}$$

$$<=> \mathbb{E}_{Q(Z)}[\log(P(Z,X))] - \mathbb{E}_{Q(Z)}[\log(Q(Z))] + KL(Q(Z)||P(Z|X)) = \log(P(X)) \tag{2.15}$$

Rewriting the first two terms as the Evidence Lower Bound (ELBO) [ELB, 2020], or $ELBO(Q) = \mathbb{E}_{Q(Z)}[\log(P(Z,X))] - \mathbb{E}_{Q(Z)}[\log(Q(Z))]$, we get:

$$ELBO(Q) + KL(Q(Z)||P(Z|X)) = \log(P(X)) \tag{2.16}$$

$$<=> KL(Q(Z)||P(Z|X)) = \log(P(X)) - ELBO(Q) \tag{2.17}$$

The gap between the evidence and the ELBO is thus the KL-Divergence between $Q(Z)$ and $P(Z|X)$. Maximizing the ELBO is thus equivalent to minimizing the KL-Divergence.

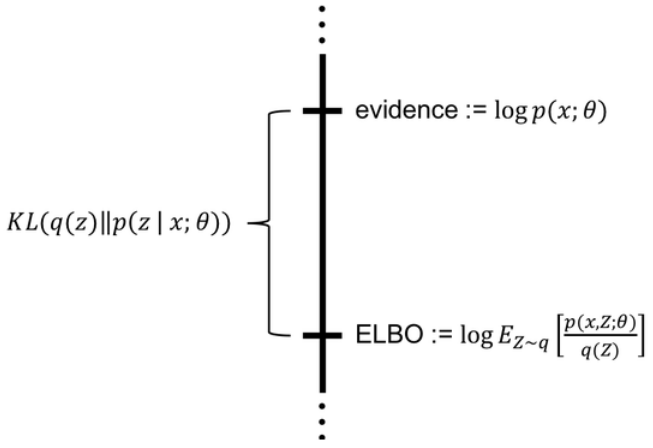

evidence := $\log p(x;\theta)$
$KL(q(z)\|p(z \mid x;\theta))$
ELBO := $\log E_{Z\sim q}\left[\frac{p(x,Z;\theta)}{q(Z)}\right]$

Figure 2.11: The relation between evidence, KL-Divergence and ELBO as shown in [Kullback and Leibler, 1951]

We can rewrite ELBO into a more familiar format:

$$ELBO(Q) = \mathbb{E}_{Q(Z)}[\log(P(X|Z))] - KL(Q(Z)||P(Z)) \tag{2.18}$$

The two distributions can now be substituted with encoder $q_\theta(z|x)$ and our decoder $p_\Theta(x|z)$ [Prokhorov et al., 2019] to acquire a loss function:

$$L_{VAE} = \sum \mathbb{E}_z[\log(p_\Theta(x|z))] - KL(q_\theta(z|x)||p_\theta(z)) \tag{2.19}$$

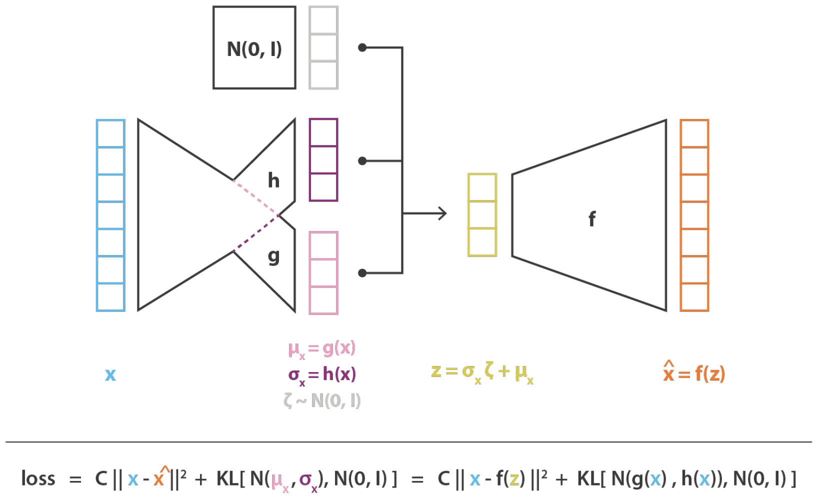
The first term in the loss is effectively a Reconstruction Loss, where the model tries to recreate the input sequence in its output. The second term is a KL Loss that makes our approximate posterior closer to the prior. To control the balance of these two terms, we can place a hyperparameter $\lambda$ on the KL term.

$$L_{VAE} = \sum \mathbb{E}_z[\log(p_\Theta(x|z))] - \lambda KL(q_\theta(z|x)||p_\theta(z)) \tag{2.20}$$

We can then use an annealing strategy by modulating $\lambda$ to avoid KL term collapse in the training of our VAEs [Lucas et al., 2019].

### Reparameterization Trick

One problem with modelling the prior and the posterior as mentioned previously is that we use sampling of a stochastic distribution to obtain a latent variable for our decoder. In a Deterministic Autoencoder, there is no stochastic part of the model and thus there is an unbroken pipeline through the model through which we can perform backpropagation, back through the arrows in the other direction in Figure 2.9. In the case of a VAE, this pipeline is broken, meaning that we cannot perform backpropagation from the decoder to the encoder directly.



N(0, I)
h
g
f
x
$\mu_x = g(x)$
$\sigma_x = h(x)$
$\zeta \sim N(0, I)$
$z = \sigma_x \zeta + \mu_x$
$\hat{x} = f(z)$
$\text{loss} = C \| x - \hat{x} \|^2 + KL[\, N(\mu_x, \sigma_x), N(0, I)\,] = C \| x - f(z) \|^2 + KL[\, N(g(x), h(x)), N(0, I)\,]$

Figure 2.12: Representation of a VAE, also from [VAE, 2019]

The solution to this is to use the Reparameterization Trick [Kingma et al., 2015] to provide us with an unbroken route through the model through which we can perform backpropagation.

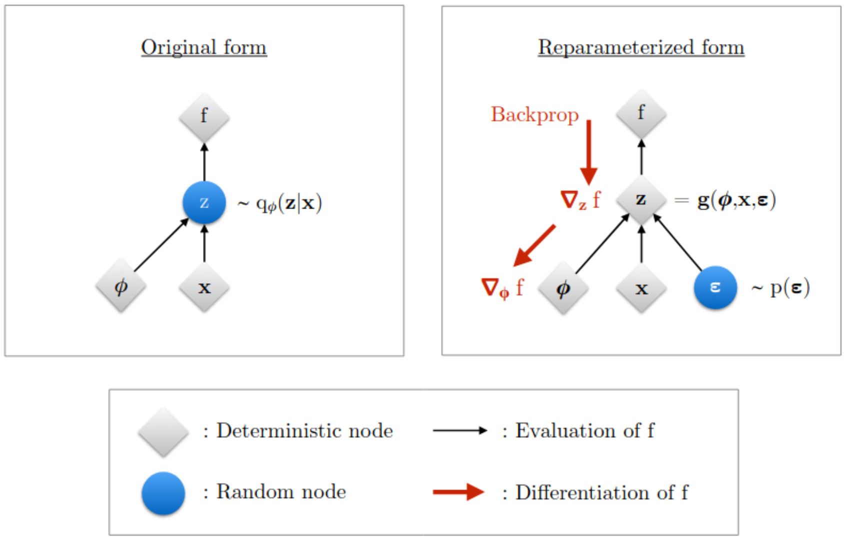


Figure 2.13: Illustration of the reparameterization trick from [Kingma, 2017]

As seen in Figure 2.13, our goal is to create a deterministic node that substitutes for the stochastic one, whose function is given as $g$ in Figure 2.12. The original method directly samples from $\boldsymbol{Z}$, which we can denote as $\boldsymbol{z} \sim \mathcal{N}(\mu, \sigma^2)$. In stead of doing this directly, we can sample $\boldsymbol{\epsilon}$ from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$, perform simple transformations on the encoder output to receive both $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ and combine them into $\boldsymbol{z}$ with the formula:

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \otimes \boldsymbol{\epsilon} \tag{2.21}$$

This satisfies all the conditions mentioned above, giving us a node through which our gradient can pass all the way from the decoder to the encoder and we can properly back-propagate to update the model.

# Chapter 3

# Methodology

When generating lyrics, there is generally a desire to be able to control certain aspects of the output during generation. With many text generation tasks, there is a desire to be able to influence the semantics while adhering to syntax and grammaticality. In the case of lyrics we would like to be able to make the text "musical" or "lyrical" while retaining coherency. The proposed method for doing so uses the hidden state from a Bag-of-Words VAE for controlling the semantic aspect of generated lyrics. A second VAE conditioned on the phonetic structures of lyrics will hopefully provide a hidden state that can be used as a guide to make more musical text, guided by phonetic patterns gleaned from the lyrics. These two hidden states can then be used in tandem to produce the final lyrical structure. We will start by describing the loss functions used to guide the models and help them achieve their objective.

## 3.1 Loss functions

When developing ML models, there are many qualities we want a model to possess. In our case, we have several goals or qualities that we want our model to embody. For example, we want it to neither underfit nor overfit on the data presented. The model should generalize and learn from the data rather than memorize the training examples outright. It should also adhere to the semantic qualities given by one VAE while properly following the phonetic pattern set out by the other VAE. All this while producing output that is grammatical and coherent. The best way to achieve all these goals is to take each requirement and produce a loss function that can guide the model towards its goals. We can then balance out how much weight each of these loss functions carries and thus make it take into account our concerns.

### 3.1.1 Cross-Entropy Reconstruction Loss

One of the classic losses used in many ML models is Cross-Entropy Loss [Shannon, 1948] [Zhang and Sabuncu, 2018]. A short summary of this loss is that it is a general classification loss that can be used to guide any model performing classification. In the case where we have an original sequence $\boldsymbol{x}$ and we are training a reconstructive model to produce $\hat{\boldsymbol{x}}$, we can express the Cross-Entropy Loss of a model thusly:

$$\mathcal{L}_{rec} = -\sum_{c=1}^{K} \boldsymbol{x}_c \log(\hat{\boldsymbol{x}}_c)$$

Where $K$ is the number of classes, or the number of words in the vocabulary in our case. The effect that this has is to generate a loss for classification, where more loss is accrued when the prediction strays further from the class of the ground truth. Here, the term ground truth refers to the sequence the model is attempting to reconstruct.

### 3.1.2 L2 Regularization Loss

When developing ML models, we usually want to make certain that it is not overfitting for the dataset we are training on. There are many methods for addressing this concern and one of the more straight-forward ones is L2 Loss [Ng, 2004]. If our model has a simple Reconstruction Loss $\mathcal{L}_{rec}$ and weights $\boldsymbol{W}$, we can simply add the L2 Loss:

$$\mathcal{L}_{model} = \mathcal{L}_{rec} + \lambda \boldsymbol{W}^2$$

Where $\lambda$ is a hyperparameter controlling how much L2 Loss we wish to apply to the model. As can be seen, the overall idea here is to penalize the model for selecting the exact weights necessary to get a perfect score for the samples it trains on. In essence, we are adding a little bit of noise to the scoring function so that it cannot hone in on the perfect score and hopefully gains a better grasp on generalization instead. It will also make the model use its weights more evenly, possibly preventing it from carving out a section solely focused on memorization of certain samples rather than using all of its weights in a generalizable fashion. However, we must take care to set $\lambda$ appropriately. Too high and the model runs the risk of becoming unstable and underfitting. Too low and we will see overfitting anyway. We will refer to this L2 Regularization Loss as $\mathcal{L}_{L2}$ henceforth.

### 3.1.3 Cycle-Reconstruction Loss

When training a model with multiple components, it can be hard to ensure that the loss functions will adequately guide all components as needed. In our case, there is a concern that our lyrical decoder may not adequately use the information provided by the VAEs and generalize from that information. One way to further encourage the use of information obtained through another part of a model is through Cycle-Reconstruction (CR) Loss [Zhu et al., 2017].

In the case of classic Reconstruction Losses, we pass an input into a model and then compare the output to the input. Given a phonetic sequence $x$ and both the encoding and decoding functions $E_{phon}$ and $D_{phon}$, the Reconstruction Loss would be calculated from the comparisons of $\boldsymbol{x}$ and $D_{phon}(E_{phon}(\boldsymbol{x})) := \hat{\boldsymbol{x}}$. In the case of CR Loss, we take this a step further by comparing the latent variable of the encoding $E_{phon} := \boldsymbol{z}$ and the latent variable of the encoding of the prediction $E_{phon}(\hat{\boldsymbol{x}}) := \hat{\boldsymbol{z}}$. Using $L_2$ distance on these two will provide us the CR Loss:

$$\mathcal{L}_{CR} = ||\boldsymbol{z} - \hat{\boldsymbol{z}}||_2^2$$

As is readily apparent from the formulation here, the CR Loss does act very similarly to the pure Reconstruction Loss. It encourages the model to make more accurate reconstructions and encode pertinent information better. One might assume that simply giving the pure Reconstruction Loss more weight would achieve similar results, but it has been demonstrated in other domains that utilizing both losses at the same time has proven more effective than using either one on its own.

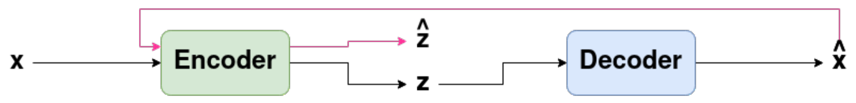


Figure 3.1: Cycle-reconstruction of an input sequence $\mathbf{x}$, denoting the second pass through the model with red arrows

### 3.1.4 Combined Losses for Models

For both the BoW and Phoneme VAEs, a combined set of losses is used in order to train the model. Namely:

$$\mathcal{L}_{Total} = \mathcal{L}_{rec} + \mathcal{L}_{L2} + \lambda\mathcal{L}_{KL}$$

The first two loss functions are the Cross-Entropy and $L2$ losses mentioned above. The thirds is the KL-Divergence used to regularize the latent space. We control its effect throughout the training of the model through adjusting the $\lambda$ hyperparameter. By controlling the value of this lambda, we can perform KL term annealing and avoid KL term collapse [Lucas et al., 2019] during the training of these models, thus ensuring both good reconstruction and a regularized latent space.

In the case of the Lyrics Decoder, a combination of the $L2$ and Reconstruction Loss are used thusly:

$$\mathcal{L}_{Total} = \mathcal{L}_{rec} + \mathcal{L}_{L2}$$

Where it relies on the pre-trained BoW and Phonetic VAEs for input. The Decoder is not variational on its own, so it does not need a KL-Divergence Loss. While it has not been done here, a Cycle-Reconstruction Loss could be added here to make the decoder focus further on utilizing the phonetic information.

## 3.2 Bag-of-Words VAE

The Bag-of-Words strategy here is similar to that mentioned in a previous chapter on Word2Vec. What we would like this model to do is take a line of lyrics as input and then encode the relevant semantic information in that sequence into a latent variable $z$. This latent variable can then be used to convey the semantic information of a line to other models.

