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

July 25, 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

Chapter 1

Introduction

Natural Language Processing, hereafter called NLP, is a subfield within artificial intelligence almost as old as the field itself. Briefly, NLP can be defined as the study of the properties of natural language and how these properties may be used to answer questions that humans who master the language can do naturally. Clearly, if we are to enable machines to cooperate with human beings, it is of utmost importance that they can speak our language, since we are not very apt in speaking theirs!

NLP has a plethora of different subfields, however, in this thesis we will limit ourselves to the field of machine translation. Machine translation has long been a cornerstone of NLP and has undergone many different guises from the beginning of the 1940's until today.

Machine translation can essentially be seen as a problem of learning a model that maps from one language $\mathscr{X}$ to another language $\mathscr{Y}$. Formally, we have a dataset $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, such that $\boldsymbol{x}_i \in \mathscr{X}, \boldsymbol{y}_i \in \mathscr{Y}$ represent the same sentence in two different languages. The goal is then to find a mapping $f : \mathscr{X} \to \mathscr{Y}$ such that $f(\boldsymbol{x}) \approx \boldsymbol{y}$ with regards to some chosen metric.

From this several different ways of looking at language has emerged. The two different ways depends on how granular you are going. For latin based languages and other languages with a phonetic writing system, this means deciding if the atomic symbols should be characters of words. Character level gives the advantage that your dictionary of characters i finite, it consists of your alphabet. Since it's

finite there's less of a worry of new characters being introduced. The data also gets bigger. For words you have a problem that the dictionary swells, often being bigger than 100000 unique words, which are troublesome when it comes to calculate the normalizer of the softmax in order to get a distribution of the output from the raw model.

From the chain rule of probability we have that any sentence $\boldsymbol{x}$ is such that the joint probability of the atomic units may be rewritten in a recursive form,

$$\begin{aligned} p(\boldsymbol{x}) &= p(x_1, \ldots, x_n) \\ &= \prod_{i=1}^{n} p(x_i | x_{i-1}, \ldots, x_1) \end{aligned}$$

, which shows that one way of modelling language is to try to capture this temporal relationship using models catered for long term dependencies. The main problem is one of scale. This can be seen from the Markovian models called *N-grams* which simply assumes that the dependency in languages can be explained by N-dimensional tables, meaning that N-grams can be seen as N-step markov models applied to language data. In practice, this means that we assume the relationship

$$p(x_i | x_{i-1}, \ldots, x_1) = p(x_i | x_{i-1}, \ldots, x_{i-N})$$

.

Although this is a reasonable assumption from a modelling point of view, given that we take N to be large enough, it doesn't work well in practice. Due to sparsity, these N-dimensional tables will be mostly filled with zeros. This is simple to see whether working on a word or character level, I will just give an example using words for maximum impact:

Assume we are trying to model the english language using a Tri-gram model, meaning we are only considering grouped word of 3. Putting together random words from the dictionary of words in the English language gives us by a huge margin gibberish, in terms of both semantic meaning and of grammar:

moucharaby epithelium sonlike
Bacchides lulliloo oneiromancer
actinology dihydroxy nonmineralogical
Homalonotus Vened dyspepsy
uncessant twee femorofibular

. This is due to the fact that only extremely few tuples of word triplets are actually valid in the sense that they can be said to exist naturally in the English language. Mathematically, this means that the tables we get from running maximum likelihood on these tables to find the actual probabilities, which is just a matter of counting the number of times the triplets occur compared to the number of times that the starting symbol occur in the corpus we have at hand.

Neural Machine Translation, hereafter called NMT, is the use of Neural Networks as the models inside the machine translation systems. NMT can be trained end-to-end by specifying the data, architecture and the various other components that make up the model specification. While NMT is very data-hungry, mostly getting its power from being able to unearth the various rules and constructs in a language (semantically and grammatically) through the use of the statistical information existing within it, it is extremely well-suited for learning these rules given enough data. This black box approach means that people without any knowledge of language $\mathscr{X}$ and $\mathscr{Y}$ can train sophisticated translation systems on par with state of the art results, solely relying on the data to speak for itself.

