Multilingual Unsupervised Word Alignment Models and Their Application

by

Anahita Mansouri Bigvand

M.Sc., University of Western Ontario, 2010
B.Sc., Sharif University of Technology, 2009

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2021

Declaration of Committee

Name:	**Anahita Mansouri Bigvand**
Degree:	**Doctor of Philosophy**
Thesis title:	**Multilingual Unsupervised Word Alignment Models and Their Application**
Committee:	**Chair:** Parmit Chilana Assistant Professor, Computing Science

Anoop Sarkar
Supervisor
Professor, Computing Science

Fred Popowich
Committee Member
Professor, Computing Science

Angel Chang
Examiner
Assistant Professor, Computing Science

Philippe Langlais
External Examiner
Professor
Département d'informatique
Université de Montréal

Abstract

Word alignment is an essential task in natural language processing because of its critical role in training statistical machine translation (SMT) models, error analysis for neural machine translation (NMT), building bilingual lexicon, and annotation transfer. In this thesis, we explore models for word alignment, how they can be extended to incorporate linguistically-motivated alignment types, and how they can be neuralized in an end-to-end fashion. In addition to these methodological developments, we apply our word alignment models to cross-lingual part-of-speech projection.

First, we present a new probabilistic model for word alignment where word alignments are associated with linguistically-motivated alignment types. We propose a novel task of joint prediction of word alignment and alignment types and propose novel semi-supervised learning algorithms for this task. We also solve a sub-task of predicting the alignment type given an aligned word pair. The proposed joint generative models (alignment-type-enhanced models) significantly outperform the models without alignment types in terms of word alignment and translation quality.

Next, we present an unsupervised neural Hidden Markov Model for word alignment, where emission and transition probabilities are modeled using neural networks. The model is simpler in structure, allows for seamless integration of additional context, and can be used in an end-to-end neural network.

Finally, we tackle the part-of-speech tagging task for the zero-resource scenario where no part-of-speech (POS) annotated training data is available. We present a cross-lingual projection approach where neural HMM aligners are used to obtain high quality word alignments between resource-poor and resource-rich languages. Moreover, high quality neural POS taggers are used to provide annotations for the resource-rich language side of the parallel data, as well as to train a tagger on the projected data. Our experimental results on truly low-resource languages show that our methods outperform their corresponding baselines.

Keywords: Word alignment, machine translation, HMM, neural HMM

Dedication

To my family!

Acknowledgements

This thesis would not have been possible without the influence and support of my supervisor, colleagues, friends and family. First and foremost, I am forever indebted to my supervisor, Prof. Anoop Sarkar, for his never-ending support and guidance throughout my PhD studies. It was an amazing experience to collaborate with such a knowledgeable, admirable and considerate advisor. I am grateful for the countless hours he dedicated to this thesis, his enthusiasm for research, his incredible patience and sense of humor. He never stopped pushing me to improve the model, paper, presentation, etc. Despite being in a competitive academic era where everyone's goal is to publish more, he always encourages his students to solve research problems that would be beneficial to the world. He helped me grow as a researcher and a person.

I am also very grateful to my co-supervisor, Prof. Fred Popowich, for his support, helpfulness and useful comments. I am also extremely thankful to my thesis committee, Prof. Philippe Langlais, and Prof. Angel Chang for taking the time to read through this thesis and their valuable feedback.

Being part of the NatLang Lab at SFU has been extremely stimulating and fun. I want to thank my amazing co-authors: Te Bu, Nishant Kambhatla and Jetic Gu. I would like to extend my thanks to other members of the SFU NatLang Lab that I am so fortunate to count as my friends: Maryam Siahbani, Golnar Sheikhshab, Ashkan Alinejad, Hassan Shavarani, Ramtin Mehdizadeh Seraj, Jasneet Sabharwal, Mehdi Soleimani, Nadia GhobadiPasha, Pooya Moradi and Bdour Alzeer. I have learned a lot from you and I will definitely miss spending time with you.

I have been so lucky to have many amazing friends in Vancouver without whom I would not want to imagine what my life would have been like. I owe special thanks to Fatemeh Riahi, who first brought my attention to the NatLang lab at SFU. She encouraged me to talk to Anoop as she found him to be a caring supervisor. We have passed sixteen years of friendship, and I know for sure that she knows me better than any other friends of mine does. I have cherished every moment that I could spend with you, before and after my dear Kimia and Kaveh came to our world. I am extremely grateful to Ali Kamali for being a great friend, teaching me so many games and teasing me and Fatemeh for girl talks.

I would like to thank my wonderful friends Monir Hajiaghayi and Ali Mahmoudzadeh. I am grateful for all the beautiful memories we have shared over the past few years. All the

campings, trips, Shelems, volleyballs, picnics, burgers, etc. I am grateful to have you and Saba in my life.

I would like to thank Atieh Sarraf, Ali Rahmani, Setareh Sajadi, Mohammad Jazayeri and Noushin Saeedi for being my wonderful friends for so many years and for being so genuine and supportive. I am also thankful to my friends Shabnam Shariaty and Sina Ghotbi.

I would like to express my deepest gratitude to Mohammad Malekian for being so generous in sharing his wisdom with many people. For all the Rumi's Masnavi sessions every Thursday night that enlighten our mind, thank you. I have been fortunate to find amazing friends through attending these sessions: Faezeh Mohammadbeigi, Shirin Edalatfar, Ali Fayazbakhsh, Hoda Bagheri, Masoud Labbani, Zohreh Sharafian, Amir Sharafian, Maryam Hayati and Ehsan Iranmanesh. I am very grateful to all of you that kept me sane over the past few years and especially during the COVID19 pandemic. My special thanks go to Faezeh and Mohammad for always being there for me and for all the walks and talks. And, to Maryam Siahbani: thank you for your friendship, love, support and encouragement. I am still not over the fact that we do not have you around as before. Thanks to Sepehr Yadegarzadeh for all the movie nights at SFU.

I would like to thank my lovely grandmother for all her prayers, love and support, and my caring, selfless and giving uncle for all his phone calls even though we are thousands of miles apart from each other. Having you two in my life is such a precious gift!

To my dearest sister Mandana who has been my source of inspiration since I was a child, thank you. How lucky I am to have you in my life; a true artist, thinker and writer just by my side. Thank you for your unconditional love and support and for always providing a fresh perspective. Your academic life in Vancouver has made the past three years here the most magical and happiest.

To my parents, Parichehreh Mansouri and Sohrab Mansouri Bigvand: words cannot express how grateful I am for all you have done for me. Thank you for your eternal love and support and for believing in me when I did not believe in myself. None of my achievements would have been possible if it was not for you. To my dad who still wishes to drive me to school and my mom who has expected "at least a PhD", this is for you.

And finally, the one person who has made this all possible is my husband, Mojtaba Kheiri. He has been immensely patient and supportive through the years that led up to this dissertation. Thanks for taking the time to proofread the drafts of my papers and thesis and for your many constructive suggestions. Thanks for enduring the long distance between Montreal and Vancouver for more than two years, for your emotional support, for being my shoulder to cry on, for being so understanding and for being such a great friend.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Word alignment is the task of discovering a word-to-word correspondence in a pair of sentences that are translations of each other. Word alignment is an essential component in a statistical machine translation (SMT) system. Even though neural machine translation (NMT) does not use explicit alignments, word alignment is useful for analysis of translation errors (Ding et al., 2017; Li et al., 2019), guiding NMT models during training (Chen et al., 2016; Liu et al., 2016; Alkhouli and Ney, 2017; Alkhouli et al., 2018), introducing the coverage and fertility models (Tu et al., 2016) and improving NMT decoding (Alkhouli et al., 2016). Apart from machine translation, word alignment has been used for many NLP tasks including bilingual lexicon extraction, projecting linguistic annotations from one language to another (Yarowsky and Ngai, 2001; Hwa et al., 2005), creating multilingual embeddings (Faruqui and Dyer, 2014b; Guo et al., 2016; Dufter et al., 2018) and learning paraphrases in a source language by doing round-trips from source to target and back using word alignments (Ganitkevitch et al., 2013).

Generative word alignment models, IBM models 1-5 (Brown et al., 1993) and HMM (Vogel et al., 1996) are the most widely used word alignment approaches. Neural network-based alignment models have been explored in the literature, including the early works (Yang et al., 2013; Tamura et al., 2014; Legrand et al., 2016) and the more recent ones (Zenkel et al., 2019; Garg et al., 2019).

Models for word alignment depend on the way they decompose this problem. The classic IBM Models 1-5 (Brown et al., 1993) and the HMM model (Vogel et al., 1996) have underpinned the majority of the SMT systems to date. HMMs have been applied to numerous problems in NLP, such as part-of-speech tagging. The key attraction of HMMs is the existence of well-known tractable algorithms for EM parameter estimation (Baum, 1972) and maximization (Viterbi, 1967). HMMs have been widely studied for the word alignment problem (Och and Ney, 2000a; Toutanova et al., 2002).

The HMM-based word alignment model has been shown to significantly outperform IBM Models 1, 2, and 3 (Och and Ney, 2000a, 2003). IBM 4, 5 and the Och and Ney (2003) variant called IBM Model 6 all outperform IBM Model 3. However, only IBM Model 4, which is a probabilistically deficient version of IBM Model 5, is computationally tractable. HMMs are not deficient (no missing probability mass) and are computationally tractable. On the other hand, IBM model 4 alignment as produced by GIZA++ (Och and Ney, 2000b) are often the best that can be obtained for large parallel corpora. Despite its modelling power and widespread use, IBM model 4 has its own limitations. The parameter estimation of this model is implemented by approximate hill-climbing methods and hence can be very slow, memory-intensive and difficult to parallelize. The performance of a refined HMM model is comparable to that of IBM Model 4, and it is also much easier to understand. Practically, we can train the HMM model by the Forward-Backward algorithm, and by parallelizing the parameter estimation, we can control memory usage, reduce the time needed for training, and increase the corpus used for training. For all these reasons, the focus of this thesis is on the HMM-based word alignment model and the extensions and neural variants of this model.

All previous studies on word alignment have assumed that word alignments are untyped. To our knowledge, the alignment types for word alignment provided by the Linguistics Data Consortium (LDC) as annotations on word alignment links, have never been used to improve word alignment. We introduce a new probabilistic model for word alignment where word alignments are associated with linguistically motivated alignment types.

Despite the rapid rise of neural approaches in different areas of NLP, neural word alignment approaches have not matured enough, and traditional unsupervised statistical models remain the most widely used approaches for word alignment. Using neural networks allows for seamless integration of additional context that cannot be easily done in traditional models. We present an unsupervised neural Hidden Markov Model for word alignment, where emission and transition probabilities are modeled using neural networks.

We then focus on the investigation of using our neural HMM aligner for the part-of-speech tag projection task. We consider the zero-resource scenario where no POS annotated training data is available for the low-resource language. We use high quality neural POS taggers to tag the resource-rich language side of the parallel data, and to train a tagger on the projected data.

1.2 Contributions

The main contributions of this thesis can be summarized in the following directions:

- **Joint prediction of word alignment with alignment types:**
  A new probabilistic model for word alignment where word alignments are associated

with linguistically motivated alignment types, as well as a novel task of joint prediction of word alignment and alignment types and a novel semi-supervised learning algorithms for this task.

- **Unsupervised neural Hidden Markov Model for word alignment:**
  An unsupervised neural Hidden Markov Model for word alignment, where emission and transition probabilities are modeled using neural networks.

- **Cross-lingual annotation projection with neural HMM for low-resource languages:**
  A method that induces part-of-speech taggers for low-resource languages using cross-lingual tag projection that relies on our proposed neural HMM aligners to obtain high quality word alignments and neural POS taggers to tag the resource-rich language side of the parallel data as well as to train a tagger on the projected data.

1.3 Thesis Outline

Chapter 2 gives an overview of the relevant literature, starting with an explanation of word alignment. After reviewing IBM models in section 2.3, we thoroughly present the original HMM-based word alignment model (section 2.4). We overview the extensions to this model in section 2.5. We then discuss neural machine translation (section 2.6) and evaluation measures (section 2.7).

