Latent Variable Models for Machine Translation and How To Learn Them

Philip Schulz

Latent Variable Models for Machine Translation and How To Learn Them

ILLC Dissertation Series DS-2020-08

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/

The investigations were supported by the Dutch Organisation for Scientific Research (NWO) VICI Grant nr. 277-89-002.

ISBN: 978-3-00-065633-0

Latent Variable Models for Machine Translation and How To Learn Them

Academisch Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in de Aula der Universiteit
op donderdag 18 juni 2020, te 11.00 uur

door

Philip Schulz

geboren te Berlijn, Duitsland

Promotiecommisie

Promotor:	Prof.dr. K. Sima'an	Universiteit van Amsterdam
Co-promotor:	Dr. W.F. Aziz	Universiteit van Amsterdam
Overige leden:	Prof. Dr. M. Welling	Universiteit van Amsterdam
	Prof. Dr. P.D. Grünwald	Centrum Wiskunde & Informatica Amsterdam
	Prof. Dr. L.W.M. Bod	Universiteit van Amsterdam
	Dr. I.A. Titov	University of Edinburgh
	Dr. C. Monz	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Abstract

This thesis concerns itself with variation in parallel linguistic data and how to model it for the purpose of machine translation. It also reflects the paradigm shift from phrase-based to neural machine translation in that it addresses the variation phenomena in both frameworks.

Machine translation is the task of automatically translating text between different languages. As such, any trainable machine translation system needs to be exposed to a lot of parallel training data, i.e. sets of sentences in two or more languages that we know to be translations of one another. This data is expensive to generate since translations need to be produced by human translators first. Of course, not all human translators perform their task identical. On the contrary, their translation outputs may vary wildly. This has to do with the personal style that every translator injects into their work as well as the translators proficiency in a particular language or domain. A more experienced translator will likely produce more accurate results than a newly trained one. Similarly, a translator who specialises in sports news may not be qualified to translate legal documents. Besides these differences between translators, a translator's performance can vary from day to day depending on factors such as motivation, fatigue, stress and the like. Finally, there is variation between languages. Many romance languages allow for the omission of pronouns under certain circumstances while the use of pronouns is mandatory in English. German and many Slavic languages employ grammatical gender, a concept unknown (and deeply confusing) to the anglophone parts of the world.

For machine translation research this presents the following problem: the data is not homogeneous and a verbatim translation that may be appropriate in one context may be wrong in another. Moreover, translation systems are usually trained on a variety of documents from different sources which means that they encounter different linguistic styles. Users of machine translation systems these days expect not only an accurate translation that carries all the information of the original text, they also expect it to be grammatical and well-sound. I have

therefore taken to modelling at least some of the variation found in translation data. My main contention is that this improves the output translation as it relaxes the assumption of data homogeneity which we know to be false. Measuring translation quality with the commonly used BLEU I show experimentally that this indeed the case.

This thesis begins with a short motivating introduction in Chapter 1. It then provides the necessary mathematical background in Chapter 2. Since the probabilistic models presented in this thesis necessitate the use of approximate inference techniques, a particular emphasis is places on these methods. The chapter also provides the reader with an introduction to phrase-based and neural machine translation.

Chapter 3 introduces a new latent variable model to handle variation in word alignment. Word alignment is the first step in the phrase-based machine translation pipeline. It connects words across two parallel sentences which are likely translations of each other. These word-level translations are expanded to phrases in a later step which are in turn memorised by the translation system. A common assumption of many word alignment models is that each word in one of the languages needs to have a counterpart on the other side. This is of course false, since languages vary in how they express concepts. As mentioned earlier, some languages omit pronouns while others may omit prepositions. The reason that a pronoun occurs in sentence A and not in sentence B is thus entirely due to the grammatical requirements of language A and has nothing to do with translation. I therefore present a latent variable model that is a mixture of a classical alignment models and a language model component. The language model can account for grammatically induced words and thus prevents the alignment models from producing erroneous alignments. Experiments show that the resulting alignments lead to improved translations.

In model presented in Chapter 4 approaches variation phenomena more holistically as it is embedded into an end-to-end neural machine translation system. The hypothesis underlying that model is that the sources of variation in translation are too numerous to annotate explicitly. The model therefore attributes all variation at a given word position in the translation data to a common noise source. The innovation here is that the noise sources evolves together with the translation. Noise is modelled on a word (or sub-word) level and changes according to the hitherto produced translation. The model is an instance of a deep generative model, in particular a variational autoencoder, and uses recent variational inference techniques that allow for gradient flow through stochastic computation graphs. Not only does the model outperform its baselines, it is also shown to produce different but accurate translations when the noise source is varied stochastically.

The thesis concludes with Chapter 5. That chapter also provides an outlook on future research avenues for which I hope to have provided some of the groundwork.

Samenvatting

Deze dissertatie gaat over variatie in parallelle taalkundige data en over hoe dit gemodelleerd kan worden ten behoeve van machinevertaling. De dissertatie laat ook de paradigmaverschuiving zien van frase-gebaseerde naar neurale machinevertaling door het fenomeen van variatie te beschouwen vanuit beide paradigma's.

Machinevertaling is het automatisch vertalen van tekst tussen verschillende talen. Om een machinevertalingssysteem goed te trainen, moet het blootgesteld worden aan veel parallelle trainingsdata, d.w.z., verzamelingen zinnen in twee of meer talen waarvan we weten dat ze vertalingen zijn van elkaar. Het is duur om deze data te genereren omdat zulke vertalingen door menselijke vertalers geproduceerd moeten worden. Natuurlijk produceren niet alle menselijke vertalers precies dezelfde vertalingen. Integendeel, hun vertalingen kunnen enorm uiteenlopen. Dit heeft zowel te maken met de persoonlijke stijl van iedere vertaler als met verschillen in expertise over een bepaald domein of in een bepaalde taal. Een meer ervaren vertaler produceert over het algemeen nauwkeurigere resultaten dan een minder ervaren vertaler. Ook is een vertaler die gespecialiseerd is in sportverslaggeving wellicht bijvoorbeeld niet gekwalificeerd om wetgeving te vertalen. Bovendien kan de kwaliteit van vertalingen verschillen van dag tot dag afhankelijk van factoren zoals de motivatie van een vertaler, vermoeidheid, stress, enzovoorts. Ten slotte is er ook nog variatie tussen talen. Veel Romaanse talen staan bijvoorbeeld toe dat voornaamwoorden weg worden gelaten in bepaalde gevallen, terwijl dit niet mag in bijvoorbeeld het Engels. Het Duits en veel Slavische talen maken gebruik van grammaticaal geslacht, wat een concept is dat onbekend (en hoogst verwarrend) is in Engelstalige delen van de wereld.

Voor onderzoek naar machinevertaling levert dit de volgende uitdaging op: de data is niet homogeen en een letterlijke vertaling die gepast is in de ene context kan verkeerd zijn in de andere. Bovendien zijn vertalingssystemen vaak getraind op diverse documenten van verschillende bronnen, wat betekent dat ze verschillende taalkundige stijlen tegenkomen. Gebruikers van moderne machinevertalingssystemen verwachten niet alleen een accurate vertaling die alle informatie

van de oorspronkelijke tekst bevat, maar ze verwachten ook dat de vertaling grammaticaal is en goed klinkt. Daarom heb ik een model gemaakt van de variatie in vertalingsdata, of althans van een deel hiervan. Mijn hoofdstelling is dat dit vertalingen verbetert, omdat het de aanname dat de data homogeen is versoepelt (we weten tenslotte dat deze aanname niet klopt). Ik laat experimenteel zien dat dit inderdaad verbeterde vertalingen oplevert, middels de algemeen gebruikte BLEU I-maatstaf.

De dissertatie begint met een korte motiverende introductie in Hoofdstuk 1. Daarna beschrijft het de noodzakelijke wiskundige achtergrond in Hoofdstuk 2. Omdat de probabilistische modellen in deze dissertatie gebruik maken van benaderende inferentietechnieken wordt er een bijzondere nadruk gelegd op deze technieken. Hoofdstuk 2 biedt ook een introductie in frase-gebaseerde en neurale machinevertaling.

Hoofdstuk 3 introduceert een nieuw latente-variabele-model om om te gaan met variatie in woorduitlijning. Woorduitlijning is de eerste stap in de frase-gebaseerde machinevertalingsprocedure. Het verbindt woorden, tussen twee parallelle zinnen, die waarschijnlijk vertalingen zijn van elkaar. Deze vertalingen op woordniveau worden dan uitgebreid naar frases in een volgende stap, die vervolgens door het vertalingssysteem onthouden worden. Een veelgebruikte aanname van veel woorduitlijningsmodellen is dat ieder woord in een van de talen een tegenhanger moet hebben in de andere taal. Dit is natuurlijk niet waar, omdat talen verschillen in hoe ze concepten uitdrukken. Zoals eerder genoemd laten sommige talen voornaamwoorden weg terwijl andere talen voorzetsels weglaten. De reden dat een voornaamwoord voorkomt in zin A en niet in zin B is daarmee volledig afhankelijk van de grammaticale structuur van taal A en heeft niets te maken met vertaling. Ik introduceer daarom een latente-variabele-model dat een combinatie is van een klassiek uitlijningsmodel- en een taalmodelcomponent. Het taalmodel kan grammaticaal geïntroduceerde woorden verklaren en voorkomt hiermee dat de uitlijningsmodellen een verkeerde uitlijning produceren. Experimenten laten zien dat de resulterende uitlijningen leiden tot verbeterde vertalingen.

Het model dat geïntroduceerd wordt in Hoofdstuk 4 benadert variatiefenomenen op een meer holistische manier omdat het ingebed is in een integraal neuraal machinevertalingssysteem. De hypothese die onder dit model ligt is dat de bronnen van variatie in vertaling te talrijk zijn om expliciet te annoteren. Het model kent daarom alle variatie per woord positie in de vertalingsdata toe aan een enkele bron van ruis. De innovatie hierbij is dat de bron van ruis samen met de vertaling evolueert. Ruis wordt gemodelleerd op woordniveau (of onder woordniveau) en verandert aan de hand van de tot dan toe geproduceerde vertaling. Het model is een voorbeeld van een diep generatief model (in het bijzonder van een variationele autoencoder), en gebruikt recente variationele inferentietechnieken die hellingsstroom door stochastische berekeningsgrafen mogelijk maken. Niet alleen overtreft dit model qua prestatie de baseline, het produceert ook verschillende doch accurate vertalingen als de ruisbron stochastisch gevariëerd wordt.

De dissertatie sluit af met Hoofdstuk 5. Dit hoofdstuk biedt ook een blik op richtingen voor vervolgonderzoek, waarvoor ik hoop dat ik een deel van de basis heb gelegd.

Acknowledgments

My path as a scientist (and in part my outlook on life) has been shaped by a few remarkable individuals who were true mentors to me. Here, I would like to acknowledge them in the order that I met them.

The first one is Wolfgang Sternefeld who taught me what it means to be a researcher. Never satisfied with any given finding, Wolfgang would encourage enquiry in all directions and often be his own and other people's harshest critic. He also does not shy away from voicing his concerns about commonly agreed on scientific findings. The enquiring nature of Wolfgang serves me as an inspiration to this day.

Yi Xu was my supervisor in London and he introduced me to experimental work. He instilled the believe in me that no naturally occurring phenomenon can be explained solely by theorising about it. He also made me appreciate a baby steps approach to experimental research which often leads to tedious, incremental results in which one can place a lot of confidence and credulity, however.

Back in Tübingen I met Michael Ramscar. Michael is one of the most outspoken people I have ever met and one of the sharpest minds ands keenest rhetoricians. Both his persuasiveness and limitless enthusiam have shaped my way of thinking greatly. He made me appreciate the importance of statistical methods for empirical research and thereby laid the ground work for this thesis.

Having Wilker Aziz join the University of Amsterdam after the first year of my PhD was a blessing beyond compare. Wilker quickly became my supervisor and has done a superb job in that role. Always eager and willing to learn, Wilker would patiently listen to my proposals and ideas and round out their rough edges. He also gave me enormous freedom to pursue my interests and encouraged me to learn a lot of new things, even if they were not directly relevant for my PhD work.

Finally, there is Trevor Cohn at the University of Melbourne. I only got to work with Trevor for the final four months of my PhD but those have been some of the happiest and most productive months during the entire four years. Trevor

is an amazing researcher how never stops exploring new questions. Bouncing around crazy ideas with him has been an absolute pleasure.

There are a some more people in Amsterdam whom I would like to thank: Khalil Sima'an for giving me the opportunity to pursue this PhD and Jenny Batson and the rest of the ILLC admin team who are keeping the ship afloat. I am grateful to Iris and Ronald for their quick reaction to my request to have the abstract translated into Dutch. Outside academia, I much appreciated the workout companionship and cooking of Eddy.

Around the world there has been a large group of people who have helped me in different ways. These are of course my amazing friends in Berlin, London, Spain and Melbourne. Thanks to Anna, Carlos, Greta, Gursh, Manuel, Niko, John, Rachel, Regina, Sarah, Steven, Udo, Ziad. And of course, for sharing the path at the end of my PhD and beyond, to Dinali.

Zu guter Letzt will ich mich natürlich auch bei meinen Eltern bedanken, die meinen Lebensweg durch beinahe 20 Jahre Erziehung und ihre fortwährende Unterstützung danach geebnet haben. Im Hinblick auf meinen Beruf als Wissenschaftler habt ihr mir geholfen, indem ihr eure Meinung immer klar vertreten und auch immer Widerspruch zugelassen habt.

Contents

Abstract v

Samenvatting vii

Acknowledgments xi

1 Introduction 1

2 Background 7
- 2.1 Mathematical Background and Notation . . . 7
  - 2.1.1 Notation . . . 7
  - 2.1.2 Statistical Concepts . . . 8
- 2.2 Bayesian Probabilistic Modelling . . . 12
- 2.3 Approximate Inference Methods . . . 14
  - 2.3.1 Markov Chain Monte Carlo . . . 15
  - 2.3.2 Variational Inference . . . 20
- 2.4 Phrase-based Statistical Machine Translation . . . 26
  - 2.4.1 History . . . 26
  - 2.4.2 The SMT Pipeline . . . 31
  - 2.4.3 Word Alignment . . . 37
  - 2.4.4 Evaluation . . . 40
- 2.5 Neural Machine Translation . . . 42
- 2.6 Deep Generative Models . . . 45

3 Word Alignment Without NULL Words 49
- 3.1 Bayesian IBM models 1 and 2 . . . 51
- 3.2 Removing the NULL Word . . . 53
  - 3.2.1 Problems with the NULL Word . . . 53
  - 3.2.2 Word Alignment with a Language Model . . . 54
- 3.3 Bayesian HMM Alignment Model . . . 55

3.4 Inference by Gibbs Sampling . . . 57
3.4.1 The Gibbs Sampler . . . 57
3.4.2 Hyperparameter Inference . . . 60
3.5 Reducing Sampling Complexity . . . 61
3.6 Experiments . . . 63
3.6.1 Aligners . . . 63
3.6.2 Data Sets . . . 64
3.6.3 Translation Systems and Preprocessing . . . 64
3.6.4 Translation Results . . . 65
3.6.5 Analysis of the Auxiliary Variable Sampler . . . 65
3.7 To AER or not to AER? . . . 68
3.8 Related Work . . . 68
3.9 Summary . . . 69

4 Latent Variable NMT **71**
4.1 Deep Generative Models . . . 72
4.2 A Stochastic Decoder for NMT . . . 77
4.2.1 Conditional VAE and Stochastic RNN . . . 79
4.2.2 Model Formulation . . . 81
4.2.3 Inference Model . . . 83
4.3 Strong Generators . . . 86
4.3.1 Formal Analysis of Strong Generators . . . 87
4.4 Related Work . . . 89
4.5 Experiments . . . 90
4.6 Future Work . . . 93
4.7 Summary . . . 94

5 Conclusion and Outlook **95**

A The Gibbs Distribution and Exponential Families **99**
A.1 Log-Linear Models . . . 99
A.2 The Gibbs Distribution and Exponential Families . . . 100

B Gaussian Reparametrisation **103**
B.1 Change of Continuous Variables . . . 103
B.2 KL Divergence between Exponential families . . . 104
B.3 Gaussian KL Divergence . . . 105

Chapter 1

Introduction

This thesis presents research on variation in translation data. It uses probabilistic latent variable models to model this variation. While these models have vastly different applications (word alignment vs. end-to-end translation) a common theme underlies them. This theme is variation. Variation is ubiquituous in language. In fact, it is probably one of the defining features of human languages that they can express an idea in a multitude of ways. This is also what makes translation hard. If every utterance in a language corresponded to exactly one idea, the task of translation would boil down to writing a lexicon that lists the expressions of ideas in each language. The work presented here treats all sources of linguistic variation in translation data on par. This makes it possible to learn and evaluate these models without any need for annotations, such e.g. as the textual domain of the document that we wish to translate. The research presented here is thus applicable in all translation settings.

Interestingly, not all languages exhibit the same kind of variation. Morphologically simple languages such as English have a relatively rigid word order that limits the syntactic variation (of course, the syntactic variation available in English still is appreciable). On the other end of the spectrum there are languages like Turkish or several Slavic languages which exhibit a great amount of word order freedom. While it is hard to trace the origins of syntactic variation, many types of lexical variation are rooted in the society that uses a language and the conditions that its people live in. The Inuit famously have tens of different words for snow, for example.

As language technologists we need to account for the variation that we observe in language data. Unfortunately, it is expensive and time-consuming to gather annotations of recorded linguistic output. Even when such annotations are available they only contain restricted information. The Penn Treebank (Marcus et al., 1993), arguably the best-known annotated linguistic resource, contains information about the syntactic structure of sentences but does not reveal anything about their semantics, sentiment or sociological traits of the authors. It is furthermore

restricted to news text and may thus be uninformative with respect to text harvested from social media or recorded telephone conversations. Annotation efforts are laudable and tremendously useful to the community, however, they will never be able to provide all the information that a researcher would like to know, especially nowadays where new applications for language technology are discovered at a fast rate.

The present work acknowledges the impossibility of exhaustively listing all factors of linguistic variation and instead opts to model the dynamics underlying linguistic diversity stochastically. This approach has both advantages and drawbacks. Its main advantage is certainly that it does not have to rely on annotated data although an annotated data subset may still be useful. This makes it easy to scale the probabilistic approach to very large collections of text. The probabilistic approach also builds on a strong theoretical basis that has been developed for centuries and is still an active topic of research at this time. Probabilistic models are thus easy to design in a principled manner and one can often choose from an array of inference algorithms that have proven their worth over and over again. Furthermore, new ideas from mathematics constantly spill over into applied fields such as NLP and machine learning and lead to better models.

A downside is that the accuracy with which probabilistic models infer latent factors of variation is lower than that of a human annotator. However, considering that tasks like topic modelling (Blei et al., 2003) would be impossible to perform by humans, this can hardly be seen as a substantial criticism. The concern that practitioners often have about probabilistic models is their lack of interpretability. It is indeed hard to map stochastically inferred distributions over categories or continuous influences on variation to any commonly accepted cause of variation. Constructing such mappings, e.g. in the form of Bayesian hierarchical models (Gelman and Hill, 2007) that try to correlate induced factors of variation with observed changes in the linguistic output, certainly is an interesting task that deserves to be studied in its own right. As far as the engineering problem is concerned I would argue that interpretability of the induced representations is a minor concern as long as the engineering objective (e.g. to build better translation systems) is met. This is of course not to say that engineering cannot benefit from a better analysis of latent representations induced by a model.

The goal of this thesis is to probabilistically model sources of variation in translation data. Importantly, I make no claim that there is a causal effect from those sources on the output. Rather, I assume the linguistic variation encountered in training time correlates with unknown non-linguistic factor (such as the identity of the translator) and design models that are able to leverage that correlation to better account for the observed variation in the text. As such, the latent variables employed in this work are but a means to bestow an inductive bias onto the model and thereby better control its learning process and not to explain why variation occurs in translation data.

In Chapter 2 I provide the mathematical background on which the rest of this

thesis builds. I start by introducing basic notions of probability theory. I then give an overview of Monte Carlo methods and variational inference, currently the two most popular choices for approximating functions of random variables. The chapter then gives a historic overview of machine translation (MT) and review both phrase-based and neural machine translation. Both flavours of MT are used in this thesis.

Chapter 3 reports a modelling innovation that extends the IBM models of Brown et al. (1993) for word alignment. Based on the observation that certain words in the source sentence do not have a lexical correspondence in the target sentence, I introduce a binary latent variable that governs the generative process for each source word. Depending on the value of that variable, a source word is produced either from the alignment model or from a bigram language model. Words that are generated from the language model do not receive any links in the output alignment.

The most important consequence of this new model is that it does not need to stipulate NULL words. This leads to a much cleaner modelling approach that conditions solely on observed data. The model also uses prior distributions on all the parameters of the likelihood. This allows us to incorporate two important prior assumptions: 1) the lexical parameters of the translation table are likely to be sparse since a single word usually only has a handful of translations and 2) the alignment model should be used more often than the language model. Experimental results show that the proposed model outperforms strong baselines on several language pairs.

The second contribution of Chapter 3 is beneficial to all Bayesian word alignment models or indeed any model that uses Gibbs sampling for inference. A naïve Gibbs sampler needs to run for too long until the sampled posterior is of satisfactory quality. This makes the Bayesian word alignment models useless in practice as the increased alignment quality is not worth the additional waiting time. In this work, we augment the Gibbs sampler with an auxiliary variable that samples one or more competitors to the current alignment link. It thereby makes the time required for resampling a link constant, leading to vast runtime improvements. In fact, many Bayesian alignment models run faster than some commonly used settings of Giza++ (Och and Ney, 2003).

The work reported in Chapter 3 has previously been published in Schulz et al. (2016) and Schulz and Aziz (2016).

In Chapter 4 I report research on latent variable models for neural machine translation. The main point of contention in that chapter is that current NMT models are viewing translation as a deterministic process. However, when we look at real translation data we observe a lot of variation. In fact, most source sentences will have several equally adequate and valid translations. A translation corpus often contains only one of these possible translations. When training current NMT models it is unclear how to make them robust to the translations (and inputs) that have not been observed at training time. This robustness is crucial

since NMT models, unlike their phrase-based counterparts, do not explicitly store sentence segments and often end up producing segments that were not present in the training data. A model that is robust will produce translations that exert a degree of innovation but which are, importantly, adequate.

I propose to use a latent variable per target position. The latent variables are meant to capture variation on the word level. We use simple Gaussian latent variables that influence the decoder dynamics by serving as an additional input to its recurrent network. Since other parts of the decoder, such as the attention mechanism, depend on the RNN output the entire decoder is effectively a stochastic model. We show experimentally that the proposed model indeed improves over the standard recurrent architecture (Bahdanau et al., 2014) and is able to produce diverse translation when we sample its latent variables.

Through the use of recurrent neural networks the joint over target words and latent variables factorises exactly. This of course means that the posterior over latent variables is not analytically computable. Rather, it is approximated through variational inference. Additionally, we apply the Gaussian reparametrisation presented in Kingma and Welling (2014); Rezende et al. (2014); Titsias and Lázaro-Gredilla (2014) to enable parameter updates by backpropagation. We also discuss problems encountered during training in detail and theoretically justify the idea of scaling the KL term in the variational objective (Bowman et al., 2016).

The work presented in Chapter 4 was carried out while I was visiting the University of Melbourne from October 2017 until March 2018. It is the result of collaboration with Trevor Cohn and Wilker Aziz and has been submitted for publication at the time of writing.

Of course, non of this work has happened in isolation, although prior work on variation in machine translation is scarce in relationship to the vast amount of publications in the field. Early attempts at capturing variation in translation data were mostly based on formal grammars, and applying them to phrase-based MT has led to improvements, especially when translating into Mandarin. A good example that applies several grammatical frameworks simultaneously is Mylonakis and Sima'an (2011). More targeted models of variation are provided by Sennrich et al. (2016) and Rabinovich et al. (2017), who focus on politeness and gender traits, respectively. Both papers use annotations of these types of variations as additional inputs to their translation systems. Thus, they don't model the variation but rather take it as an input to their system. In word alignment specifically, Rios et al. (2018) have built on the work presented here and designed an alignment system that models the variation at each alignment point while also inducing word embeddings. Furthermore, earlier work by Cuong and Sima'an (2015) had already explored the idea of using latent variables to capture variation in word alignment. They fitted a sentence-level variable to the HMM alignment model of Vogel et al. (1996) that represented one out of a known number of textual domains. The alignment parameters where then conditioned on these domain

indicators, improving the resulting alignments.

Among more holistic approaches, Zhang et al. (2016) have layed the groundwork for the research presented in Chapter 4 by formulating a conditional translation model that accounts for variation through a single noise source on the sentence level. Shah and Barber (2018) and Eikema and Aziz (2019) extended this work by formulating a joint translation model that again used sentence-level noise.

Chapter 2

Background

In this chapter I introduce the technical background on which this thesis builds. I first introduce various mathematical concepts used in this thesis and provide a summary of the mathematical notation used. I then introduce concepts from Bayesian statistics which are needed to understand the contributions presented here. Thereafter I provide a survey of approximate inference methods for probabilistic models whose posterior distributions are intractable. The second part of the chapter is an overview of standard methods used in phrase-based statistical machine translation (SMT) and neural machine translation (NMT) with a focus on the issues that I have worked on.

2.1 Mathematical Background and Notation

I use several concepts from probability theory and statistics in this thesis. I presuppose a basic understanding of statistical methods, in particular maximum likelihood estimation. This section introduces some more advanced concepts that are needed to follow the contributions made in later chapters. First, however, the notational conventions used are introduced.

2.1.1 Notation

I use upper case Roman letters to denote random variables whose outcomes are (hidden) data. The Greek alphabet is reserved for random variables whose outcomes are parameters. Outcomes are denoted by the lower case version of the corresponding random variable. All random variables are assumed to be vectors of random variables (this includes one-dimensional vectors). Sequences of random variables X_1 to X_n are denoted by X_1^n and sequences of outcomes x_1 to x_n are denoted x_1^n. I often use x when it can be understood that $x = x_1^n$.

Parameters are always assumed to be real vectors. Dimensions are indicated with subscripts. Thus, the k^{th} dimension of the parameter vector θ is denoted by

θ_k. These indexing conventions extend to matrices. I use the words *parameter* and *parameter vector* interchangeably.

I notationally distinguish between probability mass functions (pmfs) and probability density functions (pdfs) by using $P(\cdot)$ for the former and $p(\cdot)$ for the latter. I use densities in the general setting and pmfs whenever they are part of a specific model. All probability functions used in this thesis depend upon some parameters θ and thus should be written $p(X = x|\Theta = \theta)$. To ease notation, I will sometimes drop the parameter vector as a conditioning event when there is no risk of confusion. The conditioning on the parameters should, however, be understood to always take place. Finally, I use the notation equivalence $p(x) := p(X = x)$.

2.1.2 Statistical Concepts

Sufficient Statistics A statistic $t(x)$ is any function of some data x (which is the outcome of a random variable X). Being a function of a random variable, $t(X)$ is itself random. Another term for the same concept is *estimator*. To avoid confusion, I use *estimator* only for functions of the form $x \mapsto f(\hat{\theta})$, i.e. estimators of (functions of) parameters. I reserve the term *statistic* for all other functions of the data.

A statistic $t(x)$ is called sufficient if the probability function $p(x|\theta)$ factorises as in Equation (2.1) (for a derivation see Bijma et al., 2013, Ch. 6.2) for two arbitrary non-negative functions g and h. Notice that g depends on the parameter while h does not. The only interaction between the data and the parameter happens in g and is mediated by $t(x)$.

$$p(x|\theta) = g(t(x), \theta) \cdot h(x) \tag{2.1}$$

Intuitively, a sufficient statistic captures all the information that the data contains about the parameter of the probabilistic model assumed for that data. This means that after the sufficient statistics have been obtained, the data can be ignored for the purposes of parameter estimation.

To illustrate this concept with an example, consider the binomial distribution with parameters $k \in \mathbb{N}$ and $\theta \in [0, 1]$ and a data set of n i.i.d. data points x_i $(1 \leq i \leq n)$ drawn from that distribution. The sufficient statistic for the binomial is $t(x_1^n) = \sum_{i=1}^n x_i$. This can be seen by considering the likelihood term

$$P(x_1^n|\theta) = \left(\prod_{i=1}^n \binom{n}{x_i} \right) \theta^{\sum_{i=1}^n x_i} (1-\theta)^{n - \sum_{i=1}^n x_i} \tag{2.2}$$

and setting $g(t(x), \theta) = \theta^{\sum_{i=1}^n x_i} (1-\theta)^{n-\sum_{i=1}^n x_i}$ and $h(t(x), x) = \prod_{i=1}^n \binom{n}{x_i}$.

Exponential Families Exponential families are families of distributions whose probability function depends only on the sufficient statistics and the natural parameters. By definition, a probability distribution is part of an exponential family

if it can be written as

$$p(x|\theta) = h(x) \exp\left(t(x)^\top \eta - a(\eta)\right) \tag{2.3}$$

where

- $h(x)$ is a reference measure
- $t(x)$ is a vector of sufficient statistics
- η is the natural parameter vector
- $a(\eta)$ is the log-normaliser.

Notice that for the log-normaliser $a(\eta)$ the following equality holds,

$$\exp(a(\eta)) = \int h(x) \exp\left(t(x)^\top \eta\right) \mathrm{dx} \tag{2.4}$$

Notice that generally the natural parameter is a function of the canonical parameter, i.e. $\eta(\theta)$. Since the canonical parameter may be a function of a conditioning context, the definition of exponential families applies to both marginal and conditional distributions. Since writing $\eta(\theta(y))$ is burdensome, we usually drop the conditioning context.

To exemplify the concept of exponential families I again make use of the binomial distribution over k outcomes. We first need to transform its canonical formulation into its exponential family formulation.

$$\begin{aligned}
P(x|\theta) &= \binom{k}{x} \theta^x (1-\theta)^{k-x} && \text{(2.5a)}\\
&= \exp\left(\log\left(\binom{k}{x}\right) + x\log(\theta) + (k-x)\log(1-\theta)\right) && \text{(2.5b)}\\
&= \binom{k}{x} \exp\left(x\log\left(\frac{\theta}{1-\theta}\right) + k\log(1-\theta)\right) && \text{(2.5c)}\\
&= \binom{k}{x} \exp\left(x\log\left(\frac{\theta}{1-\theta}\right) + k\log\left(\frac{1-\theta}{1-\theta+\theta}\right)\right) && \text{(2.5d)}\\
&= \binom{k}{x} \exp\left(x\log\left(\frac{\theta}{1-\theta}\right) - k\log\left(1+\frac{\theta}{1-\theta}\right)\right) && \text{(2.5e)}
\end{aligned}$$

It can now readily be seen that

- $h(x) = \binom{k}{x}$
- $t(x) = x$
- $\eta = \log\left(\frac{\theta}{1-\theta}\right)$

- $a(\eta) = k \log\left(1 + \frac{\theta}{1-\theta}\right)$.

An important concept within the theory of exponential families is *conjugacy*. Intuitively, two distributions are conjugate if the sufficient statistics of one are equal to the natural parameter and log-normaliser of the other. If a prior is conjugate to a likelihood, that prior can be updated by adding the sufficient statistics of the likelihood to the natural parameters of the prior. Formally, let $t(x)$ be the sufficient statistics of the likelihood and $\eta(\theta)$ be its natural parameter. Further, let $t(\theta)$ be the sufficient statistics of the prior and γ its natural parameter. The likelihood takes exactly the form of Equation (2.4). The prior has the following exponential form,

$$p(\theta) = h(\theta) \exp\left(t(\theta)^\top \gamma - a(\gamma)\right) \ . \tag{2.6}$$

Since we stipulated that for conjugacy, the sufficient statistics of the prior need to be equal to the natural parameter and log-normaliser of the likelihood, we can rewrite this as,

$$p(\theta) = h(\theta) \exp\left([\eta(\theta), a(\eta(\theta))]\gamma - a(\gamma)\right) \ . \tag{2.7}$$

This in turn leads us to rewrite the likelihood using the sufficient statistics of the prior as,

$$\begin{aligned} p(x|\theta) &= h(x) \exp(t(\theta) \times [t(x), 1]^\top) && \text{(2.8a)} \\ &= h(x) \exp\left([\eta(\theta), a(\eta(\theta))] \times [t(x), 1]^\top\right) && \text{(2.8b)} \end{aligned}$$

In order to compute the posterior first we need to update the natural parameter of the prior so as to yield the joint distribution.

$$\begin{aligned} p(x|\theta)p(\theta) &= h(x) \exp\left([\eta(\theta), a(\eta(\theta))] \times [t(x), 1]^\top\right) && \text{(2.9a)} \\ &\quad \times h(\theta) \exp\left([\eta(\theta), a(\eta(\theta))]\gamma - a(\gamma)\right) \\ &= h(x)h(t) \exp\Big([\eta(\theta), a(\eta(\theta))] \times ([t(x), 1] + \gamma)^\top - a(\gamma)\Big) && \text{(2.9b)} \\ &= h(x)h(t) \exp\Big(t(\theta) \times ([t(x), 1] + \gamma)^\top - a(\gamma)\Big) && \text{(2.9c)} \end{aligned}$$

As we can see, the natural parameters of the joint are the prior natural parameters plus the sufficient statistics of the likelihood. Interestingly, the part of the natural parameters that is multiplied with the likelihood's log-normaliser gets a count of 1 added to it, thereby keeping track of the number of data points which have already been processed.

