Latent Variable Models in Neural Machine Translation

John Isak Texas Falk

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Master of Science

of

University College London.

The Centre for Computational Statistics and Machine Learning

University College London

August 26, 2017

I, John Isak Texas Falk, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Neural Machine translation is a direction in automated translation which learns the mapping from the source to the target language directly in an end-to-end fashion using the framework of neural networks. As NMT hinges on advances in neural network methods and architectures, it is able to directly use improvements there in its own domain.

We use recent improvements in latent variables models in order to train a completely probabilistic generative model with a latent variable representing a language-agnostic representation of a sentence mapping through deep neural networks to two different output languages. Following recent advances in training deep generative latent variable models we approximate the posterior of the latents given output with a recognition model, mimicking a VAE. Following the SGVB method to find a stochastic lower bound to the true log-likelihood of the observed data, we train the parameters of the generative model and the variational recognition model jointly to optimise this bound.

Since the recognition model acts as a pseudo-posterior (it approximates it given constraints on the distributional form of the recognition model) we can use this to translate from one language to another by finding the posterior q-distribution over z and then from sampled z find the most likely output of the languages.

Acknowledgements

I would like to thank the help from my supervisor Harshil Shah for helping me make this thesis possible, and my parents, for always being there for me.

Contents

List of Figures

List of Tables

Nomenclature

GD	Gradient Descent
MCMC	Markov Chain Monte Carlo
MLE	Maximum Likelihood Estimation
NLP	Natural Language Processing
NMT	Neural Machine Translation
SGD	Stochastic Gradient Descent
VI	Variational Inference

$\boldsymbol{\theta} \in \boldsymbol{\Theta}$	Parameters $\boldsymbol{\theta}$ belonging to the parameter space $\boldsymbol{\Theta}$
$\boldsymbol{v}_{\mathrm{w}}$	Word embedding of w
$\boldsymbol{w}$	One-hot encoding of w
$\boldsymbol{x}_i \in \mathcal{Y}$	Sentence $\boldsymbol{x}_i$ belonging to language X
$\boldsymbol{y}_i \in \mathcal{Y}$	Sentence $\boldsymbol{y}_i$ belonging to language Y
$\mathcal{M}$	A model $\mathcal{M}$
$\mathcal{X}$	Set of sentences belonging to language X

$\mathcal{Y}$ Set of sentences belonging to language Y

w The word as a symbol

$\sigma(x)$ An activation function with respect to x

N Number of data points

V Dictionary; set of all words in vocabulary

$\mathbf{0}_d$ The d-dimensional zero vector

$\mathbf{0}_{d\times d}$ The $d \times d$-dimensional zero matrix

$\boldsymbol{A} \in \mathbf{R}^{n\times m}$ Real matrix $\boldsymbol{A}$ of dimensions $n \times m$

$\boldsymbol{A} \odot \boldsymbol{B}$ Elementwise multiplication of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$

$\boldsymbol{I}_d$ The $d \times d$ unit matrix

$\boldsymbol{x}$ A real column vector

$\mathbb{R}$ Set of real numbers

$\mathbb{R}^d$ Real vector space of dimension d

$\nabla_{\boldsymbol{x}} Q(\boldsymbol{x})$ Gradient of function Q with respect to vector $\boldsymbol{x}$

$\mathrm{pa}(x)$ Set of parent nodes for node x in a graphical model

$\mathbf{0}_d$ The d-dimensional zero vector

$\mathbf{0}_{d\times d}$ The $d \times d$-dimensional zero matrix

$\boldsymbol{A} \in \mathbf{R}^{n\times m}$ Real matrix $\boldsymbol{A}$ of dimensions $n \times m$

$\boldsymbol{A} \odot \boldsymbol{B}$ Elementwise multiplication of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$

$\boldsymbol{I}_d$ The $d \times d$ unit matrix

$\boldsymbol{x}$ A real column vector

$\mathbb{R}$ Set of real numbers

$\mathbb{R}^d$ Real vector space of dimension d

$\nabla_{\boldsymbol{x}} Q(\boldsymbol{x})$ Gradient of function Q with respect to vector $\boldsymbol{x}$

$\mathrm{pa}(x)$ Set of parent nodes for node x in a graphical model

$\boldsymbol{\theta} \in \boldsymbol{\Theta}$ Parameters $\boldsymbol{\theta}$ belonging to the parameter space $\boldsymbol{\Theta}$

$\boldsymbol{v}_{\mathrm{w}}$ Word embedding of w

$\boldsymbol{w}$ One-hot encoding of w

$\boldsymbol{x}_i \in \mathcal{Y}$ Sentence $\boldsymbol{x}_i$ belonging to language X

$\boldsymbol{y}_i \in \mathcal{Y}$ Sentence $\boldsymbol{y}_i$ belonging to language Y

$\mathcal{M}$ A model $\mathcal{M}$

$\mathcal{X}$ Set of sentences belonging to language X

$\mathcal{Y}$ Set of sentences belonging to language Y

w The word as a symbol

$\sigma(x)$ An activation function with respect to x

N Number of data points

V Dictionary; set of all words in vocabulary

GD Gradient Descent

MCMC Markov Chain Monte Carlo

MLE Maximum Likelihood Estimation

NLP Natural Language Processing

NMT Neural Machine Translation

SGD Stochastic Gradient Descent

VI Variational Inference

$\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{y})$ Multivariate covariance with respect to $\boldsymbol{x}$ and $\boldsymbol{y}$

$\mathrm{Cov}(x, y)$ Covariance with respect to x and y

$\mathbb{E}_x[f(x, y)]$ Expectation of a function $f(x, y)$ with respect to a random variable x. If no ambiguity to which variable we are taking expectation with respect to exists the subscript may be left out

$D_{KL}(p, q||)$ Kullback-Leibler divergence from distribution q to p

$\mathcal{D}$ Set of data points

$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ Normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix Σ

$p(x)$ Probability distribution with respect to random variable x

$\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{y})$ Multivariate covariance with respect to $\boldsymbol{x}$ and $\boldsymbol{y}$

$\mathrm{Cov}(x, y)$ Covariance with respect to x and y

$\mathbb{E}_x[f(x, y)]$ Expectation of a function $f(x, y)$ with respect to a random variable x. If no ambiguity to which variable we are taking expectation with respect to exists the subscript may be left out

$D_{KL}(p, q||)$ Kullback-Leibler divergence from distribution q to p

$\mathcal{D}$ Set of data points

$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ Normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix Σ

$p(x)$ Probability distribution with respect to random variable x

Chapter 1

Introduction

Natural language processing (NLP) is an old field with roots in many different disciplines including but not exclusive to electrical engineering, artificial intelligence and linguistics [2, p. 10-15]. Machine learning often view language as a statistical problem. A language model is a statistical model that trained on a training corpus assign probabilities to new sentences. Using latent variable models enables these language models to generate novel sentences from an underlying stochastic variable. While using probabilistic models enables us to do inference in a principled manner, it is often non-trivial to find the optimal parameters of latent variable models. Variational Inference approximates the log-likelihood through the Evidence Lower Bound (ELBO) which bound it from below, by introducing an approximate distribution of the conditional probability of the latent variable, $p(\boldsymbol{z}|\boldsymbol{x})$ [3].