First, we need a method of picking out semantically important words in a sequence. A good method for doing so is Term Frequency-Inverse Document Frequency (TF-IDF) [Qaiser and Ali, 2018]. As the name suggests, for each word we calculate both its term frequency and its inverse document frequency and use that to calculate the likelihood of the word being semantically important. Originally, TF-IDF was designed for a corpus of separate documents.

The Inverse Document Frequency (IDF) evaluates the inverse ratio of documents in a set using a given word, intended to gauge how common a word is. Given a corpus of $d$ number of documents $D$ and a word $w$, we can count the number of documents containing the word and then use that to calculate IDF:

$$IDF = \frac{d}{\sum_{i=1}^{d} w \in D_i}$$

The more common the usage of the word, the lower the IDF. Next we calculate Term Frequency (TF), how often a word $w$ occurs in a given document $D$ versus how many words are in the document:

$$TF = \log\left(\frac{\texttt{count}(w \in D)}{\texttt{length}(D)}\right)$$

What we have now is a representation of how unique a word is to a given document (the IDF) and how often that word appears as a proportion of said document (the TF). We can then take the product of these two numbers to get the TF-IDF score:

$$\text{TF-IDF score} = TF \cdot IDF$$

If either term is too low, i.e. the word occurs in all documents or does not get used enough in the documents where it occurs, the score decreases for a word. The higher the score, the likelier it is that it is a semantically important word. The two terms effectively normalize each other and give us a score for the importance of a word within a document. For this approach, excluding stop words is recommended, even though the IDF should mitigate their effect. Instead of using documents here, we can use lines of lyrics and get a score for how important a word is in a line, contained within a corpus of lyrics.

With these scores at our disposal, we can now use them in our BoW VAE. We take each line, pick out the individual words, calculate the TF-IDF score for each and then create a vector where each index corresponds to a word. The model then learns to reconstruct these vectors. Extracting the most semantically important words is then as simple as taking the top-$k$ from the models prediction and translating the indices back into words. Now we can use the hidden state $\boldsymbol{z}_{BoW}$ from this model as previously discussed.

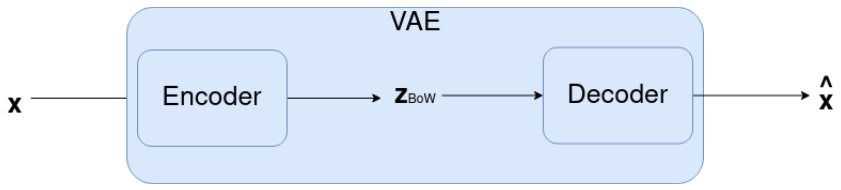


Figure 3.2: Simplified overview of the BoW VAE model

## 3.3 Phonetic VAE

This VAE is similar to the one mentioned above. In this case, we are dealing with embeddings of phonetic tokens in stead of words and not weighting the tokens with anything like TF-IDF. When dealing with standard word embeddings, we can obtain pre-trained word embeddings like Word2Vec or GloVe as mentioned previously. With phonetic embeddings, there are no widely available pre-trained embeddings and so we had to create our own.

We experimented with different phonetic representations. More details on each are provided in section 4.1. The simplest has a vocabulary of 3 different phoneme categories, a slightly more complex vocabulary of 8 different phonetic categorizations and then 2 closely related phonetic vocabularies of actual phonemes rather than vocabularies themselves.

For each of the different vocabularies used, we had separate embeddings trained. PyTorch provides an embedding layer interface that creates a trainable embedding layer for us. This was used during training and each value in the embedding vectors was initialized as a sample from $\mathcal{N}(0,1)$. From there, each line is translated into phonetic tokens. Once we have a sequence of tokens, they are transformed into our embeddings and passed through the VAE as normal. When training is over, we can obtain the latent value $\boldsymbol{z}_{phon}$ from our VAE and use in our decoder.

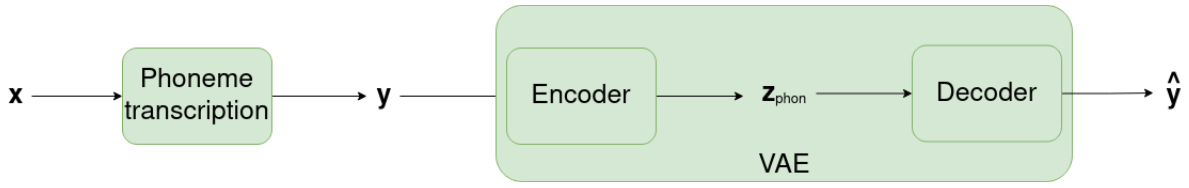


Figure 3.3: Simplified overview of the Phonetic VAE model

## 3.4 Lyrics Decoder

The third model is a stand-alone LSTM decoder. At each time-step, the output from the previous time-step is concatenated together with the 2 hidden states from the other VAEs to create the input. As with most decoders, we start off by using start-of-sentence tokens to begin the decoding process at time step 0. Here we use Word2Vec embeddings as both part of the input and the output when generating the lyrics. Seeing as the output from one time step is the input for another in these circumstances, we could symbolize the input for time-step $i$ as being:

$$\boldsymbol{y}_i = \boldsymbol{z}_{BoW} \oplus \boldsymbol{z}_{phon} \oplus \boldsymbol{y}_{i-1}$$

Where $\oplus$ is vector concatenation. It should be noted that the 2 latent variables are for the entire sequence and remain the same through the decoding process for each sequence. We want the model to focus on using the latent $\boldsymbol{z}$ variables, so before each decoding step, a dropout layer is applied to $\boldsymbol{y}_{i-1}$ to mask some of the information of the previous output. Since the model no longer has all the information from the previous time-step's output, it is forced to use and generalize from the $\boldsymbol{z}$ variables provided.

The standard method of using an Autoencoder entails having a single latent variable $\boldsymbol{z}$ for the entirety of the sequence, which is then decoded auto-regressively. That single variable will then contain both phonetic and semantic information. In our case of using two latent variables, the hope is that the bag-of-words variable will contain enough semantic information and the phonetic variable enough phonetic information to be able to reconstruct the sequence.

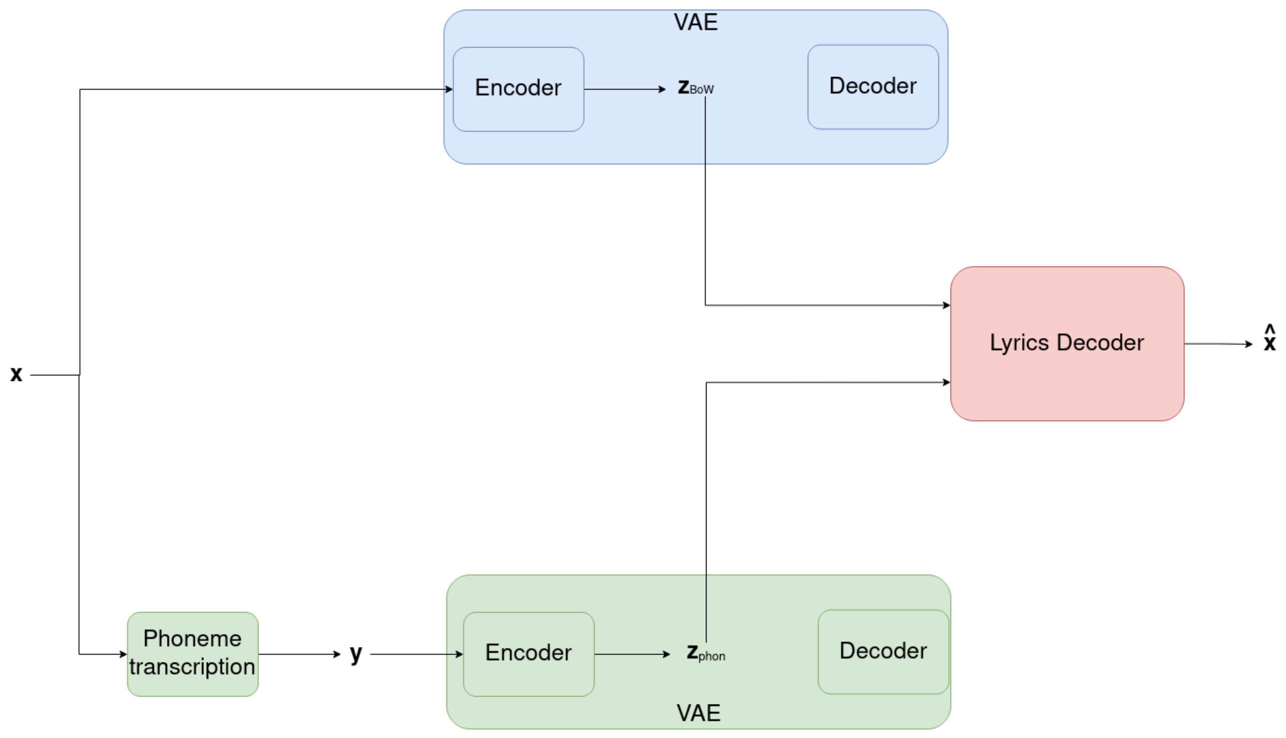


Figure 3.4: Simplified overview of the entire model's pipeline

We also utilize Teacher Forcing [Goyal et al., 2016] to guide the model towards better generation. This is a technique often employed in the training of auto-regressive models. In Teacher Forcing, instead of passing the output of the model $\boldsymbol{y}_{i-1}$ in as the input for the next step, we pass in the corresponding word from the ground truth $\boldsymbol{x}_{i-1}$ so the input would become:

$$\boldsymbol{y}_i = \boldsymbol{z}_{BoW} \oplus \boldsymbol{z}_{phon} \oplus \boldsymbol{x}_{i-1}$$

As the name suggests, this forces the model to learn the original sequences and can help guide the model towards better predictions. The best thing to do here is to perform Teacher Forcing on a percentage of training examples so that the model gains experience both using its own output and the best possible ground truth output.

## 3.5 Model Training Methodologies

Since we are using multiple models when training our decoder, we can use different strategies. One strategy would be that of pre-training our Phonetic VAE and BoW VAE. This would entail training each VAE separately on the same dataset and then freezing the weights of the models. Once frozen, we move on to getting the outputs of the VAEs (without backpropagating the losses) and feeding those results into the Lyrics Decoder to reconstruct the original lyrics. The loss acquired from this step is backpropagated only through the Lyrics Decoder.

If a VAE needs further reinforcement and is not performing well enough at the stage of training the Lyrics Decoder, we can utilize one of the loss functions discussed earlier, namely a Cycle-Reconstruction Loss. In theory, this should lead to better reconstruction of the lyrics. The risk here is that we might destabilize a previously fine-tuned VAE. A Cycle-Reconstruction Loss should encourage better reconstruction, but does not provide any reinforcement or regularization of the latent space, leading to a possible collapse.

Another strategy we tested involves training all three models in tandem in an end-to-end fashion. In stead of pre-training the two VAEs, we start with fresh, untrained VAEs and their loss is backpropagated at the same time as the loss from the Lyrics Decoder. The main difference here is that along with receiving feedback from the BoW and phonetic sequence reconstructions, the VAEs will also receive feedback from the process of decoding information back into lyrics. To go into even more detail, the decoders in the VAEs receive feedback from phonetic and BoW reconstruction, the Lyrics Decoder receives feedback only from the reconstruction of the lyrics and the encoders in the VAEs receive feedback from reconstructing the lyrics, BoW and phonemes.

This extra feedback has the possibility of improving the model's overall performance, but there is no guarantee this will be the case. Another concern is that the VAEs will have to be KL-annealed during the training of the model as a whole. When the VAEs are pretrained, we can focus on each one and fine-tune them without worrying about influences from the outside. In the case of end-to-end training, there are multiple influences on the VAEs and they might even be able to have an effect on each other through the Lyrics Decoder. These side-effects might make the training process and balancing of the KL Loss harder to manage.

# Chapter 4

# Experiments and Results

## 4.1 Datasets

Two main datasets were used during the implementation of our model, one containing lyrics and another containing phonetic transcriptions of various words. The lyrics dataset we had at our disposal consisted of around 34,000 lines of lyrics, mainly from rock and its adjacent genres. We performed a 90%/5%/5% split for train/validation/test, which gave us 30907/1717/1718 samples respectively.

The other dataset was the CMU Pronouncing Dictionary [CMU, 2014], often abbreviated to CMUdict. This is a large dictionary containing phonetic spellings of various words in English. E.g. the word "question" would be transcribed as "K W EH1 S CH AH0 N" and the word "figurehead" is "F IH1 G Y ER0 HH EH2 D", with the numbers on the vowels being 0 for unstressed, 1 for primary stress and 2 for secondary stress within a word. As noted by the dictionary authors, each of these tokens is a phone rather than a phoneme [CMU, 2014], but we will use the term phoneme for the sake of simplification here. Each phonetic token is thus a representation of the pronunciation of a part of a word (or possibly the whole word for very short words).

In our case, several datasets were extrapolated here as well. We created datasets with analogous transcriptions using simpler phoneme categorizations. One set has the phonemes "consonant, monophthong, diphthong" as the only phonetic tokens used, meaning that vowels are categorized into two different tokens depending on whether they are composite or singular vowels and all consonants are grouped under the same token. Another set uses the more complex categories "vowel, stop, affricate, fricative, aspirate, liquid, nasal, semivowel" that are based on where in the mouth a phoneme is formed and its sonic characteristics. Both "monophthong" and "diphthong" from the previous set are now under the same token "vowel", but the consonants now have much more granularity based on where they are produced, rather than all falling under the same token for "consonant". The third set uses the 39 phonemes given in the CMU dict without the numbers indicating stress. This has even more detail than the other two sets combined, since we are now directly using phonetic representations and not overarching categories over multiple tokens. We will go into more detail on how these were used in the Phonetic VAE later in this chapter.

## 4.2 Hyperparameters

### 4.2.1 BoW VAE

For the BoW VAE, a hidden size of 256 and a latent size of 128 was used. A dropout rate of 0.2 was used between each linear layer in the decoder and encoder. The sampling temperature (a measure of the amount of noise added to a sample during encoding) during training was 1.0 and a batch size of 200 was used. Annealing was performed up until iteration 3500, for about 1000 iterations.

### 4.2.2 Phonetic VAE

The Phonetic VAE consisted of a single-layer, bidirectional LSTM encoder and a single-layer, unidirectional LSTM decoder. The embeddings were all trained from scratch and each one had a dimensionality of 60. Both the hidden size and the latent size were of size 100. The L2 regularization factor was 1e-5 and a gradient clipping threshold of 50 was used. A maximum sequence length of 40 was used, note that these are 40 phonemes and not words. The dropout rate was 0.6 and the sampling temperature during training 1.0. A batch size of 200 and annealing was performed up until iteration 2500, with the annealing happening over 1000 iterations.