Chapter 3 gives a general framework for the unsupervised neural Hidden Markov Model. We then demonstrate this approach for the task of part-of-speech tag induction.

Chapter 4 presents a new probabilistic model for word alignment where word alignments are associated with linguistically motivated alignment types. We propose a novel task of joint prediction of word alignment and alignment types and propose novel semi-supervised learning algorithms for this task. We also solve a sub-task of predicting the alignment type given an aligned word pair. The generative models we introduce significantly outperform the models without alignment types, in terms of word alignment and translation quality.

Chapter 5 introduces an unsupervised neural Hidden Markov Model for word alignment, where emission and transition probabilities are modeled using neural networks. We incorporate BERT representation into our model in order to include the full target context for the emission model. In the experiments, we demonstrate improvements over GIZA++ IBM4 model, which is still a strong baseline, on Romanian-English and Chinese-English datasets.

Chapter 6 tackles the part-of-speech tagging task for the zero-resource scenario where no POS-annotated training data is available. We present a method that induces part-of-speech taggers for low-resource languages using cross-lingual tag projection. The proposed method relies on our neural HMM aligners presented in Chapter 5 to obtain high quality word alignments between resource-poor and resource-rich languages. Moreover, we leverage neural POS taggers to provide annotations for the resource-rich language side of the parallel data, as well as to train a tagger on the projected data. We provide experimental results on truly low-resource languages for POS tagging task.

Chapter 7 concludes this thesis and discusses a few of possible future directions.

Chapter 2

Background

This chapter provides an overview of the backgrounds for the research described in this thesis. We start with statistical machine translation (section 2.1) and word alignment (section 2.2). After reviewing IBM models in section 2.3, we go over the original HMM-based word alignment model (section 2.4). We present some of the extensions to the HMM alignment model in section 2.5. We discuss neural machine translation (section 2.6) as we will be comparing our neural word alignment models with NMT-based systems in terms of alignment quality. We discuss the evaluation measures that are used in this thesis (sections 2.6.3 and 2.7).

2.1 Statistical Machine Translation

In statistical machine translation, the goal is to translate from a source language F into a target language E. We assume that the source language is French and the target language is English. We have a French sentence $\boldsymbol{f} = f_1^J = f_1, f_2, \ldots, f_J$, and we want to translate it into an English sentence $\boldsymbol{e} = e_1^I = e_1, e_2, \ldots, e_I$. Among all English sentences, we are looking for the one which has the highest probability $Pr(\boldsymbol{e}|\boldsymbol{f})$. Using Bayes rule, we can write:

$$Pr(\boldsymbol{e}|\boldsymbol{f}) = \frac{Pr(\boldsymbol{f}|\boldsymbol{e})Pr(\boldsymbol{e})}{Pr(\boldsymbol{f})} \tag{2.1}$$

As the denominator is independent of $\boldsymbol{e}$, finding the most probable translation $\boldsymbol{e}^*$ will lead to the noisy channel model for statistical machine translation:

$$\boldsymbol{e}^* = \arg\max_{\boldsymbol{e}} Pr(\boldsymbol{e}|\boldsymbol{f}) \tag{2.2}$$

$$= \arg\max_{e} Pr(\boldsymbol{f}|\boldsymbol{e})Pr(\boldsymbol{e}) \tag{2.3}$$

where $Pr(\boldsymbol{e})$ is called the language model, while $Pr(\boldsymbol{f}|\boldsymbol{e})$ is called the translation model. The score for a potential translation $\boldsymbol{e}$ is the product of two scores: the language model

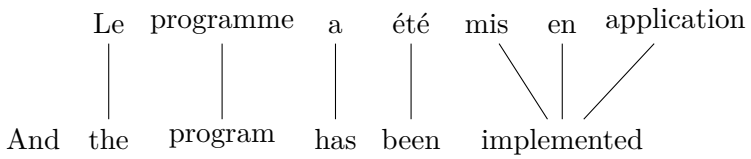


Figure 2.1: An alignment between a French sentence and its translation in English.

score which gives a prior probability of the sentence in English, and the translation model score, which indicates the likelihood of seeing the French sentence $\boldsymbol{f}$ as the translation of English sentence $\boldsymbol{e}$. The advantage of using the noisy-channel model is that it allows us to benefit from the language model which is helpful in improving the fluency or grammaticality of the translation. In the next section, we will present approaches towards introducing structures into the probabilistic dependencies in order to define translation model $Pr(\boldsymbol{f}|\boldsymbol{e})$ and estimating its parameters from the training data.

We now focus on the problem of modelling the translation probability. A key idea in IBM Models (Brown et al., 1993) was to introduce alignment variables to the problem.

2.2 Word Alignment

An alignment $\boldsymbol{a} = a_1, \ldots, a_J$ is a vector of alignment variables where each a_j can take any value in the set $\{1, 2, \ldots, I\}$. The alignment vector specifies the mapping for each French word to a word in the English sentence. Therefore, $a_j = i$ specifies that f_j is aligned to e_i.

Figure 2.1 shows an alignment between a pair of French and English sentences. In this example, $J = 7$ and $I = 6$. Since the length of the French sentence is 7, we have alignment variables $a_1, a_2, \ldots, a_7$ and $a_1, a_2, \ldots, a_7 = \langle 2, 3, 4, 5, 6, 6, 6 \rangle$ specifies the alignment depicted in the figure.

Note that the alignment is many-to-one; i.e. more than one French word can be aligned to an English word while each French word can be aligned to exactly one English word. The fertility ϕ_i of an English word e_i at position i is defined as the number of aligned French words:

$$\phi_i = \sum_{j=1}^{J} \delta(a_j, i) \tag{2.4}$$

where δ is the Kronecker delta function.

Models describing these alignments are called alignment models. The seminal work on word alignment is based on IBM Models 1-5 (Brown et al., 1993). We will briefly discuss IBM Models 1 and 2 in section 2.3.1 and 2.3.2, respectively. We then focus on the HMM-based word alignment model (section 2.4).

We follow the notational convention of Vogel et al. (1996); we use the symbol $Pr(.)$ to denote general probability distributions without any specific assumption, whereas $p(.)$ is used for model parameters, also known as model-based probability distributions.

2.3 Statistical Alignment Models

In SMT, we model the translation probability $Pr(\boldsymbol{f}|\boldsymbol{e})$. Having introduced alignment, rather than modelling $Pr(\boldsymbol{f}|\boldsymbol{e})$, we can try to model $Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e})$ which is called alignment model. We can then calculate the translation model by summing over all possible alignments:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) = \sum_{\boldsymbol{a}} Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) \tag{2.5}$$

Our statistical model depends on a set of parameters θ that can be learned from training data. We use the following notation to show the dependence of the model on the parameters:

$$Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) = Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}, \theta) \tag{2.6}$$

We assume that we have a source of bilingual training data $\boldsymbol{D} = (\boldsymbol{f}^{(k)}, \boldsymbol{e}^{(k)})$ for $k = 1, ..., n$ where $\boldsymbol{f}^{(k)}$ is the k'th French sentence and $\boldsymbol{e}^{(k)}$ is the k'th English sentence, and $\boldsymbol{e}^{(k)}$ is a translation of $\boldsymbol{f}^{(k)}$. We can estimate the parameters of our model using these training examples. The parameters θ are computed by maximizing the likelihood on the parallel training corpus:

$$\theta^* = \arg\max_{\theta} \prod_{k=1}^{n} \Big[\sum_{a} Pr(f^{(k)}, a|e^{(k)}) \Big] \tag{2.7}$$

To perform this maximization, we need to have the alignments. But, the alignments are hidden. Typically, EM algorithm (Dempster et al., 1977) is used in such a scenario. In general, the EM algorithm is a common way of inducing latent structures from unlabelled data in an unsupervised manner. For a given sentence pair, there are many possible alignments. We can find the best alignment $\boldsymbol{a}^*$ as follows:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}, \theta) \tag{2.8}$$

The best alignment $\boldsymbol{a}^*$ is called the *Viterbi alignment* of sentence pair $(\boldsymbol{f}, \boldsymbol{e})$. We can then evaluate the quality of this Viterbi alignment by comparing it to a manually produced reference alignment using the measure explained in section 2.7.

To summarize, given a bilingual training data, we estimate the parameters of our model using the EM algorithm. Using the parameters, we find the best alignment for each sentence pair. The output of word alignment is the given bilingual data with the predicted alignments for each sentence pair. In the following sections, we will describe IBM Models 1 and 2 in a formulation similar to the one for HMM-based alignment model.

2.3.1 IBM Model 1

Introducing the alignment variables allows us to model $Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e})$ instead of $Pr(\boldsymbol{f}|\boldsymbol{e})$. The alignment model can be structured without loss of generality as follows:

$$Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) = \prod_{j=1}^{J} Pr(f_j, a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) \tag{2.9}$$

$$= \prod_{j=1}^{J} Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) Pr(f_j|f_1^{j-1}, a_1^j, e_1^I) \tag{2.10}$$

In IBM model 1, we assume a uniform probability distribution for $Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I)$, which depends only on the length of the English sentence:

$$Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) = \frac{1}{I+1} \tag{2.11}$$

where we define e_0 to be a special NULL word that is responsible for generating English words without any corresponding words in the French sentence; $a_j = 0$ specifies that word f_j is generated from the NULL word.

Furthermore, we assume that $Pr(f_j|f_1^{j-1}, a_1^j, e_1^I)$ depends only on f_j and e_{a_j}. Hence, we have:

$$Pr(f_j|f_1^{j-1}, a_1^j, e_1^I) = p(f_j|e_{a_j}) \tag{2.12}$$

Putting everything together, we have the following translation model for IBM Model 1:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) = \frac{1}{(I+1)^J} \sum_{a} \prod_{j=1}^{J} p(f_j|e_{a_j}) \tag{2.13}$$

Parameter estimation and decoding is explained for IBM Model 2 since Model 1 is a special case of Model 2.

2.3.2 IBM Model 2

In IBM Model 2, we make the same assumptions as in Model 1 except that $Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I)$ is assumed to depend only on j, a_j, J and I. A set of alignment parameters is defined as follows:

$$Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) = p(a_j|j, J, I) \tag{2.14}$$

Hence, we have the following translation model for IBM Model 2:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) = \sum_{a} \prod_{j=1}^{J} p(a_j|j, J, I) p(f_j|e_{a_j}) \tag{2.15}$$

Parameter Estimation

As explained in the previous section, the expectation maximization algorithm or EM algorithm (Dempster et al., 1977) is used for partially-observed data. This algorithm initializes the model parameters (typically with uniform distribution). The algorithm iterates over the expectation and maximization steps until convergence. In the expectation step, it applies the model to the data; in the maximization step, it learns the model from data. The output of this algorithm is the set of translation parameters and alignment parameters. Once we estimate these parameters, we can use them to find the most probable alignment, also called the Viterbi alignment, for any given training example as follows:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} Pr(\boldsymbol{a}|\boldsymbol{f}, \boldsymbol{e}) \tag{2.16}$$

For IBM model 2, the solution to equation 2.16 is as follows:

$$a_j{}^* = \arg\max_{i \in \{0,\ldots,I\}} \; (p(i|j, J, I) \times p(f_j|e_i)) \tag{2.17}$$

2.4 Hidden Markov Alignment Model

In this section, we present the HMM-based alignment model proposed by Vogel et al. (1996). We begin with the motivation behind this model. The training and decoding algorithms for this model will be explained in the following sections. We will present the forward-backward and the Viterbi algorithm for the word alignment problem and hence we will explain how to compute the expected counts and posterior probabilities for a parallel training data.

2.4.1 Motivation

Translation of sentences in parallel texts, especially for language pairs from Indo-European languages, shows a strong localization effect. Words close to each other in a sentence in the source language remain close in the translation sentence. Figure 2.2 shows this effect for the language pair German-English. Note that we visualize the word alignment task by a matrix where alignments between words are represented by points in the alignment matrix. Each word in the German sentence is aligned to a word in the English sentence. Alignments tend to preserve their local neighbourhood when going from German to English, as shown in Figure 2.2.