The posterior is then derived by computing the log-normaliser with the updated natural parameters.

$$p(\theta|x) = h(x)h(t) \exp\Big(t(\theta) \times ([t(x), 1] + \gamma)^\top - a([t(x), 1] + \gamma)\Big) \tag{2.10}$$

This result is of great importance for inference purposes. Observe that the posterior has the same sufficient statistics as the prior and hence is in the same

distribution family. This means that we can compute the posterior in closed form. Moreover, there are efficient sampling algorithms for most exponential family distributions, making the posterior thus easy to sample from. Finally, all results pertaining to exponential families (e.g. about expectation, variance, entropy etc.) can readily be applied to the posterior which greatly simplifies reasoning with it. I make extensive use of conjugacy in Chapter 3. An extensive treatment of conjugacy can be found in (Bernardo and Smith, 2008, ch 5.2).

Bias-Variance Decomposition The standard measure of goodness of a statistical estimator in the frequentist paradigm is the mean squared error (MSE). It measures the expected Euclidean distance in parameter space. Since the likelihood manifold may be shaped in highly non-linear fashion, the MSE may not always be a good indicator of model fit. Parameters that are close in Euclidean space may be separated by a dent in the manifold. This would of course make it hard to move from one parameter setting to the other during optimisation and may lead to vastly different outcomes when sampling data from a model using the respective parameters. Still, the MSE is appealing because it is relatively easy to reason about.

Ideally, one would like to find an estimator such that the MSE is minimized for all possible parameters; however, this turns out to be hard in practice. It is possible, however, to decompose the MSE into a bias and a variance term. This allows us to choose estimators according to properties pertaining to their bias and variance. In particular, it is often regarded as desirable for an estimator to be unbiased, i.e. to have 0 bias. The bias-variance decomposition plays a crucial role in the inference methods used in this thesis. I therefore derive it here for an estimator g of a function f when the true parameter is θ. The data variable is $X = X_1^n$.

$$\begin{aligned}
\mathrm{MSE}(g(X)) &= \mathbb{E}\left[(g(X) - f(\theta))^2\right] && (2.11)\\
&= \mathbb{E}\left[g(X)^2 - 2g(X)f(\theta) + f(\theta)^2\right] && (2.12)\\
&= \mathbb{E}\left[g(X)^2 - \mathbb{E}\left[g(X)\right]^2 + \mathbb{E}\left[g(X)\right]^2 - 2g(X)f(\theta) + f(\theta)^2\right] && (2.13)\\
&= \underbrace{\mathrm{var}\left(g(X)\right)}_{\text{variance}} + \underbrace{\left(\mathbb{E}\left[g(X)\right] - f(\theta)\right)^2}_{\text{bias}} && (2.14)
\end{aligned}$$

All expectations are taken with respect to $p(x \mid \theta)$. Notice that both the bias and variance term are non-negative, meaning that the MSE can be lowered by lowering either one or both. This is an important idea that is a necessary prerequisite for Section 2.3. I also refer to the bias-variance decomposition repeatedly when discussing inference algorithms throughout this thesis.

Consistency While the bias-variance decomposition addresses to theoretical properties of an estimator, consistency is concerned with its asymptotic behaviour. In particular, an estimator is consistent iff it converges to the true value

in the limit of samples. To express this formally, let f be a function of the true parameter θ and let g be an estimator for f. Furthermore, assume we observe i.i.d. outcomes x_1^n that have been drawn from a distribution with parameter θ. The estimator g is consistent iff

$$\lim_{n\to\infty} g(x_1^n) = f(\theta) \ . \tag{2.15}$$

Acknowledging that the true parameter θ is usually not known, we can formulate consistency as convergence in probability. That means that for an arbitrary distribution $p(\theta)$ on the true parameter a consistent estimator fulfills

$$\lim_{n\to\infty} P\left(|g(x_1^n) - f(\theta)| > \epsilon\right) = 0 \ , \tag{2.16}$$

for a every $\epsilon > 0$.

2.2 Bayesian Probabilistic Modelling

In this section I give an overview of the principles behind the Bayesian modelling techniques used in this thesis. There are two major differences between the Bayesian and frequentist statistical paradigms. First, in Bayesian statistics we view all parts of a model as random variables, including parameters. Parameters are therefore not treated as special entities anymore but as just another random variable. This conceptually simplifies learning, since parameter learning reduces to inference of latent variables.

The second difference is that in Bayesian statistics all experiments are viewed as deterministic and thus probabilities are only used to quantify our uncertainty about the outcome of an experiment. Under this view, there are no truly random processes, only a lack of information that causes uncertainty on the side of the experimenter[1]. A standard measure of uncertainty is the variance of a distribution. Under a distribution with high variance, many outcomes are reasonably likely and it is therefore much harder to make a prediction in that setting.

Coming back to parameter inference, this means that each parameter estimate is accompanied by an uncertainty estimate. In fact, Bayesian statistics largely eschews point estimates for parameters and instead updates distributions over parameters or, more generally, latent variables. A distribution over latent variables that has been updated in the light of data is called a *posterior distribution*. It should be stressed, however, that it is only a posterior with respect to the data that was used to infer this distribution. With respect to new data, that same distribution can be used as *prior*, i.e. a distribution that encodes our belief about the outcome of an experiment **before** observing that outcome.

[1] When rolling a die, for example, the result is determined by the physical processes involved in the roll. It is only because we do not know the exact nature of these processes that we cannot predict that result.

The concept of prior and posterior are derived from Bayes' rule, which is given below.

$$p(\theta|x,\alpha) = \frac{p(x|\theta)p(\theta|\alpha)}{\int p(x|\theta)p(\theta|\alpha)d\theta}$$
$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \tag{2.17}$$

The parameter vector of the prior α is often called a hyperparameter. It is important to note that it is this hyperparameter that gets adjusted in the light of the data[2] and not the model parameter θ.

After the posterior distribution has been inferred, decisions are to be made using that distribution. In machine learning our main goal is to predict the outcome of new experiments. Ideally, this would be done by multiplying the posterior with the likelihood of the new data and integrating over the parameters. In this way, we would obtain a distribution over new data given already observed data. Such a distribution is commonly referred to as *posterior predictive distribution* which I will often simply call *predictive posterior*. For a model with parameters θ it is computed as

$$p(x_i|x_1^{i-1}) = \int p(x_i|\theta)p(\theta|x_1^{i-1})\mathrm{d}\theta \ . \tag{2.18}$$

Unfortunately, the predictive posterior often is intractable to compute, especially in structured prediction tasks. In discrete models, the reason is that the number of outcomes of the new experiment may be exponential in the number of variables involved in that experiment. In continuous models, the typical set of the posterior can live on a complex manifold. Integrating over the area of such a manifold is generally not doable analytically (for details see Betancourt, 2017). One therefore has to regress to using point estimates of parameters or other approximations most of the time.[3]

The point estimate of choice in the Bayesian setting is the Bayes estimator. It is simply the expectation of the posterior. Interestingly, given a prior distribution, it can be shown to be optimal on average for all data sets and parameters (see Bijma et al., 2013, Ch. 3.5). Intuitively, this estimator can be seen as a form of model averaging under the posterior where each model is specified by a parameter. Because we compute the integral, we average over infinitely many models, where each model's contribution is weighted according to the posterior probability of its parameter.

Hierarchical Models A further advantage of the Bayesian modelling paradigm is that it allows us to formulate hierarchies of parameters. Since parameters

[2] In a true Bayesian treatment we would of course also like to impose a prior on α. Whenever this is not done explicitly, this prior on α is assumed to be a delta function, i.e. a distribution that is degenerate in the value of α.

[3] The exception to this rule are Gaussian linear models such as Gauss Processes (Rasmussen and Williams, 2005).

are just random variables, dependencies between them can be specified as the modeller sees fit. In particular, a model can specify population-level parameters and group-level parameters where the group-level parameters depend on the population-level parameters. Notice that frequentist models cannot express such dependencies since they do not view the parameters as random variables.[4] Because they relate groups to each other through a population parameter, hierarchical models can naturally account for known groupings of the data (Gelman and Hill, 2007). For example, if we would like to model the average height of men and women in a population, we can relate the two using a hierarchical model.

$$\begin{aligned} \mu_1 &\sim \mathcal{N}(\mu_0, \sigma) \\ \mu_2 \sim \mathcal{N}(\mu_1, \sigma_1) \quad & \mu_3 \sim \mathcal{N}(\mu_1, \sigma_2) \\ m \sim \mathcal{N}(\mu_2, \sigma_3) \quad & f \sim \mathcal{N}(\mu_3, \sigma_4) \end{aligned} \tag{2.19}$$

In this example, μ_1 is the population-level average height, while μ_2 and μ_3 are the average heights per group. The expectations for the group heights under the population estimate are equal to the population average. Thus, for the estimate of a groups mean height to deviate from the population average, the data needs to provide clear evidence in the direction of the deviation.

In NLP, the best-known hierarchical Bayesian model is Latent Dirichlet Allocation or LDA (Blei et al., 2003). This model aims at labelling words in a document with topics. Each document thus is a mixture of topics. LDA learns a global prior over topic mixtures and then independently draws a mixture for each document. A further level was added to the hierarchy by Teh et al. (2006) who replaced the document-specific topic mixtures with Dirichlet Processes whose base distribution is in turn drawn from a global Dirichlet Process (Ferguson, 1973). The novelty of this construction lies in the fact that the mixtures across documents share their components. This results because the global base mixture is guaranteed to be finite. The proposal of Teh et al. (2006) presented a significant step forward in non-parametric (mixture) modelling exactly because of this sharing of components. Non-parametric hierarchical mixture models had been proposed earlier (Escobar and West, 1995), however, components were not shared between mixtures.

2.3 Approximate Inference Methods

In probabilistic modelling it is often not possible to exactly compute a probability distribution. The normalisation constant for most distributions is an integral or sum which turns out to be intractable. A point in case was discussed in Section 2.2 where the introduction of global parameters makes it impossible to

[4] Frequentist mixed effects models instead add independent noise as a latent variable to each group's likelihood (see e.g. Baayen et al., 2008).

analytically compute the normalisation constant in many cases. It should be stressed that intractable models are not exclusive to Bayesian approaches to statistical modelling but are just as common in the frequentist paradigm. In fact, most interesting probabilistic models with rich dependencies between their variables need to resort to approximate inference methods as they are not computable otherwise. For non-Bayesian examples from NLP that use different approximation algorithms[5], see Ghahramani and Jordan (1997); Zhao and Gildea (2010); Brown et al. (1993); Smith and Eisner (2005). In the broader context of machine learning, Hinton (2002) is a well-known example. More recently, variational techniques have been extended to estimate parameters via likelihood maximisation in probabilistic neural networks (Kingma and Welling, 2014; Rezende et al., 2014).

In this section I introduce the basics of the approximate inference algorithms used in this thesis. The specific algorithms for the models presented in this thesis are given in the chapters in which I present the models. The inference algorithms used can be classified as Markov Chain Monte Carlo (MCMC) and variational inference (VI) algorithms. MCMC methods estimate the exact target quantity through sampling and then evaluating a histogram. VI turns the estimation problem into an optimisation problem. To make optimisation possible it maximises a lower bound on the target quantity.

2.3.1 Markov Chain Monte Carlo

Before presenting Markov Chain Monte Carlo algorithms, I will first introduce the idea of Monte Carlo sampling. Assume we are given a density $p(x)$ and we would like to compute the expectation of $\mathbb{E}[f(X)]$ of a function f under that density. Further assume that this expectation is not analytically computable (Monte Carlo methods can of course be used to approximate analytically computable quantities but this would be a vain exercise). Then we need to approximate the expectation somehow. If we are able to draw independent samples from the density $p(x)$, we can form a Monte Carlo (MC) estimate. The Monte Carlo estimator performs the approximation as follows:

$$\mathbb{E}[f(X)] \approx \frac{1}{S}\sum_{i=1}^{S} f(x_i),\ x_i \sim p(x)\ . \tag{2.20}$$

The Monte Carlo estimator is a genuinely frequentist estimator. Interestingly, it has some properties that are considered desirable within the frequentist framework. First of all, it is unbiased. Second, it is consistent, meaning that it returns the true value of the expectation in the infinite limit of samples. This second point is closely connected to the variance of the estimator which can be

[5]The works of Brown et al. (1993) and Smith and Eisner (2005) use non-principled approximations, meaning that their inference algorithms do not provide any guarantees and their applicability therefore needs to be established on a case-by-case basis.

controlled by the number of samples. I will first show the unbiasedness of the estimator and then explain the relationship between its variance and the number of samples. As usual we assume that X_1 to X_n are i.i.d. which allows us to replace the expectations of all variables by the expectation of X_1.

$$\mathbb{E}\left[\frac{1}{S}\sum_{i=1}^{S} f(X_i)\right] = \frac{1}{S}\sum_{i=1}^{S}\mathbb{E}\left[f(X_1)\right] = \mathbb{E}\left[f(X_1)\right] \tag{2.21}$$

It follows directly from the bias-variance decomposition (Equation (2.14)) that the Monte Carlo estimator is unbiased. This also implies that the only source of error (in the MSE sense) comes from the estimator's variance. The variance of the estimator is given in Equation (2.22).

$$\operatorname{var}\left(\frac{1}{S}\sum_{i=1}^{S} f(X_i)\right) = \frac{1}{S^2}\sum_{i=1}^{S}\operatorname{var}\left(f(X_1)\right) = \frac{\operatorname{var}\left(f(X_1)\right)}{S} \tag{2.22}$$

The variance of the MC estimator is thus the variance of f under the density $p(x)$ scaled down by a factor logarithmic in the number of samples. Consistency now follows directly from the weak law of large numbers (for details see Schulz and Schaffner, 2015, Ch. 5.4). If we knew $\operatorname{var}\left(f(X_1)\right)$, we could compute the desired number of samples needed to achieve a desired MSE. However, if we were able to compute the variance, we would not need to approximate the function in the first place.

The dependence of the MC estimator's variance on sample size has important implications. First, we can make our estimates arbitrarily precise simply by taking more samples. Second, the law of diminishing returns is at play here insofar as each further sample contributes less to the variance reduction. Third, if we are able to reduce the variance of the estimator before taking any samples, e.g. by reducing the variance of the sampling distribution $p(x)$, we will need to take fewer samples. This is important since the time taken to get an MC estimate increases linearly with the number of samples taken.

Notice that in theory we can determine the desired number of samples by deciding on a desired variance and then solving Equation (2.22). In most applications, this variance is not analytically computable since the density $p(x)$ is intractable. In fact, most of the time we cannot even sample directly from $p(x)$. This may have two reasons:

1. The sampling density $p(x)$ is not computable, not even up to proportionality. On the other hand, marginal or conditional densities for some of the dimensions in x may be computable.

2. The sampling density $p(x)$ is computable but it is low in regions where we want to evaluate f. This means that most samples will be essentially useless for the purpose of learning about f. Thus, while we can sample from

$p(x)$ directly, we would rather obtain samples from another distribution and assess those under $p(x)$.

Several variations of Monte Carlo have been developed to address these problems. A good overview is provided by Besag (2004). I adapt his exposition here. In the following I focus on Markov Chain Monte Carlo (MCMC) methods. These methods assume that we cannot draw independent samples directly from the density $p(x)$. Instead, they treat each sampled outcome as a state in a Markov chain and define a transition kernel $\mathcal{K}(x^{(t)}, x^{(t+1)})$ for time steps $t \in \mathbb{N}$. Because a Markov Chain is a probabilistic process, we further require that $\mathcal{K}(x^{(t)}, x^{(t+1)})$ be a conditional pmf for states or, in the continuous case, measurable sets of states.

There are two requirements for an MCMC sampler: first, the Markov chain should converge and second, it should converge to the desired sampling density p. These requirements have to be fulfilled by the transition kernel. Notice that convergence to p implies that p is the stationary distribution under $\mathcal{K}$, i.e.

$$\mathcal{K}p = p \ . \tag{2.23}$$

Markov chains that converge to their stationary distributions are called *ergodic*. That the sampling density is the stationary distribution has to be ensured when constructing the kernel. Transition kernels of MCMC samplers are designed specifically so that they have the sampling density as their stationary distribution. In order to guarantee that the Markov chain converges to that distribution, the chain also needs to be irreducible, meaning that there are no basins of states that the chain cannot escape. If the states are discrete, the kernel can be represented by a row-stochastic matrix. Irreducibility then simply means that the matrix must not have 0-columns.

In practice, ergodicity can be checked through the *detailed balance* property. Equation (2.23) is known as the general balance property. Detailed balance is its state-wise equivalent, defined with respect to sampler states x and x'.

$$\mathcal{K}(x, x')p(x) = \mathcal{K}(x', x)p(x') \tag{2.24}$$

This property is often easier to verify. Notice that by summing over x' on both sides, we immediately retrieve general balance, thus ensuring stationarity. Ergodicity follows because of our assumption that $\mathcal{K}$ is a conditional distribution. Irreducibility is a further condition for ergodicity under the detailed balance property.

While MCMC methods make sampling from complex densities possible, the Markov property introduces a dependency between states. The obtained samples are therefore not independent anymore. The effective sample size of an MCMC sampler is thus smaller than the total number of samples. This means that we may have to run the Markov chain for a long time before we obtain enough samples.

Gibbs Sampling Probably the most famous MCMC algorithm is the Gibbs Sampler. It was introduced for optimization in Geman and Geman (1984) and Kirkpatrick et al. (1983). It was later introduced as a tool for approximate Bayesian inference in Gelfand and Smith (1990). In order to be able to apply Gibbs sampling, we assume that we cannot sample from the joint density $p(x)$ but can easily sample from conditional densities $p(x_i|x_{-i})$ $(1 \leq i \leq n)$. Here and in the following, I use x_{-i} to denote all dimensions but the i^{th} one. Instead of transitioning from one state into the next, the Gibbs sampler transitions coordinate-wise. It chooses a coordinate in random order and then updates it according to the conditional density. While the random choice of coordinates may improve the sampler's performance, in practice one often traverses a data collection in linear order. This procedure still yields satisfactory results.

The Gibbs sampler can be formally derived from another more general MCMC sampler known as the Metropolis-Hastings sampler, however, following (Besag, 2004, p. 32) I derive it directly. Assume a k-dimensional random vector x' that is distributed according to the posterior of interest. In order to transition to the next state using the Gibbs operator, we only need to resample one randomly chosen dimension i $(1 \leq i \leq k)$. Let us denote the vector containing all dimensions but i as x'_{-i}. The crucial observation here is that there is only one such vector. We can thus write the marginal probability of the new state x under the Gibbs transition operator as

$$\sum_x p(x')p(x|x') = p(x_i|x'_{-i})p(x;_{-i}) = p(x') \ . \tag{2.25}$$

Because the conditional over the i^{th} dimension is exact, the stationary distribution stays unchanged. Notice that we exploited the fact that there is only one state which differs from the new state in dimension i in the first equality of Equation (2.25). In order to change all dimensions, we need to apply the Gibbs kernel iteratively. Of course, each subsequent application still leaves the stationary distribution unchanged and thus the product of coordinate-wise Gibbs kernels defines a valid transition kernel for the Markov chain. Formally, the Gibbs kernel for the entire sample space is

$$\mathcal{K}(x^{(t)}, x^{(t+1)}) = \prod_{i=1}^{k} p(x_i^{(t+1)}|x'_{-i}) \tag{2.26}$$

where the conditioning variables have been drawn either in step t or step $t+1$.

Slice Sampling The slice sampler (Neal, 2003) is an extension of the Gibbs sampler that only samples from standard distributions such as the uniform distribution. To achieve this, it augments the Gibbs sampler with an auxiliary variable (Tanner and Wong, 1987). In order to update a coordinate, the slice sampler first samples the auxiliary variable and then samples the coordinate value conditioned

on the auxiliary variable. While this induces one more sampling step, all samples can now be taken with great ease.

Concretely, one can simply define the distribution of the auxiliary variable U_i to be uniform on the interval $[0, p(x_i|x_{-i}))$. In a second step, the new value for x_i is sampled uniformly from a set of values V such that $p(x_i^*|x_{-1}) \geq u_i$ for all $x_i^* \in V$. Notice that by construction the previous value of x_i is in V and thus V will be non-empty, guaranteeing irreducibility. Once the new value for x_i has been sampled, the auxiliary variable can be discarded.

Since both $p(u_i|x_i^{(t)})$ and $p(x_i^{(t+1)}|u_i, x_{-i})$ are valid transition kernels we can take their product. Marginalising over u for all coordinates yields a valid kernel of the form $\pi(x^{(t)}|x^{(t+1)})$. For a detailed proof of correctness for slice sampling see Neal (2003).

It is also worth mentioning that V is computed randomly by expanding the slice containing the previous value of x_i. In this way, slice sampling avoids having to compute $p(x_i|x_{-i})$ over the entire support of X_i (in fact the conditional distribution is only computed up to a constant). While this limits the applicability of slice sampling to distributions whose support has a natural order and whose modes are not too far apart, it makes Gibbs sampling on distributions with arbitrarily large support possible.

Criticism MC estimators have been criticised from a Bayesian perspective for being frequentist estimators (see e.g O'Hagan, 1987; MacKay, 1998; Rasmussen and Ghahramani, 2003). The main points of this criticism are:

- The variance of MC estimators only decreases logarithmically in the number of samples even though those samples may affirm prior beliefs that we had about the sampled function.
- Repeated samples of $f(x)$ may be obtained from input points that are very close to each other and thus hardly convey any new information. However, they are weighted equally to more informative samples by the MC estimator.

Rasmussen and Ghahramani (2003) offer an alternative that places Gaussian process (GP) priors (Rasmussen and Williams, 2005) on the sampled function. Prior information about the form of the function can be incorporated in the Bayesian MC estimator by adjusting the kernel function and length scale of the GP prior. The authors of that study showed that this leads to much faster convergence of the estimator to the true function value.

While the criticism against frequentist MC methods is certainly valid, I use them in this thesis without further modification. After all, they have been the workhorse of approximate statistical inference for decades and have produced reliable results over this period. As we have seen, we can make the MC estimator arbitrarily precise by taking more samples. This also avoids the extra effort incurred by implementing the GP prior and estimating its hyperparameters.

2.3.2 Variational Inference

While sampling methods give us unbiased, consistent estimators of the quantity of interest, they may be slow. The Gibbs sampler in particular, doing coordinate updates on a high-dimensional manifold, gets stuck in local optima of that manifold and tends to move out of them very slowly.

Variational inference (VI) methods offer a fast alternative to sampling. VI turns the inference problem into an optimization problem. In order to be efficiently computable, variational algorithms bound the target log-density from below and try to approximate it iteratively. This means that in theory they do not enjoy the same guarantees as samplers such as being unbiased. Much recent work has shown, however, that variational methods may match the performance of sampling methods (e.g Kucukelbir et al., 2017) and thus be preferred, especially when scalability is a critical factor. They are potentially also more memory efficient than MCMC methods since they only collect sufficient statistics and do not need to keep a state.

Here, I introduce the main ideas behind variational inference as well as some recent innovations that I use in this thesis. Variational inference has been an active field of research in recent years, especially in connection with the training of deep generative models (Kingma and Welling, 2014; Rezende et al., 2014). In non-neural machine learning VI has paved the way to applying well-known models such as LDA (Blei et al., 2003) to large collections of documents (Hoffman et al., 2010) and possibly even document streams (Broderick et al., 2013). A good review of these recent developments and of open problems is provided in Blei et al. (2016). I largely follow their derivation of VI here.[6]

The basic problem of approximate inference is that we want to approximate a (usually intractable) posterior $p(z|x)$ where z is the vector of all latent variables in a model which under a Bayesian view also includes the model parameters. In VI we define a surrogate variational distribution $q(z)$ as an approximation to $p(z|x)$. The variational distribution $q(z)$ can often be chosen to be tractable. This is not necessary for VI to work, however, as long as we can sample from $q(z)$ (for details see Paisley et al., 2012; Ranganath et al., 2016).

The goal of VI is to minimize the Kullback-Leibler divergence or relative entropy $KL(q(z)||p(z|x))$. As I show below, this is equivalent to maximizing a lower bound on the model's marginal log-likelihood. First, however, I derive the

[6] Some basic knowledge of information theory is needed to understand the following development of VI. A short introduction containing all the information-theoretic concepts needed can be found in (Schulz and Schaffner, 2015, Ch. 6).

evidence lower bound[7] (ELBO) which is the function that we optimise in VI.

$$\mathrm{KL}\left(q(z) \,||\, p(z|x)\right) = \int q(z)\left[\log q(z) - \log p(z|x)\right] \mathrm{dz} \tag{2.27a}$$

$$= \int q(z)\left[\log q(z) - \log p(x,z) + \log p(x)\right] \mathrm{dz} \tag{2.27b}$$

$$= \int q(z)\left[\log q(z) - \log p(x,z)\right] \mathrm{dz} + \log p(x) \tag{2.27c}$$

We have used the chain rule in line (2.27b). Under this factorisation, we see that minimising the KL divergence with respect to $q(z)$ only depends on the the first two summands in Equation (2.27c). For the purpose of maximization, we simply use function (2.28b) which is the negative of these first two summands. From here on I will simply call this function the ELBO. Importantly, it is a functional of $q(z)$. This is where variational inference derives its name from: the optimization is done using the calculus of variations, a form of calculus that operates on functions rather than points.

$$\mathrm{ELBO}(q(z)) = -\int q(z)\left[\log q(z) - \log p(x,z)\right] \mathrm{dz} \tag{2.28a}$$

$$= \mathbb{H}\left(q(z)\right) + \mathbb{E}_{q(z)}\left[\log p(x,z)\right] \tag{2.28b}$$

Here, I use $\mathbb{H}(\cdot)$ to denote the entropy of a distribution.

From Equations (2.27a)–(2.27c) and the non-negativity of KL divergence we immediately see that the ELBO lower bounds the log-evidence. The gap in that lower bound is exactly the KL divergence between the variational distribution and the model posterior.

$$\log p(x) = -\int q(z)\left[\log q(z) + \log p(x,z)\right] \mathrm{dz} + \mathrm{KL}\left(q(z) \,||\, p(z|x)\right) \tag{2.29a}$$

$$= \mathrm{ELBO}(q(z)) + \mathrm{KL}\left(q(z) \,||\, p(z|x)\right) \tag{2.29b}$$

Interestingly, when the bound is tight, i.e. when $\mathrm{KL}\left(q(z) \,||\, p(z|x)\right) = 0$, we recover the EM algorithm (Dempster et al., 1977)[8]. EM is therefore a special case of VI in which the model posterior can be computed exactly. This has inspired several modifications of EM, notably the non-stochastic mini-batch version of EM by Neal and Hinton (1999). The ELBO formulation of EM also led Hathaway (1986) to interpret EM as line search or coordinate ascent. Indeed, most versions of VI are still coordinate ascent algorithms nowadays (Blei et al., 2016).

In recent years, another representation of the ELBO that is due to Kingma and Welling (2014); Rezende et al. (2014) has become popular. It separates the

[7] The term *evidence* is another expression for *marginal likelihood*.

[8] The KL divergence is 0 iff $q(z) = p(z|x)$. For variational inference this means that the variational approximation exactly matches the posterior.

prior density from the likelihood and uses it to compute its KL divergence from the variational approximation. It is given below.

$$\text{ELBO}(q(z)) = \mathbb{E}_{q(z)}\left[\log p(x|z)\right] - \text{KL}\left(q(z) \,||\, p(z)\right) \tag{2.30}$$

Chapter 4 relies heavily on this formulation.

Mean Field VI The intractability of probabilistic models often results from complex dependencies between variables. This is true especially in the Bayesian case where many variables depend on the same parameter and are thus marginally dependent. A standard way to deal with intractabilities of this kind in variational inference is to make a mean field assumption for the variational distribution. A mean field assumption states that all variables are independent. This is obviously not true under intractable models. As a consequence, the ELBO can never become tight. On the other hand, the mean field assumption makes fast inference possible as it allows us to compute functions of a set of variables independently per variable. Formally, the mean field assumption for a variational distribution is given in Equation (2.31).

$$q(z_1^n) = \prod_{i=1}^{n} q(z_i) \tag{2.31}$$

VI with mean field variational distributions is by far the most commonly used variational inference method. Nevertheless, researchers have repeatedly tried to relax the mean field assumption and exploit dynamic programming algorithms for the relaxed distributions. For examples, see Ghahramani and Jordan (1997) and Hoffman and Blei (2015).

Variational Bayes Bayesian modelling is one of the machine learning paradigms that often require the use of approximate inference methods.[9] It is thus unsurprising that VI finds a lot of applications in Bayesian modelling. Though the general VI procedure (minimizing the KL divergence between the approximate and model posterior) stays unchanged, the application of VI to Bayesian inference is often referred to as *variational Bayes* (Attias, 2000; Beal, 2003).

Before the technical development, it is worth pointing out the conceptual difference between variational Bayes and variational inference applied to maximum likelihood models. In ML models the parameters cannot be inferred by the model and need to be optimized independently of the variational parameters. The algorithm thus alternates between updating the variational and model parameters. In Bayesian models, the parameters are modelled and thus can be inferred. Variational Bayes only optimises the variational parameters and alternates between

[9] Two famous counter-examples to this generalisation are Gaussian Processes with Gaussian Likelihoods (Rasmussen and Williams, 2005) and Kalman filters (Kalman, 1960) both of which admit exact inference.

computing the expected approximate log-evidence and updating those parameters.

Following Blei et al. (2016), I describe Bayesian graphical models in terms of local and global latent variables rather than latent variables and parameters. In particular, I use the variables z_i to denote the local latent variables associated with observed data point x_i $(1 \leq i \leq n)$ and the variable β to denote the global variables.[10] A general Bayesian model can then be written as

$$p(x, z, \beta|\alpha) = p(\beta|\alpha) \prod_{i=1}^{n} p(x_i, z_i|\beta, \alpha) \tag{2.32}$$

where α is a hyperparameter vector. For notational convenience I drop this vector from the following equations. The variational distribution for this general model is given in Equation (2.33) where λ are global variational parameters and ϕ_i are local variational parameters. The variational distribution is further factorized using the mean field assumption (see Equation (2.31)).

$$q(z_1^n, \beta|\lambda, \phi_i^n) = q(\beta|\lambda) \prod_{i}^{n} q(z_i|\phi_i) \tag{2.33}$$

The corresponding ELBO is

$$\begin{aligned} &\text{ELBO}(q(z_1^n, \beta|\lambda, \phi_i^n)) = \\ &\mathbb{E}\left[\log p(\beta|\alpha)\right] + \mathbb{H}\left(q(\beta|\lambda)\right) + \sum_{i=1}^{n} \mathbb{E}\left[\log p(x_i, z_i|\beta, \alpha)\right] + \mathbb{H}\left(q(z_i|\phi_i)\right) \ . \end{aligned} \tag{2.34}$$

This ELBO can be optimized using coordinate ascent. The coordinate updates are derived in full detail in (Beal, 2003, Ch. 2). I will only state the update equations here. I use superscripts to indicate the time step at which each update is performed.

$$q\left(z_i|\phi_i^{(t+1)}\right) \propto \exp\left(\mathbb{E}_{q\left(\beta|\lambda^{(t)}\right)}\left[\log p(\beta|\alpha)p(x_i, z_i|\beta, \alpha)\right]\right) \tag{2.35}$$

$$q\left(\beta|\lambda^{(t+1)}\right) \propto p(\beta|\alpha) \exp\left(\sum_{i=1}^{n} \mathbb{E}_{q\left(z_i|\phi_i^{(t+1)}\right)}\left[\log p(x_i, z_i|\beta, \alpha)\right]\right) \tag{2.36}$$

The above updates may be hard to implement in any particular model because the expectation terms may not be tractable. In that case further approximations are needed (for examples see Paisley et al., 2012; Ranganath et al., 2014;

[10] This description of probabilistic graphical models allows for great generality since parameters and non-parameter latent variables may occur on both the local and the global level. This is important for hierarchical models such as LDA which contain group-level parameters.