Although Variational Inference specifies ways of approximating the log-likelihood by a variational distribution q it does not specify the form of this q. Variational Autoencoder (VAE) [4], also under the name of stochastic backpropagation [5], is a framework for training deep generative models with latent variables by introducing a recognition model $q_{\boldsymbol{\phi}}$. Except for the most basic of models the ELBO function is not tractable, however, using the reparametrisation trick and the law of large numbers, it is possibly to approximate the ELBO by sampling the latent variable instead of integrating it out, giving us the Stochastic Gradient Variational Bayes (SGVB) algoritm. The recognition model may then be optimized jointly with the generative model p_θ with respect to the SGVB objective [4]. As the recognition model maps the input sentences to the latent space, it effectively tries to compress information about the sentences through $\boldsymbol{z}$, resulting in distributed latent representations of sentences [6]. After learning the recognition mapping from $\boldsymbol{x}, \boldsymbol{y}$ to $\boldsymbol{z}$ we may use this form to do translation by omitting one of the input languages, effectively letting a source language decide the distribution over the latent space, which can be mapped back through the generative model to both $\mathcal{X}$ and $\mathcal{Y}$ performing both reconstruction and translation concurrently.

Chapter 2

Background Knowledge

This chapter will introduce the necessary background knowledge to follow the theoretical derivations later.

2.1 Probability Theory and Statistics

We go over the rules of probability and statistics.

2.1.1 Rules and Theorems

We here lay out definitions and theorems that we will make use of in the thesis. For all parts below, assume that $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ are random variables and $p(X)$ respectively $p(Y)$ the probability distributions of these variables. We interpret the integral over X to be a sum over $\mathcal{X}$ if X is discrete and a regular Lebesgue integral over $\mathcal{X}$ if X is continuous. Then:

Definition 2.1.1 (Unit Volume)**.**

$$\int_X p(X)\,\mathrm{d}X = 1$$

Definition 2.1.2 (Non-negativity)**.**

$$p(X) \geq 0$$

Most manipulations of random variables may be reduced to the following three rules of probability:

Theorem 2.1.3 (Sum Rule).

$$p(X) = \int_{\mathcal{Y}} p(X, Y)\, \mathrm{d}Y$$

Theorem 2.1.4 (Product Rule).

$$p(X, Y) = p(Y|X)p(X)$$

Theorem 2.1.5 (Bayes Rule).

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

Two very important operations involving probabilities of random variables are those of *Expectation* and *Covariance*.

Definition 2.1.6 (Expectation). *Let $f : \mathcal{X} \to \mathbb{R}$, then*

$$\mathbb{E}_X[f] = \int_{\mathcal{X}} f(X)p(X)\, \mathrm{d}X$$

Definition 2.1.7 (Covariance).

$$\mathrm{Cov}(X, Y) = \mathbb{E}_{XY}[(X - \mathbb{E}_X[X])(Y - \mathbb{E}_Y[Y])]$$

The variance operator is straightforwardly defined:

Definition 2.1.8 (Variance).

$$\mathrm{Var}(X) = \mathrm{Cov}(X, X)$$

The generalisation from $f : \mathcal{X} \to \mathbb{R}$ to $f : \mathcal{X} \to \mathbb{R}^n$ is defined in the straightforward manner. If $\boldsymbol{f} = f(X)$ then:

$$\mathbb{E}_X \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} \mathbb{E}_X f_1 \\ \vdots \\ \mathbb{E}_X f_n \end{bmatrix}$$

similarly $\mathrm{Cov}(\boldsymbol{f})$ is a $D \times D$-dimensional matrix where $\mathrm{Cov}(\boldsymbol{f})_{i,j} = \mathrm{Cov}(f_i, f_j)$[7, 8].

2.1.2 The Gaussian Distribution

For a D-dimensional random vector $\boldsymbol{X}$, the multivariate Gaussian distribution takes the form

$$\mathcal{N}(\boldsymbol{X}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{X} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{X} - \boldsymbol{\mu})\right) \tag{2.1}$$

where $\boldsymbol{\mu}$ is a D-dimensional mean vector, Σ is a $D \times D$ dimensional positive definite covariance matrix and $|\Sigma|$ denotes the determinant of Σ. It is straightforward to show that these parameters correspond to the mean and covariance as defined in 2.1.6 and 2.1.7[8].

The Gaussian distribution can be seen as a unit D-dimensional cube which is translated, sheared and rotated, giving rise to the fact that we can write any Gaussianly distributed random variable $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma)$ as a linear combination of a unit Gaussian random variable $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{0}_D, \boldsymbol{I}_{D\times D})$. If we let $\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top = \Sigma$ be the Cholesky decomposition[9, p. 100-102] of Σ, then we also have that

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{\Lambda} \boldsymbol{z} \tag{2.2}$$

, where the equality is in terms of distribution. If we further assume that $\boldsymbol{x}$ is parametrised by $\boldsymbol{\mu}$ and Σ such that Σ is diagonal positive definite with diagonal $\boldsymbol{\sigma}$, then $\Sigma = \boldsymbol{\sigma} \odot \boldsymbol{I}_{D\times D}$ where we let $\boldsymbol{\sigma} \odot \boldsymbol{I}_{D\times D}$ represent elementwise multiplication of $\boldsymbol{\sigma}$ with the diagonal of $\boldsymbol{I}_{D\times D}$. Finally this means that if we want to sample a random variable $\boldsymbol{x}$ with diagonal covariance structure, then we can do this by sampling a unit normal $\boldsymbol{z}$ which we then transform, which we can express as

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma} \odot \boldsymbol{I}_{D\times D}) \tag{2.3}$$

As the Gaussian distribution is part of the exponential family[8], the density of the joint distribution of iid Gaussian variables are themselves Gaussian distributed where the natural parameters of this joint distribution is the sum of the natural parameters of each random variable in the joint. In particular for the Gaussian distribution, this means that if we have a collection of iid gaussian random variables $\{\boldsymbol{x}_i\}_i^n$, such that $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_i, \Sigma_i)$, then the joint can be found to be Gaussian distributed as

$$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

, where

$$\Sigma = \left(\sum_i^n \Sigma_i^{-1}\right)^{-1} \tag{2.4}$$

$$\boldsymbol{\mu} = \Sigma\left(\sum_i^n \Sigma^{-1}\boldsymbol{\mu}_i\right) \tag{2.5}$$

[7, p. 78-84].