In this thesis we will explore fully probabilistic models, meaning that any statement about output languages can be given a probability score. Using the language of probability enables use to make statements about the plausibility of sentences and logical statements about these sentences using the laws of probability. Practically, it means that we get a model which is generative, that is, we can sample random variables of the hidden variable that encodes the sentences in a language-agnostic which through the models can map back to the sentences in the original languages. This gives us some hope that the model have some kind of internal language model and not only learns the specific output relation when mapping from $\mathscr{X}$ to $\mathscr{Y}$.

In essence, we will use recent techniques that enables us to train latent-variable models, that is models where the observed output, in our case the language sentence tuples depend on a hidden variable $\boldsymbol{z}$ that encodes the information about the sentence in a language-agnostic way. Consider the sentence *The quick brown fox jumps over the lazy dog*. The sentence is written using the English language, but it's easy to imagine the if we disregard the language we are saying it in, whether it be German, *Der schnelle braune Fuchs sprang über den faulen Hund* or in Latin, *Lorem ipsum vulpes salit super piger canis*, there is some underlying meaning which all languages are trying to convey. Our model tries to encode this meaning in terms of a hidden stochastic variable $\boldsymbol{z}$.

This leads to the realisation given that we can encode $\boldsymbol{z}$ properly we should be able to translate from language $\mathscr{X}$ to $\mathscr{Q}$ without the model having ever seen a sentence pair of the form $(\boldsymbol{x},\boldsymbol{q}),\boldsymbol{x}\in\mathscr{X},\boldsymbol{q}\in\mathscr{Q}$.

1.1 Things to talk about

- NLP (History, challenges)
- Deep Learning (What it is)
- Neural Machine Translation

Chapter 2

Background Knowledge

From here on I will assume familiarity with some concepts which will be important for the experiments that we will conduct and analyse.

2.1 Probability Theory and Statistics

2.1.1 Probability Theory as a Reasoning System

Although often seen as a self-contained mathematical theory of stochastic systems and reasoning about the random, probability theory has a prominent role within machine learning since it gives us a principled way of reasoning about the world. As proved by Cox, if we are to accept that any theory of reasoning is to satisfy the Cox desiderata,

1. Degrees of plausibility are represented by real numbers.

2. Qualitative correspondence with common sense

3. If a conclusion can be reasoned out in more than one way, then every possible way must lead to the same result.

, then we must accept that this theory is isomorphic to probability theory and effectively the same.

This is a formal way of stating that probability theory is the best way to organize our reasoning if we want to make sure that we are consistent in the way we reason.[1, p. 3-23]

2.1.2 Rules and Theorems

Rules of probability that will be useful to us are the following axioms

Unit volume

$$\int_{\mathscr{X}} p(X) = 1 \tag{2.1}$$

Non-negativity

$$p(X) \geq 0 \tag{2.2}$$

where $\mathscr{X}$ is the domain of X and the generalized integral is interpreted as the Lebesgue integral if X is continuous and as a sum over the possible values of X if it is discrete.

I will assume the notion of a random variable X in an intuitive sense. However, this X might be represented in many forms and in our case there is not reason to not be able to put probabilities over sentences, words and/or characters and conditioning thereof. Similarly I will assume familiarity with the different notions of continuous, discrete and categorical random variables. Finally

Most manipulation of statements about random variables can be stated as a consequence of the two following fundamental rules

Sum rule

$$p(X) = \int_{\mathscr{Y}} p(X,Y) \tag{2.3}$$

Product rule

$$p(X,Y) = p(Y|X)p(X) \tag{2.4}$$

such that X, Y are two random variables defined on the domains $\mathscr{X}, \mathscr{Y}$. The integral $\int_{\mathscr{Y}}$ is to be understood in the general sense, if Y is continuous it is the ordinary Lebesgue integral as commonly used throughout mathematics and calculus, while if Y is discrete then it is the sum over the possible values of Y. $p(X,Y)$ is the joint distribution of X and Y, $p(Y|X)$ is the probability of Y conditioned on X and $p(X)$ is the marginal distribution of X. Using these rules it is easy to Bayes theorem, one of the most integral (and simple) theorem of probability theory

Bayes Theorem

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} \tag{2.5}$$

Two very important operations involving probabilities of random variables are those of *Expectation* and *Covariance*. These take as input a function f and maps to the real number line $\mathbb{R}$, and are defined implicitly with regards to some random variable X and its probability distribution $p(X)$.