Kingma and Welling, 2014). Fortunately, these expectations are easily computable whenever the model global and local distributions are conjugate[11]. In order to be conjugate both kinds of distributions need to be in an exponential family which is the case in most machine learning models.

If the global and local distributions are indeed conjugate, the update in Equation (2.35) can be computed analytically since the approximate predictive log-posterior (the exponent in that equation) depends only on the hyperparameters α which are the natural parameters of an exponential family distribution. The expectation of the sufficient statistics of such a distribution is simply the first derivative of the log-normalizer and therefore analytically computable.

The update in Equation (2.36) is a sum of the natural parameters α of the global distributions and the expected sufficient statistics of the local distributions. Hence, in the case of conjugacy the updates can be rewritten as follows:

$$q\left(z_i|\phi_i^{(t+1)}\right) \propto \exp\left(t(x_i, z_i)^\top \frac{d}{d\lambda} a\left(\lambda^{(t)}\right)\right) \tag{2.37}$$

$$q\left(\beta|\lambda^{(t+1)}\right) \propto \alpha + \sum_{i=1}^{n} \mathbb{E}_{q\left(z_i|\phi_i^{(t+1)}\right)}\left[t(x_i, z_i)\right] \ . \tag{2.38}$$

For more details, see again chapter 2 of Beal (2003).

Stochastic Variational Inference Stochastic Variational Inference (SVI) was introduced by Hoffman et al. (2013). It stochastically approximates the ELBO by data subsampling. The theory of stochastic approximation was first laid out by Robbins and Monro (1951). The idea is to sample a subset of data points uniformly from the data set. Subsequently, the ELBO and its gradients with respect to the variational parameters are computed on that subset only. The results are then scaled to the full data set size to ensure that on expectation they are equal to the actual values. Let S be the size of the subsample and N be the data set size. The stochastic estimate of the ELBO is therefore

$$\frac{N}{S} \sum_{s=1}^{S} \text{ELBO}_{x_i}\left(q(z_i)\right) \ , \tag{2.39}$$

where I use ELBO_{x_i} to denote the ELBO computed with respect to one data point only.

Notice that this scaling is what theoretically prevents SVI from being applied to streaming data. When observing a data stream we do not know the data size a priori and thus cannot compute the scaling factor. In practice, however, not scaling the gradients may still work. A more sophisticated approach to handling

[11] In the case of discrete distributions we also require that the support be efficiently enumerable.

streaming data has been developed by Broderick et al. (2013). Notice further that this problem only occurs in approaches that search for point estimates (VI is one of them). In true Bayesian inference, updating beliefs with observations from a data stream is no problem at all. It boils down to computing the (unnormalised) posterior. As I have pointed out repeatedly in this section computing the posterior is generally intractable and thus we have to make use of point estimation methods more often than we would like to.

Once we are able to obtain unbiased estimates from subsampling, we can use Monte Carlo methods to compute the value of interest. The remaining problem is variance. While in classical Monte Carlo, we use the histogram obtained from the samples to perform further reasoning, in SVI we want to use each individual sample to perform a parameter update, and thus increase the ELBO. In SVI we use gradient-based updates. Due to the variance of the sampled gradients, we may move in vastly different directions at each update and never converge. Robbins and Monro (1951) give necessary and sufficient conditions to ensure convergence. In particular, at update step t we use the following update equation,

$$\theta^{(t)} = (1 - \rho_t)\theta^{(t-1)} - \rho_t g^{(t-1)} \tag{2.40}$$

where g is the gradient estimate computed under $\theta^{(t-1)}$. In order for this update to converge to the true ELBO the following conditions on ρ need to be fulfilled,

$$\sum_{t=1}^{\infty} \rho_t = \infty \tag{2.41a}$$

$$\sum_{t=1}^{\infty} \rho_t^2 < \infty \ . \tag{2.41b}$$

See Robbins and Monro (1951) for a proof.

Modern deep learning systems rely heavily on the theory of stochastic approximation. Their training workhorse is the backpropagation algorithm (Rumelhart et al., 1986) which applies the chain rule for derivatives to arbitrary composed differentiable functions. Since deep neural nets contain tens of millions of parameters they need to be trained on very large data sets. Updating their parameters only after processing the entire data set would be prohibitively slow. Instead of using gradient descent, modern neural network libraries employ *stochastic gradient descent* (SGD) where the objective function and the gradients are computed on a data subsample. The SGD update equations are exactly those derived by Robbins and Monro (1951).

With the development of SVI, integrating variational methods into deep neural networks has become exceedingly simple. One can easily exploit the stochastic update already implemented in neural network training software to ascend the ELBO. How this is done precisely is spelled out in Chapter 4.

Criticism Just like MC methods, VI does have its shortcomings. The most frequently raised concern is that it underestimates the variance of the distribution it seeks to approximate (see e.g. Blei et al., 2016; Rezende et al., 2014). The reason for this being that it minimises the KL divergence in Equation 2.27a. This divergence becomes infinite whenever $supp(q) \not\subseteq supp(p)$. Because the optimiser needs to ensure that the support of the variational approximation is contained in the support of the model posterior, it often makes the approximation's support smaller than the posterior's which of course results in lower variance.

A second point of criticism of classical variational inference is that the KL divergence is not symmetric. It does therefore not measure the distance between the approximation and the posterior in parameter space but merely the divergence of the posterior from the approximation. Currently, significant research efforts are aimed at overcoming this limitation. One line of research uses more general divergence measures than the simple KL divergence (Nowozin et al., 2016). The other line eschews computable divergences altogether and instead opts to use divergences that depend on a learnable function. That function is then optimised discriminatively such as to minimise the divergence between the approximation and the posterior. An example of such a method is Ranganath et al. (2016). Notice that this latter approach potentially also solves the problem of variance underestimation since the discriminatively learned function can be constrained to be finite.

2.4 Phrase-based Statistical Machine Translation

In this section I give an overview of the phrase-based variant of statistical machine translation (SMT). I start out by giving a short history of the field and then proceed to summarise the training pipeline used in modern SMT systems. I then discuss in more detail the parts of the pipeline that I improve in this thesis. To complete the overview, I conclude by describing how machine translation systems are usually evaluated.

2.4.1 History

The idea to use computers to do translation is almost as old as the field of information theory. In his seminal paper, Shannon (1948) laid the foundations of information theory. In particular, he made it possible to quantify the information content of a random source and to compute the minimal amount of bits needed to encode a random message without loss of information[12]. To illustrate his theory, Shannon (1948) introduced the concept of a noisy channel (see Figure 2.1). A

[12]This result is commonly known as the source-coding theorem.

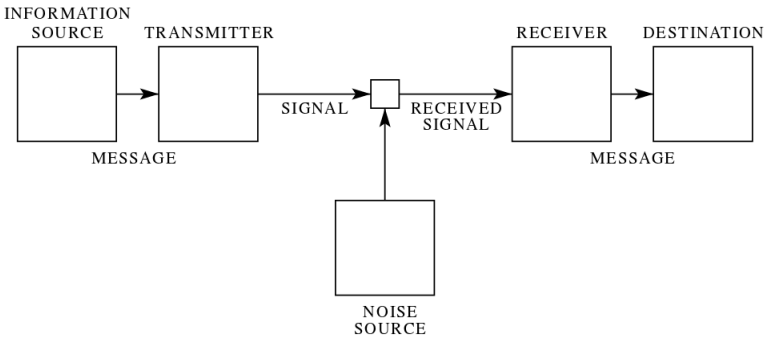


Figure 2.1: A diagram of the communication process as envisioned by Shannon (Figure taken from Shannon, 1948). The information source creates a message which is then encoded by the transmitter. The encoded message is passed through a noisy channel (the noise source is shown explicitly in the diagram). The receiver then receives the possibly distorted code and decodes it into an output message which can be read by the destination. Notice that information theory is not concerned with the information source and destination.

noisy channel is simply a collection of conditional distributions of the form $p(y|x)$ for each outcome of a random variable X. The channel is noisy whenever at least one of these distributions is not degenerate, i.e. when one source symbol can be mapped to more than one target symbol. The encoded source message x is transformed into a received encoded message y when passed through a communication channel. If that channel is noisy, the received encoded message may be different from the sent code. The receiver thus needs to make an educated guess at what the original code may have been.[13] In order to be able to make such an educated guess he needs to work with the posterior $p(x|y)$ (or an approximation to it). Based on the noisy channel model, the task of reconstruction the originally sent message is known as *decoding*, a concept that we will encounter at various points in this thesis.

Shannon's co-worker Warren Weaver soon realised that the noisy channel formulation could in principle be exploited for the task of translation (Weaver, 1949/1955). His idea was that a message (the semantic content of a sentence) could be encoded in English and then be passed through a noisy channel which generated a Russian encoding of that same message. The translation task thus reduces to inference of the source code in that particular noisy channel.

[13]The distinction between message and code is important. Shannon's theory is centred around the code. According to Shannon, communication is efficient if a) the sent code can be recovered by the receiver and b) as few bits as possible have been used to encode the message. Whether the message is sensible or of any use to the destination is a question outside of this theory. In Shannon's own words:"These semantic aspects of communication are irrelevant to the engineering problem." (Shannon, 1948, p.1)

Research on SMT stalled in the following decades. In the late 1980's/early 1990's a research group at IBM began publishing papers that adapted Weaver's idea to real-world data. This group first proposed a noisy channel formulation of machine translation in Brown et al. (1988). Their research culminated in Brown et al. (1993) where they introduced the famous IBM models.

The idea of Brown et al. (1993) was to augment the noisy channel with latent variables known as *alignments* which would discover word-to-word correspondences in pairs of translated sentences. They approached the problem as a maximum likelihood estimation problem. The noisy channel parameters were learned separately from a distribution over source encodings $P(e)$ which was modelled using an n-gram language model (Chen and Goodman, 1996/98). The product of the noisy channel and the language model, which is proportional to the posterior by Bayes's rule, was then used in decoding a foreign sentence. Since Brown et al. (1993) worked with English-French data from the proceedings of the Canadian parliament the source and target language are often generically called French and English, respectively.[14]

Because the IBM models work with categorical distributions in their canonical instead of their natural parametrisation, they cannot include features. Feature-rich models, however, have the capability to model correlations between observations through feature sharing. Potentially, they also require fewer parameters. The late 1990's saw a surge in the use of feature-rich log-linear models[15] in NLP. These were soon adapted to machine translation (Och and Ney, 2002). Since features in log-linear models can be defined without restriction, researchers soon started including pairs of contiguous word sequences into the translation model. These pairs are known as *phrase pairs* and were fruitfully exploited in Koehn et al. (2003). This idea caused a shift in the SMT paradigm. Instead of using the noisy channel approach, researchers started to focus on designing more informative features for the log-linear decoder or translation model.

A further development that had a large impact on the SMT community was the idea to induce a tree structure over phrase pairs (Chiang, 2005, 2007). This approach still used log-linear model for scoring, however, it changed the atomic units of the decoder from phrases to tree fragments and consequently also some of the input features. In particular it introduced a grammar over possibly gappy n-gram pairs. This allowed the decoder to capture richer relationships between the phrases in a sentence pair. Whereas Koehn et al. (2003) set a fixed limit

[14] I adopt the current SMT nomenclature where the source language is the language we translate from and the target language is the language that we translate into. Under the noisy channel model these two names would of course be reversed, i.e. we would translate from target to source. Indeed, Brown et al. (1993) still used this latter naming convention.

[15] A log-linear model can be seen as a Boltzmann distribution where the Boltzmann constant is dropped and the temperature is set to 1. The energy function is then defined as a linear combination of the input features. The coefficients in that linear combination are equivalent to the natural parameters in the exponential family formulation of the categorical distribution.

on how far apart a source phrase and its translation can be and scored their ordering as a linear function of that distance, the hierarchical reordering system of Chiang (2005) would allow phrase ordering over longer spans. Moreover, each ordering would be associated with several new features, allowing the decoder to make better decisions on how to order translated phrases.

Unfortunately, the probability distributions induced by the log-linear models used in SMT are not efficiently computable, owing to the fact that the model can generate too large a number of translations per source sentence. To better understand this, let us look at the partial derivatives of a log-linear model, where we use θ to denote the parameter vector and $f(x, y)$ to denote the feature vector, both of which are of dimensionality m. The vectors are given as row vectors. We compute these derivatives on a corpus of n i.i.d. data points (x_i, y_i).

$$\frac{\partial}{\partial \theta_k} \log(P(y|x)) = \sum_{i=1}^{n} \frac{\partial}{\partial \theta_k} \log \left(\frac{\exp\left(f(x_i, y_i)^\top \theta\right)}{\sum_{y_i'} \exp\left(f(x_i, y_i')^\top \theta\right)} \right) \tag{2.42a}$$

$$= \sum_{i=1}^{n} f(x_i, y_i)_k - f(x_i, y_i)_k \frac{\exp\left(f(x_i, y_i)^\top \theta\right)}{\sum_{y_i'} \exp\left(f(x_i, y_i')^\top \theta\right)} \tag{2.42b}$$

$$= \sum_{i=1}^{n} f(x_i, y_i)_k - \mathbb{E}\left[f(x_i, y_i)_k\right] \tag{2.42c}$$

These partial derivatives are needed to do gradient descent on the energy surface of the Boltzmann distribution which is equivalent to doing maximum likelihood estimation.[16] In order to compute them we need to sum over all possible outcomes (see Equation (2.42b)) which is impossible in SMT (Knight, 1999).

Och and Ney (2002) chose to approximate expectations by summing over only an n-best list of translations. This list is obtained by beam searching the space of possible translations. The n-best translations remaining in the beam are then summed over. Unfortunately, until now this methodology is still widely adopted in SMT, even though it introduces bias and variance into the gradient estimate. The bias stems from the fact that only a subset of the support is summed over and from arbitrary pruning of that subset during beam search. The variance results because this subset changes at each update iteration. Neither the bias nor the variance are easily quantifiable, leading to a potentially unbounded MSE.

Blunsom and Osborne (2008) tried to remedy this problem by introducing an MCMC approximation to the critical sum. As we have seen, this gives an unbiased estimate. Although they also needed to include some heuristics in constructing their sample space, their approach is arguably more justified than the one taken by Och and Ney (2002) and they also report better empirical performance.

Nowadays, phrase-based SMT systems are often surpassed by neural machine translation systems. These treat translation as conditional language modelling.

[16] In practice only local optima are found as this a gradient-based iterative algorithm.

They compress the source sentence into a n-dimensional real vector that can then be decoded into a target language sentence by the decoding module (Sutskever et al., 2014; Bahdanau et al., 2014). The decoder module typically is a recurrent neural language model (Elman, 1990; Hochreiter and Schmidhuber, 1997) that predicts the target sentence one word at a time. Crucially, the decoder has access to information from the encoder either through the final encoder state (Sutskever et al., 2014) or through the addition of an attention mechanism (Bahdanau et al., 2014) that learns a per-target-state weighting over hidden states in the encoder. The latter technology allows the decoder to arbitrarily access information from the encoder and yields good translation results. I provide more details on recurrent NMT systems in 2.5.

Neural MT (NMT) systems are more memory-efficient than phrase-based systems during decoding since they do not need to store a large number of phrase pairs. However, they can consume quite a lot of memory during training since the backpropagation algorithm requires all intermediate computations to be stored. The memory requirements during training thus grow linearly in the input sequence size. Moreover, NMT systems also take much longer to train and are highly sensitive to the choice of hyperparameters such as activation functions, dimensionality of word representations, hidden states etc. See Britz et al. (2017) for a comparison of hyperparameter choices.

Same same but different Phrase-based systems and NMT models are often regarded as vastly different when in fact they are not. In particular, the distinction between statistical and neural systems is void. Both are statistical systems that are just parametrised differently. They also model sequences in different ways. The neural systems (true to their nature as conditional language models) factorise the sequence probability on the word level (see Equation (2.55)). Most phrase-based systems assign a probability to the entire sequence through their conditional random field (CRF Lafferty et al., 2001) decoder.

Interestingly, the CRF is nothing but a categorical distribution over sequences of a given length. The categorical is parametrised by its natural parameters (logits). Those are in turn computed as an inner product of the feature functions and the CRF weights. Since the same feature functions are applied to all outcomes and each outcome has its own set of weights, the logits are a linear combination of the input feature values. We further recognise the function that computes the log-probability in Equation (2.42a) as the softmax function which maps the logits onto the probability simplex. Thus, the probabilistic model of phrase-based translation can be straightforwardly implemented as a perceptron with a softmax output, albeit one whose outcomes are sentences of a given length. NMT models also assign probabilities to sentence and their main distinguishing feature is that they factorise this probability on the level of atoms such as words or characters. An architectural difference is that NMT systems have more layers than the simple

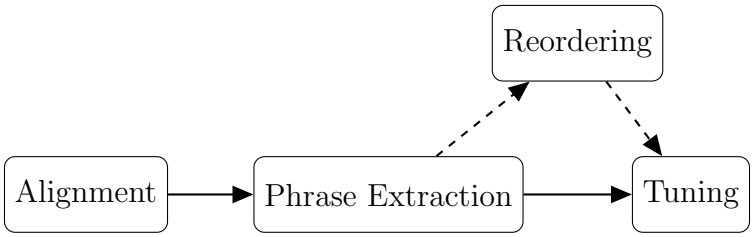


Figure 2.2: Schmematic representation of the SMT pipeline. Extracting a reordering table is optional (for example, a simple distortion-based reordering score may be used).

perceptron employed by phrase-based systems. That deep neural networks can generally be viewed as stacked generalised linear models has recently been argued by Polson and Sokolov (2017).

This insight also has important consequences for the present thesis. While I do use a phrase-based system in Chapter 3 and a neural model in Chapter 4, the model proposed for the phrase-based system can easily be "neuralised" using the techniques presented in Titsias and Lázaro-Gredilla (2014); Kingma and Welling (2014); Rezende et al. (2014).

2.4.2 The SMT Pipeline

In this section I discuss the SMT pipeline for the training of an SMT system. I first briefly describe each step in the pipeline and then give a more detailed description of the components relevant to this thesis. A depiction of the SMT pipeline is provided in Figure 2.2 to help understand its workflow.

In SMT we are given a parallel corpus of sentence pairs which are translations of each other (information about which of the sentences is the source and which is the translation is not provided). The first step in the SMT pipeline is then to align the words in each sentence pair, so as to find word-to-word translations. These word alignments are exactly the latent variables introduced into the translation process by Brown et al. (1993).

At a second step, phrase pairs are extracted from the aligned sentence pairs. The word alignments serve as constraints on the extraction process. Notice that each word pair is also a phrase pair at the same time. Then, a reordering table is optionally extracted. Reordering refers to the distance between a source and its corresponding target phrase in the linear order of the sentences. The simplest reordering model only measures this linear distance (Koehn et al., 2003) and does not require such a table.

After all the necessary information has been extracted from the training corpus, the parameters of the log-linear decoder are estimated using a small development corpus. Translation then proceeds as a discriminative task where the best translation for a given source sentence needs to be found.

Next, I give more detail on each of the components of the pipeline.

Word Alignment One standardly assumes that all sentence pairs in the parallel corpus are independent. I thus describe the word alignment process only on the sentence level. The extension to a full corpus is trivial because of the independence assumption between sentences.

Many word alignment models assume that the source sentence is generated from the target sentence, i.e. they assume the opposite generative process of translation. This implies that they need to "explain" the presence of the source words given the target sentence. This explanation happens in the form of alignment links. Based on the assumption that each source word is the translation of exactly one target word, these links connect the source words to their generating target counterparts. The set of alignment links for a sentence pair is referred to as an alignment. I will now turn to a more formal description.

For a sentence pair consisting of source words f_1^m and target words e_1^l, Brown et al. (1993) defined a probabilistic model containing a noisy channel.

$$P(f_1^m, e_1^l) = P(e_1^l)P(f_1^m|e_1^l) \tag{2.43}$$

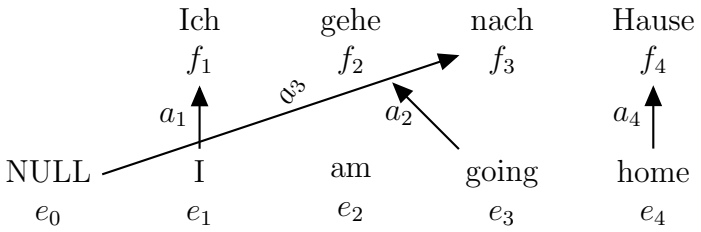
The channel $P(f_1^m|e_1^l)$ contains latent alignment variables which connect each French (source) word with the English (target) word that it is assumed to be the translation of. In order to account for the fact that not all source words may have translations in the target language, the target sentence is padded with a hypothetical NULL word e_0. Source words without lexical target translations can be generated from this NULL word. The complete channel is shown below.

$$P(f_1^m|e_0^l) = \sum_{a_1^m} P(a_1^m)P(f_1^m, a_1^m|e_0^l) \tag{2.44}$$

Here, a_1^m denotes the sequence of alignment links whose length m is identical to that of the French sentence. Since the alignment links connect French to English words, they take values between 0 and l. In their simplest form, the word alignment models of Brown et al. (1993) assume that the individual alignment links a_j $(1 \leq j \leq m)$ are independent and that the French words are independent given their alignment link and the target sentence. This leads to a further factorisation of the alignment models.

$$P(f_1^m|e_0^l) = \prod_{j=1}^{m} \sum_{a_j} P(a_j)P(f_j|e_{a_j}) \tag{2.45}$$

Here, each alignment link a_j takes on a value in $\{0, 1, 2, \ldots, l\}$ and thus if $a_j = i$ we have $e_{a_j} = e_i$.



(a)

Link	a_1	a_2	a_3	a_4
Value	1	3	0	4

(b)

Lexical		**Phrasal**	
English	German	English	German
I	Ich	I am	I
going	gehe	I am going	Ich gehe
going	gehe nach	I am going	Ich gehe nach
home	Hause	am going	gehe
home	nach Hause	am going	gehe nach
		going home	gehe nach Hause

(c)

Figure 2.3: An example of word alignment is given in 2.3a. The alignment links and there values are given in 2.3b. Notice that the alignment links are indexed with the source word positions. Words generated by the NULL word are treated as unaligned during phrase extraction. The resulting lexical and phrasal translations of English are given in 2.3c. Notice that all lexical translations are phrase pairs at the same time.

The final word alignment to be output is the most likely one under the inferred channel given the sentence pair.

$$a^* = \arg\max_{a_1^m} P(a_1^m | f_1^m, e_0^l) \tag{2.46}$$

If a French word is aligned to the NULL word, no link is set for this word in the output alignment a^*.

An example word alignment is given in Figure 2.3a. All German (source) words are aligned to exactly one English (target) word. The alignment links are shown as arrows. For example, $a_2 = 3$, meaning that the second German word is aligned to the third English word. The German preposition *nach* which does not have an English equivalent in this context, is generated from the NULL word. Notice that the English word *am* does not participate in any alignment. It does not need to since the model does not generate the English sentence. The linguistic motivation for leaving *am* unaligned is that it is an English gerund marker which does not get expressed in the German translation.

Since the channel model only provides directional alignments in which each French word can be aligned to at most one English word, it is often beneficial to train two channel models in different directions (exchanging the roles of source and target sentence) and to then combine their output alignments. There are several heuristics to perform this combination. The most important of these are described in Och and Ney (2000) and Och and Ney (2003).

Phrase Extraction In standard SMT models, phrase extraction is done heuristically per sentence pair based on the output word alignment a^*. The alignment links impose constraints on the phrase pairs that can be extracted. The most commonly used heuristic treats all aligned word pairs as phrase pairs and then builds additional phrase pairs by extending each side of an existing phrase pair one word at a time. The process continues as long as all alignment links connect a source word f_j to a target word e_i within the same phrase pair. Unaligned words that border a phrase pair are also attached to that phrase pair to create a new phrase pair. There often is a fixed limit to how many words a phrase pair can contain. While this procedure increases the coverage of the model, it also inflates the phrase table with many potentially useless phrase pairs. The phrase extraction heuristics are illustrated in Table 2.3c using the word alignment from Figure 2.3a. Notice that the NULL alignment has been removed for the purpose of phrase extraction. An algorithm that implements the heuristic can be found in (Och and Ney, 2004, Figure 3).

Once the phrases have been extracted, their translation probabilities are estimated heuristically, by dividing their co-occurrence counts (summed over all extracted phrase pairs) of a given source and target phrase by the occurrence count of the source phrase (see Equation (2.47)).This heuristic is often referred to as *relative frequency estimation*. While it is an estimator in the most general

sense, it is not connected to any known statistical framework and thus comes without any guarantees of optimality, consistency or the like. In fact, through slight modification of the argument in Johnson (2002) it can easily be shown that this estimator is biased and inconsistent. Some efforts have been made to create a consistent estimator by segmenting both sentences into allowable spans defined by word alignments (DeNero et al., 2006; Mylonakis and Sima'an, 2008).

If we denote a source phrase of length k as f_1^k and a target phrase of length p as e_1^p and let $c(\cdot)$ be a count function, we can write the heuristic estimator as

$$P(e_1^p|f_1^k) = \frac{c(e_1^p, f_1^k)}{c(f_1^k)} \ . \tag{2.47}$$

In practice, the phrase translation probabilities are estimated in both directions and both estimates are used as features in decoding.

Reordering The reordering component accounts for the displacement of target phrases relative to their corresponding source phrases. In its simplest form, the reordering score is simply the linear distance between the end of source phrase k and the beginning of phrase $k+1$ (in a translation consisting of at least $k+1$ phrases), assuming that their translations are linearly adjacent. This simple score of course does not take into account reordering patterns motivated by the presence of particular lexical items.

Instead of phrase distances, one can also use the orientation of two source phrases whose translations are adjacent as a distortion feature (Tillmann, 2004). This leads to three separate distortion features, namely the monotone, swap and neutral features. In the monotone and swap case, the source phrases are adjacent. Monotone applies whenever they appear in the same linear order as their translations, and swap when this order is reversed. The neutral configuration is invoked whenever there is one or more source phrases separating the two source phrases in question. The neutral configuration may also be annotated with *left* and *right* so as to capture a high-level reordering trend at the very least.

These phrase-orientation based reordering models are also lexicalised, meaning that the reordering decisions are informed by a language model component. Galley and Manning (2008) point out that this procedure is unprincipled since the reordering language model parameters are trained on the word level but subsequently used to reorder entire phrases. The same authors also provide an extension to the phrase orientation model. In their model, lexicalisation is implemented such that phrases are treated as atomic units. More importantly, they introduce a shift-reduce parser that allows them to score orientation between a phrase and any of its adjacent phrase groupings. That way, they are able to account for reordering patterns more globally. At the same time, the use of a shift-reduce parser ensures that the time overhead incurred by their model is minimal.

An entirely different approach is taken by Chiang (2005, 2007) who induces a

grammar[17] over phrase pairs where all but two rules need to contain at least one terminal phrase (the grammar only has one non-terminal symbol). The grammar is an extension of inversion transduction grammar (Wu, 1997) and can thus express monotone and inverted phrase reorderings at any level in the syntactic tree. Translation is reduced to constrained parsing in that framework, where the rules are scored locally based on their associated feature and the decoder is again a global log-linear model. A drawback of this framework is that because of the large number of rules and phrases, huge phrase tables are needed that put a high demand on working memory. Moreover, the source sentence needs to be parsed, making decoding complexity at least cubic.

At this point a remark on the actual implementations of these reordering approaches is due: while they can all handle phrase reorderings that go arbitrarily far in the linear size of the source sentence, in practice they are all constrained by rather restrictive reordering limits. This is necessary because decoding complexity would become too high otherwise. Knight (1999) provides a proof for this in the word-based translation case. Since a source sentence of m words can be split into phrases in m^2 ways, it is clear that this problem only gets exacerbated in the phrase-based translation case.

The present exposition is certainly not exhaustive. The interested reader is referred to Bisazza and Federico (2016) for a recent survey.

Tuning The task of estimating the weights of the log-linear decoder is often called *tuning* in SMT parlance. As mentioned earlier, tuning was originally performed under a maximum likelihood objective with heuristic approximation of the normalisation constant (Och and Ney, 2002). Soon after that, the community instead adopted the practice of Och (2003) and started directly optimizing for BLEU score (see Section 2.4.4). The MERT algorithm proposed by Och (2003) also is a coordinate ascent algorithm that uses the beam search heuristic of Och and Ney (2002) in order to get an approximate k-best list of translations.

Tuning has also been cast as a max-margin problem by Watanabe et al. (2007) and Chiang (2012). In that approach, the decoder parameters are adjusted so as to make the possibly best decoder output highly discriminable from the possibly worst output (again under the beam search heuristic).

Yet another tuning methodology was introduced by Hopkins and May (2011) who treated tuning as a pair-wise ranking task. They subsampled pairs of decoder translations from the k-best list of translations according to a heuristic that prefers easy-to-discriminate (in terms of BLEU score) pairs. These pairs then need to be ranked correctly by a linear support vector machine (SVM). Crucially, the parameters (and input features) of that SVM are the same as of the

[17] Notice that the rule probabilities of the grammar are estimated with the heuristic estimator in Equation (2.47) and are extracted in the same heuristic way that normal phrases are. A model-based estimation of these parameters could up to now not be performed simply because of the size of the grammar.

log-linear decoder. By adjusting the SVM parameters, the decoder parameters are adjusted as well. The procedure is repeated with a newly decoded k-best list[18] at each iteration until the k-best list does not change or a fixed number of iterations is reached.

This variant of tuning has recently been improved by Dreyer and Dong (2015). They exploit advances in research on linear rank SVMs (Lee and Lin, 2014) that allow them to do pairwise comparisons of the entire k-best list and thus avoid subsampling. This has an interesting consequence: given initial parameter settings and certain hyperparameters the APRO algorithm of Dreyer and Dong (2015) will always yield the same parameter estimate. This makes it convenient to work with especially if one wants to later test statistical hypotheses related to BLEU scores. All other tuning algorithms introduce additional randomness (through the order or parameter updates or through subsampling), that a statistical test will conflate with effects caused by the translation model. APRO's influence on the final translation performance, on the other hand, is deterministic.

The above summary is again not exhaustive. A recent survey of SMT tuning methods is provided in Neubig and Watanabe (2016).

2.4.3 Word Alignment

In this section I discuss several word alignment models that have been proposed in the literature.

The classical IBM word alignment models were introduced in Brown et al. (1993). I have already discussed the basic generative model that they are build on above (Equations (2.43)–(2.45)). There are five IBM models, however, I will focus on the first 2 here as the more advanced IBM models require some modifications to the generative model. IBM models 1 and 2 only differ in their probability mass function for the alignment links. For IBM1, this function is a constant, for IBM2 it is

$$P(A_j = i) = P(i|j, l, m) \ . \tag{2.48}$$

This formulation leads to estimation problems, especially for long sentences which only occur rarely. Vogel et al. (1996) therefore suggested a modification that uses the relative distance between source and target words scaled to the target sentence length l.

$$P(A_j = i) = P\left(i - j\frac{l}{m}\right) \tag{2.49}$$

This modification has been widely adopted since its publication (see e.g. Liang et al., 2006; Mermer et al., 2013).

The most immediate extension to the IBM word alignment models is reported by Vogel et al. (1996) who use a Markov chain over alignment links to relax the

[18] In practice, the k-best lists of each run are accumulated, yielding a bigger k-best list at each iteration.

independence assumption between those links. Their alignment link distribution is

$$P(a_j|a_{j-1}) = P(a_j - a_{j-1}) \tag{2.50}$$

meaning that they only take the relative distance between the links into account. Since alignment to NULL would induce unusually long distance, their model aligns all source words lexically. This restriction was later remedied by Och and Ney (2003) who stipulate identical NULL words at each target position. Notice that the HMM alignment model has a crucial advantage over the more complex IBM models 3-5 in that its parameters can be optimised exactly using an EM algorithm known as the Baum-Welch algorithm (Baum et al., 1970).