2.4.2 Alignment with HMM

As explained previously, introducing the alignment variables allows us to model $Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e})$ instead of $Pr(\boldsymbol{f}|\boldsymbol{e})$. The alignment model can be structured without loss of generality as in equation 2.10. In the Hidden Markov alignment model, we assume a first-order dependence

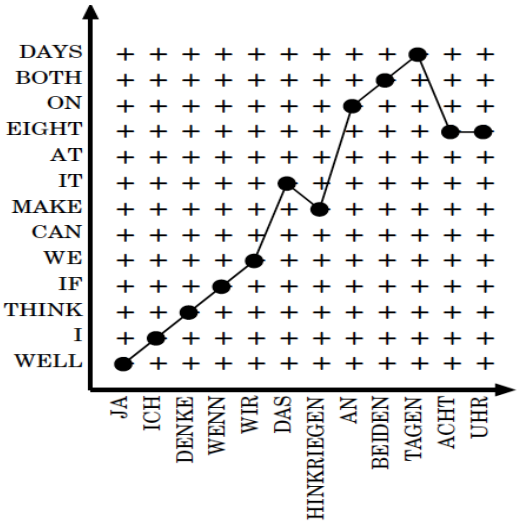


Figure 2.2: Word alignment for a German-English sentence pair (figure from Vogel et al. (1996))

for the alignments a_j; also, the translation probability is assumed to be dependent on the word at position a_j and not on the one at position a_{j-1}. Hence, we have:

$$Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) = p(a_j|a_{j-1}, I) \tag{2.18}$$

$$Pr(f_j|f_1^{j-1}, a_1^j, e_1^I) = p(f_j|e_{a_j}) \tag{2.19}$$

Putting everything together, we have the following basic HMM-based model:

$$Pr(f|e) = \sum_a \prod_{j=1}^{J} [\ p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j})\ ] \tag{2.20}$$

with two components: the alignment probability (transition probability) $p(a_j|a_{j-1}, I)$ or $p(i|i', I)$, and the translation probability (emission probability) $p(f_j|e_{a_j})$. Figure 2.3 shows the graphical model of the HMM-based word alignment model.

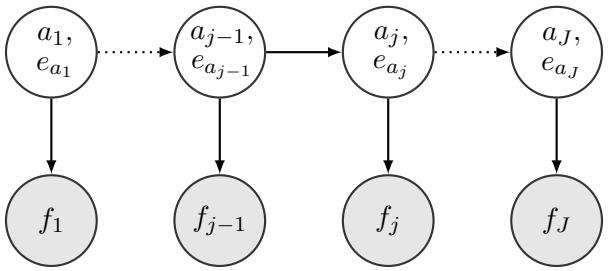


Figure 2.3: Graphical model of the basic HMM-based word alignment model

Estimating the alignment parameters $p(i|i', I)$ using the conventional maximum likelihood (ML) criterion depends on the expectation of consecutive French words generated from the English word at position i' and i with English sentence length I. This clearly could suffer from sparsity within the available bitext. To address this problem, Vogel et al. (1996) make the alignment parameters independent of the absolute word positions. It is assumed that the alignment probabilities $p(i|i', I)$ depend only on the jump width $(i - i')$. Hence, the alignment probabilities are estimated using a set of distortion parameters $c(i - i')$ as follows:

$$p(i|i', I) = \frac{c(i - i')}{\sum_{i''=1}^{I} c(i'' - i')} \tag{2.21}$$

where at each iteration $\{c(i - i')\}$ is the fractional count of transitions with jump width $d = i - i'$:

$$c(d) = \sum_{j=1}^{J-1} \sum_{i=1}^{I} Pr(a_j = i, a_{j+1} = i + d|\boldsymbol{f}, \boldsymbol{e}, \theta) \tag{2.22}$$

where θ is the model parameters obtained from the previous iteration and the terms $Pr(a_j = i, a_{j+1} = i + d|\boldsymbol{f}, \boldsymbol{e}, \theta)$ can be efficiently computed using the Forward-Backward algorithm (Rabiner, 1989).

To illustrate how the alignment model is computed for a given sentence pair and its alignment, consider the following example. We visualize the word alignment task by a matrix as in Figure 2.4. Here, alignments between words are represented by points in the alignment matrix. We are given the alignment vector $\boldsymbol{a} = \langle 1, 3, 4, 5 \rangle$. To assign a probability to the alignment model using the HMM model, we consider the first alignment point in the figure, i.e. (je, I).[1] Hence, the emission probability $p(je|I)$ is considered for this point. The second alignment point is (voudrais, like). As the previous French word is aligned to English word I, which is at position 1 in the English sentence, we have a transition to position 3 in the English sentence for the word *like*. Therefore, we have to multiply $p(3|1, 5)$ and also $p(voudrais|like)$ for the emission probability. Similarly, we have a transition probability $p(4|3, 5)$ because of jumping to position 4 for a and an emission probability for $p(un|a)$. Finally, we multiply transition probability $p(5|4, 5)$ and emission probability $p(croissant|croissant)$.

Suppose we are given $c(1) = 2$ and $c(2) = 1$. Computing the transition probabilities is straightforward; for example, $p(3|1, 5) = \frac{c(2)}{c(1)+c(2)} = \frac{1}{3}$.

2.4.3 Training

Parameter estimation of the HMM-based word alignment model can be done using the Baum-Welch (Baum, 1972) algorithm which makes use of the forward-backward algorithm.

[1]For the sake of simplicity, we have not included the initial state probabilities in this example.

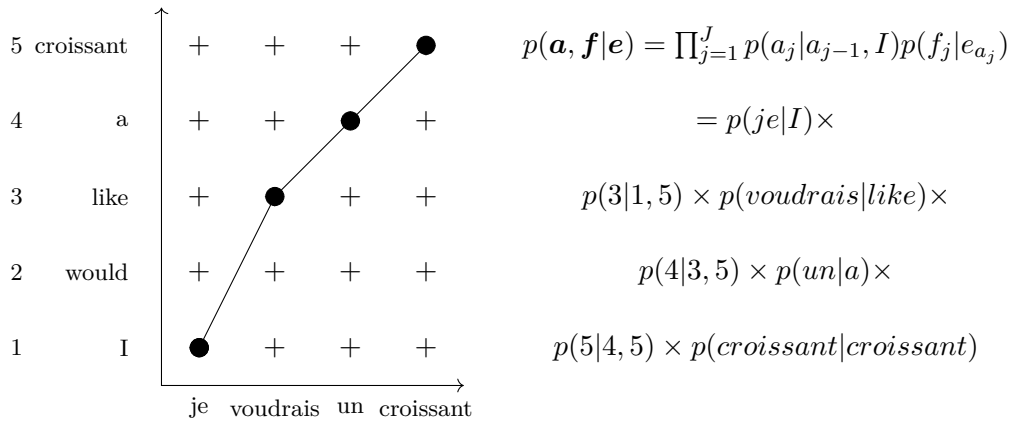


Figure 2.4: Alignment model computation for a HMM-based model for a French-English sentence pair.

Forward-backward algorithm is a dynamic programming algorithm that makes it possible to avoid the summation over all alignments. We will look at this algorithm for the HMM-based word alignment model here. A compact representation of the HMM model for alignment is $\theta = \{p(i|i', I), p(f|e), p(i)\}$ where $\{p(i|i', I)\}$ are the transition probabilities, $\{p(f_j|e_i)\}$ are the emission probabilities and $\{p(i)\}$ are the initial state probabilities. Note that we include the initial state probabilities here to give a more precise HMM model. The initial state distribution is defined as $p(i) = Pr(a_1 = i)$ (i.e. $p(i)$ is the probability of aligning the first word in the French sentence to the i-th word in the English sentence). The forward and backward probabilities can be computed efficiently using the recursive definitions, as shown in Figure 2.5.

Posterior Probabilities

With the forward and backward probabilities, we can calculate the posterior probabilities. The probability of aligning the j-th French word to the i-th English word is defined as follows:

$$\gamma_i(j) = Pr(a_j = i|\boldsymbol{f}, \theta) = \frac{\alpha_i(j)\beta_i(j)}{\sum_{l=1}^{I} \alpha_l(J)} \tag{2.23}$$

The probability of aligning the j-th French word to the i-th English word and $(j-1)$-th French word to the i'-th English word is defined as follows:

$$\xi_{i'i}(j-1) = Pr(a_{j-1} = i', a_j = i|\boldsymbol{f}, \theta) = \frac{\alpha_{i'}(j-1)p(i|i', I)\beta_i(j)p(f_j|e_i)}{\sum_{l=1}^{I} \alpha_l(J)} \tag{2.24}$$

Algorithm 1 Forward-backward($\boldsymbol{f}, \boldsymbol{e}, \theta$)

-given: A French sentence $f_1 \dots f_J$, an English sentence $e = e_1 \dots e_I$ and the HMM parameter set $\theta = \{p(i|i', I), p(f_j|e_i), p(i)\}$
Let $\alpha_i(j) = Pr(f_1 \dots f_j, a_j = i)$ be the forward probabilities
Base case:

$$\alpha_i(1) = p(i)p(f_1|e_i) \text{ for } i \in 1, \dots, I$$

Recursive case :

$$\alpha_i(j) = p(f_j|e_i) \sum_{i'=1}^{I} \alpha_{i'}(j-1).p(i|i', I) \text{ for all } i \in 1, \dots, I \text{ and } j \in 2, \dots, J$$

Let $\beta_i(j) = Pr(f_{j+1} \dots f_J|a_j = i)$ be the backward probabilities
Base case:

$$\beta_{i'}(J) = 1 \text{ for all } i' \in 1, \dots, I$$

Recursive case :

$$\beta_{i'}(j-1) = \sum_{i=1}^{I} \beta_i(j).p(i|i', I).p(f_j|e_i) \text{ for all } i' \in 1, \dots, I \text{ and } j \in 2, \dots, J$$

Figure 2.5: The FORWARD-BACKWARD algorithm

Parameter Re-estimation

The Baum-Welch algorithm adjusts the model parameters using the expected couts. Let $\boldsymbol{D}$ denote the parallel data. Let $c(f, e)$ be the posterior count accumulated over all training sentences of the English word e generating the French word f. Hence, we have

$$c(f, e) = \sum_{(\boldsymbol{f}, \boldsymbol{e}) \in \boldsymbol{D}} \sum_{i,j} \gamma_i(j) \delta(f_j, f) \delta(e_i, e) \tag{2.25}$$

The translation probabilities (emission parameters) are then estimated as

$$p(f|e) = \frac{c(f, e)}{\sum_{f'} c(f', e)} \tag{2.26}$$

Let $c(i', i, I)$ be the posterior count of transitions with jump width $(i - i')$ over all training sentences:

$$c(i', i, I) = \sum_{(\boldsymbol{f}, \boldsymbol{e}) \in \boldsymbol{D}} \sum_{i', i} c(i - i') \delta(|\boldsymbol{e}|, I) \tag{2.27}$$

where $c(i - i')$ is computed using equation 2.22 and $|\boldsymbol{e}|$ denotes the length of the sentence $\boldsymbol{e}$. Note that the terms in this equation are the transition posterior probabilities ξ. The

Algorithm 2 Viterbi($\boldsymbol{f}, \boldsymbol{e}, \theta$)

-given: A French sentence $f_1 \dots f_J$, an English sentence $e = e_1 \dots e_I$ and the HMM parameter set $\theta = \{p(i|i', I), p(f_j|e_i), p(i)\}$

Base case:

$$V[i, 1] = p(i)p(f_1|e_i) \quad \text{for } i \in 1, \dots, I$$

Recursive case :

$$V[i, j] = \max_{i'}\{V[i', j-1] \cdot p(i|i', I) \cdot p(f_j|e_i)\} \quad \text{for all } i \in 1, \dots, I \text{ and } j \in 2, \dots, J$$

The best score is $\max_i V[i, J]$

To find the most probable alignment, trace back using the V matrix

Figure 2.6: The VITERBI algorithm

alignment probabilities (transition parameters) are then estimated as

$$p(i|i', I) = \frac{c(i', i, I)}{\sum_{i''=1}^{I} c(i', i'', I)} \tag{2.28}$$

2.4.4 Decoding

After training, Viterbi decoding is used to find the best alignment $\boldsymbol{a}^*$:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} \prod_{j=1}^{J} [\ p(a_j|a_{j-1}, I).p(f_j|e_{a_j})\ ] \tag{2.29}$$

The Viterbi algorithm is shown in Figure 2.6. Both the Viterbi and the forward-backward algorithms can be computed in $O(I^2 J)$.