In the case of two independent random variables distributed according to the form as laid out in (2.3), $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu_x}, \boldsymbol{\sigma_x} \odot \boldsymbol{I})$ and $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu_y}, \boldsymbol{\sigma_y} \odot \boldsymbol{I})$, we have that the resulting distribution $p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x})p(\boldsymbol{y})$ is distributed such that

$$p(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu_{x,y}}, \boldsymbol{\sigma_{x,y}} \odot \boldsymbol{I})$$

where

$$\boldsymbol{\sigma_{x,y}} = \frac{1}{\boldsymbol{\sigma_x}^{-1} + \boldsymbol{\sigma_y}^{-1}} \tag{2.6}$$

$$\boldsymbol{\mu_{x,y}} = \frac{\boldsymbol{\sigma_x}^{-1}\boldsymbol{\mu_x} + \boldsymbol{\sigma_y}^{-1}\boldsymbol{\mu_y}}{\boldsymbol{\sigma_x}^{-1} + \boldsymbol{\sigma_y}^{-1}} \tag{2.7}$$

, with the inverse and division operators being done elementwise.

2.1.3 Maximum Likelihood Estimation

Assume we have a model $\mathcal{M}$ parametrised by $\boldsymbol{\theta}$ constrained to live in the parameter space $\boldsymbol{\Theta}$. Given data $\mathcal{D}$ we want to be able to fit the parameters $\boldsymbol{\theta}$ such that the model generalise to unseen data.

We define the likelihood function Using the common assumption of i.i.d datapoints

$$\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\boldsymbol{\theta}) = \prod_i^n p(\boldsymbol{x}_i|\boldsymbol{\theta}) \tag{2.8}$$

The MLE of of the parameters of the model is then defined to be

$$\hat{\boldsymbol{\theta}}_{ML} = \operatorname*{argmax}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) \tag{2.9}$$

.

While the original MLE is defined in terms of the likelihood function $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$, it's often more practical to work with the logarithm of this function, the log-likelihood function $\ell(\boldsymbol{\theta}; \mathcal{D})$. Using the log-likelihood we transform this product into a form involving sums

$$\ell(\boldsymbol{\theta}|\mathcal{D}) = \log \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \sum_i^n \log p(\boldsymbol{x}_i) \tag{2.10}$$

[10].

Besides from simplifying notation and calculation, the log-likelihood has the added benefit of reducing the risk of arithmetic underflow due to the small magnitude of individual probabilities. In models with long dependency such as language models, working in the log-space is needed to enable computation efficiently and robustly.

2.1.4 Graphical models

Graphical models are powerful tools for specifying the dependency relations between random variables in models. Each graph specifies a class of distributions possible to for the model. In particular they provide a clear way of reasoning about model assumptions and provide a framework for doing inference and learning [8].

Directed graphical models specify how the joint distribution of a set of random variables $\mathcal{X}$ factors in a conditional manner. In general, the relationship between a given directed graph and the corresponding distribution over the variables in $\mathcal{X}$ is such that the join distribution defined by the graph is given by the product, over all vertices of the graph, of a conditional distribution for each vertex conditioned on the variables corresponding to the parents of that vertex in the graph. So for a graph with K vertices, the joint distribution is give by

$$p(\mathcal{X}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}(x_k)) \tag{2.11}$$

where $\mathrm{pa}(x_k)$ is defined as the set of random variables corresponding to the parent vertices of the random variable x_k.

Although a graphical model is completely defined in terms of it's vertex and edge set, it is really the most powerful when visualized as a diagram. I will show this with an example using all of the graphical model parts found in the later section when we specify our models.

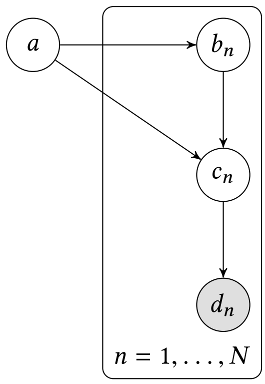


Figure 2.1: Example of a graphical model

There are two different vertices in this graphical model, the greyed out vertex indicates that the random variable is *observed* such that it's value is fixed and known. The white vertices indicates latent random variables which we don't observe.

The edge around some of the vertices specifying that $n = 1, \ldots, N$ is an instance of plate notation. Plate notation signifies that the plate is duplicated N times and is used for cases where the same subgraph reoccurs many times as is the case with for example iid data.

For this example we have the following factorisation of the joint distribution, following the rules laid out in equation (2.11),

$$p(a, \{(b_n, c_n, d_n)_{n=1}^{N}\}) = p(a) \prod_{n=1}^{N} p(d_n|c_n)p(c_n|b_n, a)p(b_n|a)$$

.

2.2 Deep Learning

Until recently the field of NLP were dominated by older machine learning techniques utilising linear models trained over very high-dimensional and sparse feature vectors. Recently the field has switched over to neural networks over dense inputs instead using embeddings[11, p. 1 - 2].

Neural networks may be seen from many angles, but from a mathematical point of view a neural network parametrised by parameters (weights) $\boldsymbol{\theta}$ specifies a non-linear functional relationship between the input to the network $\boldsymbol{x}$ and the output $\boldsymbol{y}$. As such each neural network with unspecified parameters $\boldsymbol{\theta}$ in a parameter space Θ specifies a set of functions [7]. It has been shown that this set of functions of several architectures are universal approximators and thus are able to approximate a big set of nice functions [12, 13], justifying their use theoretically.

2.2.1 Multilayer Perceptron

MLP's are neural networks represented by functional composition, where each function is interpreted as a layer of the network. The original MLP can be defined in terms of a recurrence relation such that if we have input vectors of the form $\boldsymbol{x} \in \mathbb{R}^{d_{in}}$ and output vectors of the form $\boldsymbol{y} \in \mathbb{R}^{d_{out}}$ and $\sigma_i(\cdot)$ represents an arbitrary activation function, then an MLP with L layers have the functional form

$$f(\boldsymbol{x}|\boldsymbol{\theta}) = \sigma_L(\boldsymbol{W}_L \boldsymbol{z}_{L-1} + \boldsymbol{b}_L) \tag{2.12}$$

where for any $l \in \{2, \ldots, L-1\}$

$$\boldsymbol{z}_l = \sigma_l(\boldsymbol{W}_l \boldsymbol{z}_{l-1} + \boldsymbol{b}_l) \tag{2.13}$$

and with the base case

$$\boldsymbol{z}_1 = \sigma_1(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1) \tag{2.14}$$

[14].