### 4.2.3 Lyrics Decoder

The Lyrics Decoder consists of a single-layer, unidirectional GRU decoder using 300 dimensional Word2Vec embeddings and a hidden size of 200. The maximum sequence size is of length 15. The L2 regularization factor used was 1e-9, with a gradient clipping threshold of 50. A dropout rate of 0.2 was employed here.

## 4.3 Evaluation and Metrics

### 4.3.1 BLEU Scores

All models used BLEU scores [Papineni et al., 2002] to measure how effective the models are at the reconstruction of unseen examples from the corpus. BLEU scores use so called $n$-gram overlaps to calculate how true a reconstructed sequence is to the original. The metric has proved to be a good substitute in evaluating how good a given sequence is with regards to coherence and grammar. Given an original sequence of tokens $\boldsymbol{x}$ and a reconstructed sequence $\hat{\boldsymbol{x}}$, we split $\hat{\boldsymbol{x}}$ into single words or tokens, the set of all 2 adjacent tokens, the set of all 3 adjacent tokens and all 4 adjacent tokens. A series of $n$ number of adjacent tokens or words like this is called an $n$-gram.

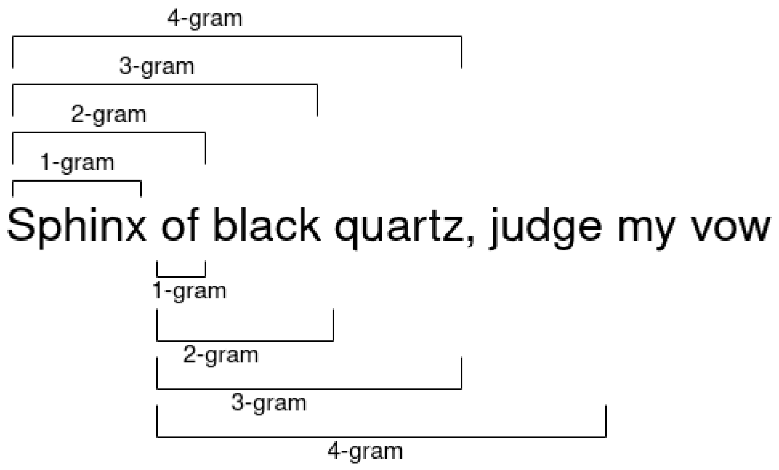


Figure 4.1: An example of different n-grams within a sentence

We then keep track of which of these n-grams occur in $\boldsymbol{x}$. The more of these sequences that occur in the ground truth, the higher the BLEU score. If we have a perfect reconstruction where it is the same as the ground truth, the BLEU score is 1.0. In the case of none of the words in the reconstruction being present in the ground truth, we get the worst possible score of 0. Thus the BLEU score is the ratio of how many of these word sequences or $n$-grams occur in the ground truth. The standard practice of using BLEU scores can be thought of as a double summation, counting the $n$-grams in a ground truth sequence $\boldsymbol{x}$ and the prediction of the model $\hat{\boldsymbol{x}}$:

$$BLEU = \frac{\sum_{n=1}^{4} \sum_{n-\texttt{gram} \in \boldsymbol{x}} \texttt{count}(n - \texttt{gram})}{\sum_{n=1}^{4} \sum_{n-\texttt{gram} \in \hat{\boldsymbol{x}}} \texttt{count}(n - \texttt{gram})}$$

### 4.3.2 Perplexity

Given an LM as detailed in a previous chapter (subsection 2.4.1), we can use it to estimate how well a sequence conforms to the rest of the corpus. Instead of feeding the sequence into the LM directly, we can derive a metric called Perplexity [Clarkson and Robinson, 1999]. The Perplexity (PPL) of a sequence $s$ of length $n$ using our LM can be defined as:

$$PPL(s) = \sqrt[n]{\frac{1}{P(x_1, x_2, ..., x_n)}} = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(x_i | x_1, x_2, ..., x_{i-1})}}$$

A lower PPL score means that a sequence conforms to the corpus the LM was trained on and is a better fit. A higher PPL score means that the sequence is not similar to what we expect and thus likely not a good fit. Another thing that must be kept in mind is that PPL scores are also related to the novelty of a sequence. When a sequence has a higher PPL score, it is not necessarily because it is grammatically incorrect or semantically incoherent, it is simply a sequence that the model does not expect. Thus if an extremely low PPL score is achieved, we might want to make sure that the model is not overfitting and that it does produce somewhat novel and unique sequences.

### 4.3.3 Levenshtein Distance

When comparing two sequences of tokens, it can sometime be useful to use Levenshtein Distance [Levenshtein, 1965]. In essence, Levenshtein Distance measures the minimum amount of tokens that need to be changed in one sequence to be identical to the second, how many single-token edits need to be made on one sequence until the second sequence is obtained. The lower the Levenshtein Distance, the more similar the two sequences are, indicating a better prediction. Levenshtein Distance is a so-called edit distance measure and belongs to a family of such methods. A simple, recursive way of computing the Levenshtein Distance of two sequences is given by:

**Algorithm 1** Calculate Levenshtein Distance between two given sequences $a$ and $b$

```
  procedure LevenshteinDistance(a,b)
Require: $|a| > 0$ and $|b| > 0$
      if $a[0] == b[0]$ then
          LevenshteinDistance(a[1:], b[1:])
      else
          return 1 + min(LevenshteinDistance(a[1:], b)
      end if
      LevenshteinDistance(a, b[1:])
      LevenshteinDistance(a[1:], b[1:]))
  end procedure
```

While BLEU scores have been proven to provide good feedback with word tokenization, it may be prudent to have another similar (but not identical) measure to confirm that the model is on the right track. This is especially true when the vocabulary is not as well studied, such as in the case of the Phonetic VAE. A mismatch of a high BLEU score and a high average Levenshtein Distance of the original and reconstructed sequences could thus be an indicator that the model is not behaving as well as expected.

### 4.3.4 Precision and Recall

When using a classifier, one must pay attention to the rate of correct classifications and incorrect classifications. We can glean even further insight into performance by paying attention to the percentage of true and false positives and negatives. The idea is possibly best understood via a simple graph.

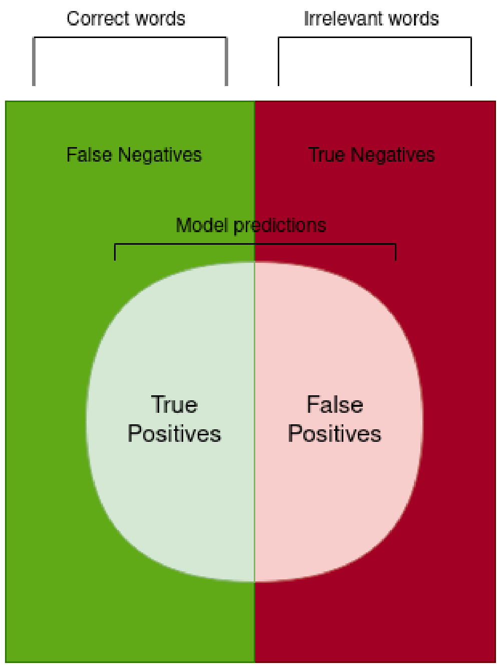


Figure 4.2: Model predictions and their relations to negatives and positives

Informally, Precision will tell us the percentage of predictions made are actually correct. On the other hand, Recall tells us: Of all the possible correct predictions, how many did the model actually get? These can also be expressed as ratios between false positives, true positives and false negatives. Precision is thus defined as the ratio:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

High precision means that when a model makes a prediction, it is usually correct. Recall can be represented in a similar manner:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

High recall means that when a model makes predictions, it is very likely that it will not leave out any correct predictions and mislabel them as false. These two metrics can both be optimized, but usually end up being at odds with each other later during training. Fully optimizing for one will usually lower the other.

### 4.3.5 Distinct Scores

Ideally, we would also like to have a method of judging the diversity of sequences. One way to do so is to use Distinct scoring as proposed in [Li et al., 2016]. For this method, we require a set $\boldsymbol{S}$ which contains $s$ number of sequences. Iterating through all $s$ sequences, we count the total number of n-grams and then count how many distinct n-grams occur. Note that the n-grams do not need to be unique within $\boldsymbol{S}$, only distinct. This means that recurring n-grams will only be counted once, no matter how often they occur. Once the counting is complete, we can calculate the scores as:

$$\text{Distinct-1} = \frac{\text{Number of distinct unigrams}}{\text{Number of total unigrams}}$$

$$\text{Distinct-2} = \frac{\text{Number of distinct bigrams}}{\text{Number of total bigrams}}$$

This is similar to using BLEU scores, but what these metrics are measuring is the overlap of words within a corpus. The higher the Distinct scores, the higher the diversity of the sequences within that corpus, meaning that we would like to aim for higher distinct scores in our generated sequences.

### 4.3.6 Self-BLEU Scores

Finally we have Self-BLEU [Zhu et al., 2018], which is another metric to ascertain the diversity within a set of sequences. We start with a set $\boldsymbol{S}$ of sequences and iterate over them all. Each time, we take sequence $S_i$ and remove it from the set $\boldsymbol{S}$. We then use $S_i$ as the candidate and $\boldsymbol{S} \setminus \{S_i\}$ as the reference. We then do this for all sequences $i$ and take the average of the BLEU scores. A lower Self-BLEU means there is less overlap between sequences in the corpus, indicating higher diversity.

## 4.4 Pre-Training Versus End-to-End Training

As mentioned before, we tried out pre-training the VAEs before training the Lyrics Decoder and also tried training all the models end-to-end at the same time. In short, the challenges mentioned in 3.5 about balancing the different losses in end-to-end training proved hard to overcome. As such, pre-training the VAEs proved a more successful approach, as can be seen in the respective BLEU scores.

| Model | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 |
|---|---|---|---|---|
| End-to-End | 0.4752 | 0.3628 | 0.2924 | 0.2398 |
| Pre-Trained | **0.5802** | **0.4591** | **0.3822** | **0.3250** |

Table 4.1: Comparison validation scores of an end-to-end trained model and one utilizing pre-training

In both methodologies, the VAEs did manage to stabilize and KL-annealing was successful. However, ensuring proper KL-annealing proved harder in the case of end-to-end training and balancing the losses yielded lower BLEU scores. The end-to-end trained decoder was more prone to repetition of words than the pre-trained one and the VAEs yielded lower BLEU scores as well. Since all the models performed worse in the end-to-end scenario, we will focus on the models using the pre-training strategy in the following chapters.

## 4.5 Phonetic VAE

### 4.5.1 Different phonetic vocabularies

As mentioned in section 4.1, we experimented with a few different types of phonetic representations when training the Phonetic VAE. The simplest of the phonetic representations used was a vocabulary of three tokens: monophthongs, diphthongs and consonants. Each line of lyrics used in training was translated into a sequence of these three tokens and then fed to the VAE in order to train it. Several different sets of hyperparameters were experimented with that all yielded similar results. The models obtained using this vocabulary tended to have fairly high BLEU scores and good reconstruction, but the latent space was not very smooth and random sampling yielded poor results. Several methods of consolidating the latent space were tried, but with little success for this simple phonetic vocabulary.

| Model | BLEU-1 ↑ | BLEU-2 ↑ | BLEU-3 ↑ | BLEU-4 ↑ | Avg. Levenshtein distance ↓ |
|---|---|---|---|---|---|
| Smaller vocab | 0.7607 | 0.7180 | 0.6658 | 0.6035 | 5.43 |
| Larger vocab | 0.7006 | 0.6226 | 0.5268 | 0.4238 | 6.52 |

Table 4.2: Reconstruction metrics for smaller vocabulary and larger vocabulary

The scores are slightly better with the smaller vocabulary, but we must also consider the fact that since the vocabulary is smaller, the likelihood of making a correct prediction should be much higher. Thus we would like to see much better scores, which has obviously not happened. With a VAE, not only would we like to ensure good reconstruction, but also a smooth latent space. The latent space for the smaller vocabulary is not unusable, but not as smooth as we had liked, as can be seen from some random sampling. Taking pairs of random samples from the latent space and then linearly interpolating samples in between the two gives us an idea how smooth the latent space is. By interpolation, we mean taking a pair of samples from the latent space, traversing the distance between the two and decoding the intermediary points or samples as we go from one to the other. We effectively split the distance into equal parts and thus expect to see a gradual shift from the first sample to the last, if the latent space has been properly smoothed and regularized. Examples of this can be seen in the appendices in section A.1.

Another concern with this smaller vocabulary is the quality of the individual samples from the latent space itself. Special tokens have been omitted from the samples above to make them more readable, but the model frequently produces samples with tokens after end-of-sentence tokens and uses padding tokens erratically.

| **Random samples from simple vocabulary** |
|---|
| cons di cons di cons mono cons cons di |
| mono cons mono cons cons mono cons cons cons mono cons cons mono cons |
| cons mono cons mono cons cons di cons mono cons mono cons di cons |
| cons di cons mono cons cons mono cons |
| cons mono cons |
| cons mono cons cons di cons mono cons |
| cons mono cons di cons di cons mono cons |
| cons mono cons mono cons cons di cons cons mono cons cons |
| cons di cons mono cons mono cons mono cons di cons mono cons cons mono cons di cons mono cons di cons di cons |
| di mono |

Table 4.3: Random samples from latent space of smaller vocabulary Phonetic VAE

Using the more complex vocabulary with the phonetic categorization proved a better choice for phonetic representations. Here there were 8 different tokens used and the results were better. BLEU scores were high, and thus reconstruction was good, as well as yielding a more smooth latent space. The slightly larger vocabulary size seems to have given the model more variance and given it more information from which to extract and learn patterns. This can be seen in better BLEU scores and lower Levenshtein distance in our latter model. Applying the same interpolation strategy from before, we can obtain several samples to test the smoothness of the latent space. Examples of this can be found in the appendices in section A.2

Taking random samples from the latent space also shows us that the samples themselves tend to be less problematic. They are not completely free from interspersing tokens when the line should have ended, but it happens less and the model is more consistent in using only end-of-sentence tokens.

| **Random samples from larger vocabulary** |
|---|
| stop vowel vowel stop fricative vowel |
| stop vowel fricative vowel fricative vowel stop vowel fricative stop stop vowel stop vowel |
| vowel stop vowel stop vowel nasal stop vowel fricative stop |
| vowel fricative nasal vowel liquid stop nasal vowel nasal fricative stop vowel |
| fricative vowel liquid liquid vowel nasal vowel stop vowel liquid semivowel vowel liquid vowel nasal stop vowel nasal vowel liquid |
| fricative vowel nasal vowel nasal vowel nasal |
| vowel nasal vowel fricative stop vowel nasal vowel fricative stop vowel nasal |
| vowel fricative vowel liquid vowel nasal fricative vowel vowel nasal vowel liquid vowel nasal vowel |
| vowel stop liquid vowel stop liquid vowel nasal vowel stop vowel nasal vowel semivowel vowel |
| stop vowel nasal stop vowel nasal stop vowel nasal stop vowel nasal vowel nasal stop nasal vowel nasal vowel nasal stop |