To avoid the summation over all possible alignments in equation 2.20, Vogel et al. (1996), use the maximum approximation where only the Viterbi alignment is used to collect the counts:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) \cong \max_{\boldsymbol{a}} \prod_{j=1}^{J} [\ p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j})\ ] \tag{2.30}$$

Later on, Och and Ney (2000a) use the Baum-Welch algorithm to train the parameters of the model efficiently.

2.5 Extensions and Improvements

In this section, we will present some of the extensions to the HMM alignment model. We first look at a group of methods that propose refined alignment models in section 2.5.1. Then, we give an overview of the methods that address empty word problem for the HMM model in section 2.5.2. We discuss methods that model fertility in an HMM-based model in section

2.5.3, and a method that incorporates part of speech tag information in the translation model in section 2.5.4. We also explore a word-to-phrase HMM alignment model in section 2.5.5. A feature-enhanced HMM model is presented in section 2.5.6.

2.5.1 Refined Alignment Models

The motivation behind these methods is that the transition model in the HMM-based alignment model is coarse. Transition probabilities only depend on the jump width from the last state to the next state. They, therefore, enhance the transition model in the HMM. We are going to look at three methods : Och and Ney (2000a), Och and Ney (2003) and He (2007).

Class Dependent Transition Models

Och and Ney (2000a) focuses on improving the transition probability. The motivation behind their approach is that the count table $c(i - i')$ has only $2 \cdot I_{max} - 1$ entries which is suitable for a small corpus, but for a large one, it is possible to give an improved model for $Pr(a_j|f_1^{j-1}, a_1^{j-1}, I)$. For instance, the effect of dependence on the surrounding words such as $e_{a_{j-1}}$ or f_j is analyzed. As conditioning on all English words (or French words) would result in a huge number of parameters, equivalence classes G over the English and French words are used. G is a mapping of words to classes. The categorization of words into classes (here, 50 classes) is performed using the statistical learning procedure described in Kneser and Ney (1993). Och and Ney (2000a) extend the transition probabilities to be class dependent as follows:

$$p(a_j - a_{j-1}|G(e_{a_j}), G(f_j), I) \tag{2.31}$$

Och and Ney (2003) extend the alignment parameters to include a dependence on the class of the preceding English word:

$$p(a_j|a_{j-1}, G(e_{a_{j-1}}), I) \tag{2.32}$$

Word-Dependent Transition Model

Knowledge of transition probabilities given a particular English word e is not modelled in the original HMM model. To put it more simply, knowledge of jumping from e to another position, e.g., jumping forward (monotonic alignment) or backward (non-monotonic alignment) is not modelled. To improve the transition model, He (2007) extends the transition probabilities to be word-dependent. The proposed word-dependent HMM model is as follows:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) = \sum_{a} \prod_{j=1}^{J} [\ p(a_j|a_{j-1}, e_{a_{j-1}}, I).p(f_j|e_{a_j})\ ] \tag{2.33}$$

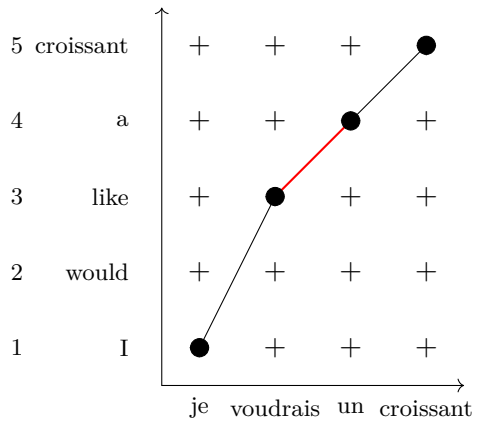


Figure 2.7: Word-dependent transition probability $p(4|3, like, 5)$ for the red edge.

Figure 2.7 shows an example of a word-dependent transition probability $p(4|3, like, 5)$ for the red edge. We need to estimate the transition parameter $p(a_j|a_{j-1}, e_{a_{j-1}}, I)$. Consequently, the full parameter set that we need to estimate is $\theta = \{p(i|i', e_{i'}, I), p(f_j|e_i)\}$. The estimation for word-dependent transition probabilities $p(i|i', e_{i'}, I)$ can be carried out using maximum likelihood training:

$$p(i|i', e, I) = \frac{c(i - i'; e)}{\sum_{i''=1}^{I} c(i'' - i'; e)} \tag{2.34}$$

where at each iteration the word-dependent distortion set $\{c(i - i'; e)\}$ is computed as follows:

$$c(d; e) = \sum_{j=1}^{J-1} \sum_{i=1}^{I} \delta(e_{a_j}, e) Pr(a_j = i, a_{j+1} = i + d|\boldsymbol{f}, \boldsymbol{e}, \theta) \tag{2.35}$$

where $d = i - i'$ is the jump width and δ is the Kronecker delta function. For non-frequent words, the data samples for $c(d; e)$ is very limited and hence leads to data sparsity problem. To address this problem, maximum a posteriori (MAP) framework is applied (Gauvain and Lee, 1994) where an appropriate prior $g(\theta|\boldsymbol{e})$ is used to incorporate prior knowledge into the model parameter estimation:

$$\theta_{MAP} = \arg\max_{\theta} p(\boldsymbol{f}|\boldsymbol{e}, \theta) g(\theta|\boldsymbol{e}) \tag{2.36}$$

The relation between ML and MAP estimation is through the Bayes theorem:

$$p(\theta|\boldsymbol{f}, \boldsymbol{e}) \propto p(\boldsymbol{f}|\boldsymbol{e}, \theta) g(\theta|\boldsymbol{e}) \tag{2.37}$$

As the transition model $p(i|i', e_{i'}, I)$ is a multinomial distribution, its conjugate prior is a Dirichlet distribution with the form

$$g(p(i|i', e_{i'}, I)|\boldsymbol{e}) \propto \prod_{i=1}^{I} p(i|i', e_{i'}, I)^{v_{i',i}-1} \tag{2.38}$$

where $v_{i',i}$ is the set of hyper-parameters of the prior distribution. By substituting 2.38 in 2.37, the iterative MAP training formula for transition model is obtained as

$$p_{MAP}(i|i', e, I) = \frac{c(i-i'; e) + v_{i',i} - 1}{\sum_{i''=1}^{I} c(i''-i'; e) + \sum_{i''=1}^{I} v_{i',i''} - I} \tag{2.39}$$

Hyper-parameter set $\{v_{i',i}\}$ of the prior distribution is set to word-independent transition probabilities:

$$v_{i',i} = \tau.p(i|i', I) + 1 \tag{2.40}$$

where τ is a positive parameter which needs to be tuned on a held-out dataset. Substituting 2.40 in 2.39, we obtain the MAP-based transition model:

$$p_{MAP}(i|i', e, I) = \frac{c(i-i'; e) + \tau.p(i|i', I)}{\sum_{i''=1}^{I} c(i''-i'; e) + \tau} \tag{2.41}$$

In this new formulation, for frequent words with a substantial amount of data samples for $c(d; e)$, the summation in the denominator is large; therefore, $p_{MAP}(i|i', e, I)$ is dominated by the data distribution. On the contrary, for rare words with low counts of $c(d; e)$, $p_{MAP}(i|i', e, I)$ will approach to the word-independent model. Note that we can vary τ to control the contribution of the prior in this model. For instance, for a small τ, a weak prior is applied, and the transition probability is more dependent on the training data of that word. Conversely, for a large τ, a stronger prior is applied, and the model will approach to the word-independent model.

2.5.2 Empty Word

Some words in a French sentence may have no relation to any of the words in the corresponding English sentence. To model this in the IBM word alignment models, Brown et al. (1993) define e_0 to be a special NULL (empty) word that is treated just like another English word; hence, $a_j = 0$ specifies that f_j is generated from a NULL word. The inclusion of a NULL word is helpful since we still want to align each French word to an English word. If we do not model NULL, we have to align those French words with no correspondences in the English sentence, to arbitrary unrelated words in the English sentence. This results in a model with a high alignment error rate.

Figure 2.8 shows an example of a pair of sentences where the French word *des* has no corresponding word in the English sentence. IBM models align *des* to NULL.

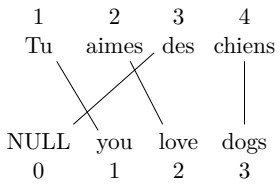


Figure 2.8: Introducing a NULL word at position 0 as Brown et al. (1993) to generate the French word *des* with no corresponding translation in the English sentence.

In the original formulation of the HMM model, there is no empty word that is responsible for generating French words with no corresponding English words. A direct inclusion of the NULL word in the HMM model by adding an e_0 as in Brown et al. (1993) is problematic if we want to model the jump distances $i - i'$, as the position $i = 0$ for the NULL word is chosen arbitrarily. We explain two methods that address this issue: Och and Ney (2000a) and Toutanova et al. (2002).

Adding a Group of NULLs

Och and Ney (2000a) extend the HMM network by I NULL words e_{I+1}^{2I} such that the English word e_i has a corresponding NULL word e_{i+I}. The NULL position chosen by the model is determined by the previously visited English word. The following constraints are enforced for the transition probabilities in the HMM network ($i \leq I, i' \leq I$):

$$p(i + I|i', I) = p_0.\delta(i, i') \tag{2.42}$$

$$p(i + I|i' + I, I) = p_0.\delta(i, i') \tag{2.43}$$

$$p(i|i' + I, I) = p(i|i', I) \tag{2.44}$$

The parameter p_0 is the probability of transitioning to the NULL word, which has to be optimized on a held-out dataset. Och and Ney (2000a) set $p_0 = 0.2$ for their experiments. To illustrate how the NULL word is chosen in this approach, we use the following example. Consider the sentence pair given previously. As the length of the English sentence is 3, we extend the HMM network by 3 NULL words as in Figure 2.9. These NULL words are added to the end of the English sentence. The French word *des* has no corresponding word in the English sentence; hence, it should be aligned to a NULL. However, there are three possible choices. Since the previous French word *aimes* is aligned to a non-NULL word, we fall into the case 2.42 . The word *aimes* is aligned to *love* which is at position 2. Therefore, the position of the NULL word for *des* is $2 + 3 = 5$, as shown in Figure 2.9.

The purpose of the above constraints is to preserve the locality in the HMM when an alignment to NULL is necessary. Suppose, we use the IBM NULL model for our HMM, as in Figure 2.8. The jump sequence $\{a_j - a_{j-1}\}$ is $\{1, -2, 3\}$ where -2 is for jumping to

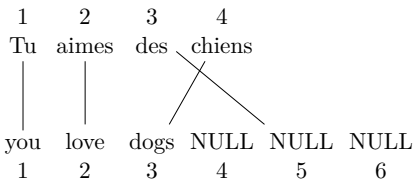


Figure 2.9: Introducing a group of NULL words as Och and Ney (2000a) to generate the French word *des*.

NULL (from position 2 to position 0) and 3 is for jumping back to position 3 from NULL. Note that transition probabilities for jump widths of -2 and 3 are caused by the inclusion of empty word at position 0, and these probabilities should not be included in the alignment model computation. The method proposed by Och and Ney (2000a) resolves this problem. In this method, the transition probabilities used in the computation of the alignment model are $p(2|1,3)$, $p(5|2,3)$ and $p(3|5,3)$ where $p(5|2,3)$ is p_0 due to constraint 2.42 and $p(3|5,3)$ is $p(3|2,3)$ due to constraint 2.44. The locality is preserved even though we align *des* to NULL because the last English word (at position 2) is remembered by the HMM model so that the next alignment to a non-NULL word is computed using $p(3|2,3)$.