$\boldsymbol{W}_l$ and $\boldsymbol{b}_l$ may be of any dimension as long as it is dimensionally consistent with the input and output of the previous and next layers and conform to the original input and output dimensions. In this case we have that the parameters of the network are all of the biases and weights for the layers, $\boldsymbol{\theta} = \{(\boldsymbol{W}_l, \boldsymbol{b}_l)_{l=1}^{L}\}$.

Activation functions generally are monotonically increasing function mapping real numbers to some interval. Activation functions which will be useful to us are

Sigmoid

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{2.15}$$

The sigmoid activation function maps any real number to a value in $(0, 1)$.

Hyperbolic tangent

$$\tanh(x) = \frac{2}{1 + \exp(-2x)} - 1 = 2\sigma(2x) - 1 \tag{2.16}$$

SELU Scaled Exponential Linear Units have the form

$$\mathrm{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha(\exp(x) - 1) & \text{if } x \leq 0 \end{cases} \tag{2.17}$$

[15].

Softmax The Softmax is a mapping from an input space $\mathbb{R}^{d_{in}}$ to an output space $\mathbb{R}^{d_{out}}$, the form given by

$$\text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_j)}{\sum_{k=1}^{d_{out}} \exp(z_k)} \tag{2.18}$$

. Since it is normalized it is a natural candidate for defining categorical distributions [8] and we will use it for this purpose in our models.

2.2.2 Recurrent Neural Networks

RNN's have traditionally been the neural network architecture of choice for NLP due to it being able to handle long-term dependencies well [16]. As we only use it briefly we will not describe it in depth. For in depth treatment of the architecture with respect to nlp see [17, 18].

2.2.3 Convolutional Neural Networks

The CNN we will use will be highly specialised for our case. We use the WaveNet as laid out in the paper by DeepMind [1].

WaveNet works on data of the form $\boldsymbol{x}$, were we describe the distribution as

$$p(\boldsymbol{x}) = \prod_{l=1}^{L} p(\boldsymbol{x}_l | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1})$$

which is the form common in NLP and NMT. The output of the model has the same time dimensionality as the input, meaning that the model has the ability to output a categorical distribution over the next value x_l with a softmax layer. In our case this is a parametrisation of the output probability at point l.

As WaveNet uses dilated convolutions, a type of convolution where the filter is applied to an area larger than its length by skipping input values with a certain step, it is possible to stack the dilation layers. By stacking them in such a way that the dilations increase with the depth of the network by a power of two, we get a receptive field which grows exponentially [1].

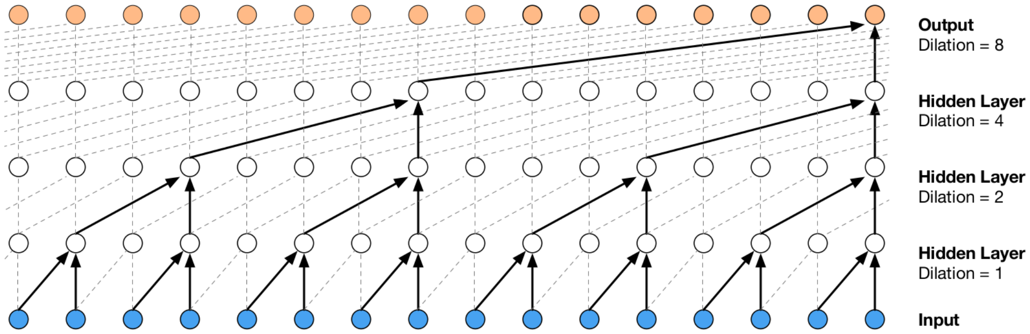


Figure 2.2: Dilated Convolutions in WaveNet [1]

2.3 Natural Language Processing

Humans use natural language every day to convey concepts and abstractions to each other in an efficient manner. Compared to formal languages found in mathematics and programming, the natural languages we use are often ambiguous systems filled with rules and exceptions [19, 20].

Natural Language Processing is an old field that for a long time developed in parallel with the field of machine learning and computational statistics which deals with how to process information coming from human languages and is split up into several subfields such as Machine Translation, statistical parsing and sentiment analysis [20].

We will here lay out the necessities for understanding the neural language model we will build.

2.3.1 Language model

We define a sentence to be a vector of words $\boldsymbol{w}_{1:L} = (\mathrm{w}_1, \ldots, \mathrm{w}_L)^\top$ such that each word is an atomic element $\mathrm{w}_i \in V$, where V is the dictionary of words in our language. Repeated use of the rule laid out in 2.1.4 enables us to rewrite the probability of a sentence in terms of how it is composed of words

$$P(\boldsymbol{w}_{1:L}) = \prod_{l=1}^{L} P(\mathrm{w}_l | \mathrm{w}_1, \ldots, \mathrm{w}_{l-1}) \tag{2.19}$$

where w_l is the l'th word of the sentence $\boldsymbol{w}_{1:L}$[21].

This form of the probability of a sentence lays bare how the words in a sentence relate to words that has previously occurred in the sentence. We will assume that all sentences in a dataset $\mathcal{D} = \{(\boldsymbol{w}^n)_{n=1}^N\}$ are iid which enables us to express the log-likelihood in the form of equation (2.10), and also the form that WaveNet uses to model probabilities.

2.3.2 Word embeddings

Breaking down sentences at a word level and processing them into a form that encodes information efficiently is a problem which has gained notorious recognition [22], leading to algorithms such as word2vec [23] and Glove [24]. However, these techniques work less well in a neural network setting where instead finding the best embedding jointly with the parameters of the model is preferred [11, p. 5-7].

A straightforward way to represent the various words of the dictionary is as one-hot-encoded vectors such that a word $\mathrm{w} \in V$, such that size of V is $|V|$, with an index i given by its place in the dictionary sorted alphabetically in descending order will have the vector representation

$$\boldsymbol{w} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.20}$$

[11, p. 6] such that $\boldsymbol{w}_j = \delta_{ij}$.

While this is a form conceptually easy to understand, it fails to account for the curse of dimensionality as the size of the vocabulary grows very big in practice and the fact that the cosine similarity of two words $\mathrm{v}_1, \mathrm{v}_2 \in V$ is zero unless they are the same word

$$\cos_{similarity}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \delta_{\mathrm{v}_1\mathrm{v}_2} \tag{2.21}$$

. This means that no meaning is embedded in the vector space except for the location in the sorted dictionary. Instead we would like to associate each word in the vocabulary with a distributed *word feature vector*, a dense, real-valued vector in $\mathbb{R}^m$ where m is the dimension of our embedding space. After training this embedding jointly with the parameters of the network words which share similarities such as `Dog, Puppy` would have a higher similarity score than with an unrelated concepts such as `Dog, Bulwark` [21].

We may represent this in a mathematical form by trying to find a linear map C from any element $\mathrm{w} \in V$ such that $C(\mathbf{w}) \in \mathbb{R}^m$. Using the canonical basis of the Euclidean space, we can express this linear map in terms of a matrix $\boldsymbol{C} \in \mathbb{R}^{m \times |V|}$, thus the word feature vector of the learned embedding can be represented by the matrix multiplication $\boldsymbol{C}\boldsymbol{w}$.