Table 4.4: Random samples from latent space of larger vocabulary Phonetic VAE

Reconstruction using this expanded vocabulary is also better. Several examples of lines, their phonetic transcription and reconstructions have been provided below.

| **Reconstructions using larger vocabulary** |
|---|
| Original line: for my senses<br>Phonetic transcription: fricative vowel liquid nasal vowel fricative vowel nasal fricative vowel fricative<br>Reconstruction: fricative vowel liquid nasal vowel fricative vowel nasal fricative vowel fricative |
| Original line: no more free steps to heaven<br>Phonetic transcription: nasal vowel nasal vowel liquid fricative liquid vowel fricative stop vowel stop fricative stop vowel aspirate vowel fricative vowel nasal<br>Reconstruction: nasal vowel nasal vowel liquid fricative vowel vowel fricative stop vowel stop fricative vowel vowel fricative vowel fricative vowel nasal |
| Original line: that part of me isn't here anymore<br>Phonetic transcription: fricative vowel stop stop vowel liquid stop vowel fricative nasal vowel <unk> stop vowel aspirate vowel liquid vowel nasal vowel nasal vowel liquid<br>Reconstruction: fricative vowel stop stop vowel fricative stop fricative fricative liquid vowel liquid vowel vowel liquid vowel liquid vowel nasal vowel nasal vowel |
| Original line: that s the shape i'm in<br>Phonetic transcription: fricative vowel stop vowel fricative fricative vowel fricative vowel stop vowel vowel nasal vowel nasal<br>Reconstruction: fricative vowel stop vowel fricative vowel vowel vowel vowel stop vowel vowel nasal fricative nasal |
| Original line: she made me feel alive<br>Phonetic transcription: fricative vowel nasal vowel stop nasal vowel fricative vowel liquid vowel liquid vowel fricative<br>Reconstruction: fricative vowel nasal vowel stop nasal vowel fricative vowel liquid vowel fricative fricative fricative |

Table 4.5: Reconstructions of phonetic sequences using large vocabulary

The third dataset, using the CMU dictionary without the stresses, was not experimented with extensively. The main problem with this dataset is that even though we are utilizing phonetic representations, they are legible transcriptions of the words they are supposed to represent. If we take an example from earlier, the word "question" would be transliterated as "K W EH S CH AH N". It is readily apparent that this is the word "question" and can only be the word "question". Using our previous vocabulary, the word would be transliterated as "stop semivowel vowel fricative affricate vowel nasal". It is much harder to figure out what the original word is and there may be multiple words that fit that pattern. The former vocabulary does not obfuscate any of the semantic meanings of words, while the second vocabulary does. Thus if we were to train a model using this modified CMU dictionary, it would be very similar to training a VAE using byte-pair encodings [Gage, 1994] of words and reconstructing lines with word segments in stead of words. Since our goal here is to train a model to extract phonetic patterns and ignore semantics, this is not a desirable approach.

### 4.5.2 Hyperparameter experimentation

Taking the above into account, we limited more extensive experimentation to the simple representations and the phonetic category representations. Between each step of decoding, a dropout layer was used to mask some of the embedding information, in the hopes of making the model rely more on the latent sequence variable than the embeddings of the phonetic information. A dropout rate of 0.2 has usually proved enough to encourage the model to encode information in the latent space rather than the embeddings, but rates from 0.2 to 0.9 were tested here. The rate did not have much effect on the reconstruction ability of the model, but a significantly higher rate was needed to make the model to properly utilize the latent space. In the end, a rate of 0.6 was settled on for the dropout layer.

As is the case with many RNN-based models, it is much harder for the model to learn from longer sequences. A maximum sequence length of 20 seems to yield models better at reconstruction, but a maximum sequence length of 20 phonemes would yield exceedingly short lyrics. A sequence size of 40 proved a more suitable balance between reconstruction and length of actual lyrics.

Another aspect that required experimentation was the size of the latent variable and the size of the trainable embeddings. The standard dimensionality for pre-trained word embeddings is 300, but we had much smaller vocabularies than in those cases. Much smaller vocabularies should mean that less information needs to be packed into each embedding, so we experimented with smaller embedding sizes from 80 to 20, before finally deciding on a dimensionality of 40. On the upper end of the scale, a size of 80 was excessive and made the model more prone to memorizing sequences and ignoring the latent variable, even with higher levels of dropout. When given a dimensionality of 20, the model did not manage to generalize and performed worse in both reconstruction and smoothing out its latent space. A size of 40 provided a good compromise with accurate reconstruction without issues of memorization.

## 4.6 Bag-of-Words VAE

Our BoW VAE performed well in extracting semantically meaningful words from the given lyrics. For each line, we extracted the most semantically important words using TF-IDF scoring and used them as the ground truth. Our model then made a prediction of the 6 most important words in the line and we evaluated based off of that output. For the BLEU score, we focused on BLEU-1 and omitted longer n-grams, since the order of words does not matter in a BoW. We also evaluated the precision and recall with respect to the ground truth. A high precision in this case would mean that there are fewer incorrect predictions of the words found in the ground truth. In other words, the number of false positives becomes lower as precision increases. While a higher precision is good, we are more interested optimizing for higher recall in this instance. A higher recall would mean that the model is managing to predict more of the relevant words, leaving fewer of the important ones out. These metrics should give us a good idea as to how well the model is performing at parsing meaning from each sequence.

| BLEU-1 $\uparrow$ | Precision $\uparrow$ | Recall $\uparrow$ |
|---|---|---|
| 0.4426 | 0.4525 | 0.7941 |

Table 4.6: Scores from reconstruction in BoW VAE

From these scores we can tell that the model is not particularly accurate when predicting important words, but it is quite good at making sure that most of the important words are present in predictions.

| |
|---|
| Original line: that you could sink so low<br>Bag of words: ['could', 'low', 'sink']<br>Reconstruction: ["n't", 'could', 'light', 'low', 'sink', 'all-time'] |
| Original line: it s a drive in saturday<br>Bag of words: ["'s", 'saturday', 'drive-in']<br>Reconstruction ['yeah', "'s", 'saturday', 'judge', 'thorns', 'drive-in'] |
| Original line: and find a distant man a-waving his spoon ?<br>Bag of words: ['man', 'find', 'distant', 'spoon', 'a-waving']<br>Reconstruction: ['pretend', 'find', 'distant', 'absolution', 'spoon', 'a-waving'] |
| Original line: i don't run around with no mob<br>Bag of words: ["n't", 'around', 'run', 'mob']<br>Reconstruction: ["n't", 'wo', 'around', 'run', 'ai', 'mob'] |
| Original line: no more free steps to heaven<br>Bag of words: ['free', 'heaven', 'steps']<br>Reconstruction: ['free', 'heaven', 'key', 'mother', 'beckons', 'steps'] |

Table 4.7: Reconstruction examples from BoW VAE

Performing the same tests for the latent space in the BoW VAE as in the Phonetic VAE, we can see that the interpolations between BoW samples have smooth transitions from one to the next. Random samples from the latent space also show that the VAE produces varied sets of words.

| **Interpolation 1** | **Interpolation 2** |
|---|---|
| bad darlin end goodbye things done | money sun door world around one |
| bad darlin end goodbye things done | money sun door world around one |
| done end let fast darlin take | world door money around sun one |
| right take done let n't see | world door around love money sun |
| see right take n't sun oh | world back door 's around love |
| see sun right oh n't take | back world know 's door love |
| sun see oh right town come | back know world time 's going |
| sun see town oh lost got | back time know going 's better |
| sun see town lost mindless shining | back time know better going home |
| sun see town lost mindless shining | back time know better going home |
| **Interpolation 3** | **Interpolation 4** |
| 's try set get waiting know | right 's picture got life till |
| 's try set get waiting know | right 's picture got life till |
| 's know get right try set | right picture got life 's ca |
| know get right 's night see | right got picture life ca want |
| know get right night see dance | right want got life n't ca |
| know night get right see got | right want got see n't ca |
| got night know get see baby | want right see got moving broken |
| got getting old night things sky | want fear see turns broken moving |
| got getting old things ears sky | fear health turns kept race charade |
| got getting old things ears sky | fear health turns kept race charade |
| **Interpolation 5** | **Interpolation 6** |
| confusion wear humanity go chain thorns | every minute move night little love |
| confusion wear humanity go chain thorns | every minute move night little love |
| go confusion wear dissident chain humanity | every night know love ... little |
| go take dissident ca touch turn | night every ... n't know oh |
| take go oh n't ca way | night ... n't oh want give |
| take oh n't ca go see | night ... got give 's n't |
| take oh n't 're see water | night got 's young give ... |
| take oh n't 're water time | got young night 's dark give |
| take oh time 're girl water | young got dark 's night tower |
| take oh time 're girl water | young got dark 's night tower |

| **Interpolation 7** | **Interpolation 8** |
|---|---|
| n’t tenderly sorrow best hurricane faith | world new free used blue endless |
| n’t tenderly sorrow best hurricane faith | world new free used blue endless |
| n’t feel best sorrow tenderly leave | world ’m free new used night |
| n’t feel get line right best | world ’m free night see new |
| n’t love feel get line right | world ’m never free night give |
| love feel n’t way line get | ’m world never ... give oh |
| love way feel line life go | ’m never world ... back oh |
| love way life line new everywhere | never ’m back world ... walk |
| love way new everywhere cross life | back never learning ’m walk huh |
| love way new everywhere cross life | back never learning ’m walk huh |
| **Interpolation 9** | **Interpolation 10** |
| baby free way tell little ’ll | better shoulder song time man singing |
| baby free way tell little ’ll | better shoulder song time man singing |
| baby way tell free ’ll little | man better time running world alone |
| baby way ’ll tell n’t girl | man world time running got get |
| n’t baby ’ll girl take way | man world got time know running |
| n’t girl take make ’ll said | man got world see could know |
| girl n’t take make understand said | got man world change see could |
| girl understand brain thought take n’t | change got hope man see still |
| understand brain unknown thought girl fame | change hope lose still got turning |
| understand brain unknown thought girl fame | change hope lose still got turning |

Table 4.8: Interpolations of latent space of BoW VAE

| **Random samples** | |
|---|---|
| 'll love stumble let around footsteps | ride highway demand little things might |
| never n't head tried thing mean | head 'll n't let please ignite |
| blues hold happening way broken reverse | lonely day everything one alright dark |
| children please best '87 mind cry | 's wish goodbye heaven lonely life |
| love want take 's road say | get rise woman keep could mess |
| like n't guess sweet arrow see | children buried blur bones roll begins |
| go skin slow 's peel shoes | good like tie last opened take |
| see baby way waiting loving means | see 's road walk unread ports |
| see life try 's want something | 're like hard ... hit next |
| day ... 'll hive need names | find difference universe heartache focus path |
| heart get big dancing street 'll | say n't something coming real harm |
| years n't long forest wasted trouble | hair beach fear ... hand dress |
| 's fun memory got back well | tears 'm away cry hate friend |
| knew 're know drums heard 's | gone spent somebody side see long |
| see wave gave ghost ride sure | time away ... ca flame around |
| gray young port swallow tied bell | n't 'm care could one believe |
| little 's five dark ragged believe | help 're people n't away truth |
| yeah place oh right one insignificance | 's go kind man know get |
| comes sleep sun shine 'm get | n't higher ca close 'm read |
| tell surrender see quest n't serve | magic frightened white really money ching-a-ling |
| time found 're let tv night | long know years brings hill crawl |
| gloves tied love shining door silenced | never hearts pierce way lovers man |
| girl got eye ta little pretty | feel walls sin ... fell words |
| take 's 'm na train gives | get back look 'm 'll wait |
| see ... 're falling feel live | n't want 're bag ca paper |

Table 4.9: Random samples from latent space of BoW VAE

The model clearly has a strong affinity towards certain words, which is good in this case. Songs often follow similar ideas and patterns, often leading to similar themes and word choices. Emotions feature heavily in the samples and objects or conditions that are idiomatic or evocative in their own right appear frequently. As can be seen, a majority of the samples also have a notable sentiment, mood or theme. E.g. "'ll love stumble let around footsteps" suggests someone stumbling around, hardly able to walk straight because of love. Another could be "lonely day everything one alright dark" suggesting feeling like everything has gone dark on a lonely day, a much more somber mood than the previous set of words.

## 4.7 Lyrics Decoder

As a baseline, the lyrics decoder was first set up as a Deterministic Autoencoder to see how well it could reconstruct lyrics from a single latent variable $\boldsymbol{z}$ without any variational components or disentanglement. Once that was complete, a variational encoder was added to create a VAE that could reconstruct lyrics. In addition to being able to reconstruct, this VAE can also directly generate lyrics by taking random samples from its latent space and decoding them. These models can be used as a comparison to our Lyrics Decoder with regards to how well it can reconstruct the original lyrics. Our Lyrics Decoder differs from the DAE and the VAE in that it does not have a single variable $\boldsymbol{z}$ distilled from the information within a sequence, but rather receives a concatenation of the two latent variables from our Phonetic VAE and BoW VAE, $\boldsymbol{z}_{BoW}$ and $\boldsymbol{z}_{phonetic}$.

| Model | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | Avg. BLEU | Avg. PPL ↓ | Median PPL ↓ |
|---|---|---|---|---|---|---|---|
| Ground Truth | 1.0000 | 1.0000 | 1.0000 | 1.000 | 1.000 | 235.85 | 46.12 |
| DAE | 0.6272 | 0.5171 | 0.4385 | 0.3755 | 0.4896 | 424.64 | 100.87 |
| VAE | 0.5710 | 0.4344 | 0.3431 | 0.2732 | 0.4054 | 439.26 | 110.245 |
| Decoder | 0.5490 | 0.4103 | 0.3206 | 0.2549 | 0.3837 | 443.44 | 113.66 |

Table 4.10: Reconstruction scores from comparable decoder models

As can be seen from the scores in Table 4.10, adding a variational component lowers the ability of the model to reconstruct somewhat compared to a DAE. This is to be expected, since a DAE only focuses on reconstruction while a VAE has the added task of regularizing its latent space [Bahuleyan et al., 2018]. The Lyrics Decoder that utilizes the BoW and phoneme latent variables to reconstruct performs on par with a classic VAE. The VAE has a latent size of 200 and the latent size of the Lyrics Decoder is comparable at 228, the combined sizes from the BoW VAE and the Phonetic VAE. The VAE receives the entire line as an input while our Lyrics Decoder only has the BoW and phonetic information, suggesting that with this reduced amount of information, reconstruction is still feasible in the case of lyrics. The averages of the PPL scores are higher than one might expect, mainly due to a few outliers with very high scores. Since this is the case, the median provides a more accurate idea as to how the model is faring. Examples of reconstructions of lines from each model can be seen in Table 4.11. The DAE and VAE predictions use latent variables of each line, while the Lyrics Decoder predictions only rely on the BoW and phonetic representations of each line.