The HMM alignment model was further augmented with *fertility* distributions (distributions over the number of aligned words for each target word) in (Zhao and Gildea, 2010). Fertility distributions make the HMM intractable because the number of aligned words for target word e_i depends on the fertilities of all other target words. The distribution thus does not factorise per target or source word. In order to still be able to perform EM updates, Zhao and Gildea (2010) used an MCMC estimator (see Section 2.3.1) to approximate the needed expectations (Wei and Tanner, 1990). This led to fast training of their model. Unfortunately, the model was not particularly useful for word alignment since the authors did not manage to find the best alignment in a reasonable amount of time.

The problem of having to symmetrise the output of two directional alignment models was attacked by Liang et al. (2006) (inter alia). Their proposal was to train two models in opposite directions using EM. In the E-step they would take the product of the likelihoods of the two models and renormalise, effectively giving them a product of experts (Hinton, 2002). As is well-known, products of experts usually cannot be optimized using exact maximum likelihood estimation since the normalisation constant is intractable to compute. Liang et al. (2006) therefore settled for a heuristic in which they would simply take the product of posterior probabilities per source position and then renormalise. Notice that this does actually lead to a EM algorithm in the case of IBM models 1 and 2 where the alignment probability factorises over links. It becomes a heuristic, however, when this particular independence assumption is broken, for example in the alignment HMM. The reason is that HMM states become conditionally dependent in the posterior when used in a product of experts. Thus, the forward-backward dynamic program cannot be used separately per chain anymore.[19]. Nevertheless, Liang et al. (2006) reported good results, with the caveat that the output alignments needed to be able to align a single source word to several target words and vice versa. This requires setting a threshold on alignment link posteriors, which the

[19]The same problem, conditional dependence between states of different Markov chains, occurs when doing inference in factorial HMMs (Ghahramani and Jordan, 1997), where it is solved with the help of a structured VI algorithm that re-enables the use of forward-backward calculations

authors decided to set via grid search on a held-out data set.

One of the most practically useful models, fastAlign, was proposed by Dyer et al. (2013). This model uses a distance feature between aligned source and target words that takes values in $[0, 1]$. This feature is then fed into a log-linear model, whose parameter is optimised. The lexical translation probabilities are drawn from a Dirichlet prior and optimized using variational Bayes (see Section 2.3.2). A training iteration in their model thus consists of a) gathering expected sufficient statistics and then b) updating the variational lexical parameters as well as the log-linear parameters (the latter is done using gradient ascent). The log-linear component of fastAlign is given in Equation (2.51) where i and j and l and m are the target and source positions and sentence lengths.

$$P(A_j = i|l, m) \propto \exp\left(-\theta \left|\frac{i}{l} - \frac{j}{m}\right|\right) \tag{2.51}$$

In addition, fastAlign contains a NULL alignment probability that is not trained but set manually (in Dyer et al. (2013) it was found using grid search). Moreover, fastAlign is again directional, requiring alignment symmetrisation. This overhead is offset by fastAlign's speed that results because it can be parallelised and because the computation of the normaliser for the model in Equation (2.51) is carried out efficiently using standard results on geometric series (for details, see Dyer et al., 2013).

Bayesian Word Alignment Of particular interest in the context of this thesis is the use of Bayesian models for word alignment. This approach was pioneered by Mermer and Saraçlar (2011) who imposed Dirichlet priors on the lexical categorical distribution of IBM model 1. They would then integrate over the model analytically parameters and infer alignments using Gibbs sampling. This led to large improvements in translation quality. The same authors later extended their model to IBM2 (Mermer et al., 2013) where they would also put Dirichlet priors on the categorical distribution over alignment links. Since the alignment distribution is global and thus has many observations, the Dirichlet prior on it is rendered meaningless, though, and the posterior is dominated by the likelihood. A model without Dirichlet priors on the alignment links would work almost as well.

A particularly fine example of using hierarchical Bayesian models for word alignment is Gal and Blunsom (2013) who use hierarchical Pitman-Yor processes (HPYPs) (Teh, 2006) to extend the IBM models 1,3 and 4 and the HMM alignment model. The quality of translation systems using their word alignments improves drastically. However, apart from the empirical improvements their model is also theoretically appealing as it leverages the full power of Bayesian hierarchical models (Gelman and Hill, 2007).

The above Bayesian models, while theoretically attractive, suffer from slow inference algorithms (e.g. Gal and Blunsom (2013) used a Gibbs sampler based

on the Chinese Restaurant Process (CRP) representation of the PYP). The slow inference renders the above models practically useless. Interestingly, Riley and Gildea (2012) have presented VI algorithms (see Section 2.3.2) for Bayesian IBM models 1 and 2. These models underperform their MCMC competitors but are still much better than the original IBM models and, crucially, only take marginally longer to train than their maximum likelihood counterparts.

2.4.4 Evaluation

Evaluating SMT models is notoriously difficult. Since any given source sentence can potentially have an unbounded number of translations, no exhaustive enumeration of those translation is possible. The set of gold translations is therefore only a tiny subset of the possible valid translations.

Many NLP models are compared in terms of their F1 score, the harmonic average[20] of precision and recall. Since not all valid translations are known, recall cannot be measured for translation output. Thus, the F1 score cannot be used in the case of SMT.

Human evaluation of translation output is still considered the best evaluation possible but it is expensive to obtain and takes time to collect thereby hindering system development. During the past two decades there have been several attempts to define appropriate automatic metrics of SMT quality. The quasi-standard metric[21] is BLEU (Papineni et al., 2002).

BLEU computes the n-gram precision with respect to one or more provided gold translations. The count of each n-gram in the output translation is reduced to the maximal count of that n-gram in any reference translation (this is referred to as *clipping* by Papineni et al., 2002). Precision is then computed for each order n and the geometric average of the precisions for the different n-gram orders is computed as a surrogate for precision. This penalizes translations that are a) too long or b) contain lexical items not in any reference or c) contain the correct lexical items but in the wrong order in a window of any n-gram. Notice that the geometric average ensures that BLEU is only positive if the precision at all orders is positive. Obviously, it becomes harder to match reference n-grams of higher order. In practice, one therefore restricts the maximal n-gram order to 4.

The geometric mean of n-gram precisions does not penalize sentences that are too short. In fact, an output translation that only consists of one perfectly matching 4-gram achieves a precision score of 1. Papineni et al. (2002) introduced

[20] The harmonic average of a set of numbers is the reciprocal of the sum of their reciprocals.

[21] The term *metric* is an unfortunate choice of word that is deeply entrenched in the SMT community. BLEU is of course not a metric as it does not directly relate the output of two SMT scoring systems but instead assigns a score to each of them independently. It would thus be more appropriate to speak of a score. In the following I use BLEU score throughout.

a factor which they called *brevity penalty* to penalise sentences that are too short.

$$\mathrm{BP}(|o|,|r|) = \begin{cases} \exp\left(1 - \frac{|r|}{|c|}\right) & \text{if } |o| < |r| \\ 1 & \text{otherwise} \end{cases} \tag{2.52}$$

Above, $|o|$ is the length of the corpus of output translations (in words) and $|r|$ is the length of the reference corpus.[22] Both lengths are computed on the corpus level so as to not exaggerate the penalty incurred on short reference translations. Notice that this makes BLEU as a whole a corpus-level score. BLEU as a function of output and reference corpora o and r is given below. Here, p_i is the precision for n-grams of order i after clipping.

$$\mathrm{BLEU}(o,r) = \mathrm{BP}(|o|,|r|)\left(\prod_{i=1}^{4} p_i\right)^{\frac{1}{4}} \tag{2.53}$$

Because both the precision and brevity penalty terms can at most be 1, BLEU is bounded from above by 1. Likewise, both terms are at least 0. It follows that BLEU lies in $[0,1]$.

Despite its simplicity, BLEU regularly performs on par with other evaluation scores when correlated with human judgements on translation quality. Since it is easy to implement and virtually takes no time to compute, it is still the standard choice for translation quality scoring.

Statistical Evaluation Apart from finding an adequate evaluation scoring function, another problem in the evaluation of SMT (and more generally NLP) results is statistical evaluation. The statistical evaluation of SMT usually relies on the comparison of BLEU scores of different systems. Because of the previously discussed randomness induced by most tuning algorithms, these BLEU scores are variable and usually too few of them are obtained to make statistical tests reliable.

There is a more severe problem however: even assuming that the tests were powerful enough to discriminate between systems accurately, they still a) only compare systems holistically instead of measuring effects[23] and b) the comparisons are littered with what Gelman and Loken (2013) (inter alia) call "researchers' degrees of freedom" or, less politely, "p-hacking". In the context of NLP, this same phenomenon has been shown to exist by Søgaard et al. (2014). In a nutshell, it means that there is variability between systems that is purely due to chance. If a researcher is given the freedom to choose the comparisons he does

[22] In the case of multiple reference translations, the reference that is closest in length to the output is found per sentence. These closest lengths are then summed to yield the effective size of the reference corpus.

[23] SMT systems virtually never differ in only one component and thus holistic comparison is uninformative and inadequate.

and, equally important, does not perform, he is essentially given the freedom to look for comparisons that turn out to be significant. Because of chance variability, the researcher is guaranteed to find such comparisons, even if there is no actual difference between systems.

As Gelman and Loken (2013) point out, there need no be a vicious intent on the side of the researcher for this problem to arise, nor does the researcher need to perform and subsequently discard several comparisons. Simply by not performing possible comparisons, his results may be based on chance outcomes.

Notice that this problem is a peculiarity of significance test. In a Bayesian framework, where we simply update our beliefs, this problem does not arise because our beliefs result from observing the outcomes of several experiments. It is very unlikely that the same chance outcome would occur consistently across experiments. In the context of significance testing, this problem can be addressed through replications. If a replication fails, the previously reported significant results were likely due to chance variation in the data. Søgaard et al. (2014) state 50%(!) as a conservative estimate of how many NLP results are likely to not be reproduced across data sets.

2.5 Neural Machine Translation

Neural Machine Translation (NMT) models are a new modelling paradigm that has conquered the field of machine translation in recent years. NMT views the task of translation as conditional language modelling. This development is enabled by the capacity of neural networks to generate adequate sequences. Recurrent neural networks, and in particular the long-short-term memory (LSTM Hochreiter and Schmidhuber, 1997), are capable of conditioning the generation of a random outcome on arbitrarily long prefixes. This is achieved by the use of a memory cell in the LSTM which can encode the prefix. The memory cell is a k-dimensional real vector.[24] As the LSTM unit is applied sequentially to each item in a sequence, this memory cell gets updated.

The update mechanism is governed by gating functions. The input to the LSTM is modulated by a squashing function and then downscaled by an input-dependent gating value. The gated input is then added to the memory cell. The output is a squashed and gated version of the memory cell (although output squashing is usually omitted in current implementations). Besides gating, the major invention in the LSTM is that the memory cell has a constant self-connection of value 1. During the forward pass, this means that the information in the memory cell can be propagated across time steps (or elements in the sequence) in

[24] In the original work of Hochreiter and Schmidhuber (1997), the term *memory cell* was used for only one neuron. What I refer to as a memory cell is a *memory block* in their nomenclature. They also suggested that there may be several memory blocks in one LSTM unit, however, this is usually not done today.

a loss-free manner. The self-connection's main motivation is the backprop pass, however. Let m be the memory cell, x the input and $\sigma(\cdot)$ be the sigmoid function. Then we define an LSTM with output gate o and input gate i as,

$$[i, o] = \sigma\left(W_g[x, m]^\top + b_g\right) \tag{2.54a}$$
$$m = m + i \odot \tanh\left(W_m x + b_m\right) \tag{2.54b}$$
$$o = o \odot \tanh(m) \tag{2.54c}$$

The weight matrices W and bias vectors b are indexed according to their function. The operator $\odot$ denotes the element-wise or Hadamard product.

The backpropagation algorithm (Rumelhart et al., 1986) builds on the insight that neural networks compose functions. Each layer in a neural net learns a function and its output serves as the input to the next layer. Thus, we can represent the computation of an n-layer neural network as $f_n \circ f_{n-1} \circ \ldots \circ f_1$. In order to compute the weight gradient of the lower layers with respect to the error of the output of the n^{th} layer we need to apply the chain rule of derivatives. Since each layer contains many nodes the error derivative distributes per layer. If the derivative at each layer has an absolute value smaller than 1, the derivative arriving at the lower layers diminishes at an exponential rate. Consequently, parameter updates in lower layers will be small to none. This phenomenon is called *vanishing gradient problem* (Hochreiter and Schmidhuber, 1997). It is particularly pronounced in recurrent neural networks. Applying a recurrent unit k times across a sequence is equivalent to building a k-layer neural net, with the important addition that each layer may receive additional input from outside the net itself. For long sequences, the vanishing gradient problem therefore tends to be very pronounced.[25]

The LSTM tackles the vanishing gradient problem through the self-connection of the memory cell. The gradient flow along the memory cell is constant and thus ensures that the updates to the LSTM are performed even for positions at the beginning of the sequence. Interestingly, the LSTM still works well and even improves its performance when the self-connection is also turned into a gate (Gers et al., 2000). This is likely due to the fact that this gate enables the network to purge its memory on occasion. In this thesis I focus on LSTMs, however, it is worth pointing out that a modification of LSTMs called the gated recurrent unit (GRU Chung et al., 2014) is also frequently used in NMT.

The general architecture of NMT models consists of two modules, an encoder and a decoder (Sutskever et al., 2014). The encoder uses a recurrent neural net, usually a bi-LSTM (Graves et al., 2005) to build a context-dependent representation of each word in the source sentence. This information is then used by the

[25]The opposite problem of gradients growing too large when their absolute value at each layer is greater than 1 is called the *exploding gradient problem*. It can, however, be rather easily addressed by gradient rescaling (Pascanu et al., 2013).

decoder to produce the output sentence word by word. The decoder also conditions on the prefix it has generated so far. It is therefore often a standard LSTM. The probabilistic model employed by the encoder-decoder framework is given in Equation (2.55).

$$P(e_1^l|f_1^m) = \prod_{i=1}^{l} P(e_i|f_1^m, e_1^{i-1}) \tag{2.55}$$

Notice that the factorisation of the sequence probability is exact. This is due to the fact that the memory of the LSTM is in principle infinite. As noted above, this model is a simple conditional language model. We could replace the source sequence by an image or sound wave without changing the generative model.

The basic encoder-decoder architecture is not quite adequate for NMT. It was not until Bahdanau et al. (2014) introduced attention mechanisms that NMT systems became competitive with SMT systems. An attention mechanism computes a dynamic representation of the entire input sentence that is updated at each decoding step. This allows the decoder to selectively *attend* to parts of the input that are relevant to the current translation step. Below I abstractly summarise a standard recurrent NMT system as it is used nowadays. The function $\mathrm{RNN}(\cdot)$ can be instantiated by some recurrent neural network. The attention function is denoted $a(\cdot)$. In a slight abuse of notation I take e_{i-1} to be the vector representation of the target word in the $i-1^{st}$ position.

$$\begin{aligned}
\big[h_1, \ldots, h_m\big] &= \mathrm{RNN}\,(f_1^m) && \text{(2.56a)}\\
\tilde{t}_i &= \mathrm{RNN}\,(t_{i-1}, e_{i-1}) && \text{(2.56b)}\\
d_{ij} &= v_a^\top \tanh\left(W_a[\tilde{t}_i, h_j]^\top + b_a\right) && \text{(2.56c)}\\
\alpha_{ij} &= \frac{\exp\left(d_{ij}\right)}{\sum_{j=1}^{m} \exp\left(d_{ij}\right)} && \text{(2.56d)}\\
c_i &= \sum_{j=1}^{m} \alpha_{ij} h_j && \text{(2.56e)}\\
t_i &= W_t[\tilde{t}_i, c_i]^\top + b_t && \text{(2.56f)}\\
\phi_i &= \mathrm{softmax}(W_o t_i + b_o) && \text{(2.56g)}
\end{aligned}$$

Here, the vector c_i is the compressed representation of the input computed by the attention mechanism. It is commonly called *context vector*. The weights $W, b \in \theta$ are learnable parameters. They produce the decoder state. Besides being used to compute the next decoder state (see Equation (2.56b)), it also serves to compute the conditional distribution at output step i. To this end, the decoder state is mapped into the logit space of the categorical output distribution via a learnable affine transformation. The logit space is a real space whose dimensionality is the size of the output vocabulary. The logits are the natural parameters of the

categorical distribution.[26] The logits are mapped onto the probability simplex using the softmax transformation to yield the desired conditional distribution ϕ.

Recurrent NMT models have received the bulk of attention in research in the past years. They do, however, have the shortcoming that they necessarily process the input sequentially. It therefore takes a long time to train them. Recently, convolutional encoders and decoders have been proposed to solve this problem (Gehring et al., 2017). While efficient in training, they do not appear to beat standard recurrent architectures in translation. Another alternative that does actually supercede the RNN's performance is the self-attention network of Vaswani et al. (2017). It employs several layers of attention on both the source and the target side. While both these approaches are interesting in their own right, neither will be pursued further in this thesis.

2.6 Deep Generative Models

Deep generative models (DGMs) are a class of probabilistic models that are parametrised by a composition of several differentiable functions, usually in the form of a neural network. This section introduces DGMs and provides a taxonomy of them. It thereby lays the groundwork for the application of DGMs to NMT in Chapter 4.

The notion of "deep" in DGMs refers to the stacking of layers. Crucially, these are deterministic differentiable function layers as known from neural networks, not layers of random variables. Models that combine several layers of random variables are known as hierarchical models. It is of course possible to combine the layering of deterministic functions with the layering of random variables and thus get a hierarchical DGM (see e.g. Hinton et al., 1995; Rezende et al., 2014). In this section, we will focus on non-hierarchical DGMs. Notice that in all application scenarios of DGMs, at least one of the variables in the model is latent. This makes them perfectly suitable for unsupervised learning. We will ,however, only consider directed DGMs and thus exclude e.g. restricted Boltzmann machines.

The general idea behind (directed) DGMs is to compute a probability density or mass function with a differentiable function. This can happen in one of two ways.

- We assume a distribution whose density/mass we can write down analytically. In that case, the deep function computes the parameters of the the density.
- The function transforms an analytically available distribution into a another one whose functional form is generally unknown. For reasons discussed below, this option is only available for densities, not for mass functions.

[26] See Section 2.1.2 for an explanation of natural parameters.

	VAE	OT	GAN
analytical density/mass	yes	no	no
computable density/mass	yes	yes	no
posterior inference possible	yes	some[27]	no
continuous variables	yes	yes	yes
discrete variables	yes	no	no

Table 2.1: Comparison of DGM classes based on their capabilities.

Examples of the first instance of DGMs include sigmoid belief nets (Hinton et al., 1995, e.g.) and, more recently, variational autoencoders (Kingma and Welling, 2014; Rezende et al., 2014). Transformation-based models of the second kind include generative adversarial networks (Goodfellow et al., 2014) and models using optimal transport (OT), such as normalizing flows (Rezende and Mohamed, 2015; Kingma et al., 2016). Notice that speaking about different models is slightly misleading: all these different frameworks are able to express the same models (namely directed graphical models), however, they differ in several other dimensions. Variational autoencoders (VAE) and normalizing flows allow for inference during training while generative adversarial networks do not. This has of course an impact on the learning algorithms used. GANs take samples from a fixed distributions and transform them. Their training procedure then optimizes the match between that sample and a real data point based on a two-sample statistical test. VAEs and normalizing flows exploit insights from existing graphical models and use approximate inference algorithms (see Section 2.3) for learning. While Monte Carlo techniques have been used to learn sigmoid belief nets (Neal, 1996), the most commonly used approximate inference technique used today is variational inference. However, Monte Carlo techniques have recently been used to find better variational approximations (Salimans et al., 2015; Hoffman, 2017).

At this point I readily admit that drawing lines between these different frameworks is necessarily an oversimplification. After all, they can and have been combined. There are VAEs that use normalizing flows (Kingma and Welling, 2014; Rezende et al., 2014) and or GANs (Zhao et al., 2018) for inference. However, in order to guide the reader's thought and justify the modelling decisions taken in Chapter 4, Table 2.1 contrasts some aspects of the three DGMs classes.

One thing that stands out about VAEs is their ability to model discrete variables. This is because they are readily amenable to the score function estimator (Williams, 1992; Paisley et al., 2012). The same estimator can in principle be used for GANs, however, it has not been used as widely or successfully. On the

[27]Since OT uses invertible functions, posterior inference is in principle possible. However, some OT models move their particles in a nonparametric fashion and thus the inverse cannot be recovered (Liu and Wang, 2016).

flip-side, VAEs are constrained by the modeller's choice of variational approximation. If the approximating family is not a good fit to the target distribution, the VAE is doomed to fail.

Chapter 3

Word Alignment Without NULL Words

In this chapter I present the work originally reported in Schulz et al. (2016) and Schulz and Aziz (2016). In line with the overarching theme of this thesis, this work concerns itself with variation in word alignment. The particular variation that is being looked at is the production of untranslatable words. Examples of untranslatable words can be articles, preposition and personal pronouns amongst others. I say "can be" exactly because their being untranslatable depends on the language pair under consideration. When we are trying to align from pro-drop languages such as Spanish or Italian to English, the English subject pronouns need to be filled in because the English language requires them. There is no word on the source side that could be used to justify their occurrence; they are simply a consequence of English not being a pro-drop language.

Let us consider prepositions as another case. Whereas the English sentence *I love my dog* does not contain any prepositions, its Spanish translation *Yo amo a mi perro* does contain one. Again, the presence of the Spanish pronoun cannot be related to any of the words in the English sentence. It is required only because of how direct objects work in Spanish.

This kind of variation is not unexpected in translation. If we change the languages we are translating between, it seems plausible that certain words will translate differently or may not translate at all. For word alignment, however, this poses a significant challenge. After all, we are trying to align as many words as we reasonably can (as this will significantly constrain the number of spurious entries in the translation table).

This chapter addresses the problem of untranslatable words through close inspection of commonly used word alignment models (see Section 2.4.2 for an overview of these models). Most alignment models assume that every word in the language we are aligning to is generated from some word in the language we are aligning from. I am challenging this assumption, as it seems to me that it is not well motivated given the above insights into why untranslatable words occur. I propose to augment the existing alignment models with a language model that is

used to produce untranslatable words. This way the assumption that these words exist because of idiosyncrasies in their respective language is explicitly capture by the model.

In doing so, I make use of Bayesian modelling techniques introduced in Section 2.2 as well as a variant of the MCMC algorithms introduced in Section 2.3.1. These two techniques are combined to build a hierarchical Bayesian word alignment model that mixes between an alignment and a language model component.

Notice that this chapter considers the task of word alignment in the context of phrase-based machine translation. Word alignments find application in other domains such as word vector fitting, and I make no claim that the aligners presented here are useful to these tasks. This also means that I do not conduct a general evaluation of the resulting alignments and only verify that they positively impact the translation outcome. The main evaluation score used in this chapter thus is BLEU.

Chapter Highlights

Problem Statement

- All commonly used word alignment models stipulate a hypothetical NULL word that generates untranslatable words. The NULL word cannot be motivated linguistically. Moreover, rare words often get aligned to the NULL word although they actually do have a translation in the translated sentence.
- MCMC samplers for Bayesian word alignment models are very slow. This has prevented a wider adoption of Bayesian models of word alignment despite their superior performance.

Contributions

- This chapter formulates a new class of models for word alignment which do not need a NULL word and instead predict unaligned words from their context
- It presents a fast auxiliary variable Gibbs sampler whose sampling complexity is linear in sentence length while a naïve Gibbs samplers complexity would be quadratic
- It gives a proof that the new sampler is ergodic
- It compares the accelerated sampler to the standard Gibbs sampler by tracing their joint likelihoods

---

3.1 Bayesian IBM models 1 and 2

Before I present the new collocation-based models I introduce Bayesian IBM models 1 and 2 in this section. These models were first presented in Mermer and Saraçlar (2011) and Mermer et al. (2013). The reader is referred to Section 2.4.2 for an overview of word alignment in general and the frequentist IBM models 1 and 2 in particular.

Below I again give the pmf of a source sentence $\mathbf{f}_1^m$ given a target sentence $\mathbf{e}_0^l$ under a general IBM-style alignment model. Since I will use word types[1] to index parameters, sequences of types are bold-faced in this chapter to distinguish them from individual words. The categorical parameter set of the translation distributions is denoted by θ and θ_e denotes translation distribution of a particular type e. I will sometimes use θ_{ef} to denote the parameter for a specific translation. The parameters of the prior over alignment links is denoted by ϕ. English word are sometimes indexed by their corresponding alignment links. This means that $e_{a_j} = e_i$ iff $a_j = i$.

$$P(\mathbf{f}_1^m|\mathbf{e}_0^l, \theta) = \sum_{a_1^m} P(a_1^m|\phi) \prod_{j=1}^{m} P(f_j|e_{a_j}, \theta) \tag{3.1}$$

Recall that alignment links connect each source word to exactly one target word. The links are thus indexed In the case of IBM 1 and 2, the alignment links are assumed independent and thus the likelihood becomes

$$P(\mathbf{f}_1^m|\mathbf{e}_0^l, \theta) = \prod_{j=1}^{m} \sum_{i=0}^{l} P(a_j = i|\phi) P(f_j|e_{a_j}, \theta) \ . \tag{3.2}$$

For IBM1, the factor $P(a_j|\phi)$ is a uniform distribution and for IBM2 several formulations have been proposed. The original formulation of Brown et al. (1993) suffers from overparametrisation as it conditions on the lengths of the two sentences (m for the source, l for the target). Here, we adopt the formulation introduced by Vogel et al. (1996) which scales the source word position j to the length of the target sentence l. The same formulation was also adopted in Mermer et al. (2013).

$$P(a_j = i|\phi) = P\left(i - \left\lfloor j\frac{l}{m} \right\rfloor |\phi\right) \tag{3.3}$$

All distributions in Equation (3.2) are categorical distributions. The obvious choice of a prior distribution for these is the Dirichlet distribution as it is the conjugate prior for the categorical. Other priors are of course possible: for example, if one wishes to introduce correlations between the categorical parameters,

[1] In linguistics, a type is a lexical entry and distinct from a token, which is a realisation of a lexical entry. In the phrase "on and on" the type "on" is realised by two tokens.

one could choose a logistic-normal prior (Aitchison and Shen, 1980). Inference of the covariance matrix of such a prior is problematic in high dimensions, however. In the case of word alignment one would need a matrix of the squared size of the source vocabulary whose estimation and inversion are very costly.[2] We thus assert that the conjugate Dirichlet prior, while not modelling correlations, is the more attractive option.

Categorical-Dirichlet Posterior I now derive the posterior distribution for a model with a categorical likelihood and a Dirichlet prior. For the sake of illustration, I focus on the conditional distribution associated with a particular target type e. To make the exposition general (and useful for later sections), I write the posterior in exponential family form. I also assume that the alignment variables have been fixed beforehand. The constants $V_{f/e}$ are used to denoted the target and source vocabulary sizes. The Dirichlet's concentration parameter is denoted by α. Since the value of α is the same for all conditional distributions, I do not index it as this reduces notational clutter.

$$P(\theta_e|\mathbf{e}_0^l, \mathbf{f}_1^m, a_1^m) = p(\theta_e|\alpha) \times \prod_{j=1}^{m} P(f_j|e_{a_j}, \theta_e)^{\mathbb{1}_e(e_{a_j})} = \tag{3.4a}$$

$$\exp\left(\sum_{f=1}^{V_f} \log(\theta_{ef})(\alpha_j - 1) - a(\alpha)\right) \times \exp\left(\sum_{j=1}^{m} \log(\theta_{ef})\mathbb{1}_f(f_j)\mathbb{1}_e(e_{a_j}) - a(\theta_e)\right) \tag{3.4b}$$

$$\propto \exp\left(\sum_{f=1}^{V_f} \log(\theta_{ef})\left(\alpha - 1 + \sum_{j=1}^{m} \mathbb{1}_f(f_j)\mathbb{1}_e(e_{a_j})\right)\right) \tag{3.4c}$$

The expression in line (3.4c) is an unnormalized Dirichlet distribution. This leads us to conclude that the resulting posterior is a Dirichlet (as is expected from conjugacy) with parameters $\alpha_f + \sum_{j=1}^{m} \mathbb{1}_f(f_j)\mathbb{1}_e(e_{a_j})$ where α_f is the entry in the Dirichlet parameter vector corresponding to the concentration on type f. This means that we simply need to add the sufficient statistics of the categorical distribution to the parameter of the Dirichlet prior in order to compute the Dirichlet posterior (this is exactly what we would expect from our discussion of conjugacy in Section 2.1.2). This argument can of course be extended to a product of categoricals such as the one we encounter in the case of the IBM model. There, each English type is associated with its own distinct conditional distribution whose posterior is updated according to line (3.4c). Since the distribution over alignment links for IBM2 is also categorical and assumed to be drawn from a Dirichlet, the posterior over its parameters can be derived analogously.

[2] One can of course do inference on factorisations of the covariance matrix, such as its Cholesky factor. However, even such types of inference are still unfeasible in our case.

For the full Bayesian IBM model 2 we introduce a random parameter vector Φ over jump distances (see Equation (3.3)). This distribution over distances additionally contains a NULL event that occurs whenever an alignment link to the NULL word is used. The parameter vector ϕ is drawn from a Dirichlet distribution with symmetric parameters β. The likelihood of the model is given in Equation (3.5).

$$P(\mathbf{f}_1^m|\mathbf{e}_0^l,\alpha,\beta) = \int p(\phi|\beta) \int \prod_e p(\theta_e|\alpha) \prod_{j=1}^{m} \sum_{i=0}^{l} P(a_j = i|\phi) P(f_j|\theta_{e_{a_j}}) \mathrm{d}\theta \mathrm{d}\phi \quad (3.5)$$

The likelihood of Bayesian IBM1 is the same except that the prior on ϕ is a delta function on the uniform distribution.

3.2 Removing the NULL Word

In this section I motivate the modelling innovation presented in this chapter by discussing conceptual and practical problems that arise from stipulating a NULL word in the target sentence. I then formulate a solution to the identified problems in form of a mixture of an alignment and a language model.

3.2.1 Problems with the NULL Word

When Brown et al. (1993) designed the IBM models, they chose a noisy channel model and thus faced the challenge of having to generate *all* source words. Some source words, however, do not have translations in the target language. This is especially often in the case of function words such as determiners and prepositions. Below are two examples from French and German as translated to English where the prepositions that do not have translations in the English sentences are boldfaced.

1. (a) Ich gehe **nach** Hause.
   (b) I am going home.
2. (a) Je bois jus **d'** orange.
   (b) I drink orange juice.

The solution of Brown et al. (1993) was to introduce an invisible NULL word that they stipulated to occur in every target sentence. Since the NULL word is invisible, source words that translate to it would effectively have no translation. This choice was supposedly made out of convenience as it does not necessitate any modification of the probabilistic model. Nevertheless this choice is not without problems; those are listed below.

- **Garbage Collection:** The NULL word is so frequent that its inferred conditional translation distribution has support over the entire source vocabulary. As a consequence, the conditional's probability mass is widely spread. This undermines the intended function of the NULL word, which is to generate source words that do not have lexical translations in the target sentence. In the IBM models, it is often the case that infrequent target words are aligned to such source words, simply because they have a smaller support. This problem is commonly referred to as **garbage collection** and has already been pointed out in Brown et al. (1993). It has also been studied in depth by Moore (2004).

- **Distortion:** IBM model 2 and more developed models use a non-uniform distribution over alignment links, also known as distortion model. The NULL word occupies the 0^{th} position. If we parametrise the distortion model as done in Equation (3.3) it will induce unusually long jumps whenever a word towards the end of the source sentence is aligned to NULL. While one can partially circumvent this problem for IBM model 2 by introducing a special, distanceless NULL event into the jump distribution such a solution is not possible for the alignment HMM (see Section (3.3)).

- **Modeling:** From a statistical modeling perspective (where we seek to explain our observed translation data), the NULL word is clearly a poor choice. After all, prepositions in French and German are a product of structural requirements of those languages and need to be present after certain verbs/nouns independently of the English target sentence. These contextual effects present on the source side cannot be captured by the IBM models.