Expectation

$$\mathbb{E}_X[f] = \int_{\mathscr{X}} p(x)f(x) \tag{2.6}$$

Covariance

$$\text{Cov}(X,Y) = \mathbb{E}_{XY}[(X - \mathbb{E}_X[X])(Y - \mathbb{E}_Y[Y])] \tag{2.7}$$

We then define the variance operator as

$$\text{Var}(X) = \text{Cov}(X,X) \tag{2.8}$$

The generalisation from $f : \mathscr{X} \to \mathbb{R}$ to $f : \mathscr{X} \to \mathbb{R}^n$ is defined in the straightforward manner such that if $\boldsymbol{f} = f(X)$ then

$$\mathbb{E}_X \begin{bmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_n \end{bmatrix} = \begin{bmatrix} \mathbb{E}_X \boldsymbol{f}_1 \\ \vdots \\ \mathbb{E}_X \boldsymbol{f}_n \end{bmatrix}$$

similarly $\text{Cov}(\boldsymbol{f})$ is a $D \times D$-dimensional matrix where $\text{Cov}(\boldsymbol{f})_{i,j} = \text{Cov}(\boldsymbol{f}_i, \boldsymbol{f}_j)$.

2.1.3 The Gaussian Distribution

A very common distribution in machine learning and statistics in general is the Gaussian distribution, also called the Normal distribution. The Gaussian distribution satisfies some properties that makes it an ideal candidate from a modeling perspective including the Central Limit Theorem which says that sums of independent random variables of finite mean and variance will tend to a normal distribution,

and the fact that the joint distribution of many random variables that are Gaussianly distributed is itself Gaussian which means that conditionals, posteriors and other distributions are themselves Gaussian when we deal with Gaussian random variables.

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

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{2.9}$$

where $\boldsymbol{\mu}$ is a D-dimensional mean vector, $\boldsymbol{\Sigma}$ is a $D \times D$ dimensional covariance matrix, and $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$. The meaning of these parameters can be shown to correspond to the operations of the expected value and covariance of $\boldsymbol{x}$, $\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}] = \boldsymbol{\mu}$ and $\mathrm{Cov}(\boldsymbol{x}) = \boldsymbol{\Sigma}$.

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 $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ 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 $\boldsymbol{\Lambda}\boldsymbol{\Lambda} = \boldsymbol{\Sigma}$ be the Cholesky decomposition[2, p. 100-102] of $\boldsymbol{\Sigma}$, then we also have that

$$p(\boldsymbol{x}) = p(\boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{z})$$

. If we further assume that $\boldsymbol{x}$ is parametrised by $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ such that $\boldsymbol{\Sigma}$ is diagonal positive definite with diagonal $\boldsymbol{\sigma}$, then $\boldsymbol{\Sigma} = \boldsymbol{\sigma} \odot \boldsymbol{I}_{D\times D}$. 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

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma} \odot \boldsymbol{I}_{D\times D}) \tag{2.10}$$

2.1.4 Maximum Likelihood Estimation

The methods that we will use in our experiments stem from the tradition of *Bayesian Statistics*. In short, in Bayesian statistics we assume that the parameters we are trying to find are not fixed as is common to assume when thinking of probability state-

ments as long-term frequencies of events. Instead we assume that probabilities are statements about our *beliefs* of the parameter or random variable we are interested in.