2.3.3 Neural machine translation

For a long time the dominant paradigm within machine translation was to use phrase based machine translation systems [25, 26], recently however, modelling the word or character level directly with neural networks has overtaken phrase based translation systems [27, 28]. These systems are called Neural Machine Translation.

Most NMT models work in terms of an encoder-decoder architecture where the encoder extracts a fixed length representation $\boldsymbol{c}$, often called a context vector, from a variable length input sentence $\boldsymbol{x} \in \mathcal{X}$, and the decoder uses this representation to generate a correct translation $\boldsymbol{y} \in \mathcal{Y}$ from this representation [29] as can be seen in Figure 2.3.

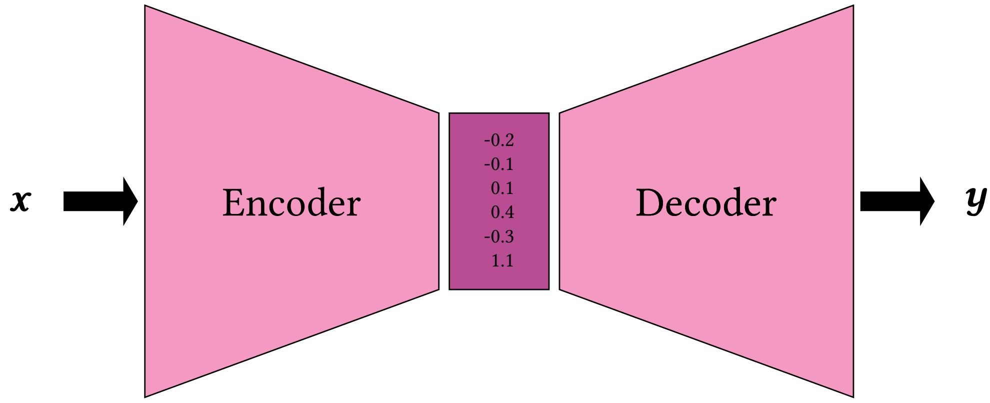


Figure 2.3: Encoder decoder schematic

2.4 Optimization

Most parts of machine learning reduces to optimization of a loss function. While optimisation of convex problems are well-understood there has been considerable interest in ways to optimize non-linear loss functions such as those used traditionally in deep learning.

While theoretical results are lacking, there has been substantial advances in various optimisation techniques fit to attack the highly non-linear, non-convex and high-dimensional optimization problems of learning in deep models, particularly from Stochastic Gradient Descent and its friends. This has led to numerous gradient descent-like algorithms used machine learning [30].

2.4.1 Stochastic Gradient Descent

For a normal probabilistic machine learning problem, we have data $\mathcal{D}$ that we try to model with a model $\mathcal{M}$ parametrised by parameters $\boldsymbol{\theta} \in \Theta$. The optimization problem in our case can be recast as an effort to find the parameters $\boldsymbol{\theta}_{ML}$ that maximizes the log-likelihood function (2.10) or equally minimizes the negative log-likelihood.

Gradient descent takes steps in the direction of the gradient, which also is the direction of steepest descent. If we let $\mathcal{D} = \{\boldsymbol{w}_i\}_{i=1}^{n}$ be our set of sentences and $-\ell$ be the negative log-likelihood that we are trying to minimize then the updates are of the form

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} (-\ell(\boldsymbol{\theta}; \boldsymbol{x}_i)) \tag{2.22}$$

, where γ is the learning rate of the algorithm.

Stochastic Gradient Descent is a similar algorithm to GD that instead of calculating the gradient with respect to the whole dataset calculates an approximate gradient, as it only takes a subset of the data into account at each update. If we let I_t be a random subset of the indices of $\{(\boldsymbol{w}_n)_{n=1}^{N}\}$ of size m then SGD does the following update

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \frac{1}{m} \sum_{i \in I_t} \nabla_{\boldsymbol{\theta}} (-\ell(\boldsymbol{\theta}; \boldsymbol{x}_i)) \tag{2.23}$$

[31][7, p. 240].

2.4.2 ADAM

SGD has found widespread use within the machine learning community due to strong experimental results and ease of use, especially in deep learning. However there are notable alternatives that try to improve on SGD such as RMSProp[32] and AdaGrad[33].

Adam takes inspiration from RMSProp and AdaGrad. Technically, Adam keeps an exponential running average of the first and second order statistics of the gradient, using these to calculate an adaptive learning rate.

As laid out in the original paper by Kingma et. al [34], ADAM operates on a stochastic objective function $f(\boldsymbol{\theta})$, which in our case would be the probability averaged over our input batch. $f_t(\boldsymbol{\theta})$ denotes this function over different time steps, equally $g_t = \nabla_{\boldsymbol{\theta}} f_t(\boldsymbol{\theta})$. m_t and v_t denotes the first and second order moments of the gradients, however, these are biased and are corrected in the algorithm. The whole algorithm is as follows.

Algorithm 1 ADAM

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates of the moment estimates
Require: $f(\boldsymbol{\theta})$: Stochastic objective function with parameters $\boldsymbol{\theta}$
Require: $\boldsymbol{\theta}_0$: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1$^{\text{st}}$ moment vector)
$v_0 \leftarrow 0$ (Initialize 2$^{\text{nd}}$ moment vector)
$t \leftarrow 0$ (Initialize timestep)
while $\boldsymbol{\theta}_t$ not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_{\boldsymbol{\theta}} f_t(\boldsymbol{\theta}_{t-1})$ (get gradients w.r.t stochastic objective at timestep t)
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
  $\hat{m}_1 \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
  $\hat{v}_1 \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \cdot \hat{m}_i / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
return $\boldsymbol{\theta}_t$ (Resulting parameters)

Experimentally Adam has shown very good results on training various deep learning models such as MLP's, CNN's and RNN's so we will use it to train our models throughout this dissertation [34].

2.4.3 Automatic Differentiation

The AutoDiff framework is a general theory for efficiently calculating gradients with respect to a loss function in general computational graphs. It includes the famed backpropagation [7, 35] algorithm as a special instance.

In machine learning we most often deal with many-to-one mappings with a high-dimensional input $\boldsymbol{x}$ to a scalar output in the form of a loss. In terms of machine learning models, AutoDiff takes a loss function $L(\boldsymbol{\theta})$ and returns an exact value up to machine accuracy for the gradient $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})|_{\boldsymbol{\theta}}$. There are two different modes of AutoDiff, forward and reverse, although in practice reverse is preferred.