  Notice further that the NULL word is not actually observed in the data and should thus be treated as a latent variable if one wishes to use it. Stipulating its presence amounts to altering the data.

The above problems have motivated us to seek a modeling solution that erases the need for a NULL word. Taking inspiration from the fact that untranslatable words are often strongly associated with their contexts, we introduce a language model component into the aligner. Our model is described in detail in the next section.

3.2.2 Word Alignment with a Language Model

Since source words that do not have translations in the target language are usually present because of requirements from the source context, we contend that such requirements should be incorporated into our probabilistic models. In NLP, a classical way to incorporate context dependencies into a probabilistic model is by means of an n-gram language model (Chen and Goodman, 1996/98). This is also the solution of choice here.

Concretely we introduce a bigram language model on the source side. Since our main goal is word alignment we wish to use the language model only when necessary. Thus, our full model is a mixture of the language model and an IBM-style translation model.

Since we wish to enjoy the improvements already achieved by the Bayesian IBM models and at the same time be able to bias the mixture weights towards the translation component, we also chose to design a Bayesian model. The IBM component of our model is exactly equal to Equation (3.5). We add one set of latent variables and two sets of parameters to it.

The latent variables are Z_j with the index being a source sentence position. The variables are binary and indicate whether the source word in position j is generated by the translation model ($Z_j = 0$) or the language model ($Z_j = 1$). The parameters ψ_f are categorical parameters over the source vocabulary. Each source word is associated with such a distribution. This yields a bigram language model. Finally there are also parameters q_f which are Bernoulli parameters governing the mixture indicators. Since each bigram context has its own Bernoulli parameter, our model can capture the tendency of certain source words to be followed by words that do not have a lexical translation in the target language. This is why we like to think of our model as a collocation model.

Since our model is Bayesian, the parameters ψ_f are drawn from a Dirichlet distribution with symmetric parameter γ. The Bernoulli parameters q_f are drawn from a Beta distribution with parameters s_1, s_2.

The joint likelihood of the collocation-based alignment model is given in Equation (3.6). The likelihood can be obtained through marginalisation. We state the joint likelihood here as we will use this term when developing our inference algorithm (Section 3.4).

$$\begin{aligned} &p(\mathbf{f}_1^m, a_1^m, z_1^m, \theta, \phi, \gamma, q | \mathbf{e}_1^l, \alpha, \beta, \gamma, s_1, s_2) \\ &= p(\phi|\beta) \prod_f p(\psi_f|\gamma) p(q_f|s_1, s_2) \prod_e p(\theta_e|\alpha) \times \\ &\prod_{j=1}^{m} P(a_j = i|\phi) \left\{ P(Z_j = 0|q) P(f_j|\theta_{e_{a_j}}) + P(Z_j = 1|q) P(f_j|\psi_{f_j}) \right\} \end{aligned} \tag{3.6}$$

We also provide a graphical depiction of the model in Figure 3.1.

3.3 Bayesian HMM Alignment Model

In addition to extending the IBM models 1 and 2 with our collocation-based approach, we also extend the HMM alignment model of Vogel et al. (1996). This model generally tends to perform better than the IBM models and in fact replaces IBM 2 in many implementations (e.g. in Giza++, Och and Ney, 2003).

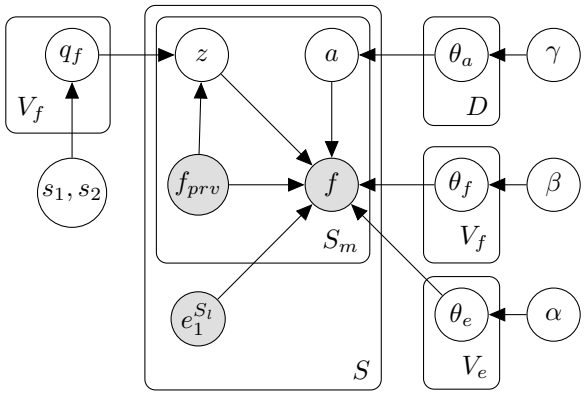


Figure 3.1: A graphical representation of our model for S sentence pairs. We use $V_{f/e}$ to denote the source/target vocabulary sizes and D to denote the number of possible alignment link configurations. Furthermore, $S_{m/l}$ is the number of source/target words in the current sentence and f_{prv} the source word preceding the one that we currently generate.

We have already described the maximum likelihood formulation of that model in Section 2.4.3. For convenience, we repeat the formulation of the transition distribution here.

$$P(a_j|a_{j-1},\phi) = P(a_j - a_{j-1}|\phi) \tag{3.7}$$

In the Bayesian formulation, we also put a Dirichlet prior on the transition parameters ϕ which replace the distortion parameters of IBM2. When we extend the HMM with our collocation-based model, the overall model is similar to Equation (3.5), however, the alignment links are not independent anymore.

$$\begin{aligned} &p(\mathbf{f}_1^m, a_1^m, z_1^m, \theta, \phi, \gamma, q|\mathbf{e}_1^l, \alpha, \beta, \gamma, s_1, s_2) \\ &= p(\phi|\beta) \prod_f p(\psi_f|\gamma) p(q_f|s_1,s_2) \prod_e p(\theta_e|\alpha) \times \\ &\prod_{j=1}^{m} P(a_j = i|a_{j-1},\phi) \left\{ P(Z_j = 0|q) P(f_j|\theta_{e_{a_j}}) + P(Z_j = 1|q) P(f_j|\psi_{f_j}) \right\} \end{aligned} \tag{3.8}$$

Notice that the HMM parameters are still global, meaning that there is only one set of them that is not conditioned on any other events. The distribution over alignment links thus remains fully structural and does not depend on any lexical information.

In our implementation of the HMM aligner without the collocation model, we chose to introduce a special event for jumps to NULL. This has two effects:

- The alignment jump distribution of the HMM does not get distorted as badly since long jumps to NULL are treated separately. This makes the model potentially more robust as it can focus its mass on jumps that are actually possible.

- The distribution over alignment links following NULL is uniform since the distance computation performed in Equation (3.7) does not apply anymore. This is because the NULL position is not associated with an integer in this formulation.

3.4 Inference by Gibbs Sampling

Inference in the Bayesian IBM models and our proposed extension of them is intractable. To see this, consider Bayesian IBM model 1. We need to compute two posterior marginals, one over alignment links and one over translation parameters. From here on we use the notation x_{-j} to denote all but the j^{th} outcome in a sequence of random outcomes.

$$P(a_j|a_{-j}, \mathbf{e}_0^l, \mathbf{f}_1^m, \theta, \alpha) = \frac{\int \prod_e p(\theta_e|\alpha) P(f_j|\theta_{e_{a_j}}) \mathrm{d}\theta}{\int \prod_e p(\theta_e|\alpha) \sum_{a_j} P(f_j|\theta_{e_{a_j}}) \mathrm{d}\theta} \tag{3.9}$$

$$p(\theta|a_1^m, \mathbf{e}_0^l, \mathbf{f}_1^m, \alpha) = \frac{\prod_e p(\theta_e|\alpha) \sum_j \sum_{a_j} P(f_j|\theta_{e_{a_j}})}{\int \prod_e \sum_j \sum_{a_j} p(\theta_e|\alpha) P(\mathbf{f}_j|\theta_{e_{a_j}}) \mathrm{d}\theta} \tag{3.10}$$

As we can see, both of these posterior marginals involve integrals over the categorical parameters of all translation distributions and summations over alignment links. Such high-dimensional integrals are extremely hard to compute analytically. To make inference feasible, we thus need to approximate the exact solution. Here, we choose to use MCMC sampling (see Section 2.3.1) as an approximation method.

3.4.1 The Gibbs Sampler

The Gibbs sampler is one of the simplest samplers to design and has found widespread applications in NLP (e.g. Goldwater and Griffiths, 2007; Heinrich, 2005). Since NLP models often use categorical likelihoods, the problem of designing a Gibbs sampler for inference in Dirichlet-categorical models has been well-studied. Here, I derive the Gibbs sampler for our collocation-based aligner. The Gibbs sampler for the standard Bayesian IBM models is a special case of it.

Since we are only interested in deriving the posterior over alignment links and less concerned with the parameters, we use a collapsed Gibbs sampler, meaning that we integrate over the parameters before starting to sample.[3] This integration makes the sampler notably faster since as it reduces its state space. Before the state space consisted of pairs of alignments and parameters, and after the

[3] Griffiths and Steyvers (2004) show how to recover parameter estimates even from a collapsed sampler. They treat their sample of latent data as observed and subsequently perform maximum likelihood estimation. While this reaps the benefits of both a collapsed sampler and knowing the posterior over parameters, it is not a valid Bayesian estimation technique.

integration it consists solely of alignments. However, at the same time it also makes the alignment links marginally dependent with the consequence that exact summation over alignment links becomes impossible (recall that it is usually possible in the IBM models 1 and 2).

It should be noted that collapsing the sampler is only possible because we are using a conjugate model. This means that we know the family in which the posterior lives and integration reduces to computing the mean. Since conjugate models are in the exponential family, posterior means can easily be found for most of them. We will soon see an example of non-conjugate inference (Section 3.4.2) in which integration over the parameters is not possible.

Recall that the general strategy in designing a Gibbs sampler is to sample a highdimensional variable (the corpus alignment in this case) dimension-wise. The intractable computation of the joint posterior can then be replaced by several computations of posterior conditionals which are assumed to be tractable.

In order to construct our sampler we first choose an arbitrary alignment link a_j and assume all other alignment links to be fixed. From those fixed alignment links we can analytically compute a conditional posterior distribution over the categorical parameters using Equation (3.4c). For the sake of the present exposition we only consider IBM model 1. We subsequently use this posterior as a prior on the likelihood of a_j (line (3.11b)). Notice that this computation is again conjugate and we can therefore integrate over the categorical parameters to obtain a posterior predictive distribution over alignment variable A_j which serves as one of the conditionals. This means that we have formed an analytically tractable conditional posterior that can be used by the Gibbs sampler.

$$P(a_j|a_{-j}, \mathbf{e}_0^l, \mathbf{f}_1^m, \alpha) \tag{3.11a}$$

$$\propto \int P(a_j)P(\mathbf{f}_j|\theta_{e_{a_j}})p(\theta|a_{-j}, \mathbf{e}_0^l, f_{-j}, \alpha)\mathrm{d}\theta \tag{3.11b}$$

$$= P(a_j) \int \theta_{f_j|e_{a_j}} p(\theta|a_{-j}, \mathbf{e}_0^l, f_{-j}, \alpha)\mathrm{d}\theta \tag{3.11c}$$

$$= P(a_j)\mathbb{E}\left[\theta_{f_j|e_{a_j}}|a_{-j}, \mathbf{e}_0^l, f_{-j}, \alpha\right] \tag{3.11d}$$

$$= P(a_j)\frac{\alpha + \sum_{k\neq j} \mathbb{1}_f(f_k)\mathbb{1}_e(e_{a_k})}{V_f\alpha + \sum_{k\neq j} \mathbb{1}_e(e_{a_k})} \tag{3.11e}$$

We only need to compute the expectation in line (3.11d) with respect to the French word that a_j aligns to since the other expectations are independent of a_j. Recall that V_f denotes the source vocabulary size. In a slight abuse of notation, the sum in Equation (3.11e) ranges over all alignment points in the corpus. This should remind the reader of the Dirichlet priors true power: it makes all alignment marginally dependent and thus enables the model to share information across sentence boundaries.

Sampling from the distribution computed in Equation (3.11) is straightforwardly obtained using inverse transform sampling since A_j only has support on the integers 1 to l (the English sentence length that may include 0 if a NULL word is present).

In the case of IBM model 2 and the HMM we also need to compute the posterior with respect to the distortion or jump parameters. Since these are categorical parameters as well, the computation is parallel to the one shown above. Furthermore, the distortion parameters are independent of the translation parameters (see Figure 3.1) and thus both integrals can be computed separately. The conditional predictive posteriors for IBM model 2 and the HMM are,

$$P(A_j = i|a_{-j}, \mathbf{e}_0^l, \mathbf{f}_1^m, \alpha, \beta) = \frac{\beta + \sum_{k \neq j} \mathbb{1}_c(i - \lfloor k\frac{l}{m} \rfloor)}{\beta \times V_a + T} \frac{\alpha + \sum_{k \neq j} \mathbb{1}_f(f_k)\mathbb{1}_e(e_{a_k})}{V_f\alpha + \sum_{k \neq j} \mathbb{1}_e(e_{a_k})} \tag{3.12}$$

$$P(A_j = i|a_{-j}, \mathbf{e}_0^l, \mathbf{f}_1^m, \alpha, \beta) = \frac{\beta + \sum_{k \neq j} \mathbb{1}_c(i - a_{k-1})}{\beta \times V_a + T} \frac{\alpha + \sum_{k \neq j} \mathbb{1}_f(f_k)\mathbb{1}_e(e_{a_k})}{V_f\alpha + \sum_{k \neq j} \mathbb{1}_e(e_{a_k})} \,. \tag{3.13}$$

We have used V_a to denote the number of all distortion events (including the jump to NULL where applicable). The constant T is the number of total alignment positions in the sentence (or, more generally, in the corpus).

Sampling the Mixture Variable Our collocation-based model presented in Section 3.2.2 additionally contains language model parameters and mixture indicators. The posterior for the language model can straightforwardly be computed in the same way as the alignment posterior in Equation (3.11e). The main difference is that the conditional posterior for the language model ranges over the entire French vocabulary,

$$P(F_j = f'|F_{j-1} = f, Z_j = 1, \gamma, \mathbf{z}_1^m) = \frac{\gamma + \sum_{k \neq j} \mathbb{1}_f(f_{k-1})\mathbb{1}_{f'}(f_k)z_k}{V_f\gamma + \sum_{k \neq j} \mathbb{1}_f(f_{k-1})z_k} \,. \tag{3.14}$$

Fortunately, all French words are observed and thus we never need to compute the entire posterior, just the probability of the observed French word f_j. This means that the language model component incurs only a small computational overhead. The posterior for the mixture variable is derived next.

We split the derivation into two parts, one where Z_j is set to 0 and one where it is set to 1. Whenever the mixture variable is set to 0, the alignment component is used and thus the likelihood is defined by the alignment component. If, however, the mixture variable is set to 1, the language model defines the likelihood. In that case, the contribution of the translation parameters can be neglected. Let us start with the case where the mixture variable is set to 0. To avoid clutter, we

collect most conditioning variables in the set $\mathcal{C} = \{a_1^m, \mathbf{e}_1^l, \mathbf{f}_1^m, \alpha, s_1, s_2\}$. Notice that we also collapse the Bernoulli parameters q.

$$p(Z_j = 0 | z_{-j}, \mathcal{C}) \propto \frac{s_1 + \sum_{k \neq j} \mathbb{1}_0(z_k)}{s_1 + s_2 + m} \frac{\alpha + \sum_{k \neq j} \mathbb{1}_f(f_k)\mathbb{1}_e(e_{a_k})\mathbb{1}_0(z_k)}{V_f \alpha + \sum_{k \neq j} \mathbb{1}_e(e_{a_k})\mathbb{1}_0(z_k)} \tag{3.15}$$

The first quotient follows directly from the fact that the mixture variable's Beta prior is nothing but a 2-dimensional Dirichlet distribution. Together with the fact that the Bernoulli parameters are independent of all other parameters, it follows that the integral is the same as in the case of the Dirichlet. The second quotient is simply the predictive posterior of the translation distribution. The predictive posterior of the distortion model does not figure in because it is the same for both values of Z_j.

The second outcome of Z_j triggers the use of the language model. The likelihood of the token f_j must thus be evaluated using that model.

$$p(Z_j = 1 | z_{-j}, \mathcal{C}) \propto \frac{s_2 + \sum_{k \neq j} z_k}{s_1 + s_2 + m} \frac{\gamma + \sum_j \mathbb{1}_f(f_{k-1})\mathbb{1}_{f'}(f_k) z_k}{V_f \gamma + \sum_j \mathbb{1}_f(f_{k-1}) z_k} \tag{3.16}$$

There are essentially two forces that determine whether the language model is used or not. First, there is the lexicalised probability that f_{j-1} be followed by a non-translating word. Second there is the likelihood ratio of the translation model and the language model. Whichever assigns higher probability to the current French token is more likely to be used.

In practice, the alignment links and mixture variables are sampled in turn. One pass over the data is made for each, such that one Gibbs sampling step consists of two passes over the data.

Finally, recall that one of the main reasons for choosing a Bayesian model was to allow us to bias the mixture towards using the alignment component and only using the language model if there is strong evidence for doing so. However, it is not entirely clear what exact values one should choose for the Beta parameter in order to produce this effect. While we may have some intuitions about reasonable values, it is usually less time-consuming to let the model decide itself. In order to do so we place Gamma priors on the shape parameters of the Beta prior. These Gamma priors are hyperparameters since they are not themselves random. However, we can optimise them by sampling them as well. In this case, our sampler performs optimisation instead of approximating an integral (see also Geman and Geman, 1984).

3.4.2 Hyperparameter Inference

We optimise the gamma parameters while running the Gibbs sampler. Notice that this changes the stationary distribution of the sampler slightly after each optimisation step. Moreover, since our optimisation procedure is random, we are

not guaranteed that any given step actually improves the setting of hyperparameters. Nevertheless, we found this method to work well in practice. There are also many precendents for it (e.g. Johnson and Goldwater, 2009; Blunsom and Cohn, 2011).

Unfortunately, we cannot use a straight-forward Gibbs sampler in this case because the Gamma distribution is not conjugate to the Beta and thus the predictive posterior is hard to compute. Instead we choose to use slice sampling. We use the stepping out and shrinkage procedures of Neal (2003) and use at most 10 steps. Whenever we cross 0 on the left side, we stop stepping out.[4] Hyperparameter updates are performed after a sample has been taken. This allows the Gibbs sampler, which samples the alignment variables, some time to come closer to equilibrium for the current hyperparameter setting.

3.5 Reducing Sampling Complexity

Bayesian word alignment models have shown remarkable improvements whenever they were applied (Mermer and Saraçlar, 2011; Gal and Blunsom, 2013). However, the Gibbs samplers used for inference in these models are prohibitively slow. Even on small data sets they may run for more than a day. The main bottle-neck for the sampling procedure lies in the fact that for each source word all target words in the aligned sentence need to be considered as alignment points.

Let l be the length of the target sentence and m the length of the source sentence. To compute the predictive posterior for the alignment link of a given source position, we need to compute its likelihood under all alignments, i.e. we need to perform l likelihood evaluations. Since we do this for *all* source word, the overall sampling complexity is $\mathcal{O}(lm)$. If we assume that both sentences are of the same length we immediately see that this makes sampling quadratic in time complexity. Since the average sentence length of standard WMT data sets is rather high, inference by Gibbs sampling becomes infeasible in practice.

In this section we seek to remedy this problem. In particular, we design a Gibbs sampler with linear time complexity per sentence. An appealing feature of our sampler is that inference speed can be traded off against convergence speed. This work was originally described in Schulz and Aziz (2016).

The main idea of our new sampler is to augment the existing Gibbs sampler with a set of auxiliary variables (Tanner and Wong, 1987). For each source position we uniformly draw one competing alignment point that is not the current alignment point. Once we have done this, we only compute the posterior over these two alignment points. Sampling complexity per source position thus becomes constant and the overall sampling complexity for a sentence pair is $\mathcal{O}(m)$. The number of competitors can be adjusted, of course. More competitors slow

[4] The Gamma has support on the positive reals and thus our posterior is guaranteed to be 0 on non-positive numbers.

down the computation of the posterior predictive distribution[5] but may lead to faster convergence since in principle they allow for a better exploration of the state space of the Markov chain.

A crucial property of the augmented sampler is the retention of the currently aligned position in the posterior. This ensures reversibility of the sampler. Reversibility is key to making sure that the sampler is ergodic (Besag, 2004).

We are now going to develop the auxiliary variable sampler formally. Let a_j be the alignment link that we wish to resample and let i be the current alignment point. We introduce an auxiliary random variable S_j over sets of integers of size k (where the value of k is fixed beforehand). We construct S_j by uniformly sampling k elements from $\{0, 1, \ldots, l\} \setminus \{i\}$ *without replacement*.[6] Of course, if we use the collocation-based model (Section 3.2.2) the set that we sample from does not include 0.

What we need to show now is that the stationary distribution of our sampler remains unchanged when using the auxiliary variable. A common way to achieve this in both the sampling literature (Tanner and Wong, 1987) and the variational inference literature (Agakov and Barber, 2004) is to define the joint posterior in such a way that marginalising the auxiliary variable respects the other marginals. In our case:

$$P(s_j, a_j|\mathcal{C}) = P(s_j|a_j)P(a_j|\mathcal{C}) \ . \tag{3.17}$$

Notice that this factorisation clearly leaves the posterior predictive distribution over a_j unchanged when we sum out S_j. Moreover, we have made very strict independence assumptions for S_j. The auxiliary variables are independent of each other and only condition on their associated alignment link.

Since the goal of our auxiliary variable method is to make inference faster, we want to choose a simple distribution over auxiliary variables. As outlined above, we choose a uniform distribution. Notice that if for models that contain a NULL word we have one more position that we can sample and hence the denominator increases accordingly.

$$P(s_j|a_j) = \prod_{i=1}^{k} \frac{1}{l - i(+1)} \tag{3.18}$$

The transition kernel for sampling a new value of a_j can consequently be split up.

$$\mathcal{K}(a'_j|a_j) = \sum_{s_j} \mathcal{K}(a'_j|s_j)\mathcal{K}(s_j|a_j) \tag{3.19}$$

The second kernel is defined by the distribution in Equation (3.18). The first kernel can be derived from the predictive posterior.

$$\mathcal{K}(a'_j|s_j) = P(a'_j|\mathcal{C}, s_j) \propto P(s_j|a'_j)P(a'_j|\mathcal{C}) \tag{3.20}$$

[5]Notice, however, that the sampling complexity remains linear as long as the size of the competitor set is fixed.

[6]If $l \leq k$ we simply use all available alignment positions, thus making $k = l$.

input: e_0^l, f_1^m, a_1^m, k
for $j \in \{1, \ldots, m\}$ **do**
  $s_j \leftarrow \{a_j\}$;
  $L \leftarrow \{0, 1, \ldots, l\} \setminus s_j$;
  while $k > 0$ *and* C *not empty* **do**
    $i \sim \mathcal{U}(C)$;
    $s_j \leftarrow s_j \cup \{i\}$;
    $L \leftarrow L \setminus \{i\}$;
    $k - -$;
  end
  $a_j \sim P(a_j = i|\mathcal{C})$ if $i \in s_j$;
end

Algorithm 1: One Sweep of the auxiliary variable Gibbs sampler.

Now clearly the equivalence in Equation (3.19) holds. This means that we have shown that the auxiliary variable leaves the posterior predictive distribution over alignment links unchanged. The augmented sampler is procedurally described in Algorithm 1. The only modification that we need to make when applying it to the collocation-based model is to remove the NULL position from the set of alignment positions. Other than that the sampler is fully general and can be used for all Bayesian alignment models presented so far. In fact, it can also be used for inference in more complex hierarchical models such as the ones proposed by Gal and Blunsom (2013). We only need to adjust the sampling process so as to make it suitable for hierarchical categorical models (see Teh et al., 2006).

3.6 Experiments

In this section I report the empirical results obtained with the different Bayesian models presented previously. The goal is to show improvements in translation quality as measured by BLEU as well as to demonstrate the speed-up attainable with the auxiliary variable sampler. Before I report on the experimental results I summarise the different aligners and the data sets.

3.6.1 Aligners

We use our own implementation of the Bayesian alignment models which is publicly available at `https://github.com/philschulz/Aligner`. The Bayesian models are:

- BIBM1: the Bayesian variant of IBM model 1
- BIBM2: the Bayesian variant of IBM model 2

Data	De	Fr	Cs
Train	200.608	182.327	146.144
Dev	2999	2997	2999
Test	3003	3003	3003

Table 3.1: Number of sentence pairs for each language paired with English.

- BHMM: the Bayesian variant of the HMM aligner
- model-Colloc: the above three models augmented with a language model component

Notice that all Bayesian aligners were run using the auxiliary variable sampler presented in Section 3.5. The initial state of the sampler was set to be the Viterbi alignment of the maximum likelihood version of IBM model 1 after 5 rounds of EM. The samplers were subsequently run for 1000 iterations.

To enable a comparison with known alignment models we use our own implementation of IBM model 2 as a baseline. Furthermore we report results obtained using 2 standard alignment tools:

- Giza++: the quasi-standard alignment toolkit that runs IBM model 1, the HMM and IBM models 3 and 4.
- fastalign: The aligner of Dyer et al. (2013). It uses a continuous distance measure presented in 2.4.3 in computing the distortion probability and a Dirichlet prior on the translation parameters. Inference is done using a mix of maximum likelihood estimation (for the distortion parameters) and variational Bayes (for the translation parameters).

3.6.2 Data Sets

We construct four language pairs for our evaluation by pairing English with German, French and Czech. The German, French and Czech data are the WMT 2014 news commentary data sets.[7]. Data statistics are given in Table 3.1.

3.6.3 Translation Systems and Preprocessing

We use the Moses machine translation toolkit[8] for translating the test data. The WMT data are tokenised and truecased. Sentences longer than 100 words are discarded. We use MERT (Och, 2003) to tune the weights of the log-linear decoder. All reported BLEU scores are averaged over 5 MERT runs.

[7] http://www.statmt.org/wmt14/translation-task.html
[8] http://www.statmt.org/moses/

The language models used by our systems are 5-gram language models trained with KenLM (Heafield, 2011) on the entire monolingual data available for WMT 2014.

The defaults of the Moses training scripts are used, however the directional word alignments are provided by the aligners which we seek to evaluate.

3.6.4 Translation Results

In Table 3.2 we report the translation results when only alignments from the target to the source language are used. In Table 3.3 the `grow-diag-final-and` heuristic of Och and Ney (2003) was used to arrive at symmetrised alignments. We report BLEU improvements relative to IBM model 2 trained using maximum likelihood estimation. The distortion parametrisation used for that model is the one of Brown et al. (1993). We boldface the best-performing model out of the Bayesian models in each column.

Unsurprisingly, the more advanced IBM2 model versions generally perform better. The performance of the HMM seems to suffer somewhat from symmetrisation. The collocation variants of our models perform best in the symmetrised scenario. This indicates that they not only lead to cleaner modelling but also have the potential to improve translation performance to a certain degree. Strikingly, they manage to outperform the more complex Giza++ on several occasions.

3.6.5 Analysis of the Auxiliary Variable Sampler

The main purpose of the auxiliary variable Gibbs sampler is to make Bayesian word alignment feasible for practical purposes. Here, we seek to explore the trade-off between speed and accuracy by comparing it to the standard Gibbs sampler. We only perform this analysis with the Bayesian IBM model 1 when aligning the German-English data set with German being the language that is generated in the generative model. This is because the standard Gibbs sampler is prohibitively slow for more complex models.

Timing Results We have shown in Section 3.5 that the improved sampler has lower time complexity than the standard Gibbs sampler. It is of course interesting to investigate how these theoretical improvements manifest themselves in practice. In Table 3.4 we show timing results for 3 runs of the auxiliary variable and standard Gibbs samplers. Notice that these results were obtained on a cluster whose nodes have different CPUs. This may explain the comparatively fast speed achieved in run 2 of standard Gibbs sampler.

The speed-up achieved by the auxiliary variable sampler is very pronounced even for this rather simple model. We figure that for more complex models this speed-up would speak even more in favour of the auxiliary sampler. The reason is that the computation of one cell in the categorical posterior predictive distribution

Model	En-De	En-Fr	En-Cs
IBM2	14.56	27.16	13.74
BIBM1	-0.09	-0.64	+0.38
BIBM2	**+1.07**	-0.17	**+1.76**
BHMM	+0.98	+0.99	+1.57
BIBM1-Colloc	-0.03	-0.79	-0.42
BIBM2-Colloc	+0.92	**+1.32**	+1.66
BHMM-Colloc	+0.95	+0.46	+1.58
Giza++	+0.96	+0.23	+1.58
fastAlign	+0.88	+0.70	+1.47

(a) Translations from English.

Model	De-En	Fr-En	Cs-En
IBM2	18.12	26.69	18.77
BIBM1	+0.32	-0.92	+0.66
BIBM2	+1.63	-1.04	+1.63
BHMM	+1.92	+1.65	**+2.44**
BIBM1-Colloc	+0.29	-1.49	+0.45
BIBM2-Colloc	+1.73	**+2.01**	+1.42
BHMM-Colloc	**+2.01**	+1.39	+2.43
Giza++	+2.27	+2.26	+1.96
fastAlign	+2.27	+1.90	+1.86

(b) Translations into English.

Table 3.2: Directional: The alignments are obtained in target to source direction.

table becomes more time-consuming as the models become more complex (that is as we add factors to their likelihood). Hence, the amount of time saved by the auxiliary variable sampler when computing each posterior predictive distribution grows with the complexity of the alignment likelihood.

Likelihood Comparison An often-used way to evaluate the movement of an MCMC sampler through state space is to trace the joint likelihood of each sample. We consider a sample to be the corpus with imputed alignments after all alignment links have been resampled. Thus, obtaining a sample requires one pass over the corpus. Since our samplers were run for 1000 iterations, we have 1000 samples. The trace plots for the likelihoods are shown in Figure 3.2. As we can see, the standard sampler reaches a low-energy configuration faster than the auxiliary variable sampler. The latter, however, steadily moves towards a low-energy

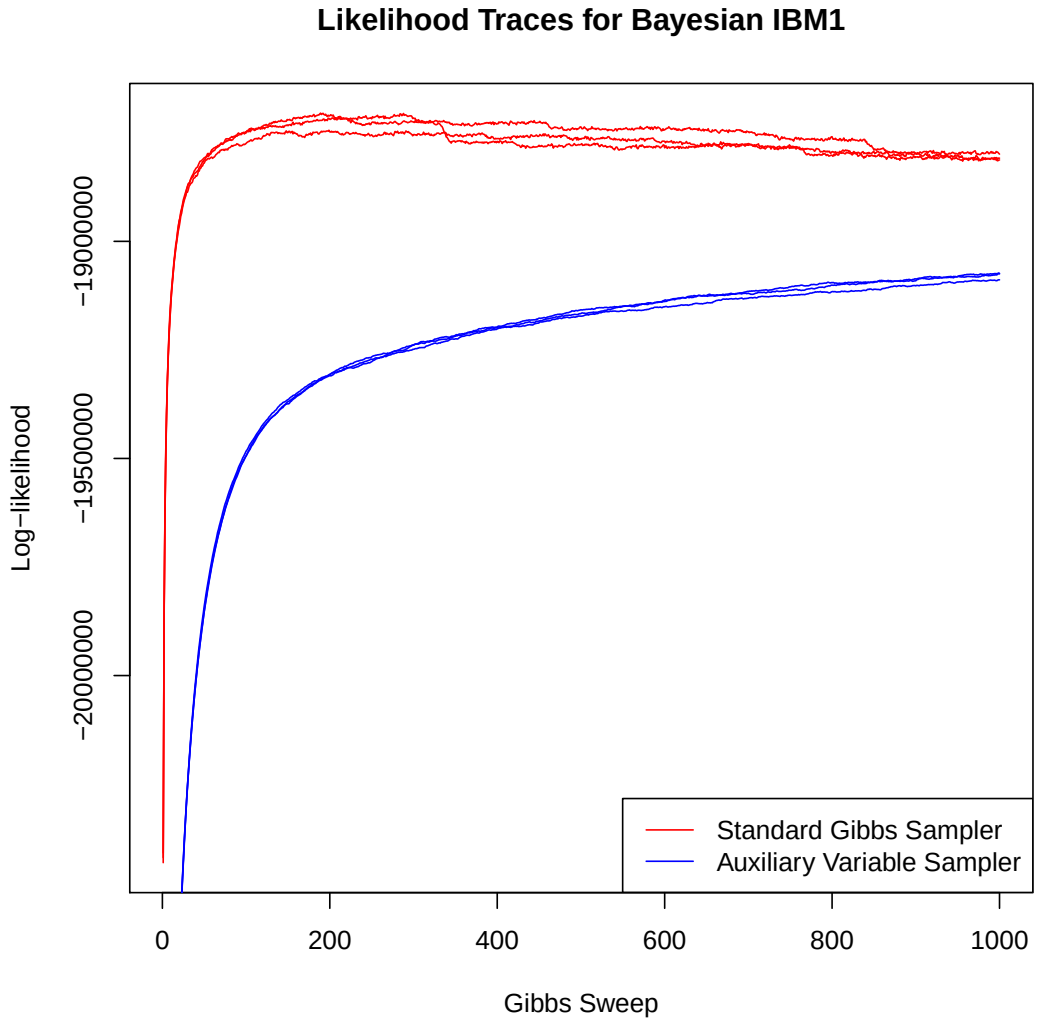


Figure 3.2: Likelihood traces for Bayesian IBM model 1. We performed 3 runs each with the standard and auxiliary variable Gibbs samplers.

region. It also shows less variance between runs. Overall, the auxiliary variable sampler leads to decreased inference performance which was not unexpected. It does, however, make up for this decrease by vastly accelerating the inference procedure. Also recall that we could make the auxiliary variable sampler perform closer to the standard sampler by increasing the number k of competitors. To which degree one prioritises speed or accuracy is an application dependent choice. Importantly, the new sampler can accomodate both demands.