| |
|---|
| Ground truth: for my senses<br>DAE prediction: for my home<br>VAE prediction: for my senses<br>Decoder prediction: for the senses |
| Ground truth: steals away in the night<br>DAE prediction: carried away every night<br>VAE prediction: away stopped in the night<br>Decoder prediction: that the last night |
| Ground truth: love will be the death of<br>DAE prediction: love will be the death of<br>VAE prediction: love is the flames of divide<br>Decoder prediction: love will be the death of |
| Ground truth: that part of me isn't here anymore<br>DAE prediction: that this game isn't free anymore here<br>VAE prediction: that's something else here to stay here<br>Decoder prediction: that don't part me anymore me anymore |
| Ground truth: show your love show your sweet sweet love<br>DAE prediction: love your show love your sweet love<br>VAE prediction: bring your love you dance with love sweet<br>Decoder prediction: so sweet your love you sweet love |
| Ground truth: she makes it real<br>DAE prediction: she makes it real<br>VAE prediction: he makes it real<br>Decoder prediction: so real makes the real |

Table 4.11: Examples of reconstructions from comparative models

### 4.7.1 Information Retention in Mixing Phonetic and BoW Samples

One way to examine how much phonetic information is retained in the latent variable from the Phoneme VAE is to try taking phonetic information from one line, BoW information from another. We can concatenate the two, decode the resulting concatenation and then observe how much the phonetic information changes between the line with the original phonetic information and the one just produced. This change should be most apparent when pairing together lines of differing length. We start off by taking lines with 8 words and lines with 14 words from our test set and perform these types of combinations.

|  | Avg. Levenshtein | Avg. line words | BoW BLEU-1 | BoW Recall |
|---|---|---|---|---|
| Shorter lines | 6.67 | 6.86 | 0.3905 | 0.7103 |
| Longer lines | 11.62 | 11.81 | 0.5310 | 0.6064 |
| Short BoW/long phonemes | 15.57 | 10.90 | 0.3143 | 0.5714 |
| Long BoW/short phonemes | 10.05 | 7.71 | 0.2883 | 0.3239 |

Table 4.12: Metrics from combining lines with 8 words together with lines with 14 words and respective reconstruction scores with 21 line pairs

First line: let the truth of love be lighted
Second line: a cop knelt and kissed the feet of a priest and a queer
First reconstruction: let love you've to love to be
Second reconstruction: it can be just the one that's cop and
BoW from first, phonemes from second: i don't let to the truth that that we love and we love
Phonemes from first, BoW from second: it can be just the one that's cop and

First line phonemes: liquid vowel stop fricative vowel stop liquid vowel fricative vowel fricative liquid vowel fricative stop vowel liquid vowel stop vowel stop
Second line phonemes: vowel stop vowel stop nasal vowel liquid stop vowel nasal stop stop vowel fricative stop fricative vowel fricative vowel stop vowel fricative vowel stop liquid vowel fricative stop vowel nasal stop vowel stop semivowel vowel liquid
First combo phonemes: vowel stop vowel nasal stop liquid vowel stop stop vowel fricative vowel stop liquid vowel fricative fricative vowel stop fricative vowel stop semivowel vowel liquid vowel fricative vowel nasal stop semivowel vowel liquid vowel fricative
Second combo phonemes: vowel stop stop vowel nasal stop vowel affricate vowel fricative stop fricative vowel vowel nasal vowel fricative vowel stop fricative stop vowel stop vowel nasal stop

First line: one is to love your neighbor till
Second line: i'd sure like to see her when the sun goes down
First reconstruction: one you're already one light
Second reconstruction: i'll see what you're sure goes like the sun goes
BoW from first, phonemes from second: i love her already you already love you already one
Phonemes from first, BoW from second: when i will see you like a little wonder

First line phonemes: semivowel vowel nasal vowel fricative stop vowel liquid vowel fricative semivowel vowel liquid nasal vowel stop vowel liquid stop vowel liquid
Second line phonemes: vowel stop fricative semivowel vowel liquid liquid vowel stop stop vowel fricative vowel aspirate vowel liquid semivowel vowel liquid fricative vowel fricative vowel nasal stop vowel fricative stop vowel nasal
First combo phonemes: vowel liquid vowel fricative aspirate vowel liquid vowel liquid liquid vowel stop vowel semivowel vowel vowel liquid liquid vowel stop vowel liquid vowel fricative semivowel vowel vowel liquid liquid vowel stop vowel semivowel vowel nasal
Second combo phonemes: semivowel vowel nasal vowel semivowel vowel liquid fricative vowel semivowel vowel liquid vowel stop vowel liquid vowel stop liquid semivowel vowel nasal stop vowell liquid

First line: don't want to wait for heaven
Second line: i look to the stars as they flicker and float in your eyes
First reconstruction: don't want to wait with heaven
Second reconstruction: all the stars that the stars that the same for me
BoW from first, phonemes from second: i look i just for the heaven i want to this heaven
Phonemes from first, BoW from second: can't you look into your eyes

First line phonemes: stop vowel nasal stop semivowel vowel nasal stop stop vowel semivowel vowel stop fricative vowel liquid aspirate vowel fricative vowel nasal
Second line phonemes: vowel liquid vowel stop stop vowel fricative vowel fricative stop vowel liquid fricative vowel fricative fricative vowel fricative liquid vowel stop vowel liquid vowel nasal stop fricative liquid vowel stop vowel nasal semivowel vowel liquid vowel semivowel fricative
First combo phonemes: vowel liquid vowel stop vowel affricate vowel fricative stop fricative vowel liquid fricative vowel aspirate vowel fricative vowel nasal vowel semivowel vowel nasal stop stop vowel fricative vowel fricative aspirate vowel fricative vowel nasal
Second combo phonemes: stop vowel nasal stop semivowel vowel liquid vowel stop vowel nasal stop vowel semivowel vowel liquid vowel fricative

| |
|---|
| First line: i'm the stain in your bed |
| Second line: i'm stuck in this dream it's changing me i am becoming |
| First reconstruction: i'm the stain in your bed |
| Second reconstruction: i'm becoming the same i can not a same my only i am |
| BoW from first, phonemes from second: i'm the stain of bed and a stain in my bed |
| Phonemes from first, BoW from second: i'm the living i'm been i |
| First line phonemes: vowel nasal fricative vowel fricative stop vowel nasal vowel nasal semivowel vowel liquid stop vowel stop |
| Second line phonemes: vowel nasal fricative stop vowel stop vowel nasal fricative vowel fricative stop liquid vowel nasal vowel stop fricative affricate vowel aspirate vowel nasal nasal vowel vowel vowel nasal stop stop vowel nasal vowel nasal |
| First combo phonemes: vowel nasal fricative vowel fricative stop vowel nasal vowel fricative stop vowel stop vowel nasal stop vowel fricative stop vowel nasal vowel nasal nasal vowel stop vowel stop vowel stop |
| Second combo phonemes: vowel nasal fricative vowel liquid vowel fricative vowel nasal vowel nasal stop vowel nasal vowel |
| First line: standing by the wall by the wall |
| Second line: feeling like there's something i never thought you'd be part of |
| First reconstruction: that's blessed to have the wall |
| Second reconstruction: there are like a part of thought i never can never go |
| BoW from first, phonemes from second: the wall of the wall and the wall that's standing |
| Phonemes from first, BoW from second: never can never be a part |
| First line phonemes: fricative stop vowel nasal stop vowel nasal stop vowel fricative vowel semivowel vowel nasal stop vowel fricative vowel semivowel vowel liquid |
| Second line phonemes: fricative vowel liquid vowel nasal liquid vowel stop fricative vowel liquid vowel fricative fricative vowel nasal fricative vowel nasal vowel nasal vowel fricative liquid fricative vowel stop semivowel vowel stop stop vowel stop vowel liquid stop vowel fricative |
| First combo phonemes: fricative vowel semivowel vowel fricative fricative vowel semivowel vowel liquid vowel nasal stop fricative vowel semivowel vowel liquid semivowel vowel stop fricative fricative stop vowel nasal stop vowel nasal |
| Second combo phonemes: vowel fricative liquid stop vowel nasal nasal vowel fricative liquid stop vowel vowel stop vowel liquid stop |

Table 4.13: Examples of combinations of 8 word lines and 14 word lines

We start of by reconstructing the shorter lines and the longer lines. The phonetic patterns for the reconstructions had average Levenshtein distances of 6.67 and 11.62 respectively while the reconstructed BoWs had BLEU-1 scores of 0.3903 and 0.5310. Then the BoW from a short line and the phonetic information from a long line are concatenated and decoded to obtain a resulting line. With the resulting line, we can now compare the resulting phonetic sequence with the phonetic sequence from the original longer line.

If the Levenshtein distance from the original reconstruction is similar to that of our new line, that would indicate that the phonetic pattern is being retained. Another metric that will provide us with some basic feedback is the length of the original longer line. Similarly, we can compare the resulting BoW with the BoW from the shorter line and see how well the semantic information was retained. The closer the BLEU-1 score of the original is to the BLEU-1 of the generated line, the better the retention of the semantic information. The inverse is true when a BoW from a long line has been added to the phonemes of a short line. Then we would compare the BoW with that of the long line and the phonemes from the short line to get an idea how well information was retained.

Judging by the comparison of the Levenshtein distances and the average number of words in the generated lines as opposed to the reconstructions, we can see that the model is guided somewhat by the phonetic structure provided. The average number of words increases with longer phonetic sequences and decreases with shorter phonetic sequences. While the Levenshtein distance of the phonetic sequences does increase, the loss in accuracy is not too bad.

| | Avg. Levenshtein | Avg. line words | BoW BLEU-1 | BoW Recall |
|---|---|---|---|---|
| Shorter lines | 3.85 | 5.03 | 0.3485 | 0.7879 |
| Longer lines | 9.55 | 9.06 | 0.4740 | 0.6008 |
| Short BoW/long phonemes | 13.00 | 8.88 | 0.3061 | 0.7010 |
| Long BoW/short phonemes | 6.71 | 5.38 | 0.2592 | 0.3437 |

Table 4.14: Another set of metrics from combining lines with 6 words together with lines with 11 words and respective reconstruction scores with 66 line pairs

First line: oh i believe in miracles
Second line: i just pay while you're breaking all the rules
First reconstruction: oh i believe in miracles
Second reconstruction: i'll be just a well who will be where the rules
BoW from first, phonemes from second: oh what was all you believe for me believe
Phonemes from first, BoW from second: i'll be a rules in here

First line phonemes: vowel vowel stop vowel liquid vowel fricative vowel nasal nasal vowel liquid vowel stop liquid fricative
Second line phonemes: vowel affricate vowel fricative stop stop vowel semivowel vowel liquid semivowel liquid stop liquid vowel stop vowel nasal vowel liquid fricative vowel liquid vowel liquid fricative
First combo phonemes: vowel semivowel vowel stop semivowel vowel fricative vowel liquid semivowel vowel stop vowel liquid vowel fricative fricative vowel nasal vowel stop vowel liquid vowel fricative
Second combo phonemes: vowel liquid stop vowel vowel liquid vowel liquid fricative vowel nasal aspirate vowel liquid

First line: i'm not looking back
Second line: no short haired yellow bellied son of georgy porgy is
First reconstruction: i'm looking back at
Second reconstruction: from the haired yellow bellied is son of georgy porgy is
BoW from first, phonemes from second: then you're looking for you back for what i'm looking for
Phonemes from first, BoW from second: i am son to yellow

First line phonemes: vowel nasal nasal vowel stop liquid vowel stop vowel nasal stop vowel stop
Second line phonemes: nasal vowel fricative vowel liquid stop aspirate vowel liquid stop semivowel vowel liquid vowel stop vowel liquid vowel stop fricative vowel nasal vowel fricative affricate vowel liquid affricate vowel stop vowel liquid stop vowel vowel fricative
First combo phonemes: fricative vowel nasal semivowel vowel liquid liquid vowel stop vowel nasal fricative vowel liquid semivowel vowel stop vowel stop fricative vowel liquid semivowel vowel stop vowem liquid liquid vowel nasal fricative vowel liquid
Second combo phonemes: vowel vowel nasal fricative vowel nasal stop vowel semivowel vowel liquid vowel

First line: and you the master dealer
Second line: ghosts crowd the young child's fragile eggshell mind
First reconstruction: and we're master master
Second reconstruction: ghosts crowd they fragile mind's fragile eggshell mind
BoW from first, phonemes from second: but there's just master for master's master master
Phonemes from first, BoW from second: and fragile's a fragile eggshell

First line phonemes: vowel nasal stop semivowel vowel fricative vowel nasal vowel fricative stop vowel liquid stop vowel liquid vowel liquid
Second line phonemes: stop vowel fricative stop fricative stop liquid vowel stop fricative vowel semivowel vowel nasal aspirate vowel liquid stop fricative fricative vowel affricate vowel liquid vowel stop fricative vowel liquid nasal vowel nasal stop
First combo phonemes: stop vowel stop fricative vowel liquid fricative affricate vowel fricative stop nasal vowel fricative stop vowel liquid fricative vowel liquid nasal vowel fricative stop vowel liquid fricative nasal vowel fricative stop vowel liquid nasal vowel fricative stop vowel liquid
Second combo phonemes: vowel nasal fricative liquid vowel affricate vowel liquid fricative vowel fricative liquid vowel affricate vowel liquid vowel stop fricative vowel liquid

First line: don't gimme no lip
Second line: i had a dream that you had a different face
First reconstruction: don't gimme no lip
Second reconstruction: i had a dream of what i had a different face
BoW from first, phonemes from second: i have been how i had to gimme a lip
Phonemes from first, BoW from second: don't dream

First line phonemes: stop vowel nasal stop stop vowel nasal vowel nasal vowel liquid vowel stop
Second line phonemes: vowel aspirate vowel stop vowel stop liquid vowel nasal vowel fricative semivowel vowel stop vowel aspirate vowel stop vowel stop vowel fricative vowel liquid vowel nasal stop fricative vowel fricative
First combo phonemes: vowel aspirate vowel fricative stop vowel nasal aspirate vowel vowel aspirate vowel stop stop vowel stop vowel nasal vowel vowel liquid stop
Second combo phonemes: stop vowel nasal stop stop liquid vowel nasal

First line: someone set a bad example
Second line: the reaction i get now thinking about it looking back
First reconstruction: someone bad that sign you're not bad
Second reconstruction: they are looking back on a back i get back down down
BoW from first, phonemes from second: the bad i bad about a bad set and i can not bad
Phonemes from first, BoW from second: then when the world won't get back

First line phonemes: fricative vowel nasal semivowel vowel nasal fricative vowel stop vowel stop vowel stop vowel stop fricative vowel nasal stop liquid
Second line phonemes: fricative vowel liquid vowel vowel stop fricative vowel nasal vowel stop vowel stop nasal vowel fricative vowel nasal stop vowel nasal vowel stop vowel stop vowel stop liquid vowel stop vowel nasal stop vowel stop
First combo phonemes: fricative vowel stop vowel stop vowel stop vowel stop vowel stop vowel stop vowel stop vowel stop fricative vowel stop vowel nasal stop vowel stop vowel nasal nasal vowel stop stop vowel stop
Second combo phonemes: fricative vowel nasal semivowel vowel nasal fricative vowel semivowel vowel liquid liquid stop semivowel vowel nasal stop stop vowel stop stop vowel stop

Table 4.15: Further examples of combinations, this time with 6 word and 11 word lines

### 4.7.2 Controlled Generation

What we have seen so far is that the Lyrics Decoder possesses good reconstruction and it can extract some phonetic information from the latent variable for a given line while retaining much of the semantic information from the provided BoW latent variable. Some interesting use cases for this decoder would now include having fixed representations of a BoW and replacing the phonemes or vice versa.