Translation Model for NULL

This method presents a new generative model for the source language words that do not have any correspondences in the target language (Toutanova et al., 2002). The translation probabilities for the French words with no correspondences in English is modelled such that these words are generated from NULL and also from the next word in the French sentence by a mixture model. For instance, in the pair *des chiens* in Figure 2.8, *chiens* contributes extra information in generation of *des*. The new formulation for the probability of a French word given that it does not have a corresponding word in the English sentence is:

$$p(f_j|a_j = 0) = \lambda p(f_j|f_{j+1}, e_{a_j} = \text{NULL}) + (1-\lambda)p(f_j|e_{a_j} = \text{NULL}) \tag{2.45}$$

The probabilities $p(f_j|f_{j+1}, e_{a_j} = \text{NULL})$ are re-estimated from the training corpus using the EM algorithm. Note that the dependence of a French word on the next French word requires a change in the generative model. First, alignments are proposed for all words in the French sentence and then French words are generated given their alignment starting from the end of the sentence towards the beginning. For this model, there is an efficient dynamic programming algorithm similar to the forward-backward algorithm that can be used for computations in EM.

2.5.3 Modelling Fertility

One major advantage of IBM models 3-5 over the HMM model is the presence of a model for English word fertility. Thus, knowledge that some French words translates as phrases in English was incorporated in these models. HMM-based word alignment model has no memory, but the previous alignment, about how many words have been aligned to an English word, and this memory is not used to decide whether to generate more words from this word. This decision is independent of the word and is estimated over all the words in the corpus as estimating the transition probability with a jump of size 0 is computed using equation 2.21. Toutanova et al. (2002) extend the HMM model such that deciding whether to generate more French words from the previous English word $e_{a_{j-1}}$ or to move to another English word depends on the word $e_{a_{j-1}}$. To do this, they introduce a factor $p(stay|e_{a_{j-1}})$, where the boolean random variable *stay* depends on the English word $e_{a_{j-1}}$. The rationale for this method is that as in most cases words with fertility more than one generate consecutive words in the source language, this method approximately models fertility. They change the baseline model, equation 2.20, as follows:

$$Pr(\boldsymbol{f}|\boldsymbol{e}) = \sum_{\boldsymbol{a}} \prod_{j=1}^{J} [\tilde{p}(a_j|a_{j-1}, e_{a_{j-1}}, I)p(f_j|e_{a_j})] \tag{2.46}$$

where

$$\begin{aligned}\tilde{p}(a_j|a_{j-1}, e_{a_{j-1}}, I) = {} & \delta(a_j, a_{j-1})p(stay|e_{a_{j-1}}) \\ & + (1 - \delta(a_j, a_{j-1}))(1 - p(stay|e_{a_{j-1}}))p(a_j|a_{j-1}, I)\end{aligned}$$

A jump of size zero in the new alignment (transition) probabilities $\tilde{p}(a_j|a_{j-1}, e_{a_{j-1}}, I)$ depends on the English word $e_{a_{j-1}}$.

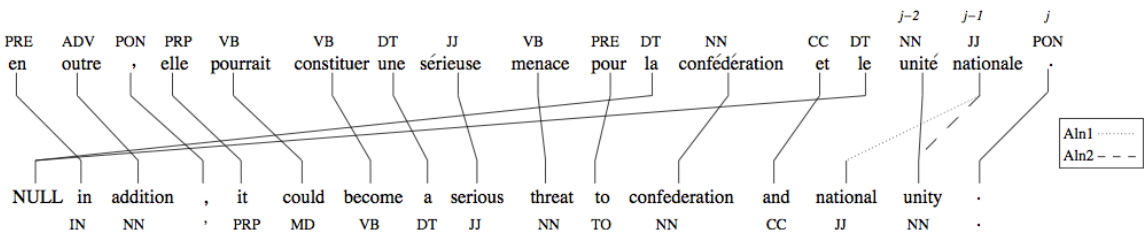
To handle the sparsity problem in estimating $p(stay|e_{a_{j-1}})$, smoothing is done using a probability of a jump zero as the prior:

$$p(stay|e_{a_{j-1}}) = \lambda p_{ZJ} + (1 - \lambda)p(stay|e_{a_{j-1}}) \tag{2.47}$$

where p_{ZJ} is the alignment probability of the baseline model for a jump of size zero $p_{ZJ} = Pr(a_j = i|a_{j-1} = i, I)$.

Modeling fertility is challenging in the HMM framework as it violates the Markov assumption. Whereas the HMM jump model considers only the prior state, fertility requires looking across the whole state space. Therefore, the standard forward-backward and Viterbi algorithms do not apply.

Zhao and Gildea (2010) build a fertility hidden Markov model by adding fertility to the HMM. Their model assumes that fertility ϕ_i for a word e_i follows a Poisson distribution



	e_{a_j}	$e_{a_{j-1}}$	$e_{a_{j-2}}$	P($a_{j-1}\|a_{j-2}, l$)	P($f_{j-1}\|e_{a_{j-1}}$)	P($a_j\|a_{j-1}, l$)	P($f_j\|e_{a_j}$)	total
Aln1	.	national	unity	0.0292	0.5741	0.05357	0.9766	$8.77\text{x}10^{-4}$
Aln2	.	unity	unity	0.0862	0.2789	0.31	0.9766	$7.2\text{x}10^{-3}$

Figure 2.10: The basic HMM model makes a simple alignment error (figure from Toutanova et al. (2002)).

Poisson($\phi_i, \lambda(e_i)$). Compared to IBM Models 3-5, this model has a much smaller number of parameters to learn; one parameter for each English word. In this method, we estimate the parameters using the EM algorithm. However, to compute the expected counts, we have to sum over all possible alignments, which is exponential. Gibbs sampling is thus used to approximate the expected counts.

2.5.4 Part of Speech Tags in a Translation Model

This method extends the HMM model by incorporating part of speech (POS) tag information of the source and target languages in the translation model (Toutanova et al., 2002). Basic HMM model introduces some alignment irregularities. For instance, the transition of the $NP \rightarrow JJ\ NN$ rule to $NP \rightarrow NN\ JJ$ from English to French[2]. There are two main reasons why translation probabilities may not catch such irregularities in monotonicity. (1) When both English adjective and noun are unknown words, the translation probabilities will be close to each other after smoothing. (2) When the adjective and noun are words that are frequently seen together in English and therefore there will be an indirect association between the English noun and the translation of the English adjective. In such cases, the word translation probabilities are not differentiating enough and alignment probabilities become the dominating factor to make a decision about which English word should be aligned to f_j. Figure 2.10 shows how the basic HMM model makes such an alignment mistake. The table shows the alignment and translation probabilities of two alignments, $Aln1$ and $Aln2$, for the last three words. $Aln1$ correctly aligns the pair of adjective and noun in French to the corresponding pair in English while $Aln2$ incorrectly aligns both French noun and adjective to the English word. (i.e. $e_{a_{j-1}} = unity$ and $e_{a_{j-2}} = unity$). Since the jump width sequences $\{(a_{j-1} - a_{j-2}), (a_j - a_{j-1})\}$ for $Aln2$ is $\{0, 1\}$ which are more probable than $\{-1, 2\}$ for

[2]Word order changes in translating an adjective-noun pair in English to French.

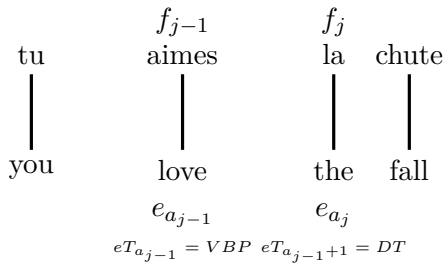


Figure 2.11: Part of speech tag information for modelling local word order variation

$Aln1$, and the translation probabilities are not differentiating enough, $Aln2$ is preferred by the HMM model.

POS Tags for Translation Probabilities

Let $\boldsymbol{eT}$ and $\boldsymbol{fT}$ be the possible part of speech tag sequences of the sentences $\boldsymbol{e}$ and $\boldsymbol{f}$. The model with part of speech tags for translation probabilities has the following form:

$$Pr(\boldsymbol{f}, \boldsymbol{fT}|\boldsymbol{e}, \boldsymbol{eT}) = \sum_{a} \prod_{j=1}^{J} [\ p(a_j|a_{j-1}, I).p(fT_j|eT_{a_j}).p(f_j|e_{a_j}) \tag{2.48}$$

This model introduces tag translation probabilities as an extra factor to equation 2.20. This factor boosts the translation probabilities for words of part of speech that are often translations of each other. Hence, it provides a prior knowledge of the translations of a word based on its part of speech. This factor should not be too sharp that dominates the alignment and translation probabilities. To avoid this potential problem, smoothing is done for tag translation probabilities:

$$p(fT_j|eT_{a_j}) = \lambda \tilde{p}(fT_j|eT_{a_j}) + (1 - \lambda)\frac{1}{T} \tag{2.49}$$

where T is the size of the French tag set and $\lambda = 0.1$. It is necessary to set λ to a small value to prevent tag translation probabilities from becoming very sharp in EM and dominating the alignment and translation probabilities.

In section 2.5.1, we discussed methods that explores conditioning the alignment probabilities on the class of the English word $e_{a_{j-1}}$ and/or that of French word f_j. Toutanova et al. (2002) investigates whether they can improve the model by conditioning on the POS tags of those words or even more words around the alignment position. For instance, consider using $p(a_j|a_{j-1}, eT_{a_{j-1}-1}, eT_{a_{j-1}}, eT_{a_{j-1}+1})$ as the alignment probability. This model is helpful in modelling local word order variations. Figure 2.11 shows an example where the probability of aligning $f_j = la$ (with $fT_j = DT$) to *the* will be boosted knowing that $eT_{a_{j-1}} = VBP$ and $eT_{a_{j-1}+1} = DT$.

2.5.5 HMM Word and Phrase Alignment

This method develops a generative probabilistic model of Word-to-Phrase (WtoP) alignment (Deng and Byrne, 2008). Suppose, we have a target sentence in English $\boldsymbol{e} = e_1^I$ and its translation in the source language French $\boldsymbol{f} = f_1^J$. Here, the term *phrase* is used to refer to any subsequence in the source sentence. We define the phrase count variable K, which specifies that the French sentence is segmented as a sequence of K phrases: $\boldsymbol{f} = v_1^K$.

Each French phrase is generated as a translation of an English word. The correspondence between English words and French phrases are determined by the alignment sequence a_1^K. The length of each phrase is specified by the random process ϕ_1^K which is constrained to satisfy $\sum_{k=1}^K \phi_k = J$. French phrases are allowed to be generated by NULL. Hence, a binary NULL prediction sequence h_1^K is introduced. If $h_k = 0$, then v_k is aligned to NULL; if $h_k = 1$, then v_k is aligned to e_{a_k}. Taking into account all these quantities, the alignment can be treated as $\boldsymbol{a} = (\phi_1^K, a_1^K, h_1^K, K)$. We can rewrite alignment model $Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e})$ as:

$$\begin{aligned} Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) = Pr(v_1^K, K, a_1^K, h_1^K, \phi_1^K|\boldsymbol{e}) &= Pr(K|J, \boldsymbol{e}) \\ &\times Pr(a_1^K, \phi_1^K, h_1^K|K, J, \boldsymbol{e}) \\ &\times Pr(v_1^K|a_1^K, \phi_1^K, h_1^K, K, J, \boldsymbol{e}) \end{aligned} \tag{2.50}$$

We describe the component distributions here:

- Phrase count distribution: $Pr(K|J, \boldsymbol{e})$ specifies the distribution over the number of phrases in the French sentence given the length of the French sentence and the English sentence. A single parameter distribution is used for phrase count distribution:

$$Pr(K|J, I) \propto \eta^K \tag{2.51}$$

  where $\eta \geq 1$ controls the segmentation of the French sentence into phrases such that larger values of η leads to French sentence segmentation with many short phrases. η is used as a tuning parameter to control the length of phrases.