3.7 To AER or not to AER?

The alignment error rate (AER) measure of Och and Ney (2003) is an often-used evaluation score for word alignment models. I have not reported it here for the reason is that improved AER likely does not correlate with improved translation quality (Fraser and Marcu, 2007). Here we are interested in investigating the effects that of our word alignment models on the downstream task of machine translation. If AER is not a good proxy for translation quality improvements, it does not contribute to our investigation.

There are also modelling motivations behind our choice. The latent alignments we seek to model are probably not the alignments detected by humans. This can be understood from the fact that the hand-aligned corpus of Och and Ney (2003) contains more probable than sure links. Modelling this distinction would be an interesting task in and of itself but is orthogonal to the modelling ideas presented here. It does show, however, that even human annotators can be highly unsure of their alignment choices. There is thus no reason to expect our models to make the same choices.

The last reason for which we do not report AER is that it is unclear that hand-labelled alignments are even useful to the translation task. If our model was able to perfectly reproduce human-generated alignments, it would still not guarantee an improvement in translation quality. It is perfectly conceivable that an alignment model whose output alignments are different form human- generated ones might lead to better translation quality in the end.

3.8 Related Work

Bayesian word alignment has been pioneered by Mermer and Saraçlar (2011); Mermer et al. (2013). It has been shown to be very effective, however, the inference procedures where often too slow. An extreme example of an elegant alignment model that uses very slow inference is Gal and Blunsom (2013). Speedups for Bayesian alignment models where achieved through variational inference in Riley and Gildea (2012); Dyer et al. (2013).

Common to all of the above alignment models is the assumption of a NULL word to model unaligned words. The idea presented here to build an explicit model of variation in alignment data has been further advanced by Rios et al. (2018) who assign an abstract latent variable to each alignment position and use a VAE to learn their model. On top of producing alignments, their model also generates (stochastic) bilingual word embeddings. Previous work by Cuong and Sima'an (2015) had already used domain latent domain indicators from a fixed set of possible domains to learn domain-dependent parameters for the HMM alignment model.

3.9 Summary

In this chapter I have presented a new class of word alignment models that do not need to stipulate a NULL word on the target side. Instead, they use a language model that generates untranslatable source words from their context. The choice of whether to use the alignment or language model component is made by a binary indicator variable. All parameters of the model's likelihood are drawn from prior distribution, thus making the model fully Bayesian. The prior on the binary indicator in particular allows us to control the degree to which we expect the language model to be used a priori.

The chapter also contributes a very fast Gibbs sampler that uses an auxiliary variable to select competing alignment points. This makes the sampling complexity for a given alignment link constant since only a fixed number of positions needs to be considered when computing the conditional posterior predictive distribution.

Experiments show that the new models do increase translation quality as measure in terms of BLEU and that the new sampler vastly improves the runtime of all Bayesian models.

Model	En-De	En-Fr	En-Cs
IBM2	15.98	27.63	16.02
BIBM1	-0.32	+0.03	+0.16
BIBM2	-0.21	-0.44	+0.80
BHMM	-0.47	+0.15	-0.48
BIBM1-Colloc	-0.43	-0.54	+0.17
BIBM2-Colloc	**+0.23**	**+0.70**	**+0.94**
BHMM-Colloc	-0.30	+0.25	-0.29
Giza++	+0.75	+0.10	+0.56
fastAlign	+0.55	+0.46	+0.43

(a) Translations from English.

Model	De-En	Fr-En	Cs-En
IBM2	21.16	29.18	21.22
BIBM1	-0.29	-0.17	+0.62
BIBM2	+0.22	-0.55	**+1.19**
BHMM	-0.89	-0.25	+0.09
BIBM1-Colloc	-0.19	-0.79	+1.72
BIBM2-Colloc	**+0.50**	**+2.19**	+0.98
BHMM-Colloc	-0.74	-0.02	-0.27
Giza++	+0.74	+0.31	+1.21
fastAlign	+0.64	+0.29	+1.06

(b) Translations into English.

Table 3.3: Symmetrised: alignments obtained in both directions independently and heuristically symmetrised (grow-diag-final-and).

Run	**Standard**	**Auxiliary Variable**
1	10h06m	01h37m
2	09h36m	01h36m
3	10h03m	01h33m

Table 3.4: Run times of the standard and auxiliary variable Gibbs samplers on 3 independent runs when aligning from English to German.

Chapter 4

Latent Variables For Neural Machine Translation

This chapter presents work on latent variable neural machine translation (NMT) models that has been published in Schulz et al. (2018). For a general introduction to NMT, the reader is referred to Section 2.5.

The topic of variation in translation continues in this chapter. In contrast to the preceding chapter, here we look at end-to-end translation and not just a specific sub-component. When translation is performed by humans, the generated translations may vary considerably. This has to do with all kinds of factors related to the identity of the translator (proficiency, social background, gender etc.) and also with the nature of the text. On the personal side, it seems obvious that an experienced translator will on average deliver better translations than a less experienced one. But even short-term influences may affect the translation. A translator may underperform due to fatigue, for example. Moreover, the particular wording and structure in a translation depends to some degree on a translator's individual taste. This may be less apparent in translations of legal text but becomes more prominent when one turns to literary text. As one can see from this argument, what kind of variation one can observe in translated text and to what extend also depends on the text genre.

From hereon I take it as a given that we observe a considerable amount of lexical and syntactic variation in translation data. Standard NMT models, that learn deterministic functions from source sentences and translation prefixes to distributions cannot account for this kind of variation, unless they receive explicit supervision.[1] I therefore propose to incorporate the underlying factors of variation in the model. This is done through a sequence of latent variables. Each output

[1] Sennrich et al. (2016) provide a case study that controls explicitly for variation due to politeness. This is an interesting avenue of research, however, it presents a challenge in terms of data acquisition. Getting annotations for large corpora is expensive. Consequently, Sennrich et al. (2016) used a simple heuristic that allows for automatising annotations but that a) is language-specific and b) surely misses a lot of instances.

word depends on this sequence. Depending on the setting of the latent variables, the translation output changes considerably (this is demonstrated in Section 4.5). The latent variables also depend on one another. This promotes consistency within the translation. Once a particular style is chosen at the beginning of the translation it is unlikely to change later on. Disruption inside a given translation is thus mitigated. The concrete formulation of the model is provided in Section 4.2.

Chapter Highlights

Problem Statement

- Neural Machine Translation models assume one output distribution that is conditioned only on the source sentence. They ignore other sources of output variation such as features pertaining to the translator or the text genre. Consequently, translations are often generic and do not convey the meaning of the source sentence adequately.

Contributions

- This chapter presents a stochastic decoder model for NMT that models variation on the word level. This contrasts with earlier approaches that attempt to model variation on the sentence level.
- It reports and analyses problems that occur during training and provide a solution whose utility is verified experimentally. The main problem when training deep generative models with strong decoders (such as NMT architectures) is that the latent variable may simply be ignored. An additional complication is that the variational posterior approximation may stay too close to the initial prior. Both problems are addressed.

4.1 Deep Generative Models

The research presented in this chapter is based on the idea of deep generative models. I therefore present the basics of deep generative models (DGMs) before turning to my own contribution. A taxonomy of DGMs has already been given in Section 2.6. Here, I am concentrating on variational autoencoders. The reason is that they a) easily allow to model discrete variables which is needed for text modelling and b) provide an easy way to perform posterior inference which we want to address the variability in the translations that we train on.

DGMs are graphical models in which the parameters of the conditional and marginal distributions are predicted by a neural network. The parameters of the

model are thus the neural network weights. Although the distributions specified by the graphical model may be simple exponential family distributions, DGMs can exploit the representational power of the underlying neural nets to model data points very precisely. The features induced by the neural net are shared between different inputs and through these features the DGMs can capture correlations between inputs even if its output density does not account for these correlations explicitly.

One problem that has long hampered the application of DGMs is the fact that it is not straighforward to optimise them with backpropagation. There are 2 reasons for this:

- Due to the non-linearities introduced by the neural network the optimisation manifold becomes very complex. Integrating over that manifold exactly is not analytically possible (see Betancourt, 2017). Thus we need to resort to approximate integration.

- When backpropagating gradients through the DGM we eventually have to differentiate the Monte Carlo estimator. The result of this differentiation step is itself not an integral, however. This means that while it is easy to approximate the model density by MC, it is not straightforward to compute gradient estimates.

The first problem is generally addressed by introducing a lower bound on the integral through the use of variational inference (see Section 2.3.2). But even the ELBO cannot be computed exactly in DGMs. Luckily, it can be efficiently approximated using Monte Carlo estimates (Schulman et al., 2015), often even with only one sample. This solution, however, leads to the second problem.

One solution to the unavailability of gradients in DMGs is to transform the variable of interest into one that is either parameter-free or whose parameters do not depend on the model parameters. We can then use the transformed measure for integration and re-express the integral as an expectation over gradients. Effectively this allows us to sample **stochastic gradients**. This idea has been formulated independently by Kingma and Welling (2014); Rezende et al. (2014); Titsias and Lázaro-Gredilla (2014).

Another way of obtaining stochastic gradients is through the *score function estimator* (Paisley et al., 2012; Ranganath et al., 2014). It is more general than the transformation-based sampling technique as it also applies to discrete variables. On the downside, its gradient estimates have much higher variance and necessitate the use of control variates (Paisley et al., 2012; Ranganath et al., 2014; Gregor et al., 2014). I will not consider it further in this chapter and instead will focus solely on transformation-based gradient samplers.

We formally derive the stochastic gradient technique for DGMs as follows: let Z be the latent variable and x the observation we wish to model. The probabilistic

formulation of a generative model with parameters θ is

$$p(x) = \int (p(x|z,\theta)p(z,\theta))\,\mathrm{dz}\ . \tag{4.1}$$

The ELBO for this model as in Equation 2.30 is given below. We use λ to denote the variational parameters.

$$\mathbb{E}_{q(z|\lambda)}\left[\log p(x|Z,\theta)\right] - \mathrm{KL}\left(q(Z|\lambda)\ ||\ p(Z|\theta)\right) \tag{4.2}$$

In the context of DGMs we usually prefer this formulation of the ELBO because we are mostly working with exponential family distributions. If both arguments of the KL divergence come from the same exponential family the KL term can be computed analytically and does not need to be approximated. See Appendix B.2 for details.

Updating the model parameters through stochastic gradient descent is straightforward since the stochastic gradient can be computed efficiently.

$$\frac{\partial}{\partial\theta}\left(\mathbb{E}_{q(z|\lambda)}\left[\log p(x|Z,\theta)\right] - \mathrm{KL}\left(q(Z|\lambda)\ ||\ p(Z|\theta)\right)\right) \tag{4.3a}$$

$$= \mathbb{E}_{q(z|\lambda)}\left[\frac{\partial}{\partial\theta}\log p(x|Z,\theta)\right] - \frac{\partial}{\partial\theta}\mathrm{KL}\left(q(Z|\lambda)\ ||\ p(Z|\theta)\right) \tag{4.3b}$$

The reason that this computation is easy is that the expectation does not depend on variables with respect to which we differentiate. Using the linearity of differentiation and expectation then allows us to exchange the two. In effect we get a doubly stochastic gradient (Titsias and Lázaro-Gredilla, 2014) – one source of stochasticity is the data sample (Robbins and Monro, 1951) and one is the MC sample of the latent value z. This is not possible when we differentiate with respect to the variational parameters since the expectation depends on the variables with respect to which we differentiate. Therefore, we need to find a way to transform the expectation such that its measure does not depend on the variational parameters.

This idea was formalised by Kingma and Welling (2014); Rezende et al. (2014); Titsias and Lázaro-Gredilla (2014). They proposed to transform the latent variable so as to make the expectation independent of the differentiation variable. Let h be the transformation that we use for this purpose. It needs to fulfil two requirements:

1. h must be invertible. This is a requirement for any transformation of continuous random variables.

2. h must be differentiable. This requirement ensures that we can correct for the change in volume induced by h. Incidentally, it also enables the use of backpropagation to compute the gradients in our model.

We are now in a position to define an outcome $\epsilon = h(z)$ whose density is derived from the standard change of variable technique for continuous random variables (Equation (4.4)). Notice that this implies that if we can compute the density of z we can also compute the density of ϵ. It is generally not the case, however, that we can also sample from that density.

$$p(\epsilon) = p(h^{-1}(\epsilon)) \left| \frac{d}{d\epsilon} h^{-1}(\epsilon) \right| = p(h^{-1}(\epsilon)) \left| \frac{d}{dz} h(z) \right|^{-1} \tag{4.4}$$

Overloading notation, I use $\frac{d}{dz}$ to denote the Jacobian of Z if Z is multivariate and $\left|\frac{d}{dz} h(z)\right|$ to denote the absolute value of the Jacobian determinant.

Gaussian Reparametrisation Finding a transformation that allows us to sample from the transformed density is generally problematic. A general strategy may be to transform the latent variable into some kind of standard variable Ruiz et al. (2016). This can most easily be done for the Gaussian, or in fact any other location-scale family. Let m and s be the location and scale. Standardising a location-scale variable then amounts to,

$$\epsilon = h(z, m, s) = \frac{z - m}{s} \ . \tag{4.5}$$

Notice that this is in fact an affine transformation. Any location-scale family is therefore closed under affine transformations, meaning that a) applying an affine transformation to a variable from that family yields another variable from the same family and b) any variable in the family can be reached from any other variable by an appropriate transformation.[2] In the case of the Gaussian, the location is the mean μ and the scale is the standard deviation σ. The resulting ϵ is standard normal, i.e. $\epsilon \sim \mathcal{N}(0, \mathrm{I})$. As is common in probability theory we denote the standard normal density by $\phi(\cdot)$.

Applying the Gaussian reparametrisation to the ELBO requires a change of the measure of integration and the infinitesimal. The transformed measure is the standard normal measure. The transformed ELBO is given in Equation (4.6). Importantly, its value does not change with the transformation. To make the derivation general, we denote the original distributions parameters (m and s in the location-scale case) by λ. Notice that λ may also be the parameters of a regression model (e.g. a neural network) that is used to predict the distribution's parameters. The interested reader is referred to Appendix B.1 for a derivation.

$$\mathbb{E}_{\phi(\epsilon)} \left[\log p(x| \underbrace{h^{-1}(\epsilon, \lambda)}_{z}, \theta) \right] - \mathrm{KL}\left(q(Z|\lambda) \,||\, p(Z|\theta)\right) \tag{4.6}$$

[2] I do not formally prove this statement here, however, it can be understood by noting that every variable is a transformation of the standard variable and that any variable can be transformed into the standard variable. Thus any variable can be reached by going through the standard variable.

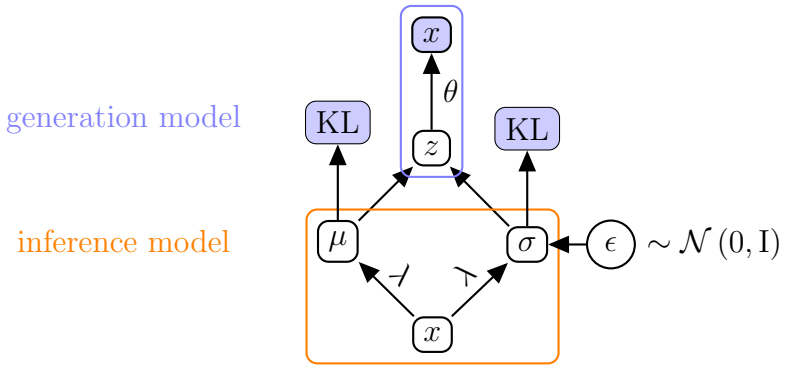


Figure 4.1: Stochastic computation graph for a VAE. Shaded nodes denote loss terms. The generation model is the graphical model that we want to train. The inference model is the neural network which predicts the posterior approximation. The parameter labels on the edges follow the parametrisation in the text. Unlabelled edges perform parameter-free computation.

The reason for not reparametrising the KL divergence is that we can compute it exactly if the two distributions come from the same exponential family.[3]

The transformation makes it possible to sample stochastic gradient estimates which can then be used to update the parameters during backprop.

$$\frac{\partial}{\partial \lambda}\left(\mathbb{E}_{\phi(\epsilon)}\left[\log p(x|\underbrace{h^{-1}(\epsilon,\lambda)}_{z},\theta)\right] - \mathrm{KL}\left(q(Z|\lambda)\;||\;p(Z|\theta)\right)\right) \tag{4.7a}$$

$$= \mathbb{E}_{\phi(\epsilon)}\left[\frac{\partial}{\partial z}\log p(x|\underbrace{h^{-1}(\epsilon,\lambda)}_{z},\theta) \times \frac{\partial}{\partial \lambda}h^{-1}(\epsilon,\lambda)\right] - \frac{\partial}{\partial \lambda}\mathrm{KL}\left(q(Z|\lambda)\;||\;p(Z|\theta)\right) \tag{4.7b}$$

Variational Autoencoders A famous example of a DGM is the variational autoencoder (VAE, Kingma and Welling, 2014). It is an autoencoder[4] whose code is a distribution instead of a point. To provide the reader with a better understanding of the computations performed in such a model, Figure 4.1 shows the stochastic computation graph (Schulman et al., 2015) for a basic VAE.

This basic scheme works well in practice even with one sample. It can also be generalised to other common continuous distributions through a form of generalised variable normalisation (Ruiz et al., 2016) or importance sampling (Naesseth

[3]Throughout this thesis we will assume that this is indeed the case. If it was not, we would approximate the KL divergence by sampling, as well.

[4]An autoencoder is a neural network that learns a bijection between data $x \in \mathbb{R}^n$ and codes $y \in \mathbb{R}^d$. Since the standard application of an autoencoder is data compression, we usually have $d \ll n$.

et al., 2017). Alternatively one can express the variable of interest as a transformation of a Gaussian variable *in the model* (as opposed to the variational distribution). The Gaussian variable can then be expressed as a transformation of a standard Gaussian during inference. This idea has been put forward by Kucukelbir et al. (2017). Finally, there have also been recent proposals that transform an initially Gaussian variable into an arbitrarily complex one through normalising flows (Rezende and Mohamed, 2015; Kingma et al., 2016).

4.2 A Stochastic Decoder for NMT

An important statistical assumption of most NMT models is that their output only depends on the input (most notably the source sentence but also additional information such as images or parse trees). In fact, the model invariably produces the same output distribution for a given input. This can be best understood when considering the probablistic model of NMT, which I repeat here.

$$P(e_1^l|f_1^m) = \prod_{i=1}^{l} P(e_i|f_1^m, e_1^{i-1}) \tag{4.8}$$

Given a parameter set θ, the distribution for a given input is computed deterministically. It is important to point out that the output of the NMT system (at least from a statistical perspective) is indeed this distribution and *not* any particular sentence. The mapping from the distribution to a sentence is done by a decision rule, such as beam search, that is chosen independently of the statistical system. Beam search, for example, is employed for both SMT and NMT systems and the beam search algorithm is agnostic to the underlying probabilistic model.

In this section I wish to address the question of whether this stringent assumption is a realistic one. After all, statistical models are meant to capture real-word phenomena (in this case human translation behaviour). Of course, we cannot hope to ever build an accurate model of complex real word processes (see Box (1979); Gelman (2016) for an in-depth discussion of this issue) but it seems counter-productive to equip our statistical models with assumptions that we can reasonably believe to be wrong.

Let us thus consider how human translators perform their task. It is almost guaranteed that for text of even moderate length, different translators will give us different translations. This fact is acknowledged by the machine translation community in the construction of their evaluation metrics which ideally compare one machine output against several different human translations (see Section 2.4.4). To get a better understanding of where these differences in translation come from, I review three sources of variation:[5]

[5]This list is not meant to be exhaustive nor do I claim that the split I chose is in any way superior to others. I simply list the sources of variation that seem the most plausible/intuitive to me.

- **Across-translator variation:** As pointed out above, different translators are almost bound to produce different translations. This variation may depend on the translator's gender (e.g.women tend to use more adjectives while men use more nouns, Johannsen et al., 2015). The education level and experience in the job are other factors which likely influence a translator's performance. Technological aids such as translation memories may also lead to vastly different translations by translators who use them as compared to those who do not use them.

- **Within-translator variation:** A given translator will produce different translations which may differ in quality on different days. This may depend on the familiarity that he has with his current assignment. One would expect a learning effect to set in if a translator is tasked with translating similar documents over a longer period of time. Even mundane influences such as exhaustion, mood or the familial situation of the translator may impact how he performs his job. Notice that these latter factors may vary within hours and to the extent that they influence the translations, their influence will be hard to capture with a statistical model due to their volatility.

- **Domain-dependent translation:** It is well-known that different textual domains require different translations. In particular, the translation style and lexical choice may depend heavily on the domain. While the domain is not directly related to the translator, it does prompt a variation in translation style within and across translators. Importantly, we would expect an interaction between textual domains and translators. A translator who excels in legal translation may be out of his depth if he was assigned a medical document.

I have identified three broad causes of variation in translation but there are certainly many more. The key contention here is that it is extremely difficult (if not impossible) to exclude the factors listed above or any other factors *a priori*. In other words, we simply don't know if and to what extent a specific factor influences human translation performance.[6] Standard NMT models are oblivious to this problem and in fact ignore all and any factors of variation (other than the source input). This of course weakens them considerably since they are not able to capture the random variation that we are sure to find in our translation data. As such, they are rather poor statistical models.

The kinds of variation that may occur in translation are diverse. Consider the following example where the English sentence has two valid German translations, with the verb being the only variation. It is not possible to change the tense or

[6]Solely considering the factors that we deem interesting will lead to essentially random outcomes when we try to assess their effects (Gelman and Loken, 2013).

personal pronoun if the semantic content of the English source sentence is to be preserved.

1. I went running.
   (a) Ich ging laufen.
   (b) Ich ging rennen.

Another example of variation is word order freedom. When translating a language with rather strict word order such as English into a morphologically richer language like German, the possible translations may vary considerably in their word order. I give an example below. Notice that the subject-verb agreement (underlined), dative case of the direct object (dashes) and accusative case of the indirect object (dots) need to be respected by all translations independent of word order.

2. I can't imagine you naked.
   (a) Ich kann mir dich nicht nackt vorstellen.
   (b) Ich kann dich mir nicht nackt vorstellen.
   (c) Dich kann ich mir nicht nackt vorstellen.

Stochastically encoding the sources of this variation helps an NMT decoder to disentangle the mandatory parts of the translation (such as case) from the randomly varying word order and thus has the potential to lead to a better model.

In Table 4.1 I present variation observed in real translations from the multi-reference Chinese-English corpus.[7] We again observe that this variation is indeed appreciable which further motivates our model development in the next section.

The research that I present next is aimed at addressing the problem of contextual variation. The basic assumption is that the translation variation can (to a certain extent) be captured as Gaussian noise that influences the dynamics of the decoder LSTM. This is of course a rather crude assumption. Importantly, it conflates the factors of variation listed above into one common source of noise. However, I do believe that this is an important step towards modelling translation data more accurately.

4.2.1 Conditional VAE and Stochastic RNN

Modelling word-level variation in the target translation could be addressed by letting the noise source be i.i.d variables such as in the STORN model of Bayer and Osendorfer (2015). Such a model, however, does not do the nature of language

[7] https://catalog.ldc.upenn.edu/ldc2002t01

The hearing is expected to last two days. The hearing will last two days. The hearings are expected to last two days. It is expected that the hearing will go on for two days.
However, the Republican complainant in the House wanted to summon 15 people including Lewinsky to testify in court. The prosecutor of Republican Party in House of Representative hoped to summons more than 15 persons, including Lewinsky, to court. The House of Representatives republican prosecution hopes to summon over fifteen witnesses including Monica Lewinsky to appear in court.

Table 4.1: Examples from the multiple-translation Chinese corpus (`LDC2002T01`), where the translations come from different translators. These demonstrate the lexical variation of the verb and variation between passive and raising structures (top), and lexical variation on the agent NP (bottom). Both examples also exhibit appreciable length variation.

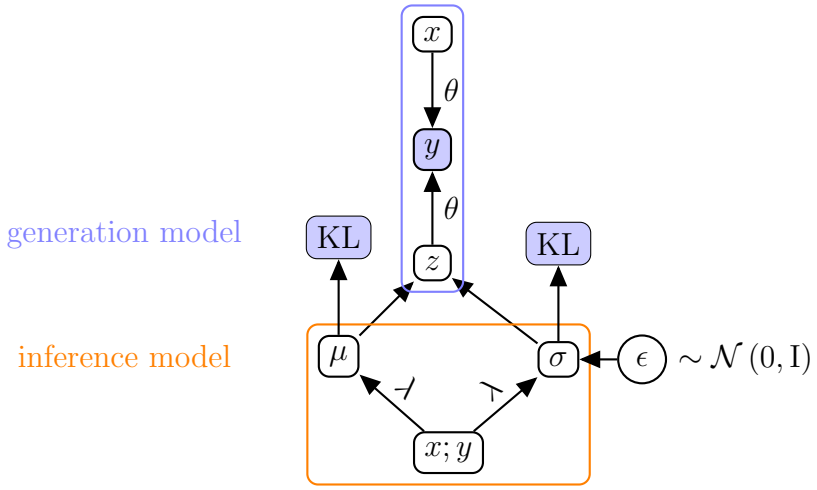


Figure 4.2: Stochastic computation graph for a conditional VAE.

justice as it neglects the contextual influence of an already produced translation prefix on the words that are to follow.[8] To capture this contextual influence, we have to make the latent variables dependent on their predecessors.

We do this by conditioning latent variable distributions on the translation prefix. This means that our model contains conditional priors over latent variables (much like an HMM). The general idea of conditional DGMs has been presented in Sohn et al. (2015). To better understand it compare the computation graphs in Figures 4.1 and 4.2. In the latter, both the variational approximate posterior and the generative model have access to side information x (in the case of NMT the source sentence). During training, the inference model takes a fully observed target sentence y as input. This means it can "look ahead into the future" and condition on target words that need yet to be modelled by the generation model. This obviously makes the posterior approximation very informative about future words. Since the KL-term in the ELBO encourages the prior to be close to the posterior, the prior may eventually learn to encode information not only about the current latent state but also about future states. This can make the generative model extremely powerful.

There are several ways of incorporating conditional VAEs into recurrent networks such as LSTMs. One is to have to separate recurrent layers, one stochastic and one deterministic, which both influence the models output. Such an approach has been proposed by Fraccaro et al. (2016). It makes inference very easy since we can first process the sequence with the deterministic model and then condition the inference of the latent state on the deterministic output during training. This also instantiates a "look-ahead" mechanism in which the variational distribution conditions on the translation suffix that still needs to be processed by the generator. Our hope, however, is to capture variation in the dynamics of translation (as modelled by an LSTM) instead of in the output. We therefore opt for a hybrid between the models of Fraccaro et al. (2016) and Chung et al. (2015). The latter work used the latent state to update the decoder state but only conditioned on the prefix and the current output. In our model the stochastic state directly influences the update of the decoder state but has no direct connection to the output. Unlike Chung et al. (2015) its inference network conditions on future words and its prior conditions on the previous latent value in addition to the decoder state. The details of our model are given in the next section.

4.2.2 Model Formulation

Before I introduce my own contribution, I repeat the basic architecture of an attention-based recurrent encoder-decoder model (Bahdanau et al., 2014) here

[8] Indeed, the model of Bayer and Osendorfer (2015) did not work well for other sequence modelling tasks, either.

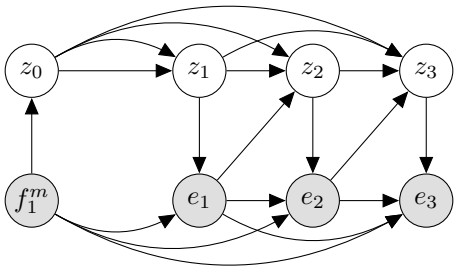


Figure 4.3: Graphical representation of the stochastic decoder model. Through the recurrent net, the model also conditions its outputs on all previous latent assignments. We omit these arrows to avoid clutter.

for the reader's convenience.

$$\left[h_1, \ldots, h_m\right] = \text{RNN}\left(f_1^m\right) \tag{4.9a}$$
$$\tilde{t}_i = \text{RNN}\left(t_{i-1}, e_{i-1}\right) \tag{4.9b}$$
$$d_{ij} = v_a^\top \tanh\left(W_a[\tilde{t}_i, h_j]^\top + b_a\right) \tag{4.9c}$$
$$\alpha_{ij} = \frac{\exp\left(d_{ij}\right)}{\sum_{j=1}^{m} \exp\left(d_{ij}\right)} \tag{4.9d}$$
$$c_i = \sum_{j=1}^{m} \alpha_{ij} h_j \tag{4.9e}$$
$$t_i = W_t[\tilde{t}_i, c_i]^\top + b_t \tag{4.9f}$$
$$\phi_i = \text{softmax}(W_o t_i + b_o) \tag{4.9g}$$

We are now ready to formally define the new model which I term the *stochastic decoder* model. Its likelihood is,

$$P(e_1^l | f_1^m) = \int p(z_0 | f_1^m) \prod_{i=1}^{l} p(z_i | z_0^{i-1}, e_1^{i-1}, f_1^m) P(e_i | z_0^i, e_1^{i-1}, f_1^m) \mathrm{dz}_0^1 \ . \tag{4.10}$$

where

$$z_0 | f_1^m \sim \mathcal{N}\left(\mu_0, \sigma_0^2\right) \tag{4.11a}$$
$$z_i | z_0^{i-1}, e_1^{i-1}, f_1^m \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right) \quad (i > 0) \tag{4.11b}$$
$$e_i | z_0^i, e_1^{i-1}, f_1^m \sim \text{Cat}\left(\phi_i\right) \ . \tag{4.11c}$$

A graphical representation of the model is given in Figure 4.3. Due to the recurrence in the decoder RNN, all variables depend on their predecessors. The initial latent state z_0 is conditioned on the source sentence and is meant to capture

translation variation solely based on the input. All other latent states should account for target-context-dependent variation. Notice that the initial variable Z_0 is the only latent variable present in the model of Zhang et al. (2016) which to the best of my knowledge has so far been the only model that has attempted to model latent sources of variation in NMT. The stochastic decoder is clearly more expressive since latent values may vary per target position.

To make our model concrete we need to decide how the underlying neural network computes the parameters of the conditional distributions. To that end we introduce functions f_{μ_0}, f_{σ_0} and $f_{\mu_i}, f_{\sigma_i}, i > 0$. The first two functions compute the parameters of Z_0 while the latter two compute the parameters of the remaining Z_i. The reason we use different functions is that their input dimensions differ. Notice that at each position i the parameters are different because the inputs differ. Each of these functions is implemented as a single hidden layer neural net whose hidden layer has a tanh activation function. The Gaussian parameters are computed as follows,

$$\mu_0 = f_{\mu_0}(h_m) \qquad \sigma_0 = f_{\sigma_0}(h_m) \tag{4.12a}$$

$$\mu_1 = f_{\mu_1}(t_{i-1}, z_{i-1}) \qquad \sigma_1 = f_{\sigma_1}(t_{i-1}, z_{i-1}) \ . \tag{4.12b}$$

The concrete implementation of our model also depends on how we choose to feed the latent input into the decoder. Inspired by Chung et al. (2015), we provide it as an additional argument to the RNN inside our decoder. This results in a simple change of the computations shown in Equation (4.9).

$$\tilde{t}_i = \text{RNN}(t_{i-1}, y_{i-1}, z_i) \tag{4.13}$$

Notice that this choice of update justifies the name *stochastic decoder*. Through its dependence on the latent variables, the decoder state is itself stochastic. Applying this argument recursively shows that the attention mechanism is stochastic, as well. We have thus designed a fully stochastic decoder.