Reverse mode AutoDiff is efficient in the way that calculating the gradient of $L(\boldsymbol{\theta})$ is guaranteed to take at most 5 times the time it takes to compute $L(\boldsymbol{\theta})$ [36], enabling neural network libraries to calculate the gradients needed for optimization algorithms such as SGD and ADAM in a swift and automatic manner.

Chapter 3

Methods and Theory

This chapter deals with the theory we will be using in order to use VAE in the our setting. It will also deal with extending the theory and necessary calculations.

3.0.1 Variational Inference

Consider the following general graphical model of a latent variable z and an observed variable $\boldsymbol{x}$:

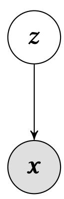


Figure 3.1: Latent variable model

implying the joint distribution

$$p(\boldsymbol{x}, z) = p(\boldsymbol{x}|z)p(z) \tag{3.1}$$

In order to fit the parameters of the model using MLE we need to be able to to find the gradient and value of $\ell(\boldsymbol{\theta})$ to use ADAM. We can rewrite the likelihood for a datapoint $\boldsymbol{x}$ as

$$\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}) = \int_X p(\boldsymbol{x}, z) \,\mathrm{d}X \tag{3.2}$$

by using the sum rule 2.1.3. For many models, this integral is either not available in a closed form or requires exponential time to compute [3] and while for some special cases it is possible to find a local optimum of the likelihood by using the EM algorithm [37] in general it is too restrictive since we need to be able to find $p(z|\boldsymbol{x})$ analytically.

While it is possible to use sampling based techniques such as MCMC [38] to sample from $p(z|\boldsymbol{x})$ in practice it is often slow and depending on the problem may be less than ideal. Especially for problems where we use complex models and have large datasets it's not possible in practice to use MCMC.

Variational inference approaches the problem of optimizing the intractable log-likelihood by introducing a variational distribution q belonging to some constrained set of predefined distributions $\mathcal{Q}$. Since the log-likelihood is intractable, we try to bound the log-likelihood from below by an expression that is easier to deal with. Using Jensen's inequality with the $\log(\cdot)$ function, we have that

$$
\begin{aligned}
\log p_\theta(\boldsymbol{x}) &= \log \int_{\mathcal{Z}} p_\theta(z, \boldsymbol{x}) \, \mathrm{d}z \\
&= \log \int_{\mathcal{Z}} q_\varphi(z) \frac{p_\theta(\boldsymbol{x}, z)}{q_\varphi(z)} \, \mathrm{d}z \\
&= \log \mathbb{E}_{q_\varphi(z)} \left[\frac{p_\theta(\boldsymbol{x}, z)}{q_\varphi(z)} \right] \\
&\geq \mathbb{E}_{q_\varphi(z)} \left[\log \frac{p_\theta(\boldsymbol{x}, z)}{q_\varphi(z)} \right] \\
&= \mathbb{E}_{q_\varphi(z)} \left[\log p_\theta(\boldsymbol{x}, z) - \log q_\varphi(z) \right]
\end{aligned}
$$

Defining the term Evidence Lower Bound Objective,

$$
\mathrm{ELBO}(q_\varphi) = \mathbb{E}_{q_\varphi(z)} \left[\log p_\theta(\boldsymbol{x}, z) - \log q_\varphi(z) \right] \tag{3.3}
$$

we express this as the following resulting inequality

$$
\log p_\theta(\boldsymbol{x}) \geq \mathrm{ELBO}(q_\varphi) \tag{3.4}
$$

. However, we note that we can rewrite the ELBO as

$$\mathrm{ELBO}(q_{\varphi})) = \log p_{\theta}(\boldsymbol{x}) - D_{KL}(q_{\varphi}||) \tag{3.5}$$

3.1 ELBO

Ideally we would like to optimize the log-likelihood of the data, $\ell(\boldsymbol{\theta}; \mathcal{D})$. For many models, it is not possible to evaluate this quantity directly due to it being computationally and/or analytically intractable.

Instead we make the following observation, letting $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{N}$

$$\log p_{\theta}(\mathcal{D}) = \sum_{i=1}^{N} \log p_{\theta}(\boldsymbol{x}_i) \tag{3.6}$$

and looking at each term in the sum we can see that we may rewrite this as

$$\begin{aligned}
\log p_{\theta}(\boldsymbol{x}_i) &= \log \int_{\mathcal{Z}} p_{\theta}(z, \boldsymbol{x}_i)\, \mathrm{d}z \\
&= \log \int_{\mathcal{Z}} q_{\varphi}(z|\boldsymbol{x}_i) \frac{p_{\theta}(\boldsymbol{x}_i, z)}{q_{\varphi}(z|\boldsymbol{x}_i)}\, \mathrm{d}z \\
&= \log \mathbb{E}_{q_{\varphi}(z|\boldsymbol{x}_i)} \left[\frac{p_{\theta}(\boldsymbol{x}_i, z)}{q_{\varphi}(z|\boldsymbol{x}_i)} \right] \\
&\geq \mathbb{E}_{q_{\varphi}(z|\boldsymbol{x}_i)} \left[\log \frac{p_{\theta}(\boldsymbol{x}_i, z)}{q_{\varphi}(z|\boldsymbol{x}_i)} \right] \\
&= \mathbb{E}_{q_{\varphi}(z|\boldsymbol{x}_i)} \left[\log \frac{p_{\theta}(\boldsymbol{x}_i|z) p_{\theta}(z)}{q_{\varphi(z|\boldsymbol{x}_i)}} \right] \\
&= \mathbb{E}_{q_{\varphi}(z|\boldsymbol{x}_i)} \left[\log p_{\theta}(\boldsymbol{x}_i|z)] - D_{KL}(q_{\varphi}(z|\boldsymbol{x}_i)||p_{\theta}(z)) \right]
\end{aligned}$$

Which leads to the following observation, if we call the lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}; \boldsymbol{x})$, that

$$\log p_{\theta}(\boldsymbol{x}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}; \boldsymbol{x}) = \mathbb{E}_{q_{\varphi}(z|\boldsymbol{x})}[\log p_{\theta}(\boldsymbol{x}|z)] - D_{KL}(q_{\varphi}(z|\boldsymbol{x})||p_{\theta}(z)) \tag{3.7}$$

.

In general this lower bound is still not tractable, however appealing to the Strong Law of Large Numbers (SLLN) we can sample $\boldsymbol{z}^l$ and use the reparametrisation trick (Include how the reparametrisation trick works) to approximate the lower bound by

$$\tilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\varphi}; \boldsymbol{x}) = \frac{1}{L}\sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, z) - \log q_{\boldsymbol{\varphi}}(z|\boldsymbol{x}) \tag{3.8}$$

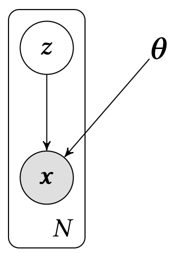
. This can also be expressed in the closed form of the KL so

$$\tilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\varphi}; \boldsymbol{x}) = (\frac{1}{L}\sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)) - D_{KL}(q_{\boldsymbol{\varphi}}(z|\boldsymbol{x})||p_{\boldsymbol{\theta}}(z)) \tag{3.9}$$

Since all quantities in this form have analytically known forms, it is simple to evaluate and sample from $\tilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\varphi}; \boldsymbol{x})$.