A musician may have written lyrics that have a cadence or meter that matches the rhythm of music, but wishes to change the subject matter to take the song in a new direction. In this case, we could feed the phonetic pattern of said lyrics to our decoder with different sets of BoWs and we will hopefully receive lyrics that still adhere to the meter while conveying different ideas and emotions. On the other hand, a musician may have settled on a theme, but may be having trouble writing lyrics that conform to the cadence of the song. Here we could extract a BoW from what is already written and match it with phonemes from a different sequence to hopefully gain lyrics that still carry the initial idea while conforming to the rhythm.

Mixing together random lines within the test set and comparing them to the original lines gives us some indication how well the model can do these tasks. Taking a pair of lines, we can extract a BoW from one and the phonemes from another and then combine the two as done in the previous chapter. Reconstructing the first line can give us a baseline to compare to.

| | Avg. Levenshtein | Avg. line words | BoW BLEU-1 | BoW Recall |
|---|---|---|---|---|
| Reconstructions | 5.66 | 6.06 | 0.3409 | 0.6801 |
| Combined lines | 8.64 | 6.08 | 0.2506 | 0.5343 |

Table 4.16: Metrics from 849 randomly combined lines and reconstructions

Similar to what we have seen before, the model seems to retain BoW information fairly well but struggle more with retaining the phonetic information. When we examine some of the examples from the model, we can see that this is the case and that the model struggles to create coherent lines with the mixed information.

| Source lines and constructed combinations: |
|---|
| First line: well no ones hurt and no ones cursed and no one needs to hide<br>Second line: if she says give it all i ll give everything to her .<br>First reconstruction: well well you can't pretend that you can go and one<br>Second reconstruction: is there i give everything i'll give everything to her<br>First combined line with BoW from first and phonemes from second: if the few i've got a free who you need<br>Second combined line with phonemes from first and BoW from second: will you give me and then you give me but then give you and |
| First line phonemes: semivowel vowel liquid nasal vowel semivowel vowel nasal fricative aspirate vowel liquid stop vowel nasal stop nasal vowel semivowel vowel nasal nasal vowel stop fricative stop vowel aspirate vowel stop<br>Second line phonemes: vowel fricative fricative vowel fricative vowel fricative stop vowel fricative vowel stop vowel liquid vowel liquid stop vowel fricative vowel fricative liquid vowel fricative vowel nasal stop vowel aspirate vowel liquid<br>First combo phonemes: vowel fricative fricative vowel fricative semivowel vowel vowel fricative stop vowel stop vowel fricative liquid vowel aspirate vowel semivowel vowel nasal vowel stop<br>Second combo phonemes: semivowel vowel liquid semivowel vowel stop vowel fricative nasal vowel vowel nasal stop fricative vowel nasal semivowel vowel stop vowel fricative nasal vowel stop vowel stop fricative vowel nasal stop vowel fricative semivowel vowel vowel nasal stop |
| First line: cut the paper to the bone<br>Second line: in the port of amsterdam<br>First reconstruction: to cut in the paper<br>Second reconstruction: in the port of amsterdam<br>First combined line with BoW from first and phonemes from second: in the paper and the paper<br>Second combined line with phonemes from first and BoW from second: to the port to be |
| First line phonemes: stop vowel stop fricative vowel stop vowel stop vowel liquid stop vowel fricative vowel stop vowel nasal<br>Second line phonemes: vowel nasal fricative vowel stop vowel liquid stop vowel fricative vowel nasal fricative stop vowel liquid stop vowel nasal<br>First combo phonemes: vowel nasal fricative vowel stop vowel stop liquid vowel nasal stop fricative vowel stop vowel stop liquid<br>Second combo phonemes: stop vowel fricative vowel stop vowel liquid stop stop vowel stop vowel |

First line: shake it up shake it up
Second line: see me comin mama
First reconstruction: shake it up shake it up
Second reconstruction: see my mama comin
First combined line with BoW from first and phonemes from second: shake me on shake me
Second combined line with phonemes from first and BoW from second: that's comin comin mama

First line phonemes: fricative vowel stop vowel stop vowel stop fricative vowel stop vowel stop vowel stop
Second line phonemes: fricative vowel nasal vowel stop vowel nasal vowel nasal nasal vowel nasal vowel
First combo phonemes: fricative vowel stop nasal vowel vowel nasal fricative vowel stop nasal vowel
Second combo phonemes: fricative vowel stop fricative stop vowel nasal vowel nasal stop vowel nasal vowel nasal nasal vowel nasal vowel

First line: awakens regrets
Second line: if you can see me i can see you
First reconstruction: oh your land
Second reconstruction: if you can see me i can see you
First combined line with BoW from first and phonemes from second: oh the land of the regrets
Second combined line with phonemes from first and BoW from second: i can see myself to see

First line phonemes: vowel semivowel vowel stop vowel nasal fricative liquid vowel stop liquid vowel stop fricative
Second line phonemes: vowel fricative semivowel vowel stop vowel nasal fricative vowel nasal vowel vowel stop vowel nasal fricative vowel semivowel vowel
First combo phonemes: vowel fricative vowel liquid vowel nasal stop vowel fricative fricative vowel liquid vowel stop liquid vowel stop fricative
Second combo phonemes: vowel stop vowel nasal fricative vowel nasal vowel fricative vowel liquid fricative stop vowel fricative vowel

First line: i walk along beside the garbagemen and i dig everything dig everything
Second line: state of integrity
First reconstruction: i do i've walk everything i've dig everything dig everything
Second reconstruction: state of integrity
First combined line with BoW from first and phonemes from second: before dig everything dig
Second combined line with phonemes from first and BoW from second: i i can't state that state of integrity come on the state

First line phonemes: vowel semivowel vowel stop vowel liquid vowel nasal stop vowel fricative vowel stop fricative vowel <unk> vowel nasal stop vowel stop vowel stop vowel fricative liquid vowel fricative vowel nasal stop vowel stop vowel fricative liquid vowel fricative vowel nasal
Second line phonemes: fricative stop vowel stop vowel fricative vowel nasal stop vowel stop liquid vowel stop vowel
First combo phonemes: stop vowel fricative vowel liquid stop vowel stop vowel fricative liquid vowel fricative vowel nasal stop vowel stop
Second combo phonemes: vowel vowel stop vowel nasal stop fricative stop vowel stop fricative vowel stop fricative stop vowel stop vowel fricative vowel nasal stop vowel stop liquid vowel stop vowel stop vowel nasal vowel nasal fricative vowel fricative stop vowel stop

First line: tell me
Second line: don t need no good advice don t need no human rights
First reconstruction: tell me
Second reconstruction: need my need and the human need to need a human need to
First combined line with BoW from first and phonemes from second: don't tell me and tell me on the tell that i
Second combined line with phonemes from first and BoW from second: need

First line phonemes: stop vowel liquid nasal vowel
Second line phonemes: stop vowel nasal stop nasal vowel stop nasal vowel stop vowel stop vowel stop fricative vowel fricative stop vowel nasal stop nasal vowel stop nasal vowel aspirate semivowel vowel nasal vowel nasal liquid vowel stop fricative
First combo phonemes: stop vowel nasal stop stop vowel liquid nasal vowel vowel nasal stop stop vowel liquid nasal vowel vowel nasal fricative vowel stop vowel liquid fricative vowel stop vowel
Second combo phonemes: nasal vowel stop

| |
|---|
| First line: a hundred thousand dreams |
| Second line: and the creatures all have fled |
| First reconstruction: a hundred thousand thousand |
| Second reconstruction: and the dead is creatures |
| First combined line with BoW from first and phonemes from second: in a thousand thousand |
| Second combined line with phonemes from first and BoW from second: i'll not the creatures |
| First line phonemes: vowel aspirate vowel nasal stop liquid vowel stop fricative vowel fricative vowel nasal stop stop liquid vowel nasal fricative |
| Second line phonemes: vowel nasal fricative vowel stop liquid vowel affricate vowel liquid fricative vowel liquid aspirate vowel fricative fricative liquid vowel stop |
| First combo phonemes: vowel nasal vowel fricative vowel fricative vowel nasal stop fricative vowel fricative vowel nasal stop |
| Second combo phonemes: vowel liquid nasal vowel stop fricative vowel stop liquid vowel affricate vowel liquid fricative |

Table 4.17: Example combinations and reconstructions

We can employ a similar strategy by manually selecting lines and then selecting bags of words and phonemes to swap out.

| |
|---|
| Line with phonetic pattern: and i stare out at the deep blue ocean with fear<br>BoW to combine with: ['sky', 'happy', 'bright', 'watch', 'morning', 'foe']<br>Line from combining phonetic pattern with BoW: and the sky it dressed to watch the sky sky<br>BoW to combine with: ['misty', 'dark', 'night', 'mountains', 'hill', 'low']<br>Line from combining phonetic pattern with BoW: in the night of the dark in the night<br>BoW to combine with: ['sky', 'bright', 'blue', 'happy', 'watch', 'jeans']<br>Line from combining phonetic pattern with BoW: and the happy and the sky sky |
| Line with phonetic pattern: i am confused why you had to go<br>BoW to combine with: ['go', 'sorry', 'problems', 'sad', 'one', 'please']<br>Line from combining phonetic pattern with BoW: i'm sorry to go with this go<br>BoW to combine with: ['joy', 'stay', 'dance', 'magic', 'charleston', 'flashy']<br>Line from combining phonetic pattern with BoW: i'm not you dance to you dance<br>BoW to combine with: ['confused', 'leave', 'footprint', 'simpler', 'start', 'sure']<br>Line from combining phonetic pattern with BoW: i'm just can you you can you |
| Line with phonetic pattern: tell me all the silent secrets you have kept<br>BoW to combine with: ['tell', 'thinking', 'infallible', 'big', 'stop', 'class']<br>Line from combining phonetic pattern with BoW: can for you that i can't tell yourself that he's<br>BoW to combine with: ['hopes', 'dreams', 'tell', 'fix', 'couple', 'washed']<br>Line from combining phonetic pattern with BoW: can a hopes of the dreams of dreams<br>BoW to combine with: ['secrets', 'tell', 'dirty', 'halls', 'coins', 'thoughts']<br>Line from combining phonetic pattern with BoW: tell the secrets that's just tell me |

Table 4.18: Phonetic pattern matched with different BoWs

Here we have several examples where a phonetic pattern is extracted and then joined with a BoW to produce a new line. We can also do the inverse where we select a line, extract its BoW and then combine it with different phonetic examples.

| |
|---|
| Line with extracted BoW: singing softly of paradise and better times<br>Extracted BoW: [’times’, ’better’, ’singing’, ’hundred’, ’paradise’, ’ways’]<br>Line with phonetic pattern: it has never been better<br>BoW from fixed line with phonemes from others: it has can it i can me<br>Line with phonetic pattern: this is good<br>BoW from fixed line with phonemes from others: she she’s she times<br>Line with phonetic pattern: what i would not give to hear that again<br>BoW from fixed line with phonemes from others: what it’s better than you can me? |
| Line with extracted BoW: i just want to dance with you until the night is over<br>Extracted BoW: [’dance’, ’want’, ’night’, ’magic’, ’taught’, ’dear’]<br>Line with phonetic pattern: and have a good time<br>BoW from fixed line with phonemes from others: and i want to be night<br>Line with phonetic pattern: come with me<br>BoW from fixed line with phonemes from others: dance with me<br>Line with phonetic pattern: happiness is spending this time together<br>BoW from fixed line with phonemes from others: how just want to the night and the night to dance |
| Line with extracted BoW: these broken dreams<br>Extracted BoW: [’broken’, ’dreams’, ’another’, ’appears’, ’leave’, ’thoughts’]<br>Line with phonetic pattern: are mine<br>BoW from fixed line with phonemes from others: in my dreams<br>Line with phonetic pattern: tell me what you want<br>BoW from fixed line with phonemes from others: dreams you broken<br>Line with phonetic pattern: cast aside so very very long ago now<br>BoW from fixed line with phonemes from others: but the broken but i’ll be broken |

Table 4.19: BoWs matched with different phoneme patterns

Similar to what we have seen before, the semantics are somewhat retained in the generated sequences and the phonetic patterns guide the generation, mainly in the length of the line. However, the Lyrics Decoder does have some problems with maintaining coherency when mixing information from different lines, especially when it comes to the phonetic information. This suggests that a Cycle-Reconstruction Loss as mentioned earlier might be able to strengthen the ability of the model to utilize phonetic information.

### 4.7.3 Diversity in Generated Sequences

Another aspect worth considering when generating lines is the diversity and uniqueness of said lines. We can gain an approximate idea of the diversity of lines by using Distinct scoring and Self-BLEU as mentioned previously. A comparison can then be made to the ground truth.

| | Distinct-1 ↑ | Distinct-2 ↑ | Self-BLEU ↓ |
|---|---|---|---|
| Ground truth | 0.2524 | 0.7415 | 0.3587 |
| Reconstructions | 0.1638 | 0.6191 | 0.8400 |
| Combined lines | 0.1417 | 0.6229 | 0.8352 |

Table 4.20: Comparison of diversity metrics for generated lines

Lower Distinct scores are to be expected in the case of the reconstructions and combined lines, since the model has to generalize when constructing these lines. The fairly high Distinct scores suggest that the model is managing to learn a majority of the vocabulary being used. However, the Self-BLEU scores are much higher (indicating worse performance). Self-BLEU also takes into account larger n-grams, suggesting that the model tends to make more simplistic predictions. This in turn leads to repetitions of sequences, leading to less coherent lines overall. As has been demonstrated with samples from the decoder, this is the case, with good overall retention of semantic information but poorer retention of the phonetic information. This further suggests that the model is in need of reinforcement to utilize the phonetic information provided.