The downside of that stochasticity is that the integral in Equation (4.10) is not analytically computable. If we want to generate from the stochastic decoder, we have to draw latent variable samples. This can easily be achieved with the inverse transformation of Equation (4.5). This inverse defines a mapping from the standard Gaussian variable to any other Gaussian variable.

$$z = h^{-1}(\epsilon, m, s) = m + s \odot \epsilon \qquad \epsilon \sim \mathcal{N}(0, \mathrm{I}) \tag{4.14}$$

Here, $\odot$ denotes the Hadamard product. We use Equation (4.14) to draw all samples from our model and the inference network described in the next section.

4.2.3 Inference Model

We train the model using variational inference (see Section 2.3.2). Since we are dealing with a DGM we only consider the transformed ELBO of Equation (4.6).

I repeat it here as applied to our model at target position $i > 0$. Recall that θ are the generative parameters and λ are the variational parameters.

$$\begin{aligned}\text{ELBO}_i = \mathbb{E}_{\phi(\epsilon)}\left[\log P\left(e_i|f_1^m, e_1^{i-1}, h^{-1}\left(\epsilon_0^i, \lambda\right), \theta\right)\right] \\ - \text{KL}\left(q(Z_i|\lambda) \;||\; p\left(Z_i|f_1^m, e_1^{i-1}, z_0^{i-1}, \theta\right)\right)\end{aligned} \tag{4.15}$$

The expectation of the reconstruction term is approximated with a single sample from a standard normal distribution.[9] Most NLP models that employ DGMs use only a single latent variable (Bowman et al., 2016; Zhang et al., 2016) or assume independence between their latents (Zhou and Neubig, 2017). Our model is different in that it stacks conditional DMGs (Sohn et al., 2015). This leads to a nested ELBO similar to the one employed by Rezende et al. (2014).

$$\text{ELBO} = \text{ELBO}_0 + \mathbb{E}_{q(z_0|\lambda)}\left[\text{ELBO}_1 + \mathbb{E}_{q(z_1|z_0,\lambda)}\left[\text{ELBO}_2 + \ldots\right]\right] \tag{4.16}$$

Next, we compute the KL-term which depends on the distributions used. Here we use fully factorised Gaussian distributions for both the conditional priors and the variational approximation. In other words the covariance matrices of the Gaussians only contain variance terms on their diagonal and all off-diagonal elements are zeros. The KL-divergence between two such Gaussians is thus equivalent to the KL-divergence of a product of k independent univariate Gaussians, where k is the dimensionality of the latent variable. We drop the position index from the KL-term to enhance readability and use $\mu_q = \mathbb{E}_{q(z|\lambda)}[Z]$ and $\sigma_q = \sqrt{\text{var}_q(Z)}$ and likewise for the model distribution p. See Appendix B.3 for a detailed derivation.

$$kl = \frac{1}{2}\sum_{j=1}^{k}\left(-\log\left(\frac{\sigma_{qj}^2}{\sigma_{pj}^2}\right) - 1 + \frac{\sigma_{qj}^2}{\sigma_{pj}^2} + \frac{\left(\mu_{qj} - \mu_{pj}\right)^2}{\sigma_{pj}^2}\right) \tag{4.17}$$

Since for a given prefix the KL-term only depends on the distributions parameters (as it does for any pair of distributions from the same exponential family) we can compute it exactly.

Now that we have established the ELBO we can turn to our implementation of the inference network. As pointed out before, it is crucial to the success of our model that the inference network can make use of information from the future. At the same time we want it to share information with the generation model. This is for two reasons: a) it reduces the number of parameters necessitated by the inference model (and therefore also reduces the amount of computation performed) b) information sharing makes it easier for the conditional prior to match the approximate posterior.

[9] It may certainly be desirable to draw more samples. This would imply copying each batch for each sample, an operation that our hardware resources unfortunately do not support.

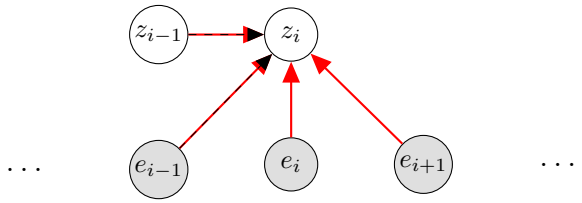


Figure 4.4: Graphical representation of the inference model used in the stochastic decoder. Red lines indicate variational parameters and dashed red-black lines indicate that feature representations are computed using the model parameters. Those feature representations are then fed into the Gaussian sampler of the inference model. The inference model is only used at training time. Dots indicate further conditioning context.

We start from a graphical representation of the inference model which is given in Figure 4.4. It is crucial to point out that the inference model is only available during training, i.e. when the target sentence is fully observed. The feature representation for the previous word is the previous decoder state. The dependence on future target words is introduced through a backward LSTM. Analogous to the generative model, the inference model uses two Gaussian samplers: one for Z_0 and one for the remaining latent states. The representations computed by the backward and bidirectional target LSTMs, which we denote by b and r, respectively, are shown below.

$$\left[b_1, \ldots, b_n\right] = \text{RNN}\left(e_1^n\right) \tag{4.18a}$$
$$\left[r_1, \ldots, r_n\right] = \text{RNN}\left(e_1^n\right) \tag{4.18b}$$

Like the generative model, the inference network uses single hidden layer networks to compute the mean and standard deviations of the latent variable distributions. We denote these functions g and again employ different functions for the initial latent state and all other latent states.

$$\mu_0 = g_{\mu_0}\left(h_m, b_n\right) \tag{4.19a}$$
$$\sigma_0 = g_{\sigma_0}\left(h_m, b_n\right) \tag{4.19b}$$
$$\mu_i = g_{\mu}\left(t_{i-1}, z_{i-1}, r_i, e_i\right) \tag{4.19c}$$
$$\sigma_i = g_{\sigma}\left(t_{i-1}, z_{i-1}, r_i, e_i\right) \tag{4.19d}$$

As before, we use Equation (4.14) to sample from the variational distribution.

A further note on the backpropagation algorithm is in order. If we applied backpropagation without any restrictions, ELBO gradients from the inference model would be used to update model parameters. This is a consequence of feeding inputs that depend on model parameters into the Gaussian samplers of the inference network. This behaviour is undesirable as it would optimise the model parameters θ for both the model and the inference network. Recall that the

performance of the inference network depends on information that the model has no access to. Optimising θ with respect to this information is thus utterly useless and potentially harmful. We prevent updates to θ with variational gradients by blocking the gradient flow from the inference model into the generative model in our implementation.

4.3 Problems in Learning DGMs with Strong Generators

The original work on DGMs by Kingma and Welling (2014); Rezende et al. (2014) focused on image modelling where the generative model assumed all pixels to be independent. This is of course a unrealistic assumption since neighbouring pixels in an image tend to be similar. The only way their models could capture information about pixel contexts was through the latent variable. In optimising the ELBO, the model's approximate likelihood (the reconstruction term) could thus be improved by learning an informative variational approximation that captures as much information as possible about the global structure of the image. In other words: the ELBO could be increased by moving the variational approximation away from the prior, even though this would at the same time increase the KL-divergence of the prior from the approximation.

The model that I have presented in the previous section is different, however. Its attention-based decoder is an excellent density model to start with. In particular, it makes good use of the contextual information provided by the source sentence and the target prefix. This means that this model does not necessarily need to rely on the information provided by the latent states in order to achieve a good fit to the data. This is especially true at the early stages of learning when the posterior approximation is still at a high energy level and contains very little information about the data. In order to achieve a high ELBO, the learning algorithm can thus simply set the approximate posterior equal to the conditional prior, thereby minimising the KL-term. The minimisation of the reconstruction term can then be achieved by solely training the generative model without regard for the latent state. This behaviour is particularly troublesome in our case, where the initial conditional prior does not capture any prior information or expected model behaviour but simply depends on the random initialisation of the model parameters. Because the conditional model priors are trained through the KL-term, they will not adjust if that term is minimal from the beginning.

This is a learning challenge that was neglected for some time because early DGMs mostly used **weak generators**, i.e. generators with strong, unrealistic indepenence assumptions that had no hope of fitting the data well by themselves. For models with **strong generators** (Alemi et al., 2017), such as the stochastic decoder model for NMT, we thus need to find ways of modifying the training procedure so as to encourage the model to make use of the latent variable and

learn useful variational approximations.

Several works have recently focused their attention on this problem. The solutions include downscaling[10] of the KL-term with an iterative increase of the scaling factor (Bowman et al., 2016) and choosing a threshold value greater than 0 as a fixed minimum for the KL-term (Kingma et al., 2016).[11] Yet another solution has been proposed by Chen et al. (2017) and Alemi et al. (2017). They suggest that one might simply decrease the generative capacity of the generator by introducing additional independence assumptions. While this is certainly true, it seems a little perverse to hurt a good generative model just to force it to make use of latent information. After all, the latent information is stipulated by the modeller but there is no direct evidence for it in the data. If a good density model cannot corroborate the modeller's stipulations, it would make sense that he rethink his assumptions instead of choosing a weaker model. Alemi et al. (2017) also suggest that using dropout in the deterministic but not the stochastic parts of the model may force it to make better use of the latent information since at each iteration the generative capacity of the model is randomly reduced. This appears to be a much sounder suggestion since dropout is only used at training time and does not affect the actual model structure. Moreover, it is already widely used and can thus easily be integrated into DGMs.[12]

4.3.1 Formal Analysis of Strong Generators

I now proceed to a more formal analysis of why strong generators are prone to ignoring the latent state. My analysis based on the mutual information (MI) between the latent variable and the data is closely related to the analyses of Chen et al. (2017) and Alemi et al. (2017) which both make use of MI, as well. Chen et al. (2017) never use the specific term MI, however their argument that the bits captured by the latent encoding (variational) distribution $q(z|y)$ can be deducted from the data encoding cost if $q(z|y)$ is known to the receiver is mathematically equivalent what I have presented below.

[10] This technique is often referred to as KL-annealing. Annealing means putting a physical system into a heat bath and slowly cooling it down (Kirkpatrick et al., 1983). The DGMs are two representations of physical systems: the generator and the inference network. The generator can be split up into a prior and likelihood. If the KL was annealed, we would put the prior into the heat bath but not the likelihood. The real problem, however, is the following: if we anneal the variational approximation (inference model) we would also need to sample from the annealed distribution. But we only sample from its non-annealed version (at temperature 1). KL-annealing is thus a misnomer and scaling (or something similar) is to be preferred.

[11] This technique is called *free bits* as it artifically introduces a certain amount of information (measured in bits) into the code for the observed data that cannot be removed (even if the learned model was perfect).

[12] Alemi et al. (2017) also propose to downscale the KL-term to a fixed level during training. This is motivated by their analysis that shows that the ELBO objective does not discriminate between parameter settings with the same ELBO value but different reconstruction and KL-terms. I do not discuss this proposal here.

The mutual information is the amount of information that two random variables Z, Y contain about each other. It is defined as

$$I(Y;Z) = \mathbb{E}_{p(y,z)}\left[\log \frac{p(Y,Z)}{p(Z)p(Y)}\right] \tag{4.20}$$

It follows from the definition that MI is symmetric. Interestingly, we can rewrite MI as the KL divergence of a distribution that assumes independence between Y and Z from their joint density.

$$I(Y;Z) = \mathrm{KL}\left(p(Y,Z) \;||\; p(Z)p(Y)\right) \tag{4.21}$$

If the joint density can be factorised into the marginals, this KL and thus the MI are 0 and the two variables do not share information about each other.

Next, we use the definition of conditional probability to rewrite the MI yet again.

$$I(Y;Z) = \mathbb{E}_{p(y,z)}\left[\log \frac{p(Z|Y)}{p(Z)}\right] \tag{4.22}$$

$$\mathrm{KL}\left(p(Y,Z) \;||\; p(Z)p(Y)\right) = \mathbb{E}_{p(y)}\left[\mathrm{KL}\left(p(Z|Y) \;||\; p(Z)\right)\right] \tag{4.23}$$

This rewriting leads to one of the best-known properties in information theory, namely that

$$I(Y;Z) = \mathbb{H}\left(Y\right) - \mathbb{H}\left(Y|Z\right) \;. \tag{4.24}$$

This means that (on average) additional information never increases uncertainty. If we have a scenario in which we want to communicate Y, side information given by Z potentially reduces the encoding cost of Y.

In our context it suffices to recognise that the MI is closely related to the KL-term in the ELBO. Since we are not able to compute the posterior $p(z|y)$ in models for which we use VI, we approximate it with $q(z|y)$. To the extent that $q(z|y)$ is a good approximation to $p(z|y)$, the MI is recovered.

At this point recall that MI is a measure of the information shared between two random variables. Ideally, we would like our DGMs to induce high MI between the data and the latent state. This would mean that the latent state is indeed useful. Unfortunately, the ELBO can be partially maximised[13] by minimising the KL-term and thus the MI between the data distribution and the distribution over latent states. The ELBO objective therefore discourages high MI.

The only way to increase the MI at the same time as the ELBO is to to change the variatonal parameters in such a way that the reconstruction term increases more than the KL-term. Only then does the ELBO increase as a whole. For

[13]By partial maximisation I mean that given a fixed reconstruction term, the ELBO is maximised by minimising the KL-term. This is merely a theoretical consideration, though, since any change to the ELBO will also induce a change in the reconstruction term.

this to happen, however, the variational parameter space needs to be explored in order to find a parameter setting that provides latent distributions that lead to an increased data likelihood. Such an exploration means that the variational distribution needs to move away from the prior. This is difficult if the KL-term is taken into full account when computing the ELBO.

The solution proposed by Bowman et al. (2016) of scaling the KL-term addresses exactly this issue. Initially, the variational parameters are optimised primarily to yield a high data likelihood. The variational parameter space can be explored without being constrained by the prior at this point. Only at later iterations, when the scaling factor approaches 1, does the constraint become effective.

Something interesting happens at this stage, however. Assume that the variational approximation provides useful latent states to the generation model (which it should do after sufficiently long training at low KL scales). The generation model's parameters are tuned to rely on the information encoded by the latent state, i.e. the latent distribution and data distribution of the model have high MI. It would now harm the reconstruction term (and thus the ELBO) to move the variational approximation back to the prior. Instead the prior is adjusted to match the variational distribution (notice that the prior has hardly been trained at low KL scales and is thus still close to its random and uninformative initialisation). This has two positive effects.

- The variational approximation can maintain high MI with the generator's output distribution. This means that during further training the generator will still make use of the latent state.
- In the case of recurrent models such as the stochastic decoder for NMT, the prior may implicitly capture information about future contexts even though it was never conditioned on those. This is because the variational approximation may condition on the future. The prior may then learn to emulate this "look-ahead" mechanism by matching the variational distribution.

The above discussion gives a sound theoretical motivation to KL scaling and also describes in which contexts we would expect it to be necessary (namely whenever the generator is a strong density model). While it was framed mostly in terms of unconditional DGMs, it is straightforward to generalise to conditional ones.

4.4 Related Work

Before I turn to the experiments, I briefly relate prior and related work, much of which has already been mentioned in passing.

The stochastic recurrent decoder can be seen as a concatenation of interdependent conditional deep generative models (Sohn et al., 2015). Other forms of

Data	Arabic	Czech	French	German
Train	224,125	114,389	220,399	196,883
Dev	6,746	5,326	5,937	6,996
Test	2,762	2,762	2,762	2,762

Table 4.2: Number of parallel sentence pairs for each language paired with English for IWSLT data.

recurrent deep generative models have previously been presented in Bayer and Osendorfer (2015) who model all random variation by i.i.d. standard Gaussian variables. Chung et al. (2015) improve upon this by making the latent variables depend on the previous variables and outputs through the RNN state. Their inference model does not condition on future outputs, however. This innovation stems from Fraccaro et al. (2016) who use a separate layer of latent variables that is independent of the RNN states. The deterministic RNN states and the latent states are combined to make output predictions. Their inference model leverages the fact that the deterministic states can be computed for the entire sequence before considering the random states. Their inference model takes future outputs into account by running a backward RNN over these deterministic states. Recently, Goyal et al. (2017) reported excellent results for an auxiliary loss that is formed by predicting the future RNN state from the latent state. While appealing in principle, this loss is hard to motivate from a probabilistic perspective.

In machine translation, DGMs have recently become more common, following the early work of Zhang et al. (2016) and the research reported here. Examples are Shah and Barber (2018), Ma et al. (2019) and Ataman et al. (2020).

In the wider field of NLP, deep generative models have been applied mostly in monolingual settings such as text generation (Bowman et al., 2016; Semeniuta et al., 2017), morphological analysis (Zhou and Neubig, 2017), dialogue modelling (Wen et al., 2017), question selection (Miao et al., 2016) and summarisation (Miao and Blunsom, 2016).

4.5 Experiments

We report experiments on the IWSLT 2016 data set which contains transcriptions of TED talks and their respective translations. We trained models to translate from English into Arabic, Czech, French and German. The number of sentences for each language after preprocessing is shown in Table 4.2.

The vocabulary was split into 50,000 subword units using Google's sentence piece[14] software in its standard settings. As our baseline NMT systems we use

[14] https://github.com/google/sentencepiece

Sockeye Hieber et al. (2017)[15]. Sockeye implements several different NMT models but here we use the standard recurrent attentional model described in Section 2.5. We report baselines with and without dropout (Srivastava et al., 2014). For dropout a retention probability of 0.5 was used.

We further report experiments with our own implementation of the model of Zhang et al. (2016). Recall that this model (SENT) uses only one sentence-level latent variable that is used in the same way as our positional variables. Their model is equivalent to our stochastic decoder (SDEC) model where $z_i = z_0$ for all $i > 0$. Our implementation differs from that of Zhang et al. (2016) in several aspects. First it uses the last encoder state as a representation for the source side while the original implementation uses the average of all encoder states. Our implementation is based on Sockeye, whereas theirs was based on the weaker groundhog system. In their experiments, Zhang et al. (2016) use latent variables that are much larger than the decoder state. Here, we use considerably smaller variables to make the latent variable sizes comparable. We did, however, perform initial experiments with larger latent variable sizes and found that our implementation failed catastrophically in that setting. It is therefore safe to assume that the chosen latent variable size did not give a disadvantage to the SENT model.

The SDEC model is also built as an extension of Sockeye. Recall that the latent functions that compute the Gaussian parameters (Equations (4.12) and (4.19)) are implemented as single hidden layer neural networks. We set the size of the hidden layer to twice that of the hidden variable. We further use KL scaling for both the SENT and SDEC models (see Section 4.3). The scaling schedule is additive and the scaling factor at batch t is $\min(t/20{,}000, 1)$.

All models use 1028 units for the LSTM hidden state (or 512 for each direction in the bidirectional LSTMs) and 256 for the attention mechansim. Training is done with Adam (Kingma and Ba, 2015). In decoding we use a beam of size 5 and output the most likely word at each position. We deterministically set all latent variables to their mean values during decoding. Monte Carlo decoding (Gal, 2016) is difficult to apply to our setting as it would require sampling entire translations.

Please download our workflow here[16] to reproduce the experiments and view all further hyperparameter settings.

Results We show the BLEU scores for all models that we tested on the IWSLT data set in Table 4.3. The stochastic decoder dominates the Sockeye baseline across all 4 languages, and outperforms SENT on most languages. Aside from German, there is a trend towards smaller latent variable sizes being more helpful. This is in line with findings by Chung et al. (2015) and Fraccaro et al. (2016) who

[15] https://github.com/awslabs/sockeye
[16] https://github.com/philschulz/SockeyeWorkflow

Model	Dropout	LatentDim	Arabic	Czech	French	German
Sockeye	None	None	8.2	6.9	23.5	14.3
Sockeye	0.5	None	8.4	7.4	24.4	15.1
SENT	0.5	64	8.4	7.3	24.8	15.3
SENT	0.5	128	8.7	7.4	24.0	15.7
SENT	0.5	256	8.9	7.4	24.7	15.5
SDEC	0.5	64	8.2	7.7	25.3	15.4
SDEC	0.5	128	8.8	7.5	24.2	15.6
SDEC	0.5	256	8.7	7.5	23.2	15.9

Table 4.3: BLEU scores for different models on the IWSLT data for translation into English. Recall that all SDEC and SENT models used KL scaling during training.

also used relatively small latent variables. This observation also implies that our model does not improve simply because it has more parameters than the baseline.

That the margin between the SDEC and SENT models is not large is not unexpected for two reasons. First, Chung et al. (2015) and Fraccaro et al. (2016) have shown that stochastic RNNs lead to enormous improvements in modelling continuous sequences but only modest increases in performance for discrete sequences (such as natural language). Second, translation performance is measured in BLEU score. We observed that SDEC often reached better ELBO values than SENT indicating a better model fit. How to fully leverage the better modelling ability of stochastic RNNs when producing discrete outputs is a matter for future research.

Qualitative Analysis The design of the stochastic decoder model was of course guided by the observation that there is considerable variation in translation data produced by human translators. Ideally, our model should be able to capture this variation and produce it in its own translations. To test if our model has this ability, we decoded randomly chosen sentences from the test set. However, instead of setting the latent variable value to its mean, we sampled all latent variables. Importantly, we still chose the most likely word per output position. A standard non-stochastic NMT system would always produce the same translation in this setting. In Figure 4.5 we show some of the translations produced by the Sockeye baseline as well as the SENT and SDEC models.

Interestingly, the SENT model always produced the same translation, thus behaving like a deterministic model. This also indicates that it does not make full use of the latent variable and thus suffers from the same problem as Bowman et al. (2016). Our SDEC model, on the other hand, produces diverse translations that differ in linguistically interesting ways. In particular, it introduces long distance

Source	Coincidentally, at the same time, the first easy-to-use clinical tests for diagnosing autism were introduced.
SENT	Im gleichen Zeitraum wurden die ersten einfachen klinischen Tests für Diagnose getestet.
SDEC	Übrigens, zur gleichen Zeit, wurden die ersten einfache klinische Tests für die Diagnose von Autismus eingeführt.
SDEC	Übrigens, zur gleichen Zeit, waren die ersten einfache klinische Tests für die Diagnose von Autismus eingeführt worden.
Source	They undertook a study of autism prevalence in the general population.
SENT	Sie haben eine Studie von Autismus in der allgemeinen Population übernommen.
SDEC	Sie entwarfen eine Studie von Autismus in der allgemeinen Bevölkerung.
SDEC	Sie führten eine Studie von Autismus in der allgemeinen Population ein.

Figure 4.5: Sampled translations from our model (SDEC) and the sentence-level latent variable model (SENT). The first SDEC example shows alternation between the German simple past and past perfect. The past perfect introduces a long range dependency between the main and auxiliary verb (underlined) that the model handles well. The second example shows variation in the lexical realisation of the verb. The second variant uses a particle verb and we again observe a long range dependency between the main verb and its particle (underlined).

dependencies that the model handles well.

4.6 Future Work

The stochastic decoder is the first NMT model that explicitly accounts for word-level contextual variation in translation data. As such it opens up several interesting opportunities for future research. Those can broadly be classified into research with different data sets, research on model design and research on distributions used in the model.

Data The stochastic decoder accounts for variation per target symbol. Such a symbol may well be a character instead of a word. Moreover, there is a vast range of data available that is not as clean as the one we have used for our experiments. Social media texts immediately come to mind. They contain a variation that may even go beyond the grammatical rules of a language. User generated content in general is of great interest to commercial applications but often comes with

some degree of noise. We reckon that the stochastic decoder would fare well when translating such data.

Model Design Our model can of course be extended in several ways. One way is to make it more hierarchical by having the word-level latent variables depend on a sentence-level latent variable. This way we can hope to capture variation at different resolutions. The second obvious extension is the use of latent factor models such as DARN (Gregor et al., 2014). The Gaussian latent variables are not suited for disentangling different factors of variation and may in fact often conflate them. Replacing or complementing them with latent factors will on the one hand increase the expressiveness of the model and on the other hand make it more interpretable. After all, it is easier to correlate atomic factors with linguistic observations than trying to do the same for arbitrary dimensions of a Gaussian vector.

Distributions The distributions of the latent variables used in the stochastic decoder are diagonal Gaussian. As such they are unimodel and cannot capture covariance. This is of course a rather strong restriction. We therefore plan to employ more expressive distributions computed by normalising flows (Rezende and Mohamed, 2015; Kingma et al., 2016). Such distributions are able to adapt to complex likelihood surfaces as they can distribute their probability over several modes in latent space.

4.7 Summary

In this chapter I have presented a stochastic decoder model for NMT. It is the first neural translation model to address word-level variation in translation data. It does so through the use of latent variables whose parameters are predicted by neural networks. I have provided an in-depth discussion of the pitfalls encountered during training. The experiments show empirically that the model is superior to its baselines. When analysing the translations produced by the model one finds that it is indeed able to produce diverse translations which vary both in their lexical realisations as well as their syntactic structure. This indicates that the proposed model fulfils its purpose and is indeed able to capture contextual variation in translation data.

Chapter 5

Conclusion and Outlook

This thesis has explored the role of variability in machine translation. We have argued that this phenomenon is evident in translation data and that its origins can likely be traced down to differences in text type as well as varying abilities of translators.

The machine translation community is well aware of this variation. This awareness manifests itself most prominently in the design of evaluation scores that are designed to take multiple references into account. It has, however, been neglected in the design of machine translation models. The work presented here fills at least a part of that gap. It also provides modelling solutions that are easily extensible with current machine learning techniques, thus paving the way for more sophisticated models that find more informed ways of dealing with the variation observed in translation data.

The thesis concentrates on twp aspects of machine translation. One is word alignment which can in fact be used for multilingual NLP in general. While it is only a subcomponent of phrase-based SMT systems, it allows for quick model exploration. Moreover, it is arguably the least disruptive component in a phrase-based MT system. This allowed us to make large changes to the word alignment component while leaving the other parts of the pipeline virtually untouched.

On the other end of the spectrum we have explored fully end-to-end neural machine translation. Because neural networks are accurate but overconfident classifiers and because NMT usually works with a rather small beam size, these models are even more prone to ignoring variation in the translation data. We have augmented them with a chain of latent variables. Conceptually, this makes the model much more flexible and we have shown that it also leads to improved translation performance.

More to the point, this thesis has made 4 core contributions:

- It has introduced a new alignment model that does not need to stipulate NULL words. It has demonstrated that this modelling idea works and often performs better than alignment models that do use a NULL word.

- It has introduced a fast Gibbs sampler for Bayesian word alignment models that makes the complexity of resampling a given alignment link constant. While variational algorithms have been used for Bayesian word alignment, this work makes sampling-based inference for these models practical and thus enables their use in real word applications.

- A stochastic decoder model for neural machine translation has been presented. It is the first model to systematically address word level variation in NMT. It does so by introducing latent variables for each target word. The latent variables are marginally dependent and therefore can capture long-range dependencies. This benefit can be demonstrated by sampling from the trained model. The variational algorithm used to the train the model also uses a "look-ahead" mechanism which allows the conditional priors over the latent variables to encode information about future symbols in the sequence.

- The problems encountered when training the stochastic decoder have been discussed at length and a principled discussion of training techniques such as scaling the KL term has been presented. It has also been related to a large set of prior and contemporary work from which it took inspiration.

These contributions open up further research avenues that are worth exploring. Recall that at the outset of this thesis, I identified variation in translation data as the main phenomenon that the models described here are designed to capture. The alignment model is very explicit about the kind of variation it assumes: words are either generated as translations or because they are required by syntactic and semantic features of the sentence they appear in. The stochastic decoder, on the other hand, models variation as Gaussian vectors and it remains difficult to fathom the exact variation exactly these vectors encode; qualitative evaluation indicates that it is syntactic as well as tense and lexical variation. The reason that it is hard to come up with a principled evaluation of the encoded variation, however, is two-fold. First, the Gaussian dimensions do not straighforwardly correspond to linguistic categories or any other kind of annotation we may have for our data. Second, the mapping from the latent space into the output distribution space (i.e. the likelihood) is highly non-linear and so it is hard to pin down what effect each dimension has on the output categorical distribution.

Both the interpretability of the model and possibly its performance can be improved if we turn it into a latent factor model. Latent factor models such as restricted Boltzmann machines Hinton (2002) and DARN Gregor et al. (2014) are essentially state-of-the-art in image modelling. Their main advantage over continuous models is that they can divide up the latent space such that different factors cover different parts of it. Because not all factors need to be active all of the time, the latent space can effectively be shrunk to a relevant subset at each

decision step. This leads to better modelling since the model is able to identify latent factors while still being able to ignore them when convenient.

In terms of interpretation, latent factors have the advantage of being easier to correlate with linguistic features of observations as they may be present for only some of those features. Designing a latent factor model for NMT based on the work presented here is straighforward: the graphical structure of the model can stay unchanged and we simply need to alter the latent variable distributions. In particular, we need to turn the Gaussian vectors into a vector of Bernoulli variables. This of course necessitates a slight change in the inference procedure. Reparametrisations are not available for discrete variables and thus one needs to fall back on the score function estimator (Paisley et al., 2012; Ranganath et al., 2014). Apart from that, the model can be trained as before.

Another modelling idea that the stochastic decoder gives rise to is the inclusion of side information. Side information in modern NMT is almost always available, be it in the form of pictures, sentiments, translator or author information. These sources of information should be exploited not only to build better general NMT system but also to advance the field towards personalised translation, an application that carries great commercial potential. Conditioning latent variables on side information can easily be done and thus computed distributions can essentially act as a funnel through which the different sources of side information are welded into a joint represention.

My hope is that this thesis has laid the groundwork for future research on existing problems in translation and paved the road for new and exciting developments and applications. I have strived to keep the presentation clear and provide all necessary details on my modelling and inference decisions so as to provide other researchers with inspiration.

Appendix A

The Gibbs Distribution and Exponential Families

This appendix discusses the Gibbs distribution[1] and it's relationship to log-linear models and the exponential family and neural networks.

Let $\mathcal{X}$ be a set of possible outcomes and $T \in \mathbb{R}^+$ be a temperature. The pmf of the Gibbs distribution over $\mathcal{X}$ then is

$$P(x|\theta) = \frac{\exp(-TE(x))}{\sum_{x \in \mathcal{X}} \exp(-TE(x))} \tag{A.1}$$

where $E(x)$ is the model-specific energy function of the distribution. In the following we will pre-multiply the energy with -1 and thus drop the minus in the exponent of Equation (A.1).

A.1 Log-Linear Models

A log-linear model is an undirected graphical model whose likelihood is given by the Gibbs distribution at temperature 1. Let $h : \mathcal{X} \to \mathbb{R}^d$ be a feature function and $\theta \in \mathbb{R}^d$ be a parameter vector. The energy function of a log-linear model is then given as $E(x) = h(x)\theta^\top$, resulting in the following pmf:

$$P(x|\theta) = \frac{\exp(h(x)\theta^\top)}{\sum_{x \in \mathcal{X}} \exp(h(x)\theta^\top)} \ . \tag{A.2}$$

Notice that this model is generally non-identifiable since scaling of θ does not affect the pmf. To avoid this problem, choose an arbitrary dimension $j \leq d$ and replace θ_i with $\theta'_i = \log\left(\frac{\theta_i}{\theta_j}\right), 1 \leq i \leq d$. The new parameter vector θ' contains

[1] The Gibbs distribution is also known as Boltzmann distribution. A true Boltzmann distribution includes the Boltzmann constant. In the Gibbs distribution we choose the unit of our quanities such that the Boltzmann constant gets cancelled out.

the log-odds of all features with respect to feature j. This parametrisation is identifiable because we necessarily have $\theta'_j = 0$. Also, if $h(x)$ is chosen to be the sufficient statistics of the categorical distribution, Equation (A.2) is exactly the exponential family formulation of the categorical. This particular model is well-known as *logistic regression*. Note that in machine learning and NLP, the log-linear model is often used as a conditional model $p(x|y, \theta)$. This changes hardly anything about its pmf, though. Only the natural parameters become dependent on the conditioning event and should be written as $\theta(y)$.