3.2 Models

The generative model follows the one laid out in [4]:



where z is the latent variable representing the latent sentence structure and the joint factorizes as $p_{\boldsymbol{\theta}}(\boldsymbol{x}, z) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)p(z)$. In all models we let $p(z) = \mathcal{N}(z|\mathbf{0}, \boldsymbol{I}_{D\times D})$.

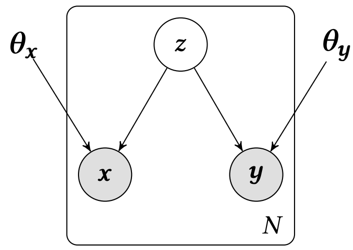
3.2.1 Reconstruction

The reconstruction model follows the original model. We assume a latent variable z that generates sentences $\boldsymbol{x}$ belonging to a language $\mathcal{X}$.

The ELBO and related calculations follow the ones laid out in the ELBO section.

3.2.2 Translation

Translation adds and extra observed variable $\boldsymbol{y}$, a sentence belonging to $\mathcal{Y}$. The probabilistic model looks like:



From the graphical model we see that the joint distribution factorises such that

$$p(z, \boldsymbol{x}, \boldsymbol{y}) = p(z)p(\boldsymbol{x}|z)p(\boldsymbol{y}|z) = \mathcal{N}(0, I)\mathcal{N}(\mu_{\theta_x}(z), \sigma_{\theta_x}(z))\mathcal{N}(\mu_{\theta_y}(z), \sigma_{\theta_y}(z)) \tag{3.10}$$

. The notation $\mathcal{N}(\mu_\theta(z), \sigma_\theta(z))$ means that the mean and variance diagonal of the normal are represented by a functional relationship $f : \boldsymbol{z} \mapsto (\mu, \sigma)$ where f is a function represented a a learnable neural network with parameters $\boldsymbol{\theta}$.

The recognition model is parametrised by the probability distribution $q_\varphi(z|x, y)$. We are free to choose this however we like, but the closer this is to the actual conditional distribution over the latent, $p(z|x, y)$ the tighter the ELBO inequality $\ell(x, y) \geq \mathcal{L}(x, y)$ will be. We let the distributions all be gaussians to have analytical tractability

$$\begin{aligned} q(z|x, y) &= q(z|x)q(z|y) \\ &= N(\mu(x), \sigma(x)I)N(\mu(y), \sigma(y)I) \end{aligned}$$

where both normals defined by the output of neural networks.

Since the product of Guassian pdf's are gaussian itself, we have that may write that

$$q(z|\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}(z|\mu(\boldsymbol{x}, \boldsymbol{y}), \sigma(\boldsymbol{x}, \boldsymbol{y}))$$

which reduces to the expressions 2.5 for the mean and 2.4 for the covariance.

Pulling all of these statements together, we get that the SGVB of the translation case reduces to the equation

$$\tilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\varphi}; \boldsymbol{x}) = (\frac{1}{L}\sum_{l=1}^{L} \log(\mathcal{N}(\boldsymbol{x}|\mu_{\boldsymbol{\theta}_y}(z), \sigma_{\boldsymbol{\theta}_y}(z))) + \log(\mathcal{N}(\boldsymbol{y}|\mu_{\boldsymbol{\theta}_y}(z), \sigma_{\boldsymbol{\theta}_y}(z))))+ \\ \frac{1}{2}\sum_{j=1}^{J}(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2)) \tag{3.11}$$

where the μ, σ is from the normal $\mathcal{N}(z|\mu(\boldsymbol{x}, \boldsymbol{y}), \sigma(\boldsymbol{x}, \boldsymbol{y}))$.

3.2.3 Training

Variational Autoencoders in NLP have a tendency to collapse the KL term in the SGVB objective to zero, effectively being reduced a RNNLM (CITE PAPER). As the KL can be seen as a measure of how much information is encoded in $\boldsymbol{z}$ this event makes the use of VAE's to model language-agnostic features in the latent space useless. One of the objectives is to make sure this KL-collapse doesn't happen.

Following the tips laid out in (CITE PAPER VAE FOR NLP) we use KL-annealing where we anneal the ELBO by a pseudo-objective

$$ELBO_{\beta_t} = \mathbb{E}_{q_{\boldsymbol{\varphi}}(z|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)] - \beta_t * D_{KL}(q_{\boldsymbol{\varphi}}(z|\boldsymbol{x})||p_{\boldsymbol{\theta}}(z)) \tag{3.12}$$

where β_t is an annealing term going from 0 to 1 as we train. We do this by choosing a warm-up β and let $\beta_t = \min(1, \frac{t}{\beta})$, letting this warm up be done linearly until we recover the original ELBO objective, similar to normal simulated annealing (CITE ANNEALING PAPER).

Chapter 4

Experiments

4.1 Data

4.1.1 Dataset

The dataset we have chosen to evaluate the models on is the Europarl dataset between languages English and French. Europarl is a dataset of the proceedings of the European Parliament, comprising in total of the 11 official languages of the European Union.

The dataset was chosen as the number of sentences for English and French is enough to be able to generalise (the uncompressed size of the full dataset is 619MB, 288MB for English, 311MB for French) to new sentences, and furthermore has established baseline for NMT in the form of BLEU scores for all the different language pairs in the full dataset, English-French in particular.[39]

4.1.2 Preprocessing

The raw data is unfit for use directly with the model. For one thing the raw data is in the form of strings and in order to use neural networks we need to transform this into a form using vectors in some space $\mathbb{R}^m$.

From a linguistical point of view, a language contain a huge number of words at any point in time. Furthermore, words never leave the language completely, rather they go in and out of fashion and differ in how commonly they are use, also depending on what context they are used in.

English: I declare resumed the session of the European Parliament adjourned on Friday 17 December 1999, and I would like once again to wish you a happy new year in the hope that you enjoyed a pleasant festive period. **French**: Je déclare reprise la session du Parlement européen qui avait été interrompue le vendredi 17 décembre dernier et je vous renouvelle tous mes vux en espérant que vous avez passé de bonnes vacances.