# Chapter 5

# Conclusion and Future Work

The proposed model preserves semantic information well in its reconstructions and generation. The information is easily preserved in longer lines, but becomes harder to incorporate in shorter lines, especially if the semantic information from a long line is being mixed with phonetic information from a shorter line for the purpose of lyric generation.

The model is capable of generating lyrics and some of the phonetic information is retained in the generated lyrics and reconstructions, but boosting the retention and usage of this information is advisable. There are several strategies that could be explored in order to improve upon the proposed model. These include:

- **Pre-training phonetic embeddings:** Word embeddings benefit from using a pre-training method like Word2Vec to refine and improve the individual embeddings. Phonetic embeddings may well be able to benefit from such a strategy as well and become more accurate in a model's use if pre-trained.

- **Cycle-Reconstruction Loss:** As mentioned in subsection 3.1.3, a Cycle-Reconstruction Loss might be able to provide further feedback to the model and guide it towards using phonetic information better. In theory, this could enable the model to become better at recognizing phonetic structures without sacrificing semantic performance.

- **Different phonetic vocabularies:** As we have seen, the choice of vocabulary for the Phonetic VAE can have an impact on its performance. There might be different phonetic representations that can provide better information for our Lyrics Decoder (e.g. including diphthong and monopthong information in our larger vocabulary) and provide stable Phonetic VAEs.

# References

[CMU, 2014] (2014). The cmu pronouncing dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict. Accessed: 2021-05-01.

[Aut, 2019] (2019). From autoencoder to beta-vae. https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html. Accessed: 2021-07-17.

[Wor, 2019] (2019). Implementing word2vec in tensorflow. https://medium.com/analytics-vidhya/implementing-word2vec-in-tensorflow-44f93cf2665f. Accessed: 2021-05-01.

[VAE, 2019] (2019). Understanding variational autoencoders (vaes). https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73. Accessed: 2021-07-17.

[ELB, 2020] (2020). The evidence lower bound. https://mbernste.github.io/posts/elbo/. Accessed: 2021-07-02.

[Pre, 2021] (2021). Precision and recall. https://en.wikipedia.org/wiki/precision_and_recall. Accessed: 2021-10-12.

[Bahuleyan et al., 2018] Bahuleyan, H., Mou, L., Vechtomova, O., and Poupart, P. (2018). Variational attention for sequence-to-sequence models. In *COLING*.

[Baldi, 2012] Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In Guyon, I., Dror, G., Lemaire, V., Taylor, G., and Silver, D., editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 37–49, Bellevue, Washington, USA. PMLR.

[Ballard, 1987] Ballard, D. H. (1987). Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, AAAI'87, page 279–284. AAAI Press.

[Bengio et al., 2001] Bengio, Y., Ducharme, R., and Vincent, P. (2001). A neural probabilistic language model. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

[Blei et al., 2017] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

[Clarkson and Robinson, 1999] Clarkson, P. and Robinson, T. (1999). Towards improved language model evaluation measures. In *Sixth European Conference on Speech Communication and Technology*.

[David E. Rumelhart, 1986] David E. Rumelhart, Geoffrey E. Hinton, R. J. W. (1986). Learning representations by back-propagating errors. volume 323, pages 533–536.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[Gage, 1994] Gage, P. (1994). A new algorithm for data compression. *The C Users Journal archive*, 12:23–38.

[Goyal et al., 2016] Goyal, A., Lamb, A., Zhang, Y., Zhang, S., Courville, A., and Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4608–4616, Red Hook, NY, USA. Curran Associates Inc.

[Heafield et al., 2013] Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. (2013). Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria. Association for Computational Linguistics.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[Jordan, 1986] Jordan, M. I. (1986). Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986.

[Joyce, 2011] Joyce, J. M. (2011). *Kullback-Leibler Divergence*, pages 720–722. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Kingma and Welling, 2014a] Kingma, D. and Welling, M. (2014a). Auto-encoding variational bayes.

[Kingma, 2017] Kingma, D. P. (2017). Variational inference  deep learning.

[Kingma et al., 2015] Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

[Kingma and Welling, 2014b] Kingma, D. P. and Welling, M. (2014b). Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

[Levenshtein, 1965] Levenshtein, V. I. (1965). Binary codes with fixes for dropouts, insertions and character replacements.

[Li et al., 2016] Li, J., Galley, M., Brockett, C., Gao, J., and 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*, pages 110–119, San Diego, California. Association for Computational Linguistics.

[Lucas et al., 2019] Lucas, J., Tucker, G., Grosse, R. B., and Norouzi, M. (2019). Understanding posterior collapse in generative latent variable models. In *DGS@ICLR*.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *ICLR*.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

[Ng, 2004] Ng, A. Y. (2004). Feature selection, l vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 78, New York, NY, USA. Association for Computing Machinery.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W. J. (2002). Bleu: a method for automatic evaluation of machine translation.

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

[Prokhorov et al., 2019] Prokhorov, V., Shareghi, E., Li, Y., Pilevar, M. T., and Collier, N. (2019). On the importance of the kullback-leibler divergence term in variational autoencoders for text generation. pages 118–127.

[Qaiser and Ali, 2018] Qaiser, S. and Ali, R. (2018). Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

[Rumelhart and McClelland, 1987] Rumelhart, D. E. and McClelland, J. L. (1987). *Learning Internal Representations by Error Propagation*, pages 318–362.

[Sak et al., 2014] Sak, H., Senior, A. W., and Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128.

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[Shannon, 1948] Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

[Silver et al., 2017] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., and Hassabis, T. G. . D. (2017). Mastering the game of go without human knowledge. *Nature*, 550.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

[Zhang and Sabuncu, 2018] Zhang, Z. and Sabuncu, M. R. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8792–8802, Red Hook, NY, USA. Curran Associates Inc.

[Zhu et al., 2017] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Zhu et al., 2018] Zhu, Y., Lu, S., Zheng, L., Guo, J., Zhang, W., Wang, J., and Yu, Y. (2018). Texygen: A benchmarking platform for text generation models. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval.*

# APPENDICES

# Appendix A

# Interpolations of BoW VAE Latent Space

## A.1 Simple Vocabulary Interpolations

| **Interpolation 1** |
|---|
| cons mono cons di cons di cons mono cons mono cons cons di cons di cons |
| cons mono cons di cons di cons mono cons mono cons cons di cons di cons |
| cons mono cons di cons mono cons di cons mono cons cons di cons cons di cons |
| cons mono cons di cons mono cons cons di cons mono cons cons di cons di cons |
| cons mono cons di cons mono cons cons mono cons cons di cons mono cons cons di cons |
| cons mono cons cons mono cons di cons mono cons cons di cons mono cons cons di cons |
| cons mono cons cons mono cons di cons mono cons cons mono cons cons di cons cons di |
| cons mono cons cons mono cons mono cons di cons cons mono cons cons di cons |
| cons mono cons cons mono cons mono cons cons di cons mono cons cons di cons cons |
| cons mono cons cons mono cons mono cons cons mono cons cons di cons cons di cons |
| cons mono cons cons mono cons cons mono cons cons mono cons cons di cons cons di cons |
| cons mono cons cons mono cons cons mono cons cons mono cons cons di cons cons di cons |

| **Interpolation 2** |
|---|
| mono cons di di |
| mono cons di di |
| mono cons di di |
| mono cons di di di |
| cons mono di di di |
| cons mono di di di |
| cons mono cons di di di |
| cons mono cons di di di di cons mono cons di cons mono cons |
| cons mono cons di di cons mono cons di cons mono cons di cons mono cons |
| cons mono cons di di cons mono cons cons di di cons mono cons di cons mono cons |
| cons mono cons di cons di cons mono cons mono cons di cons mono cons di cons mono |
| cons mono cons di cons di cons mono cons mono cons di cons mono cons di cons mono |

| **Interpolation 3** |
|---|
| cons di cons mono cons di cons mono cons cons di cons cons di cons cons di cons cons di cons mono cons cons di |
| cons di cons mono cons di cons mono cons cons di cons cons di cons cons di cons cons di cons mono cons cons di |
| cons di cons mono cons mono cons di cons cons di cons cons di cons cons cons di cons cons cons di cons cons |
| cons mono cons di cons di cons mono cons cons cons mono cons cons di cons cons cons di cons cons cons di cons cons |
| cons mono cons di cons mono cons cons di cons cons di cons di cons mono cons cons di cons cons cons di cons |
| cons mono cons di cons mono cons cons di cons cons mono cons cons di cons mono cons cons di cons cons cons di |
| cons mono cons di cons mono cons cons di cons cons mono cons cons di cons mono cons cons di cons mono cons |
| cons mono cons mono cons cons di cons di cons mono cons cons di cons mono cons cons di cons mono cons cons |
| mono cons cons mono cons cons di cons mono cons di cons cons di cons mono cons cons di cons mono cons cons |
| mono cons cons mono cons cons di cons mono cons cons di cons di cons mono cons cons di cons mono cons cons |
| mono cons cons mono cons cons di cons mono cons cons di cons di cons mono cons cons di cons mono cons cons di |
| mono cons cons mono cons cons di cons mono cons cons di cons di cons mono cons cons di cons mono cons cons di |

| **Interpolation 4** |
|---|
| cons mono cons di mono cons cons mono cons mono cons mono cons mono cons cons di cons mono cons di cons di di cons di cons |
| cons mono cons di mono cons cons mono cons mono cons mono cons mono cons cons di cons mono cons di cons di di cons di cons |
| cons mono cons mono cons di cons mono cons mono cons mono cons cons di cons mono cons di cons di cons di cons di cons |
| cons mono cons mono cons di cons mono cons cons mono cons di cons mono cons mono cons di cons di di cons di cons |
| cons mono cons mono cons di cons mono cons cons di cons mono cons mono cons cons di cons mono cons di cons cons |
| cons mono cons mono cons di cons cons mono cons di cons mono cons cons mono cons di di cons di cons cons |
| cons mono cons cons di cons mono cons mono cons di cons mono cons cons di cons mono cons di cons cons |
| cons mono cons cons di cons mono cons di cons mono cons cons mono cons di cons mono cons di cons cons |
| cons mono cons cons di cons mono cons di cons mono cons cons di cons mono cons cons di di cons cons |
| cons mono cons cons di cons mono cons di cons cons mono cons cons di cons mono cons di cons cons |
| cons mono cons cons di cons mono cons cons di cons di cons mono cons cons di cons di cons cons |
| cons mono cons cons di cons mono cons cons di cons di cons mono cons cons di cons di cons cons |

| **Interpolation 5** |
|---|
| di cons cons mono cons di di cons cons di cons di cons cons di cons di cons cons di cons di cons |
| di cons cons mono cons di di cons cons di cons di cons cons di cons di cons cons di cons di cons |
| cons di cons mono cons cons di cons di cons cons di cons di cons mono cons di cons di cons cons di cons |
| cons di cons mono cons cons mono cons cons di cons mono cons cons di cons di cons mono cons cons di cons |
| cons di cons mono cons cons mono cons cons cons di cons di cons mono cons cons di cons di cons |
| cons di cons mono cons cons mono cons cons cons mono cons cons cons di cons mono cons cons cons di cons mono |
| cons mono cons cons di cons cons mono cons cons mono cons cons mono cons cons di cons mono cons cons cons di |
| cons mono cons cons cons mono cons cons mono cons cons di cons mono cons cons cons cons di cons mono cons cons cons |
| cons mono cons cons cons mono cons cons mono cons cons mono cons cons cons mono cons cons di cons di cons mono cons cons cons |
| cons mono cons cons cons mono cons cons mono cons cons mono cons cons mono cons cons cons di cons cons cons cons cons cons cons |
| cons mono cons cons cons mono cons cons mono cons cons mono cons cons mono cons cons cons mono cons cons cons di cons cons cons cons |
| cons mono cons cons cons mono cons cons mono cons cons mono cons cons mono cons cons cons mono cons cons cons di cons cons cons cons |

| **Interpolation 6** |
|---|
| mono cons di cons di cons di cons mono cons di di cons di di cons di di |
| mono cons di cons di cons di cons mono cons di di cons di di cons di di |
| mono cons di cons di cons di cons mono cons di di cons di di cons di cons |
| mono cons di cons di cons di cons mono di di cons di cons di di cons |
| mono cons di cons di cons mono cons di di di cons di di cons di |
| mono cons di cons di cons mono cons di di di cons di cons di |
| cons mono cons di di cons mono cons di di cons cons di di cons |
| cons mono cons di di cons mono cons di cons di cons di cons |
| cons mono cons di cons di mono cons di cons mono cons di cons cons di |
| cons mono cons di cons mono cons di di cons mono cons |
| cons mono cons cons di mono cons cons di cons mono cons di cons |
| cons mono cons cons di mono cons cons di cons mono cons di cons |

| **Interpolation 7** |
|---|
| cons mono cons di mono cons |
| cons mono cons di mono cons |
| cons mono cons di mono cons cons |
| cons mono cons di mono cons cons di cons mono cons di cons mono cons |
| cons mono cons di cons mono cons mono cons di mono cons di cons mono cons cons |
| cons mono cons di cons mono cons mono cons di cons cons mono cons mono cons di cons |
| cons mono cons di cons mono cons di cons mono cons cons cons mono cons di cons mono cons |
| cons mono cons di cons mono cons cons di cons mono cons cons mono cons cons di cons |
| cons mono cons di cons mono cons cons di cons mono cons mono cons cons di cons mono cons |
| cons mono cons di cons mono cons cons mono cons cons di cons mono cons cons di cons mono cons |
| cons mono cons di cons mono cons cons mono cons cons di cons mono cons cons cons di cons mono cons |
| cons mono cons di cons mono cons cons mono cons cons di cons mono cons cons cons di cons mono cons |

| **Interpolation 8** |
|---|
| cons mono cons cons cons mono cons cons di cons cons cons cons cons cons cons |
| cons mono cons cons cons mono cons cons di cons cons cons cons cons cons cons |
| cons mono cons cons cons mono cons cons di cons cons cons |
| cons mono cons cons cons mono cons cons di cons cons mono cons cons di cons |
| cons mono cons cons di cons cons mono cons cons mono cons di cons |
| cons mono cons cons di cons cons mono cons cons mono cons di cons |
| cons mono cons di cons cons mono cons cons cons mono cons di cons |
| cons mono cons di cons cons mono cons cons cons mono cons di cons |
| di cons mono cons cons mono cons cons cons mono cons di cons |
| di cons mono cons cons mono cons cons di |
| di cons mono cons cons di cons mono cons |
| di cons mono cons cons di cons mono cons |

| **Interpolation 9** |
|---|
| cons mono cons cons mono cons mono cons cons cons cons cons cons cons cons cons cons cons cons cons cons |
| cons mono cons cons mono cons mono cons cons cons cons cons cons cons cons cons cons cons cons cons cons |
| cons mono cons cons mono cons mono cons cons cons cons cons cons cons cons cons cons cons cons |
| cons mono cons cons mono cons mono cons cons cons cons cons cons cons cons |
| cons mono cons cons mono cons mono cons cons cons mono cons cons di cons |
| cons mono cons cons mono cons mono cons cons cons mono cons cons |
| cons mono cons cons mono cons mono cons cons mono cons cons |
| cons mono cons cons mono cons mono cons |
| cons mono cons cons mono cons mono cons |
| cons mono cons cons mono cons di cons cons mono cons cons di |
| cons mono cons cons mono cons di cons cons mono cons di cons |
| cons mono cons cons mono cons di cons cons mono cons di cons |

| **Interpolation 10** |
|---|
| cons mono cons di mono cons mono cons cons mono cons cons |
| cons mono cons di mono cons mono cons cons mono cons cons |
| cons mono cons di mono cons mono cons cons mono cons cons |
| cons mono cons di mono cons cons mono cons mono cons di cons |
| cons mono cons di cons mono cons mono cons di cons mono cons cons cons |
| cons mono cons di cons mono cons mono cons di cons mono cons cons cons di |
| cons mono cons di cons mono cons mono cons di cons mono cons mono cons cons cons di |
| cons mono cons di cons mono cons di mono cons cons mono cons cons mono cons cons di cons mono cons |
| cons mono cons di cons mono cons di mono cons cons mono cons cons mono cons cons cons di cons |
| cons mono cons di cons mono cons di cons mono cons mono cons cons di cons mono cons cons di cons mono cons |
| cons mono cons di cons mono cons di cons mono cons mono cons cons di cons mono cons cons di cons mono cons |
| cons mono cons di cons mono cons di cons mono cons mono cons cons di cons mono cons cons di cons mono cons |