In our case, we won't use the full Bayesian machinery, we will instead be interested in doing *Maximum Likelihood* optimization which means that if we let $\mathscr{D} = \{(x_i, y_i)_i^n\}$ be our dataset and θ be all the parameters of our model of the distribution of the data and Θ be the domain of the model parameters, then maximum likelihood estimation in the case of supervised learning is formulated as the following problem of finding the model parameters

$$\theta_{ML} = \underset{\theta \in \Theta}{\operatorname{argmax}}\, p(y_1, \ldots, y_n | x_1, \ldots, x_n) \tag{2.11}$$

In some way this is the optimal setting of θ without prior knowledge of how they are distributed. It takes all the data available into account and reduces to finding the θ that maximizes the *likelihood*, that is the statement $p_\theta(X)$ seen as a function in θ.

2.2 Deep Learning

A ubiquitous classifier within statistics is logistic regression. Logistic regression uses an input vector $\boldsymbol{x}$ in order to give importance scores in form of probabilities to different classes $y \in \{c_1, \ldots, c_k\}$. It gets its name from the logistic function

$$\sigma(a) = \frac{1}{1 + e^a} \tag{2.12}$$

which together with an affine transformation $\boldsymbol{W}\boldsymbol{x}$ yields the layer

$$\sigma(\boldsymbol{W}\boldsymbol{x})$$

which transforms values from a feature space $\boldsymbol{X} \subset \mathbb{R}^m$ into probabilities[3].

Deep learning builds upon this intuition by recursively applying transformations and activation functions, functions which in some sense maps input on to

ON/OFF states. These functions take their functionality from an abstraction from how neurons function when firing with regards to input, mirroring how artificial neural networks have taken inspiration from how the brain operates in the past. On a very basic level, neural networks are characterised by stacked layers of affine transformations followed by activation functions, where the output of one layer serves as the input to the next layer. The final layer outputs $\hat{y}$ where the form of $\hat{y}$ depends on the application. The hope is that after training the model using backpropagation[4] that the model is able to predict satisfactory and drive down the specified loss.

Deep models are very powerful in that they are able to model complex functional relationships. In our case we are looking at Supervised and Semi-supervised learning, trying to find the relationship between $\boldsymbol{x} \in bmX$ and $\boldsymbol{y} \in \boldsymbol{Y}$ of some kind of functional form $f(\boldsymbol{x}) \approx \boldsymbol{y}$.

Besides from the straightforward models where we stack logistic regressors serially, neural networks have extended well beyond this into an extremely diverse set of models that can capture different aspects of data such as long term-dependencies through the architecture of Recurrent Neural Networks and invariances by using convolutions. Many of these models have also found use in NLP, especially in the form of RNN's which are well-suited for handling language due to how it enables information to flow through time[5][6] and more recently CNN's for finding representation over many different scales[7][8][9].

In a Bayesian setting each graphical model codifies how different random variables relate to each other in terms of independency. This is specified by the Directed Acyclic Graph where each arrow signifies a conditional relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$. A full description of how graphical models ,

2.3 Approximate Inference

In a bayesian setting we parametrise a model $\mathscr{M}$ by the parameters $\theta_{\mathscr{M}}$ which we will call θ since our model is fixed. The goal is to find the parameters that maximise the likelihood,

$$\mathscr{L}(\theta; \mathscr{X}) = p_{\theta}(\mathscr{X}) \tag{2.13}$$

.

Contrary to a normal probability which is a function of the random variable $\mathscr{X}$, the likelihood instead acts as a function of θ, hence why it's often written in the form $\mathscr{L}(\theta;\mathscr{X})$. However, since the likelihood often deals with products due to the iid assumption of the data, in practice it is much more common to use the *log-likelihood*

$$\ell(\theta;\mathscr{X}]) = \log \mathscr{L}(\theta;\mathscr{X}) \tag{2.14}$$

, since this turns the products when assuming iid data into sums

$$\log p_\theta(X_1,\ldots,X_n) = \log \prod_{i=1}^{n} p_\theta(X_i) = \sum_{i=1}^{n} \log p_\theta(X_i)$$

.

Beside for ease of notation, this also have the following advantage that when dealing with floating point arithmetic, using log-likelihoods we have a much smaller chance of over/underflowing. Arithmetic underflow and overflow stems from the fact that the computer only deals with floating point numbers, which has a smallest number that it can represent, called machine epsilon ($\varepsilon_{machine}$), and a biggest number that it can represent. When multiplying a lot of small numbers together as is common in probability theory, the resulting number might be smaller than $\varepsilon_{machine}$ which will then be rounded down to 0.0.