- Word-to-Phrase alignment: The alignment is modelled as a Markov process that specifies the length of phrases and their alignment with English words:

$$\begin{aligned} Pr(a_1^K, \phi_1^K, h_1^K|K, J, \boldsymbol{e}) &= \prod_{k=1}^K Pr(a_k, \phi_k, h_k|a_{k-1}, \phi_{k-1}, \boldsymbol{e}) \\ &= \prod_{k=1}^K p(a_k|a_{k-1}, h_k; I) d(h_k) n(\phi_k; e_{a_k}) \end{aligned} \tag{2.52}$$

  The word-to-phrase alignment a_k is a first order Markov process as in word-to-word HMM (Vogel et al., 1996). It is formulated with a dependency on the NULL prediction

variable as follows:

$$p(a_j|a_{j-1}, h_j; I) = \begin{cases} 1 & a_j = a_{j-1}, h_j = 0 \\ 0 & a_j \neq a_{j-1}, h_j = 0 \\ p_a(a_j|a_{j-1}; I) & h_j = 1 \end{cases} \tag{2.53}$$

The phrase length model $n(\phi; e)$ is a form of English word fertility. It specifies the probability that an English word e generates a French phrase with ϕ words. A distribution $n(\phi; e)$ over the values $\phi = 1, \ldots, N$ is maintained as a table for each English word. The model also has a table of transition probabilities $p_a(i|i', I)$ for all English sentence lengths I. We use $d(0) = p_0$, and $d(1) = 1 - p_0$. p_0 is a tuning parameter that controls the tendency towards the insertion of French phrases.

- Word-to-Phrase translation: The translation of words to phrases is computed as follows:

$$Pr(v_1^K|a_1^K, h_1^K, \phi_1^K, K, J, \boldsymbol{e}) = \prod_{k=1}^{K} p(v_k|e_{a_k}, h_k, \phi_k) \tag{2.54}$$

We introduce the notation $v_k = v_k[1], \ldots, v_k[\phi_k]$ and a dummy variable x_k:

$$x_k = \begin{cases} e_{a_k} & \text{if } h_k = 1 \\ NULL & \text{if } h_k = 0 \end{cases} \tag{2.55}$$

where French phrases are conditionally independent given the individual English words. two models are introduced for word-to-phrase translation: The simplest model is based on context-independent word-to-word translation:

$$p(v_k|e_{a_k}, h_k, \phi_k) = \prod_{j=1}^{\phi_k} t_1(v_k[j]|x_k) \tag{2.56}$$

A more complex model captures word context within the French language phrase via bigram translation probabilities:

$$p(v_k|e_{a_k}, h_k, \phi_k) = t_1(v_k[1]|x_k) \prod_{j=2}^{\phi_k} t_2(v_k[j]|v_k[j-1], x_k) \tag{2.57}$$

where $t_1(f|e)$ is the usual word-to-word translation probability and $t_2(f|f', e)$ is a bigram translation probability that specifies the likelihood that French word f' is followed by French word f in a phrase generated by English word e. In a nutshell, the parameter set θ of the WtoP HMM consists of the transition parameters p_a, the phrase length parameters n, the jumping-to-NULL parameter p_0, the unigram word-to-word translation parameters t_1 and the bigram translation probabilities t_2: $\theta = \{p_a(i|i', I), n(\phi, e), p_0, t_1(f|e), t_2(f|f', e)\}$.

The relationship between the current approach with the word-to-word HMM is straightforward. Constraining the phrase length model $n(\phi; e)$ to permit only phrases of one word gives a word-to-word HMM model. The extensions introduced in this method are the phrase count, phrase length model and the bigram translation model. The hallucination process addresses the NULL problem explained in section 2.5.2. The phrase length model is an alternative to fertility, and it is motivated by *stay* probabilities introduced in Toutanova et al. (2002). It is, however, more powerful than a single *stay* parameter.

WtoP HMM and IBM model 4 allow the same alignments. Both model NULL and fertility. However, distortion is not incorporated into the WtoP HMM model. Although WtoP model is more complex than the basic word-to-word HMM, the Baum-Welch and Viterbi algorithms can still be used. Hence, training can be done by forward-backward algorithm and by parallelizing we can control memory usage, reduce the time needed for training and increase the bitext for training.

2.5.6 HMM with Features

In this section, we discuss a method that shows how features can be added to a standard HMM model to inject prior knowledge to the model (Berg-Kirkpatrick et al., 2010). Each component multinomial of the generative model is turned into a miniature logistic regression model. The EM algorithm is used to learn the parameters. The E-step is unchanged, but the M-step involves gradient based training. Berg-Kirkpatrick et al. (2010) explained a general method to add features to an unsupervised model. We explain the feature-enhanced model for word alignment problem. To be consistent in the notations, we will use a slightly different notation from that of Berg-Kirkpatrick et al. (2010) for the word alignment task.

Later in this section, we discuss discriminative feature-enhanced models developed recently. We briefly describe a discriminative variant of the Berg-Kirkpatrick et al. (2010) method and mention neural-network-based models for word alignment.

Feature-enhanced HMM

There are two types of distributions in the HMM model: emission and transition probabilities which are both multinomial probability distributions. Let the emission distribution $Pr(F_j = f_j | E_j = e_{a_j}, \theta)$ be parameterized by $\theta_{f,e}$ for each French word f given English word e. For word alignment, the transition probabilities are estimated based on the jump width as explained. However, the emission factors can be expressed as the output of a logistic regression model, replacing the explicit conditional probability table by a logistic function parameterized by weights $\mathbf{w}$ and features $\mathbf{h}$:

$$\theta_{f,e}(\mathbf{w}) = \frac{exp(\mathbf{w}.\mathbf{h}(f, e))}{\sum_{f'} exp(\mathbf{w}.\mathbf{h}(f', e))} \tag{2.58}$$

In the emission, the decision f is a French word and the context e is an English word. The denominator is a normalization term to make the factor a probability distribution over French word decisions. Here, $\mathbf{h}$ is a feature function that consists of all indicator features on tuples (f, e), and $\mathbf{w}$ is the vector of weights associated to each feature that we want to optimize such that the likelihood of the data is maximized. As commonly used in log-linear models, the objective function is modified to include a regularization term:

$$L(\mathbf{w}) = \log Pr(F = \boldsymbol{f}|\mathbf{w}) - \kappa||\mathbf{w}||_2^2 \tag{2.59}$$

Note that the feature-based logistic expression, equation 2.58, is equivalent to the flat multinomial when the feature function $\mathbf{h}(f, e)$ consists of all the indicator features on tuples (f, e), which we call BASIC features. The equivalence occurs when weights are set such that $w_{f,e} = log(\theta_{f,e})$. This is known as the natural parameterization of the multinomial distribution. Optimization EM algorithm is used to learn the parameters of the model. In the E-step, expected counts are calculated for each tuple of source word f and target word e:

$$\varepsilon_{f,e} \leftarrow \mathbb{E}_\theta\Big[\sum_{j \in J} \mathbb{1}(f_j = f, e_{a_j} = e|F = \boldsymbol{f})\Big] \tag{2.60}$$

In the M-step, to re-estimate the parameters, the expected counts are normalized as follows:

$$\theta_{f,e} \leftarrow \frac{\varepsilon_{f,e}}{\sum_{f'} \varepsilon_{f',e}} \tag{2.61}$$

Similarly, we can use EM to optimize $L(\mathbf{w})$ for the model with logistic parameterizations. The E-step precomputes θ parameters from $\mathbf{w}$ for each French word f and English word e using equation 2.58. Expected counts are computed using the forward-backward algorithm. The only difference from the standard model is that the conditional probabilities θ are now functions of $\mathbf{w}$. In the M-step, we use gradient based optimization. The goal is to find the $\mathbf{w}$ that maximizes the regularized log-likelihood:

$$\ell(\mathbf{w}, \boldsymbol{\varepsilon}) = \sum_{f,e} \varepsilon_{f,e} \log \theta_{f,e}(\mathbf{w}) - \kappa||\mathbf{w}||_2^2 \tag{2.62}$$

Optimization of the objective function can be done using LBFGS. This method relies on the computation of the log-likelihood and the gradient at each step. The log-likelihood has the form given in equation 2.62, while the gradient with respect to $\mathbf{w}$ takes the following form:

$$\nabla\ell(\mathbf{w}, \boldsymbol{\varepsilon}) = \sum_{f,e} \varepsilon_{f,e}.\Delta_{f,e}(\mathbf{w}) - 2\kappa.\mathbf{w} \tag{2.63}$$

$$\Delta_{f,e}(\mathbf{w}) = \mathbf{h}(f, e) - \sum_{f'} \theta_{f',e}(\mathbf{w})\mathbf{h}(f', e)$$

Algorithm 3 Feature-enhanced EM

repeat
  compute expected counts $\boldsymbol{\varepsilon}$
  repeat
    compute $\ell(\mathbf{w}, \boldsymbol{\varepsilon})$
    compute $\nabla\ell(\mathbf{w}, \boldsymbol{\varepsilon})$
    $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \boldsymbol{\varepsilon}), \nabla\ell(\mathbf{w}, \boldsymbol{\varepsilon}))$
  until convergence
until convergence

Figure 2.12: Feature-enhanced EM

The M-step is now an iterative procedure which is much more expensive than before, because for each value of $\mathbf{w}$ considered during the search, $\theta(\mathbf{w})$ should be recomputed for each f and e. This computation is in proportion to the size of parameter space. Figure 2.12 shows the feature-enhanced EM algorithm.

Word Alignment Features: The BASIC features provide a strong baseline performance. Table 2.1 shows linguistically-motivated features that are added to the model in Berg-Kirkpatrick et al. (2010) in order to inject prior knowledge. These features are designed for Chinese-English. However, language-specific features can be designed for other language pairs similarly. Note that all the features in this table are binary.

Table 2.1: Feature templates for experiments

Template	Description
BASIC	fires for each source word and target word pair in the alignment
EDIT-DISTANCE	fires when the edit distance between source word and the target word is a specific value
DICTIONARY	fires when the source word and the target word pair is in the dictionary
STEM	fires for each source word and stem of the target word pair (for porter stemmer)
PREFIX	fires for each pair of source word and prefix of length 4 of the target word
CHARACTER	fires for each target word and ith source (Chinese here) character pair

Discriminative methods

A discriminative log-linear variant of the Berg-Kirkpatrick et al. (2010) method is introduced by Dyer et al. (2011). This method can incorporate arbitrary overlapping features, and it is used to infer word alignment. A log-linear model with parameter $\mathbf{w}$ and feature function $\mathbf{h}$ is used to model $p(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}, J)$ directly. The feature function used in this model includes word association features, positional features, lexical features and HMM-like path features. As for

the inference, Dyer et al. (2011), design their features to keep the width of tree decomposition low to allow exact inference with dynamic programming. To learn the parameters, we select $\mathbf{w}$ that minimizes the ℓ_1 regularized conditional log-likelihood. Dyer et al. (2011) use an online method that approximates ℓ_1 regularization and only depends on the gradient of the unregularized objective (Tsuruoka et al., 2009).

Blunsom and Cohn (2006) use a Conditional Random Field (CRF), a discriminative model, on a small supervised setting. The model directly encodes $p(\boldsymbol{a}|\boldsymbol{f}, \boldsymbol{e})$ with a CRF. With a first order Markov assumption, exact inference and efficient learning are possible using forward-backward and Viterbi algorithms.

Yang et al. (2013) propose a model that integrates a multi-layer neural network into an HMM-like framework. They adapt and extend the context dependent deep neural network HMM (CD-DNN-HMM) (Dahl et al., 2011) model to the HMM-based word alignment model. In their method, context dependent lexical translation score is computed by neural network, and distortion is modelled by a simple jump distance scheme. The model is discriminatively trained on bilingual corpus, in a supervised setting, while huge monolingual data is used to train word embeddings.