Table A.1: Interpolations of latent space of smaller vocabulary Phonetic VAE

## A.2 Larger Vocabulary Interpolations

| **Interpolation 1** |
|---|
| vowel fricative stop vowel nasal stop stop liquid vowel nasal stop semivowel vowel nasal semivowel vowel nasal stop |
| vowel fricative stop vowel nasal stop stop liquid vowel nasal stop semivowel vowel nasal semivowel vowel nasal stop |
| vowel fricative stop stop vowel nasal stop liquid vowel stop semivowel vowel nasal semivowel vowel nasal stop |
| vowel fricative stop stop vowel nasal stop stop semivowel vowel nasal stop semivowel vowel nasal |
| vowel fricative stop stop vowel nasal stop stop semivowel vowel nasal stop semivowel vowel nasal |
| fricative vowel stop stop vowel stop stop stop stop stop stop stop |
| fricative vowel stop stop vowel stop stop stop stop stop stop stop |
| fricative vowel stop stop vowel stop stop stop stop stop |
| fricative vowel stop stop vowel stop stop stop stop stop |
| fricative vowel stop stop stop stop stop stop stop stop |
| fricative vowel stop stop stop stop stop stop stop |
| fricative vowel stop stop stop stop stop stop stop |

| **Interpolation 2** |
|---|
| vowel nasal fricative vowel nasal stop vowel stop vowel nasal stop fricative stop vowel nasal |
| vowel nasal fricative vowel nasal stop vowel stop vowel nasal stop fricative stop vowel nasal |
| vowel nasal fricative vowel stop vowel nasal stop vowel nasal stop stop vowel stop nasal |
| vowel nasal fricative vowel stop vowel nasal stop fricative vowel stop stop vowel nasal nasal |
| vowel nasal fricative vowel stop vowel nasal stop fricative vowel stop stop vowel nasal |
| vowel nasal fricative vowel stop vowel nasal stop fricative vowel stop stop vowel stop nasal |
| vowel nasal stop vowel fricative vowel stop fricative vowel stop stop vowel stop nasal |
| vowel nasal stop vowel fricative vowel stop fricative vowel stop stop stop |
| vowel nasal stop vowel fricative vowel stop vowel stop fricative stop stop |
| vowel nasal stop vowel fricative vowel stop vowel stop fricative stop stop stop |
| vowel stop vowel fricative vowel stop stop vowel fricative stop stop stop stop stop |
| vowel stop vowel fricative vowel stop stop vowel fricative stop stop stop stop stop |

| **Interpolation 3** |
|---|
| aspirate vowel liquid stop vowel stop vowel stop stop liquid vowel stop stop stop liquid vowel |
| aspirate vowel liquid stop vowel stop vowel stop stop liquid vowel stop stop stop liquid vowel |
| aspirate vowel liquid stop vowel stop vowel liquid stop stop liquid stop stop stop |
| aspirate vowel liquid stop vowel stop vowel liquid vowel stop stop stop stop stop |
| vowel liquid stop vowel nasal stop liquid vowel stop vowel stop stop stop |
| vowel liquid stop vowel nasal stop liquid vowel stop vowel vowel stop |
| vowel liquid stop vowel nasal stop liquid vowel stop vowel aspirate vowel |
| liquid vowel nasal stop vowel liquid vowel stop vowel stop vowel nasal stop vowel |
| liquid vowel nasal stop vowel liquid vowel stop vowel nasal stop vowel liquid vowel nasal stop |
| liquid vowel nasal stop vowel liquid vowel nasal stop vowel stop vowel liquid vowel nasal stop vowel nasal |
| liquid vowel nasal stop vowel liquid semivowel vowel nasal stop vowel stop vowel liquid vowel nasal vowel stop vowel nasal |
| liquid vowel nasal stop vowel liquid semivowel vowel nasal stop vowel stop vowel liquid vowel nasal vowel stop vowel nasal |

| **Interpolation 4** |
|---|
| fricative vowel nasal vowel fricative vowel stop vowel vowel fricative vowel liquid vowel stop vowel stop vowel liquid vowel stop vowel liquid vowel |
| fricative vowel nasal vowel fricative vowel stop vowel vowel fricative vowel liquid vowel stop vowel stop vowel liquid vowel stop vowel liquid vowel |
| fricative vowel nasal vowel fricative vowel stop vowel vowel liquid vowel fricative vowel stop vowel liquid vowel stop vowel liquid vowel stop vowel |
| fricative vowel nasal vowel fricative vowel stop vowel liquid vowel vowel fricative vowel nasal stop vowel liquid vowel liquid vowel stop vowel liquid |
| fricative vowel nasal vowel fricative vowel stop vowel liquid vowel nasal vowel fricative vowel liquid vowel liquid vowel stop vowel liquid vowel |
| fricative vowel stop vowel nasal fricative vowel vowel liquid vowel nasal vowel liquid vowel fricative vowel liquid stop vowel liquid vowel |
| fricative vowel stop vowel nasal fricative vowel liquid vowel vowel nasal vowel liquid vowel fricative liquid vowel nasal vowel liquid |
| fricative vowel stop vowel nasal fricative vowel liquid vowel liquid vowel nasal vowel fricative vowel liquid vowel nasal |
| fricative vowel stop vowel nasal fricative vowel liquid vowel liquid vowel nasal vowel nasal liquid vowel fricative |
| stop vowel fricative vowel nasal stop liquid vowel vowel nasal vowel liquid fricative vowel nasal vowel nasal |
| stop vowel fricative vowel nasal stop liquid vowel liquid vowel nasal vowel nasal fricative vowel |
| stop vowel fricative vowel nasal stop liquid vowel liquid vowel nasal vowel nasal fricative vowel |

| **Interpolation 5** |
|---|
| fricative vowel stop semivowel vowel liquid fricative liquid vowel stop vowel fricative liquid vowel fricative vowel nasal semivowel vowel liquid vowel fricative vowel liquid |
| fricative vowel stop semivowel vowel liquid fricative liquid vowel stop vowel fricative liquid vowel fricative vowel nasal semivowel vowel liquid vowel fricative vowel liquid |
| fricative vowel stop semivowel vowel liquid fricative vowel liquid fricative vowel stop semivowel vowel liquid vowel fricative vowel nasal fricative vowel liquid |
| fricative vowel stop vowel liquid fricative semivowel vowel liquid semivowel vowel fricative vowel liquid fricative vowel stop vowel nasal |
| fricative vowel stop vowel liquid fricative semivowel vowel nasal vowel liquid fricative vowel stop fricative vowel liquid |
| fricative vowel stop vowel nasal fricative liquid vowel stop vowel fricative semivowel vowel liquid vowel fricative |
| fricative vowel stop vowel nasal fricative vowel liquid semivowel vowel stop fricative vowel nasal |
| fricative vowel stop vowel nasal fricative vowel stop vowel liquid fricative |
| vowel fricative stop vowel nasal vowel fricative stop vowel liquid |
| vowel fricative vowel stop nasal vowel fricative semivowel vowel |
| vowel fricative vowel stop nasal vowel fricative stop |
| vowel fricative vowel stop nasal vowel fricative stop |

| **Interpolation 6** |
|---|
| liquid vowel stop vowel nasal fricative vowel liquid vowel nasal stop vowel nasal |
| liquid vowel stop vowel nasal fricative vowel liquid vowel nasal stop vowel nasal |
| liquid vowel fricative vowel nasal stop vowel liquid vowel nasal stop vowel nasal |
| liquid vowel fricative vowel nasal stop vowel liquid vowel nasal stop vowel nasal |
| liquid vowel fricative vowel stop liquid vowel nasal stop vowel nasal vowel stop |
| liquid vowel fricative vowel stop liquid vowel nasal stop vowel stop vowel nasal |
| fricative vowel liquid vowel liquid stop vowel nasal stop vowel liquid vowel nasal |
| fricative vowel liquid liquid vowel stop vowel stop vowel liquid stop vowel nasal |
| fricative vowel liquid fricative vowel stop vowel liquid stop vowel stop vowel nasal |
| fricative vowel liquid fricative vowel stop liquid vowel stop vowel stop vowel |
| fricative vowel fricative liquid vowel stop liquid vowel stop vowel stop |
| fricative vowel fricative liquid vowel stop liquid vowel stop vowel stop |

| **Interpolation 7** |
|---|
| nasal vowel nasal fricative vowel nasal fricative vowel nasal fricative |
| nasal vowel nasal fricative vowel nasal fricative vowel nasal fricative |
| vowel nasal stop fricative vowel nasal fricative vowel nasal fricative nasal |
| vowel nasal stop fricative vowel nasal fricative vowel nasal fricative vowel nasal |
| vowel nasal stop fricative vowel nasal fricative vowel nasal fricative vowel |
| vowel nasal stop semivowel vowel fricative fricative vowel nasal vowel nasal |
| vowel nasal stop semivowel vowel fricative fricative vowel nasal vowel fricative |
| vowel nasal stop semivowel vowel fricative fricative vowel stop vowel nasal |
| vowel nasal stop semivowel vowel fricative vowel stop fricative vowel liquid |
| vowel semivowel vowel fricative stop stop vowel nasal semivowel vowel liquid fricative |
| vowel semivowel vowel fricative stop stop vowel liquid semivowel vowel fricative vowel |
| vowel semivowel vowel fricative stop stop vowel liquid semivowel vowel fricative vowel |

| **Interpolation 8** |
|---|
| liquid vowel liquid stop vowel liquid vowel vowel nasal vowel liquid vowel fricative |
| liquid vowel liquid stop vowel liquid vowel vowel nasal vowel liquid vowel fricative |
| liquid vowel liquid vowel nasal stop vowel liquid vowel vowel nasal vowel liquid |
| liquid vowel fricative vowel liquid vowel nasal vowel liquid vowel nasal vowel nasal |
| liquid vowel fricative vowel liquid vowel nasal vowel nasal vowel liquid vowel nasal |
| liquid vowel fricative vowel nasal vowel liquid vowel nasal vowel liquid vowel nasal |
| liquid vowel fricative vowel nasal vowel liquid vowel nasal vowel nasal vowel liquid |
| liquid vowel fricative vowel nasal vowel nasal vowel liquid vowel nasal vowel nasal |
| nasal vowel liquid vowel fricative vowel nasal vowel nasal vowel liquid vowel nasal vowel |
| nasal vowel liquid vowel fricative vowel nasal vowel nasal vowel nasal vowel liquid vowel |
| nasal vowel fricative vowel liquid vowel nasal vowel nasal vowel nasal vowel nasal vowel |
| nasal vowel fricative vowel liquid vowel nasal vowel nasal vowel nasal vowel nasal vowel |

| **Interpolation 9** |
|---|
| semivowel vowel liquid fricative vowel nasal fricative vowel fricative vowel nasal |
| semivowel vowel liquid fricative vowel nasal fricative vowel fricative vowel nasal |
| semivowel vowel liquid fricative vowel stop fricative vowel nasal vowel fricative |
| semivowel vowel liquid fricative vowel stop fricative vowel nasal vowel fricative vowel |
| semivowel vowel liquid fricative vowel stop nasal vowel fricative vowel fricative liquid vowel |
| semivowel vowel liquid vowel fricative stop nasal vowel fricative vowel liquid nasal vowel |
| semivowel vowel liquid vowel fricative stop vowel nasal stop fricative vowel liquid vowel nasal |
| semivowel vowel liquid vowel nasal stop fricative vowel liquid fricative vowel stop semivowel vowel nasal |
| semivowel vowel liquid vowel nasal stop semivowel vowel liquid fricative vowel fricative stop liquid vowel nasal stop |
| semivowel vowel liquid vowel nasal semivowel vowel liquid stop fricative vowel liquid fricative vowel nasal stop |
| semivowel vowel liquid vowel nasal semivowel vowel liquid stop liquid vowel fricative fricative stop vowel nasal stop |
| semivowel vowel liquid vowel nasal semivowel vowel liquid stop liquid vowel fricative fricative stop vowel nasal stop |

| **Interpolation 10** |
|---|
| vowel nasal stop vowel nasal semivowel vowel liquid semivowel vowel nasal vowel nasal nasal vowel nasal vowel fricative |
| vowel nasal stop vowel nasal semivowel vowel liquid semivowel vowel nasal vowel nasal nasal vowel nasal vowel fricative |
| vowel nasal stop vowel nasal stop vowel semivowel vowel liquid nasal vowel nasal semivowel vowel nasal |
| vowel nasal stop vowel stop semivowel vowel liquid vowel nasal nasal vowel nasal vowel nasal |
| vowel stop nasal vowel stop liquid vowel semivowel vowel nasal vowel nasal semivowel vowel |
| vowel stop stop vowel liquid semivowel vowel nasal vowel nasal stop vowel liquid |
| vowel stop stop vowel liquid vowel nasal stop vowel liquid semivowel vowel nasal |
| vowel stop stop vowel liquid stop vowel liquid vowel nasal vowel nasal |
| vowel stop stop vowel liquid stop vowel liquid vowel nasal |
| vowel stop stop liquid vowel stop vowel liquid vowel liquid stop |
| vowel stop stop liquid vowel stop vowel liquid vowel stop stop stop stop |
| vowel stop stop liquid vowel stop vowel liquid vowel stop stop stop stop |

Table A.2: Interpolations of latent space of larger vocabulary Phonetic VAE