For Maximum Likelihood learning which is the way that we ideally want to train models in the

2.4 Natural Language Processing

Chapter 3

Methods and Theory

Here we build on the theory laid out in Background Knowledge, and take it further, and tell how we use it for our experiments.

Chapter 4

Experiments

4.1 Data

4.1.1 Dataset

The dataset we have chosen to evaluate the model 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.[10]

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 leverage the mathematics easily we need to translate the raw form into a form which take place in a high-dimensional space instead, here $\mathbb{R}^N$. Equally we remove aspects of the data that will make it harder for the model to learn due to sparsity and other statistical peculiarities of the data and NLP in general.

Preprocessing the data we make the following simplifications

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

4.1.2.1 Character Level

4.1.2.2 Word Level

The problem with words is that there exist an immense quantity of them, if even just due to grammatical constructs (example: run, running, ran etc.). Similarly, for any given point in time, words go in and out of use and this necessitates choosing which words to include in the dictionary. The dictionary consists of all of the words that we consider part of the language, everything not in the dictionary are either too rare or for some other reason excluded from use.

- We only include sentences of length between 2 and 30. This makes sure that the model have long enough sentences such that it may learn from the dependencies between words, but short enough so that the parameters are able to capture the long-term dependencies of sentences.

- 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 80000 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.

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 $\boldsymbol{z}$,

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

, 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 $\boldsymbol{z}$, $p_\theta(\boldsymbol{z})$.

4.3 layout

We will perform the following experiments, building up the order of doing them from least complex to more complex.

We first have different models, depending on how we choose

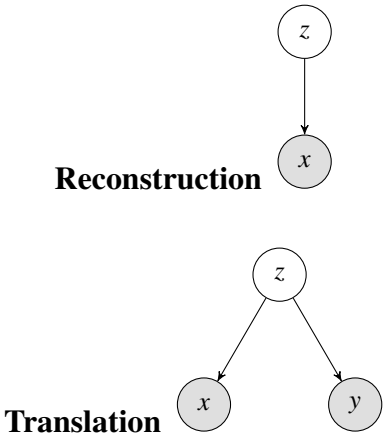
Recognition model
- WaveNet
- RNN
- MLP

Factorisation of Rec model
- Independence of $\boldsymbol{x}, \boldsymbol{y}$
- diagonal sigma
- opposite of these

Generative model
- AUTR
- WaveNet
- RNN

.

We then use these models on different types of language modelling:



While using the recognition model to do translation (We let $\boldsymbol{x}$ encode all information about $\boldsymbol{z}$, and then see what the generated $\boldsymbol{x}, \boldsymbol{y}$ correspond to).

Chapter 5

Conclusions

What have we learned from all of this?

Appendix A

An Appendix About Stuff

(stuff)

Appendix B

Another Appendix About Things

(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.

[11]

Bibliography

[1] E.T. Jaynes and G.L. Bretthorst. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[2] 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.

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

[4] David E. Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. Backpropagation. chapter Backpropagation: The Basic Theory, pages 1–34. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995.

[5] Alex Graves. Generating Sequences With Recurrent Neural Networks. *arXiv:1308.0850 [cs]*, August 2013. arXiv: 1308.0850.

[6] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]*, June 2014. arXiv: 1406.1078.

[7] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A Hybrid Convolutional Variational Autoencoder for Text Generation. *arXiv:1702.02390 [cs]*, February 2017. arXiv: 1702.02390.

[8] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved Variational Autoencoders for Text Modeling using Dilated Convolutions. *arXiv:1702.08139 [cs]*, February 2017. arXiv: 1702.08139.

[9] Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. A Convolutional Encoder Model for Neural Machine Translation. *arXiv:1611.02344 [cs]*, November 2016. arXiv: 1611.02344.

[10] 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.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*, September 2014. arXiv: 1409.0473.