Tamura et al. (2014) propose a word alignment model based on a recurrent neural network (RNN) to extend the feed-forward neural network (FNN) model of Yang et al. (2013). This RNN-based model captures long alignment history through recurrent architectures. Compared to the CD-DNN-HMM approach which can only explore the context in a window, the RNN predicts the j-th alignment a_j by conditioning on all the preceding alignments a_1^{j-1}. Moreover, noise-contrastive estimation (NCE) is applied for unsupervised training of the model.

2.6 Neural Machine Translation

2.6.1 Encoder-Decoder

Neural Machine Translation models the conditional probability $p(y|x)$ of translating a source sentence $x = (x_1, x_2, \ldots, x_J)$ into a target sentence $y = (y_1, y_2, \ldots, y_I)$. Using the encoder-decoder framework (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014), the encoder reads the input sentence into a context vector c that represents the sentence meaning. The encoder is a recurrent neural network (RNN) that reads each word of the input sentence x sequentially. At each time step t, RNN updates its hidden state h_t as follows:

$$h_t = f(x_t, h_{t-1}) \tag{2.64}$$

where $f(.)$ is a non-linear activation function that can be as simple as an element-wise logistic sigmoid or as complex as a Long Short-Term Memory (LSTM) unit (Hochreiter and

Schmidhuber, 1997). Sutskever et al. (2014) used LSTM as f and the last hidden state of the LSTM h_T as c.

The decoder is another RNN that is trained to predict the next word y_t given the context vector and all the previously predicted words $y_1, y_2, \ldots, y_{t-1}$. The decoder decomposes the conditional probability as follows:

$$p(y|x) = \prod_{t=1}^{I} p(y_t|y_1, \ldots, y_{t-1}, c) \tag{2.65}$$

where the conditional probability of predicting the next word y_t can be computed as:

$$p(y_t|y_1, \ldots, y_{t-1}, c) = g(y_{t-1}, s_t, c) \tag{2.66}$$

where $g(.)$ is a non-linear activation function that produces valid probabilities (eg. with a softmax). s_t is the hidden state of the RNN that is also conditioned on y_{t-1} and context vector c:

$$s_t = f(s_{t-1}, y_{t-1}, c) \tag{2.67}$$

Figure 2.13 illustrates an encoder-decoder architecture for NMT. The model translates a source sentence "le croissant aux amandes" into a target one "the almond croissant". The NMT model consists of two RNN networks: the encoder RNN (on the left with the red color) consumes the input source words; the decoder RNN generates a translation, one word at a time.

Training

The key benefit of NMT is that all components of the model can be trained jointly in an end-to-end fashion. This is in contrast with SMT where most components needs to be trained separately. The training objective is formulated as :

$$J_t = \sum_{(x,y)\in D} -\log p(y|x) \tag{2.68}$$

where D is the parallel training data. After training the encoder-decoder, the model can be used to generate a translation for a given source sentence. It can also be used to score a pair of source and target sentence.

Attention-based Model

In the attention-based framework, given an input source sentence, $x = (x_1, x_2, \ldots, x_J)$, and all the previously predicted target words $\{y_1, y_2, \ldots, y_{i-1}\}$, the probability of generating

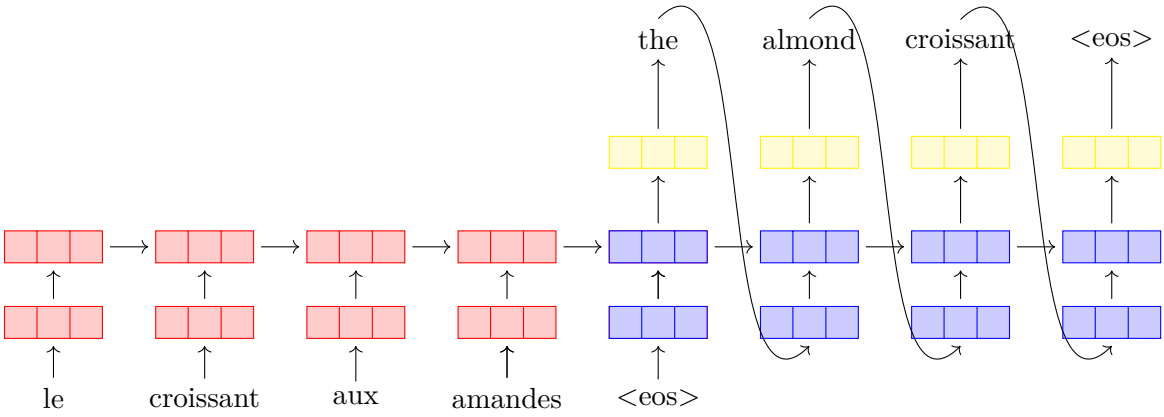


Figure 2.13: Encoder-decoder architecture for NMT. The model reads the input sentence "le croissant aux amandes" and produces "the almond croissant" as the output sentence. The model stops making predictions after predicting the end-of-sequence token <eos>. The bottom layer is an embedding layer, followed by a hidden layer on top, and another output layer (in yellow) on the decoder side.

the next target word y_i is

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i) \tag{2.69}$$

where $g(.)$ is a non-linear function and s_i is the hidden state of the decoder RNN at time step i which is computed as

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{2.70}$$

where $f(.)$ can be a non-linear function as simple as element-wise tanh or as sophisticated as a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). Bahdanau et al. (2015) use a gated recurrent unit (GRU). Unlike the Encoder-Decoder framework, a distinct context vector c_i is computed at each time i to generate target word y_i. The context vector c_i is computed as a weighted sum of source annotations h_j as follows

$$c_i = \sum_{j=1}^{J} \alpha_{ij} h_j \tag{2.71}$$

where $h_j = [\overrightarrow{h_j}^T; \overleftarrow{h_j}^T]^T$ is the annotation of x_j, which is computed using a bidirectional RNN (BiRNN) (Schuster and Paliwal, 1997) that has the information of the whole input

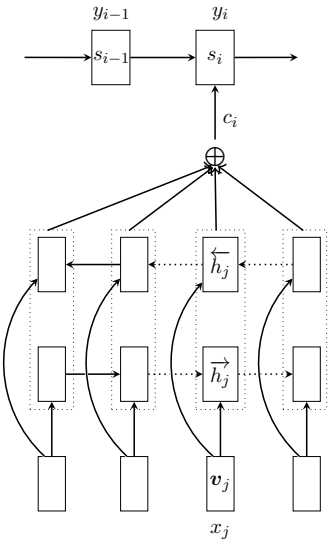


Figure 2.14: Attention model generates a target word y_t given a source sentence $(x_1, \ldots, x_J)$. In this figure, $\boldsymbol{v}_j$ is the embedding of source word x_j.

sentence with a strong focus on the words around x_j. The weight α_{ij} is computed as

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{J} \exp(e_{ik})} \tag{2.72}$$

where

$$e_{ij} = a(s_{i-1}, h_j) \tag{2.73}$$

is called an alignment model that scores how well the source words around x_j match the target words around y_i. All $\alpha_{ij} s (i = 1 \ldots I, j = 1 \ldots J)$ can be seen as an alignment-like matrix, where each row (for each target word) is a probability distribution over the source sentence $\mathbf{x}$. The alignment model is parameterized as a feed forward neural network which is jointly trained with all other components of the translation system. Since alignment model needs to be evaluated $J \times I$ times for each sentence pair of lengths J and I, a single layer multilayer perceptron is used to reduce computation:

$$a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j) \tag{2.74}$$

where $W_a \in \mathbb{R}^{n \times n}, U_a \in \mathbb{R}^{n \times 2n}$ and $v_a \in \mathbb{R}^n$ are the weight matrices.

The attention mechanism allows the decoder to select parts of the source sentence to pay attention to and frees the encoder from having to represent the entire source sequence into a single vector. The attention model is illustrated in Figure 2.14

2.6.2 Transformer

Transformer (Vaswani et al., 2017) still follows the overall architecture of the encoder-decoder, but instead of RNN layers, it uses self-attention and point-wise, fully connected layers for both the encoder and decoder. The transformer encoder is composed of six identical layers. Each encoder layer has two sub-layers: a multi-headed self attention mechanism and a fully connected feed-forward neural network. These layers are stacked to generate a final encoding of the source sentence. The structure of the transformer decoder is similar to the encoder except that it has an additional (third) sub-layer to attend to the source sentence. The novelty of the transformer architecture is in its self-attention layers in the encoder and the decoder. Self-attention is an attention mechanism that relates different positions of a single sequence to compute a representation of the sequence.

2.6.3 Evaluation

BLEU

The most common evaluation metric for machine translation is the BLEU score (Papineni et al., 2002). It is based on the n-gram matches between the candidate translation and the reference translation. Given the n-gram matches up to order N, we compute n-gram precision, p_n, with weights w_n that sums to one. BLEU is defined as:

$$\text{BLEU} = \text{BP}.\exp\Big(\sum_{n=1}^{N} w_n \log p_n\Big) \tag{2.75}$$

where N is usually up to 4, and brevity penalty (BP) reduces the score if the output is too short. BP is computed as:

$$BP = \begin{cases} 1 & \text{if } c > r \\ \mathrm{e}^{1-\frac{r}{c}} & \text{if } c \leq r \end{cases} \tag{2.76}$$

where c and r are the lengths of the candidate translation and the reference, respectively. BLEU score has a high correlation with human judgements, and ranges from 0 to 1 where 1 is achieved when output is identical to the reference.

TER

Translation Edit Rate (TER) (Snover et al., 2006) measures the minimum number of edits needed to change an output translation to exactly match one of the references, normalized by the average length of the references. Possible edits are single word insertion, deletion and substitution as well as phrasal shift (shift of word sequences). All edits have equal costs. This edit-distance based measure penalizes phrasal shifts less than BLEU, and it is intended to correlate well with human judgements while being less sensitive to the number

of references. TER score also ranges from 0 to 1. However, unlike BLEU, the lower the score the better as it indicates that less edits are required.

2.7 Word Alignment Evaluation

2.7.1 Alignment Error Rate

The quality of the alignment methods is evaluated by the performance on a test set for which a gold standard has been established by human annotators. The annotators are asked to specify alignments of two kinds: an S (sure) alignment, for alignments that are unambiguous and a P (possible) alignment, for ambiguous alignments. Thus, the reference alignment obtained may contain many-to-one and one-to-many relationships. The quality of an alignment $A = \{(j, a_j)|a_j > 0\}$ is evaluated using redefined precision and recall measures:

$$precision = \frac{|A \cap P|}{|A|}, recall = \frac{|A \cap S|}{|S|} \tag{2.77}$$

and the alignment error rate (AER), which is defined as

$$AER = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} \tag{2.78}$$

2.7.2 F-measure

Fraser and Marcu (2007b) argue that AER can be a misleading metric when we make a distinction between the sure and possible alignments ($S \subset P$). This is because AER does not penalize unbalanced precision and recall as F-measure. Therefore, it is possible to obtain good AER by guessing fewer alignment links making it a less useful metric. Fraser and Marcu (2007b) suggest an F-measure with sure and possible distinction:

$$\text{F-measure} = \frac{1}{\frac{\alpha}{Precision(A,P)} + \frac{1-\alpha}{Precision(A,S)}} \tag{2.79}$$

where α sets the trade-off between precision and recall. They show that F-measure with an appropriate setting of α is useful during the development process of new alignment models, and hence has a better ability to capture alignment quality which can also lead to a better translation performance.

2.8 Summary

In this chapter, we covered all the necessary background knowledge to understand this thesis. We started with a background about word alignment, and introduced the basic concepts such as the noisy-channel model. We looked at two statistical alignment models

(IBM Models 1 and 2) and explained training and decoding of these models. We have particularly studied the HMM-based word alignment model as it is going to be the basis of the models that will be presented in this thesis. We showed how to train and decode the HMM model using the Baum-Welch and Viterbi algorithms. A detailed explanation of the forward-backward algorithm was then given for the word alignment problem. We have categorized the extensions to the basic HMM-based model into six groups of models based on the enhancement they offer: refined alignment (transition) models, NULL-enhanced models, fertility-enhanced models, part-of-speech-enhanced model, word-to-phrase model and feature-enhanced model. We have reviewed neural machine translation and attention-based NMT and closed this chapter with a discussion of evaluation measures for translation and word alignment.