Let us try to find the maximum likelihood estimator of the model in Equation (A.2) through manipulation of the score function.

$$\frac{\partial}{\partial \theta_k} \log(P(x)) = \sum_{i=1}^{n} \frac{\partial}{\partial \theta_k} \log \left(\frac{\exp\left(h(x_i)\theta^\top\right)}{\sum_{x'_i} \exp\left(h(x'_i)\theta^\top\right)} \right) \tag{A.3}$$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial \theta_k} h(x_i)\theta^\top - \frac{\partial}{\partial \theta_k} \log \left(\sum_{x'_i} \exp\left(h(x'_i)\theta^\top\right) \right) \tag{A.4}$$

$$= \sum_{i=1}^{n} h(x_i)_k - h(x_i)_k \frac{\exp\left(h(x_i)\theta^\top\right)}{\sum_{x'_i} \exp\left(h(x'_i)\theta^\top\right)} \tag{A.5}$$

$$= \sum_{i=1}^{n} h(x_i)_k - \mathbb{E}\left[h(x_i)_k\right] \tag{A.6}$$

The MLE is thus found whenever $\sum_{i=1}^{n} h(x_i) = \sum_{i=1}^{n} \mathbb{E}\left[h(x_i)\right]$. In the general log-linear model where h can be chosen arbitrarily, there exists no closed-form MLE and iterative techniques need to be applied. However, when h yields the sufficient statistics of the categorical, the well-known categorical maximum likelihood estimator can be used.

A.2 The Gibbs Distribution and Exponential Families

The discussion in the previous section has far-reaching implications for the entire exponential family. From here on, I use $t(x)$ for the sufficient statistics to clearly set them apart form an abitrary feature vector $h(x)$. Choose an exponential family distribution and let $h(x)$ be its sufficient statistics and θ its natural parameter vector. Modulo the base measure, any exponential family distribution then takes the form in Equation (A.2) with log-normaliser $a(\theta) = \log\left(\sum_{x \in \mathcal{X}} \exp(t(x)\theta^\top)\right)$. Obviously, for continuous distributions we will have to replace the sum by an integral.

The exciting consequences, of which I make extensive use in this thesis are the following:

1. Through manipulation of the temperature, annealing schemes (Kirkpatrick et al., 1983) become immediately available without any further justification.

2. A maximum-likelihood estimate of any exponential family can be found by setting $\sum_{i=1}^{n} t(x_i) = \sum_{i=1}^{n} \mathbb{E}\left[t(x_i)\right]$.

3. From the equivalence of the last term in lines (A.4) and (A.6) it follows that $\frac{\partial}{\partial\theta} a(\theta) = \mathbb{E}\left[t(x)\right]$ for all exponential families.

Appendix B

Gaussian Reparametrisation

This appendix provides a more detailed derivation of the Gaussian reparametrisation described in Kingma and Welling (2014); Rezende et al. (2014); Titsias and Lázaro-Gredilla (2014). It also derives the KL divergence for univariate Gaussian distributions.

B.1 Change of Continuous Variables

While we can compute the density of the transformed variable at a given point, we generally cannot sample from that density. Luckily, Gaussian variables can be transformed into standard Gaussian variables. The standard Gaussian distribution is easy to sample from. Let $z \sim \mathcal{N}(\mu, \sigma^2)$. We can standardise z by subtracting its mean and dividing by the standard deviation.

$$\epsilon = h(z, \mu, \sigma) = \frac{z - \mu}{\sigma} \tag{B.1}$$

We write the standard Gaussian density of $\epsilon \sim \mathcal{N}(0, \mathrm{I})$ as $\phi(\epsilon)$.

Differentiating the reconstruction term of the ELBO with respect to the variational parameters λ poses a challenge. If we differentiate the expectation we end up with an expression that is itself not an expectation and thus cannot be approximated with MC methods (see Equation (2.28b)). In order to enable MC estimation we replace the measure of the expectation with its transformed version.

$$\begin{aligned} q(z|\mu, \sigma^2) &= \phi(h(z, \mu, \sigma)) \times \left| \frac{d}{dz} h(z, \mu, \sigma) \right| \\ &= \phi(\epsilon) \times \left| \frac{d\epsilon}{dz} \right| \end{aligned} \tag{B.2}$$

The Jacobian term cancels with its inverse produced by transforming the infinitesimal in the integral, leaving only the standard Gaussian measure (see lines (B.3b) – (B.3c)).

Taking the derivative of the ELBO with respect to the variational parameters μ, σ^2 is now easy as we can push the derivative operator inside the expectation to obtain stochastic gradient estimates.

$$\frac{\partial}{\partial\lambda}\mathbb{E}_{q(z|\mu,\sigma^2)}\left[\log p(y|x,z)\right] = \tag{B.3a}$$

$$\frac{\partial}{\partial\lambda}\int q(z|\mu,\sigma^2)\log p(y|x,z)\mathrm{dz} = \tag{B.3b}$$

$$\frac{\partial}{\partial\lambda}\int \phi(\epsilon)\times\left|\frac{d\epsilon}{dz}\right|\log p(y|x,z)\mathrm{d}\epsilon\frac{\mathrm{dz}}{\mathrm{d}\epsilon} = \tag{B.3c}$$

$$\int \phi(\epsilon)\times\frac{\partial}{\partial\lambda}\log p(y|x,z)\mathrm{d}\epsilon = \tag{B.3d}$$

$$\mathbb{E}_{\phi(\epsilon)}\left[\frac{\partial}{\partial\lambda}\log p(y|x,\underbrace{h^{-1}(\epsilon,\mu,\sigma)}_{z})\right] \tag{B.3e}$$

Notice that the inverse transformation h^{-1} is given as,

$$h^{-1}\left(\epsilon,\mu,\sigma\right) = \mu + \sigma\odot\epsilon\ . \tag{B.4}$$

B.2 KL Divergence between Exponential families

Let p and q be distributions from the same exponential family. We denote their natural parameters η_p and η_q, respectively. The KL divergence of p from q can be computed as,

$$\mathrm{KL}\left(q\ ||\ p\right) = \tag{B.5a}$$

$$\mathbb{E}_p\left[\log\left(\frac{p(X)}{q(X)}\right)\right] = \tag{B.5b}$$

$$\mathbb{E}_p\left[\log h(X) + t(X)^\top\eta_p - a(\eta_p) - \log h(X) - t(X)^\top\eta_q + a(\eta_q)\right] = \tag{B.5c}$$

$$\mathbb{E}_p\left[t(X)^\top\left(\eta_p-\eta_q\right) - a(\eta_p) + a(\eta_q)\right] = \tag{B.5d}$$

$$\mathbb{E}_p\left[t(X)\right]\top\left(\eta_p-\eta_q\right) - a(\eta_p) + a(\eta_q) \tag{B.5e}$$

As long as $a(\eta_p)$ and $a(\eta_q)$ can be computed, the KL divergence is available in closed form. This is because exponential families always allow for the computation of their parameters. Further, the expected sufficient statistic is simply the derivative of the log-normaliser. As a consequence, the KL divergence as a whole is analytically computable.

B.3 Gaussian KL Divergence

In the main text I employ k-dimensional Gaussians with diagonal covariance matrices. Their density is equivalent to a product of k independent univariate Gaussians. To compute their KL divergence it thus suffices to derive the KL divergence for univariate Gaussians.[1] I use μ and σ to denote the Gaussian mean and standard deviation and index these with their corresponding distribution.

$$\mathrm{KL}\left(q(z) \mid\mid p(z)\right) = \mathbb{E}_{q(z)}\left[\log\left(\frac{q(z)}{p(z)}\right)\right] = \mathbb{E}_{q(z)}\left[\log q(z) - \log p(z)\right] \tag{B.6a}$$

$$\begin{aligned} &= \mathbb{E}_{q(z)}\left[\log\left(\frac{1}{\sqrt{2\pi}\sigma_q}\exp\left(-\frac{1}{2}\left(\frac{\left(z-\mu_q\right)^2}{\sigma_q^2}\right)\right)\right)\right. \\ &\qquad \left. - \log\left(\frac{1}{\sqrt{2\pi}\sigma_p}\exp\left(-\frac{1}{2}\left(\frac{\left(z-\mu_p\right)^2}{\sigma_p^2}\right)\right)\right)\right] \end{aligned} \tag{B.6b}$$

$$= \mathbb{E}_{q(z)}\left[\log\sigma_p - \log\sigma_q - \frac{1}{2}\left(\left(\frac{\left(z-\mu_q\right)^2}{\sigma_q^2}\right) - \left(\frac{\left(z-\mu_p\right)^2}{\sigma_p^2}\right)\right)\right] \tag{B.6c}$$

$$= \log\left(\frac{\sigma_p}{\sigma_q}\right) - \frac{1}{2}\left(\frac{\overbrace{\mathbb{E}_{q(z)}\left[\left(z-\mu_q\right)^2\right]}^{\sigma_q^2}}{\sigma_q^2} - \left(\frac{\mathbb{E}_{q(z)}\left[\left(z-\mu_p\right)^2\right]}{\sigma_p^2}\right)\right) \tag{B.6d}$$

$$= \log\left(\frac{\sigma_p}{\sigma_q}\right) - \frac{1}{2}\left(1 - \frac{\mathbb{E}_{q(z)}\left[z^2 - z\mu_p + \mu_p^2\right]}{\sigma_p^2}\right) \tag{B.6e}$$

$$= \log\left(\frac{\sigma_p}{\sigma_q}\right) - \frac{1}{2}\left(1 - \frac{\mathbb{E}_{q(z)}\left[z^2\right] - \mu_q\mu_p + \mu_p^2}{\sigma_p^2}\right) \tag{B.6f}$$

$$= \log\left(\frac{\sigma_p}{\sigma_q}\right) - \frac{1}{2}\left(1 - \frac{\sigma_q^2 + \mu_q^2 - \mu_q\mu_p + \mu_p^2}{\sigma_p^2}\right) \tag{B.6g}$$

$$= \log\left(\frac{\sigma_p}{\sigma_q}\right) - \frac{1}{2}\left(1 - \frac{\sigma_q^2 + \left(\mu_q - \mu_p\right)^2}{\sigma_p^2}\right) \tag{B.6h}$$

$$= \frac{1}{2}\left(-2\log\left(\frac{\sigma_q}{\sigma_p}\right) - 1 + \frac{\sigma_q^2}{\sigma_p^2} + \frac{\left(\mu_q - \mu_p\right)^2}{\sigma_p^2}\right) \tag{B.6i}$$

$$= \frac{1}{2}\left(-\log\left(\frac{\sigma_q^2}{\sigma_p^2}\right) - 1 + \frac{\sigma_q^2}{\sigma_p^2} + \frac{\left(\mu_q - \mu_p\right)^2}{\sigma_p^2}\right) \tag{B.6j}$$

In line (B.6g) we have used the fact that $\mathrm{var}\left(Z\right) = \mathbb{E}\left[Z^2\right] - \mathbb{E}\left[Z\right]^2$. Rearranging

[1] We could of course derive the Gaussian KL from the KL computation for exponential families. I choose to present the direct derivation for the sake of exposition.

this equality to $\mathbb{E}[Z^2] = \mathrm{var}(Z) + \mathbb{E}[Z]^2$ yields the substitution that we have used.

Bibliography

Felix V. Agakov and David Barber. *An Auxiliary Variational Method*, pages 561–566. Springer Berlin Heidelberg, 2004. URL https://doi.org/10.1007/978-3-540-30499-9_86.

J. Aitchison and S. M. Shen. Logistic-normal distributions: Some properties and uses. *Biometrika*, 67(2):261–272, 1980. URL http://www.jstor.org/stable/2335470.

Alexander Alemi, Ben Poole, Ian Fischer, Joshua V. Dillon, Rif A. Saurous, and Kevin Murphy. An information-theoretic analysis of deep latent-variable models. 2017. URL https://openreview.net/forum?id=H1rRWl-Cb.

Duygu Ataman, Wilker Aziz, and Alexandra Birch. A latent morphology model for open-vocabulary neural machine translation. In *ICLR*, 2020. URL https://openreview.net/forum?id=BJxSI1SKDH.

Hagai Attias. A variational Bayesian framework for graphical models. In *In Advances in Neural Information Processing Systems 12*, pages 209–215, 2000. URL http://papers.nips.cc/paper/1726-a-variational-baysian-framework-for-graphical-models.pdf.

R.H. Baayen, D.J. Davidson, and D.M. Bates. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59(4):390 – 412, 2008. URL http://www.sciencedirect.com/science/article/pii/S0749596X07001398.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. 2014. URL http://arxiv.org/abs/1409.0473.

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of

Markov Chains. *Annals of Mathematical Statistics*, 41(1):164–171, 1970. URL http://dx.doi.org/10.1214/aoms/1177697196.

Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. In *ICLR*, 2015. URL https://arxiv.org/abs/1411.7610.

Matthew J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003. URL http://www.cse.buffalo.edu/faculty/mbeal/thesis/index.html.

José M. Bernardo and Adrian F. M. Smith. *Bayesian Theory*. John Wiley & Sons, Inc., 2008. ISBN 9780470316870. doi: 10.1002/9780470316870.

Julian Besag. Markov chain Monte Carlo methods for statistical inference, 2004. URL http://www-users.mat.umk.pl/~wniem/SemMgr/besag_MCMC.pdf.

Michael Betancourt. A Conceptual Introduction to Hamiltonian Monte Carlo. *ArXiv e-prints*, 2017. URL https://arxiv.org/pdf/1701.02434.pdf.

Fetsje Bijma, Marianne Jonker, and Aad van der Vaart. *Inleiding in de Statistiek*. Epsilon Uitgaven, Utrecht, 2013.

Arianna Bisazza and Marcello Federico. A survey of word reordering in statistical machine translation: Computational models and language phenomena. *Computational Linguistics*, 42(2):163–205, 2016. URL http://dx.doi.org/10.1162/COLI_a_00245.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. URL https://www.cs.princeton.edu/~blei/papers/BleiNgJordan2003.pdf.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *arXiv preprint*, 2016. URL https://arxiv.org/pdf/1601.00670.pdf.

Phil Blunsom and Trevor Cohn. A hierarchical Pitman-Yor Process HMM for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 865–874, 2011. URL http://www.aclweb.org/anthology/P11-1087.

Phil Blunsom and Miles Osborne. Probabilistic inference for machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 215–223, 2008. URL http://aclweb.org/anthology/D/D08/D08-1023.pdf.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 10–21, 2016. URL http://aclweb.org/anthology/K/K16/K16-1002.pdf.

G.E.P. Box. Robustness in the strategy of scientific model building. In Rober L. Launer and Graham N. Wilkinson, editors, *Robustness in Statistics*, pages 201 – 236. Academic Press, 1979. ISBN 978-0-12-438150-6. doi: https://doi.org/10.1016/B978-0-12-438150-6.50018-2.

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. Massive exploration of neural machine translation architectures. 2017. URL https://arxiv.org/abs/1703.03906.

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michael I Jordan. Streaming variational bayes. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1727–1735. 2013. URL http://papers.nips.cc/paper/4980-streaming-variational-bayes.pdf.

P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. A statistical approach to language translation. In *Proceedings of the 12th Conference on Computational Linguistics - Volume 1*, COLING '88, pages 71–76, 1988. URL http://dx.doi.org/10.3115/991635.991651.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of Statistical Machine Translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993. URL http://www.aclweb.org/anthology/J93-2003.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. 1996/98. URL http://arxiv.org/abs/cmp-lg/9606011.

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *ICLR*, 2017. URL http://arxiv.org/abs/1611.02731.

David Chiang. A hierarchical phrase-based model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 263–270, 2005. URL http://dx.doi.org/10.3115/1219840.1219873.

David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, 2007. URL http://dx.doi.org/10.1162/coli.2007.33.2.201.

David Chiang. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13(1):1159–1187, 2012. URL http://jmlr.csail.mit.edu/papers/volume13/chiang12a/chiang12a.pdf.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. *Empirical evaluation of gated recurrent neural networks on sequence modeling*. 2014.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2980–2988. 2015. URL http://papers.nips.cc/paper/5653-a-recurrent-latent-variable-model-for-sequential-data.pdf.

Hoang Cuong and Khalil Sima'an. Latent domain word alignment for heterogeneous corpora. In *NAACL*, pages 398–408, 2015. URL https://www.aclweb.org/anthology/N15-1043.

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*, 39:1–38, 1977. URL http://web.mit.edu/6.435/www/Dempster77.pdf.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. Why generative phrase models underperform surface heuristics. In *WMT 2006*, pages 31–38, 2006. URL https://aclanthology.info/papers/W06-3105/w06-3105.

Markus Dreyer and Yuanzhe Dong. APRO: all-pairs ranking optimization for MT tuning. In *NAACL HLT*, pages 1018–1023, 2015. URL http://aclweb.org/anthology/N/N15/N15-1106.pdf.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. A simple, fast, and effective reparameterization of IBM model 2. In *NAACL*, 2013. URL http://www.aclweb.org/anthology/N13-1073.

Bryan Eikema and Wilker Aziz. Auto-encoding variational neural machine translation. In *RepL4NLP*, pages 124–141, 2019. URL https://www.aclweb.org/anthology/W19-4315.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. URL http://dx.doi.org/10.1207/s15516709cog1402_1.

Michael D. Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90 (430):577–588, 1995. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1995.10476550.

Thomas S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973. URL http://dx.doi.org/10.1214/aos/1176342360.

Marco Fraccaro, Søren Kaae Sø nderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2199–2207. 2016. URL http://papers.nips.cc/paper/6039-sequential-neural-models-with-stochastic-layers.pdf.

Alexander Fraser and Daniel Marcu. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303, 2007. URL http://dx.doi.org/10.1162/coli.2007.33.3.293.

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016. URL http://www.cs.ox.ac.uk/people/yarin.gal/website/thesis/thesis.pdf.

Yarin Gal and Phil Blunsom. A systematic Bayesian treatment of the IBM alignment models. In *NAACL*, pages 969–977, 2013. URL http://aclweb.org/anthology/N/N13/N13-1117.pdf.

Michel Galley and Christopher D. Manning. A simple and effective hierarchical phrase reordering model. In *2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, 2008. URL http://www.aclweb.org/anthology/D08-1089.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, pages 1243–1252, 2017. URL http://proceedings.mlr.press/v70/gehring17a/gehring17a.pdf.

Alan E. Gelfand and Adrian F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85 (410):398–409, 1990. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1990.10476213.

Andrew Gelman. Intro to Bayesian data analysis and Stan, 2016. URL https://www.youtube.com/watch?v=T1gYvX5c2sM.

Andrew Gelman and Jennifer Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, 2007.

Andrew Gelman and Eric Loken. The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking" and the research hypothesis was posited ahead of time. 2013.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984. URL http://dx.doi.org/10.1109/TPAMI.1984.4767596.

Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 2000. URL http://dx.doi.org/10.1162/089976600300015015.

Zoubin Ghahramani and Michael I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29(2-3):245–273, 1997. URL http://dx.doi.org/10.1023/A:1007425814087.

Sharon Goldwater and Tom Griffiths. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P07-1094.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680. 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6716–6726. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7248-z-forcing-training-stochastic-recurrent-networks.pdf.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional LSTM networks for improved phoneme classification and recognition. In *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, 15th International Conference, Warsaw, Poland, September 11-15, 2005, Proceedings, Part II*, pages 799–804, 2005. URL https://doi.org/10.1007/11550907_126.

Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *ICML*, pages 1242–1250, Bejing, China, 2014. URL http://proceedings.mlr.press/v32/gregor14.html.

T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, 2004. URL http://psiexp.ss.uci.edu/research/papers/sciencetopics.pdf.

Richard J. Hathaway. Another interpretation of the EM algorithm for mixture distributions. *Statistics & Probability Letters*, 4(2):53–56, 1986. URL http://www.sciencedirect.com/science/article/pii/0167715286900167.

Kenneth Heafield. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, 2011. URL https://kheafield.com/papers/avenue/kenlm.pdf.

Gregor Heinrich. Parameter estimation for text analysis. Technical report, http://www.arbylon.net/publications/text-est.pdf, 2005.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. Sockeye: A Toolkit for Neural Machine Translation. *ArXiv e-prints*, 2017. URL https://arxiv.org/abs/1712.05690.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268, 1995.

Geoffrey E. Hinton. Training Products of Experts by minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002. URL http://dx.doi.org/10.1162/089976602760128018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Matthew D. Hoffman. Learning deep latent Gaussian models with Markov chain Monte Carlo. In *ICML*, pages 1510–1519, 2017.

Matthew D. Hoffman and David M. Blei. Stochastic structured variational inference. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015*, 2015. URL http://jmlr.org/proceedings/papers/v38/hoffman15.html.

Matthew D. Hoffman, David M. Blei, and Francis Bach. Online learning for latent dirichlet allocation. In *In NIPS*, 2010. URL http://papers.nips.cc/paper/3902-online-learning-for-latent-dirichlet-allocation.pdf.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *Journal of Machine Learning Research*, 14(1):1303–1347, 2013. URL http://dl.acm.org/citation.cfm?id=2502581.2502622.

Mark Hopkins and Jonathan May. Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, 2011. ISBN 978-1-937284-11-4. URL http://www.aclweb.org/anthology/D11-1125.

Anders Trærup Johannsen, Dirk Hovy, and Anders Søgaard. *Cross-lingual Syntactic Variation over Age and Gender*, pages 103–112. Association for Computational Linguistics, 2015.

Mark Johnson. The DOP estimation method is biased and inconsistent. *Computational Linguistics*, 28(1):71–76, 2002. URL http://dx.doi.org/10.1162/089120102317341783.

Mark Johnson and Sharon Goldwater. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado, June 2009. Association for Computational Linguistics.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D): 35–45, 1960.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. URL http://arxiv.org/abs/1412.6980.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014. URL https://arxiv.org/pdf/1312.6114v10.pdf.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *NIPS 29*, pages 4743–4751. 2016. URL http://papers.nips.cc/paper/6581-improved-variational-inference-with-inverse-autoregressive-flow.pdf.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. URL http://www.jstor.org/stable/1690046.

Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, 1999. URL http://www.aclweb.org/anthology/J99-4005.

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, 2003. URL http://dx.doi.org/10.3115/1073445.1073462.

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(14):1–45, 2017. URL http://jmlr.org/papers/v18/16-107.html.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001. URL https://repository.upenn.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1162&context=cis_papers.

Ching-Pei Lee and Chih-Jen Lin. Large-scale linear ranksvm. *Neural Computation*, 26(4):781–817, 2014. URL http://dx.doi.org/10.1162/NECO_a_00571.

Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. HLT-NAACL '06, pages 104–111, 2006. URL http://dx.doi.org/10.3115/1220835.1220849.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. NIPS, pages 2378–2386, 2016.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. Flowseq: Non-autoregressive conditional sequence generation with generative flow. In *EMNLP*, 2019. URL http://arxiv.org/abs/1909.02480.

D. J. C. MacKay. Introduction to monte carlo methods. In *Proceedings of the NATO Advanced Study Institute on Learning in Graphical Models*, pages 175–204. Kluwer Academic Publishers, 1998. URL http://www.inference.phy.cam.ac.uk/mackay/erice.pdf.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Coşkun Mermer and Murat Saraçlar. Bayesian word alignment for statistical machine translation. In *Proceedings of the 49th Annual Meeting of the Association*

for Computational Linguistics: Human Language Technologies: short papers-Volume 2, pages 182–187, 2011. URL http://www.aclweb.org/website/old_anthology/P/P11/P11-2.pdf#page=222.

Coşkun Mermer, Murat Saraçlar, and Ruhi Sarikaya. Improving statistical machine translation using Bayesian word alignment and Gibbs sampling. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1090–1101, 2013. URL http://ieeexplore.ieee.org/document/6425427/.

Yishu Miao and Phil Blunsom. Language as a latent variable: Discrete generative models for sentence compression. In *EMNLP*, pages 319–328, 2016. URL http://www.aclweb.org/anthology/D16-1031.

Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, pages 1727–1736, New York, New York, USA, 2016. URL http://proceedings.mlr.press/v48/miao16.html.

Robert C. Moore. Improving ibm word-alignment model 1. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. URL http://dx.doi.org/10.3115/1218955.1219021.

Markos Mylonakis and Khalil Sima'an. Phrase translation probabilities with ITG priors and smoothing as learning objective. In *EMNLP 2008*, pages 630–639, 2008. URL http://www.aclweb.org/anthology/D08-1066.

Markos Mylonakis and Khalil Sima'an. Learning hierarchical translation structure with linguistic annotations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 642–652, 2011. URL https://www.aclweb.org/anthology/P11-1065.

Christian Naesseth, Francisco Ruiz, Scott Linderman, and David Blei. Reparameterization Gradients through Acceptance-Rejection Sampling Algorithms. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 489–498, 2017.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996.

Radford M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 2003. URL http://dx.doi.org/10.1214/aos/1056562461.

Radford M. Neal and Geoffrey E. Hinton. Learning in graphical models. chapter A View of the EM Algorithm That Justifies Incremental, Sparse, and Other Variants, pages 355–368. MIT Press, 1999. URL http://www.cs.toronto.edu/~fritz/absps/emk.pdf.

Graham Neubig and Taro Watanabe. Optimization for statistical machine translation: A survey. *Computational Linguistics*, 42(1):1–54, 2016. URL http://dx.doi.org/10.1162/COLI_a_00241.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 271–279. 2016. URL http://papers.nips.cc/paper/6066-f-gan-training-generative-neural-samplers-using-variational-divergence-min pdf.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, 2003. URL http://acl.ldc.upenn.edu/acl2003/main/pdfs/Och.pdf.

Franz Josef Och and Hermann Ney. Improved statistical alignment models. In *38th Annual Meeting of the Association for Computational Linguistics, Hong Kong*, pages 440–447, 2000. URL http://www.aclweb.org/anthology/P00-1056.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for Statistical Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, 2002. URL http://www.aclweb.org/anthology/P02-1038.pdf.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003. URL http://dx.doi.org/10.1162/089120103321337421.

Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004. URL http://dx.doi.org/10.1162/0891201042544884.

A. O'Hagan. Monte Carlo is fundamentally unsound. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 36(2/3):247–249, 1987. URL http://www.jstor.org/stable/2348519.

John William Paisley, David M. Blei, and Michael I. Jordan. Variational bayesian inference with stochastic search. In *ICML*, 2012. URL http://icml.cc/2012/papers/687.pdf.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of*

the 40th Annual Meeting on Association for Computational Linguistics, pages 311–318, 2002. URL http://dx.doi.org/10.3115/1073083.1073135.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1310–1318, 2013. URL http://proceedings.mlr.press/v28/pascanu13.pdf.

Nicholas G. Polson and Vadim Sokolov. Deep learning: A bayesian perspective. *Bayesian Analysis*, 12(4):1275–1304, 12 2017. URL https://doi.org/10.1214/17-BA1082.

Ella Rabinovich, Raj Nath Patel, Shachar Mirkin, Lucia Specia, and Shuly Wintner. Personalized machine translation: Preserving original author traits. In *EACL*, pages 1074–1084, 2017. URL https://www.aclweb.org/anthology/E17-1101.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In Samuel Kaski and Jukka Corander, editors, *AISTATS*, pages 814–822, 2014. URL http://proceedings.mlr.press/v33/ranganath14.html.

Rajesh Ranganath, Dustin Tran, Jaan Altosaar, and David Blei. Operator variational inference. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 496–504. 2016. URL http://papers.nips.cc/paper/6091-operator-variational-inference.pdf.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. URL http://www.gaussianprocess.org/gpml/.

C.E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. In *Advances in Neural Information Processing Systems 15*, pages 489–496. MIT Press, 2003. URL http://papers.nips.cc/paper/2150-bayesian-monte-carlo.pdf.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1530–1538, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014. URL http://jmlr.org/proceedings/papers/v32/rezende14.pdf.

Darcey Riley and Daniel Gildea. Improving the IBM alignment models using variational Bayes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, pages 306–310, 2012. URL http://anthology.aclweb.org/P/P12/P12-2060.pdf.

Miguel Rios, Wilker Aziz, and Khalil Simaan. Deep generative model for joint alignment and word representation. In *NAACL*, pages 1011–1023, 2018. URL http://aclweb.org/anthology/N18-1092.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 1951. doi: 10.1214/aoms/1177729586. URL http://dx.doi.org/10.1214/aoms/1177729586.

Francisco R Ruiz, Michalis Titsias RC AUEB, and David Blei. The generalized reparameterization gradient. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 460–468. 2016. URL http://papers.nips.cc/paper/6328-the-generalized-reparameterization-gradient.pdf.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. *Nature*, 323: 533 EP –, 1986.

Tim Salimans, Diederik P. Kingma, and Max Welling. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. In *ICML*, 2015. URL http://proceedings.mlr.press/v37/salimans15.html.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3528–3536. 2015. URL http://papers.nips.cc/paper/5899-gradient-estimation-using-stochastic-computation-graphs.pdf.

Philip Schulz and Wilker Aziz. Fast collocation-based Bayesian HMM word alignment. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3146–3155, Osaka, Japan, 2016. URL http://aclweb.org/anthology/C16-1296.

Philip Schulz and Christian Schaffner. Basic probability and statistics. 2015. URL https://github.com/BasicProbability/LectureNotes/blob/master/fullscript/BasicProbabilityAndStatistics.pdf.

Philip Schulz, Wilker Aziz, and Khalil Sima'an. Word alignment without NULL words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

Philip Schulz, Wilker Aziz, and Trevor Cohn. A stochastic decoder for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1243–1252, 2018. URL http://aclweb.org/anthology/P18-1115.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A hybrid convolutional variational autoencoder for text generation. In *EMNLP*, pages 627–637, 2017. URL http://aclweb.org/anthology/D17-1066.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Controlling politeness in neural machine translation via side constraints. In *NAACL*, pages 35–40, 2016. URL https://www.aclweb.org/anthology/N16-1005.

Harshil Shah and David Barber. Generative neural machine translation. In *NIPS*, pages 1346–1355. 2018. URL http://papers.nips.cc/paper/7409-generative-neural-machine-translation.pdf.

C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

Noah A. Smith and Jason Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 354–362, 2005. URL http://dx.doi.org/10.3115/1219840.1219884.

Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Héctor Martínez Alonso. What's in a p-value in NLP? In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 1–10, 2014. URL http://www.aclweb.org/anthology/W/W14/W14-1601.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3483–3491. 2015. URL http://papers.nips.cc/paper/5775-learning-structured-output-representation-using-deep-conditional-generative pdf.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

M. A. Tanner and W.H. Wong. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398): 528–540, 1987. URL https://www.jstor.org/stable/pdf/2289457.pdf.

Yee Whye Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992, 2006. URL http://dx.doi.org/10.3115/1220175.1220299.

Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101: 1566–1581, 2006. URL http://www.tandfonline.com/doi/abs/10.1198/016214506000000302.

Christoph Tillmann. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 101–104, 2004. URL http://www.aclweb.org/anthology/N/N04/N04-4026.pdf.

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic Variational Bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1971–1979, 2014. URL http://jmlr.org/proceedings/papers/v32/titsias14.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 6000–6010. 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics. URL http://dx.doi.org/10.3115/993268.993313.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. Online large-margin training for statistical machine translation. In *EMNLP-CoNLL)*, pages 764–773, 2007. URL https://www.aclweb.org/anthology/D07-1080.

Warren Weaver. Translation. In *Machine Translation of Languages*, pages 15–23. MIT Press, Cambridge, MA, 1949/1955. Reprinted from a memorandum written by Weaver in 1949.

Greg C. G. Wei and Martin A. Tanner. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704, 1990. URL http://www.jstor.org/stable/2290005.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. Latent intention dialogue models. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, pages 3732–3741, International Convention Centre, Sydney, Australia, 2017. URL http://proceedings.mlr.press/v70/wen17a.html.

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

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, 1997. URL http://anthology.aclweb.org/J/J97/J97-3002.pdf.

Biao Zhang, Deyi Xiong, jinsong su, Hong Duan, and Min Zhang. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530. Association for Computational Linguistics, 2016. URL https://aclweb.org/anthology/D16-1050.

Junbo (Jake) Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. Adversarially regularized autoencoders, 2018. URL https://openreview.net/forum?id=BkM3ibZRW.

Shaojun Zhao and Daniel Gildea. A fast fertility Hidden Markov Model for word alignment using MCMC. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 596–605, 2010. URL https://www.cs.rochester.edu/~gildea/pubs/zhao-gildea-emnlp10.pdf.

Chunting Zhou and Graham Neubig. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 310–320, 2017. URL http://www.aclweb.org/anthology/P17-1029.