Table 4.1: A randomly sampled sentence from the Europarl corpus

- We specify the maximum and minimum length of the words. This means that when training and testing, we can pad the shorter sentences with Null tokens until they are of the same length as the longest sentence.
- We calculate the word frequencies in order to sort all of the words in the dataset in terms of how often it appear in absolute terms. This is then used to only retain the 30000 most common words. Words which are not part of this list gets replaced by an `<UNK>` token, specifying that it's an unknown word outside of the dictionary. This makes sure that only words which are prevalent enough such that the model can derive its relation to other words are part of the dictionary.
- Newline characters were removed and replaced by `<EOS>`, end-of-sentence tokens, signifying the end of a sentence.
- In parts where we have a dataset of 2 or more language sentences in parallel, we make sure that both of the the languages both satisfy the above criteria.

It is important to note that due to how languages differ, even though the dataset might consist of sentence pairs this will still not mean that in general both of the languages will have the same dictionary of words. Partially this is due to the different ways that languages are built up when expressing meaning, but on a more basic level, there are no bijection between languages as words have slightly different meaning and contexts, with some words only existing in one language but not the other.

4.2 Scores

We will evaluate our models on a variety of scores:

ELBO ELBO is the lower bound of the actual log-likelihood of the observed data

$$\sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}_i)$$

Qualitative Since natural language is not a formal in the sense that it is ambiguous, inconsistent and with exceptions to rules; any of these scores will be imperfect insofar as taking into account the feel of the generated sentences. Due to this we will inspect the sentences manually.

BLEU BLEU compares the generated sentences with sentences translated by professional translators, yielding a score telling us how well the generated translation does in relation to the translated benchmarks for each sentence.

We follow the thing as laid out in this link http://www.statmt.org/wpt05/mt-shared-task/#TEST.

KL Part of our investigation is about building models that take into account the latent space, enforcing the model to encode the information in the latent variable z instead of the encoder/decoder part. Luckily, we have a quantitative measure of this, the KL divergence between the prior and the posterior q-distribution over z,

$$KL[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})]$$

, where the KL is a measure of how much information is put into the q-distribution compared to just using the prior isotropic gaussian over $\boldsymbol{z}$, $p_\theta(\boldsymbol{z})$.

Perplexity After training the model we want the sentences in the test set to be probable under the probability distribution parametrised by $\boldsymbol{\theta}$. Perplexity is one measure of model fit. Given a sentence $\boldsymbol{w}_{1:L}$ the perplexity, PP, is defined as the per-word inverse probability,

$$\mathrm{PP}(\boldsymbol{w}_{1:L}) = p(\boldsymbol{w}_{1:L})^{-\frac{1}{L}} \tag{4.1}$$

. In practice, this can be expressed in the form

$$\mathrm{PP}(\boldsymbol{w}_{1:L}) = \exp(-\frac{1}{L}\prod_{l=1}^{L}\log p(\boldsymbol{w}_l|\boldsymbol{w}_1,\ldots,\boldsymbol{w}_{l-1})) \tag{4.2}$$

4.3 Reconstruction

We ran a couple of experiments on reconstruction as a proof of concept to show that the model worked on data with only one language.

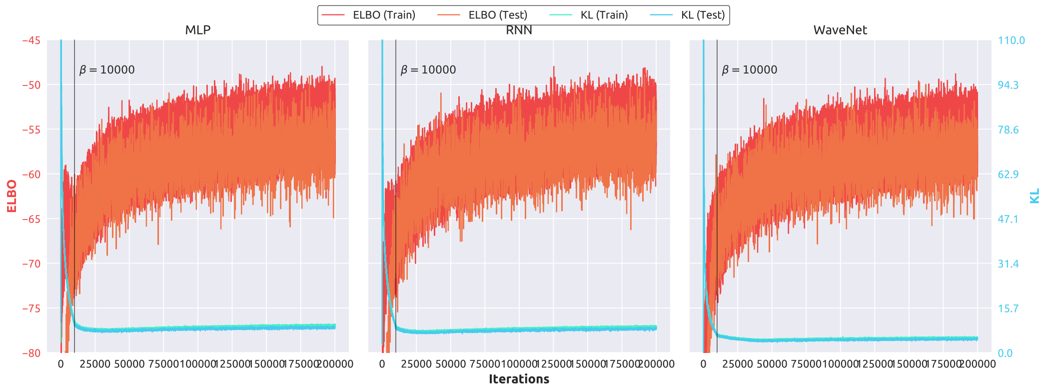


Figure 4.1: Runs with various different recognition model

4.4 Translation

Chapter 5

Conclusions

Possible extensions:

The Authors of WaveNet has introduced ByteNet, might be worthwhile looking into that as well as it is explicitly created to work well on character level generative models (https://arxiv.org/pdf/1610.10099.pdf).

Kingma et. al has come with a new extension to VAE's which uses normalising flows https://arxiv.org/pdf/1606.04934.pdf.

Appendix A

Proofs

These are the proofs for some of the results.

Appendix B

Another Appendix About Things

Appendix C

Colophon

This is a description of the tools you used to make your thesis. It helps people make future documents, reminds you, and looks good.

(example) This document was set in the Times Roman typeface using LaTeX and BibTeX, composed with a text editor.

Bibliography

[1] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

[2] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[3] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017. arXiv: 1601.00670.

[4] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, December 2013. arXiv: 1312.6114.

[5] D. Jimenez Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ArXiv e-prints*, January 2014.

[6] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. *arXiv:1511.06349 [cs]*, November 2015. arXiv: 1511.06349.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, New York, NY, USA, 2012.

[9] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007.

[10] George Casella and Roger Berger. *Statistical Inference*. Duxbury Resource Center, June 2001.

[11] Yoav Goldberg. A primer on neural network models for natural language processing, 2015. cite arxiv:1510.00726.

[12] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989.

[13] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989.

[14] David Barber. Lecture notes in applied machine learning, September 2016.

[15] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *CoRR*, abs/1706.02515, 2017.

[16] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, May 2015.

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

[18] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[19] Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here. In *Proceedings of the IEEE*, page 2000, 2000.

[20] Lenhart Schubert. Computational linguistics. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2015 edition, 2015.

[21] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[24] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.

[25] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[26] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[27] Krzysztof Wo lk and Krzysztof Marasek. Neural-based machine translation for medical text domain. Based on European Medicines Agency leaflet texts. *Procedia Computer Science*, 64:2–9, 2015. arXiv: 1509.08644.

[28] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv:1609.08144 [cs]*, September 2016. arXiv: 1609.08144.

[29] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv:1409.1259 [cs, stat]*, September 2014. arXiv: 1409.1259.

[30] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016.

[31] Léon Bottou. Stochastic gradient descent tricks. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade (2nd ed.)*, volume 7700 of *Lecture Notes in Computer Science*, pages 421–436. Springer, 2012.

[32] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[33] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010.

[34] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. arXiv: 1412.6980.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[36] David Barber. Deep learning: Autodiff, parameter tying and backprop through time, 2015.

[37] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[38] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

[39] Philipp Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT, AAMT.