Chapter 3

Unsupervised Neural Hidden Markov Models

In this chapter, we present a generative neural approach to HMMs (Tran et al., 2016). This approach can be applied to latent variable models with tractable inference that can be trained with EM. We will demonstrate this approach for the part-of-speech tag induction task (Section 3.2). In Section 3.4, we discuss the general training procedure which is a combination of forward-backward algorithm and backpropagation. We show how this framework allows easy inclusion of morphological information (Section 3.5) and integration of additional context (Section 3.6).

3.1 Framework

In a graphical model, with a set of observed variables $\boldsymbol{x}$ and latent variables $\boldsymbol{z}$, potential functions $\psi(\boldsymbol{x}, \boldsymbol{z})$ over these sets of variables are defined based on the hand-crafted features. Independence assumptions are made to simplify the model structure, and make the inference tractable. Tran et al. (2016) propose to produce the potentials using neural networks. By using neural networks, task-specific abstract representation of the data can be extracted. Furthermore, we can use LSTM-based RNNs to model unbounded contexts with much less parameters compared to one-hot feature encodings. The potentials can be reparameterized with neural networks as follows:

$$\psi(\boldsymbol{z}, \boldsymbol{x}) = f_{\text{NN}}(\boldsymbol{z}, \boldsymbol{x}|\theta) \tag{3.1}$$

In a model with a set of observed variables $\boldsymbol{x} = \{x_1, \ldots, x_n\}$, and latent variables $\boldsymbol{z} = \{z_1, \ldots, z_n\}$, we want to find θ that maximizes $p(\boldsymbol{x}|\theta)$. Using the generalized EM, the gradient is defined in terms of the gradient of the joint probability scaled by the posteriors:

$$J(\theta) = \sum_{\boldsymbol{z}} p(\boldsymbol{z}|\boldsymbol{x}) \frac{\partial \ln p(\boldsymbol{x}, \boldsymbol{z}|\theta)}{\partial \theta} \tag{3.2}$$

sentence	But	the	issue	is	stickier	than	it	seems	.
PTB	CC	DT	NN	VBZ	JJR	IN	RPR	VBZ	.

Figure 3.1: An example of a part-of-speech tagged text.

To compute the posterior probability $p(\boldsymbol{z}|\boldsymbol{x})$ in Equation 3.2, the message passing algorithm can be used. It is noted that if we can easily compute the derivative $\frac{\partial \ln p(\boldsymbol{x},\boldsymbol{z}|\theta)}{\partial \theta}$, Direct Marginal Likelihood optimization (Salakhutdinov et al., 2003) can be performed. In the next section, we present how this framework can be applied to HMMs for the part-of-speech tag induction task.

3.2 Part-of-speech Induction Task

From the unsupervised perspective, part-of-speech tagging can be viewed as a clustering problem where words are assigned to different clusters that are called the POS classes or categories of the words. For example, in English, the Penn Treebank (Marcus et al., 1994) consists of 36 POS categories and 12 other tags (for punctuation and currency symbols). Figure 3.1 shows an example of a sentence, with its POS tags, extracted from the Penn Treebank data.

3.2.1 Hidden Markov Model

The Hidden Markov Model (HMM) is a standard model for the POS induction task. An HMM-based POS tagger generates a sequence of words in order. In this model, it is assumed that an observed word x_i is generated by the latent POS tag z_i, and the latent POS tag z_i is generated independently, conditioned on the previous latent POS tag z_{i-1}. This model has two types of distributions, emission and transition. The graphical representation of an HMM is shown in Figure 3.2 where the shaded circles are the observations and the ones with the white background are the latent variables. The joint probability of a distribution over both latent and observed variables is given by

$$p(\boldsymbol{x}, \boldsymbol{z}) = \prod_{t=2}^{n} p(z_t|z_{t-1}) \prod_{t=1}^{n} p(x_t|z_t) \tag{3.3}$$

where $p(x_t|z_t)$ is the emission probability and $p(z_t|z_{t-1})$ is the transition probability.

To estimate the parameter of the HMM, the Baum-Welch (Baum, 1972) algorithm is used. In the next section, we present the neural HMM for the part-of-speech tag induction task (Tran et al., 2016).

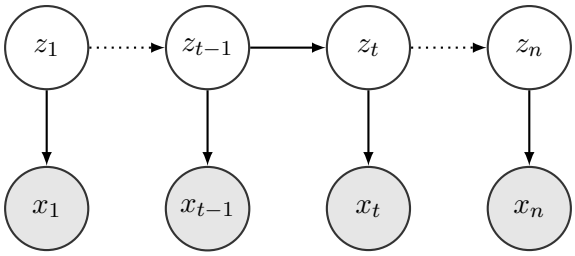


Figure 3.2: Graphical representation of a Hidden Markov Model. At time step t, latent variable (z_t) depends on the previous latent variable (z_{t-1}), and emits an observed word (x_t).

3.3 Neural HMM

We now describe the neural HMM for unsupervised POS tag induction (Tran et al., 2016). A standard POS HMM learns the emission and transition probabilities. In a neural HMM variant, emission and transition probabilities are modeled by neural networks. Enhancing each of these distributions is seamless. To achieve this, conditioning variables can be introduced as inputs to the network. We now discuss the basic feed-forward emission and transition models.

3.3.1 Emission Architecture

For an input tag z_k, the emission model gives a probability distribution over all possible words w_i. This can be implemented by a simple feed-forward neural network with an embedding matrix of $K \times D$ following by a non-linear activation (ReLU). We then apply a hidden layer of size D followed by a ReLU. Finally, an output linear layer of size V will be applied. The softmax function will be applied to provide $p(w_i|z_k)$ probabilities:

$$p(w_i|z_k) = \frac{\exp(\boldsymbol{v}_k^T \boldsymbol{w}_i)}{\sum_{j=1}^{V} \exp(\boldsymbol{v}_k^T \boldsymbol{w}_j)} \tag{3.4}$$

where $\boldsymbol{v}_k \in \mathbb{R}^D$ is the vector embedding of tag z_k, and $\boldsymbol{w}_i$ is the weight of unit i at the output layer of the network. V is the vocabulary size and K is the number of possible tags (clusters).

3.3.2 Transition Architecture

The transition model relies on a Multi-Layer Perceptron (MLP) to compute the multinomial distribution. Given a hidden layer vector $\boldsymbol{q} \in \mathbb{R}^D$ and a weight matrix $\boldsymbol{W} \in \mathbb{R}^{K^2 \times D}$, we compute the un-normalized transition matrix as follows:

$$\boldsymbol{T} = \boldsymbol{W}\boldsymbol{q} \tag{3.5}$$

The transition matrix $\boldsymbol{T}$ is reshaped to a $K \times K$ matrix. A softmax layer per row is then applied to produce transition probabilities.

3.4 Training Neural HMM

In this framework, the EM algorithm still applies where the E-step is similar to a standard HMM, but the M-step involves a gradient-based training. Whereas traditionally the expected counts are normalized during the M-step to re-estimate the parameters, in a neural HMM, they are used to re-scale the gradients. Using the HMM factorization (Equation 3.3) in the gradient (Equation 3.2), gives the following gradient update rule for the baseline neural HMM:

$$\begin{aligned} J(\theta) &= \sum_{\boldsymbol{z}} p(\boldsymbol{z}|\boldsymbol{x}) \frac{\partial \ln p(\boldsymbol{x}, \boldsymbol{z}|\theta)}{\partial \theta} \\ &= \sum_{t} \sum_{z_t} p(z_t|\boldsymbol{x}) \frac{\partial \ln p(x_t|z_t, \theta)}{\partial \theta} + \\ &\quad \sum_{t} \sum_{z_t} \sum_{z_{t-1}} p(z_t, z_{t-1}|\boldsymbol{x}) \frac{\partial \ln p(z_t|z_{t-1}, \theta)}{\partial \theta} \end{aligned}$$

where $p(z_t|\boldsymbol{x})$ and $p(z_t, z_{t-1}|\boldsymbol{x})$ are the posterior probabilities that can be computed using the Baum-Welch algorithm. The gradient terms come from the backpropagation algorithm. Note that because of the HMM factorization, it is easy to compute the joint probability $p(\boldsymbol{x}, \boldsymbol{z}|\theta)$; hence, Direct Marginal Likelihood (DML) (Salakhutdinov et al., 2003) can be used to optimize the parameters of the model. Tran et al. (2016) discuss that compared to EM, DML is faster to converge and yields to a slightly better performance.

During training, for a given batch, we compute the posterior probabilities using the neural-network based emission and transition models, perform backpropagation, and update the parameters of the networks. This procedure is repeated for a fixed number of epochs. The next two section demonstrate the extended versions of the emission and transition model.

3.5 Character-based Emission Model

Character-based representations is used to replace word embeddings to accommodate arbitrary vocabularies. Words are replaced with embedding vectors derived from a Convolutional Neural Network (CNN) (Kim et al., 2016; Jozefowicz et al., 2016). A convolutional kernel with widths from 1 to 7 is used which covers up to 7 character n-grams. The model learns the lexical representations based on prefix, suffix and stem information of a word.

3.6 Contextual Transition Model

Using neural networks allows for seamless integration of additional context. Tran et al. (2016) augment their transition model with with all the preceding words in the sentence $p(z_t|z_{t-1}, w_0, \ldots, w_{t-1})$. Incorporating this amount of context into a traditional HMM is intractable due to the exponentially large number of parameters. An LSTM is used to encode the word context $(w_0, \ldots, w_{t-1})$ into a vector that can be fed into the network when producing a transition matrix. Therefore, the transition probability becomes

$$p(z_t|z_{t-1}, w_0, \ldots, w_{t-1}) = p(z_t|z_{t-1}, h_{t-1}) \tag{3.6}$$

where h_{t-1} is the LSTM hidden state at time step $t-1$. In terms of the model architecture, the hidden layer vector $\boldsymbol{q}$ (Equation 3.5) will be replaced by h_{t-1}.

3.7 Evaluation

After training, the Viterbi algorithm is used to find the best latent POS tag sequence for every sentence. The Viterbi algorithm for POS tagging is very similar to the Viterbi algorithm for word alignment given in Figure 2.6. The evaluation can be done using the following metrics:

Many-to-One (M-1) Many-to-one measure maps each cluster to the most common gold POS tags for the words in that cluster. The tagging accuracy is then computed after the mapping is done. More than one cluster can be mapped to the same gold POS tag. This metric has been widely used in the literature. However, it tends to give higher scores as the number of clusters increases, making comparisons difficult when the number of clusters can change.

One-to-One (1-1) In the One-to-one measure, at most one cluster can be mapped to a given tag. Generally, as the number of clusters increases, fewer clusters can be mapped to their most common tag leading to a decrease in the score. Note that when the number of clusters is larger than the number of tags, some clusters remain unassigned.

V-Measure (VM) V-Measure (Rosenberg and Hirschberg, 2007) is an entropy-based measure which is analogous to F-measure that trades off the conditional entropy between the clusters and the gold tags. Christodoulopoulos et al. (2010) found VM to be the most stable and informative metric across different numbers of found and true clusters.

3.8 Summary

In this chapter, we presented a general unsupervised neural HMM and discussed this framework for part-of-speech tag induction task. The proposed neural HMM consists of neural

network-based emission and transition probabilities. We gave an overview of different network architectures used for the POS induction task. In Chapter 5, we will show how we develop a new model based on this framework and present an unsupervised neural HMM for the word alignment task.

Chapter 4

Joint Prediction of Word Alignment with Alignment Types

Current word alignment models do not distinguish between different types of alignment links. In this chapter, we provide a new probabilistic model for word alignment where word alignments are associated with linguistically motivated alignment types. We propose a novel task of joint prediction of word alignment and alignment types and propose novel semi-supervised learning algorithms for this task. We also solve a sub-task of predicting the alignment type given an aligned word pair. In our experimental results, the generative models we introduce to model alignment types significantly outperform the models without alignment types.

4.1 Introduction

Word alignment is an essential component in a statistical machine translation (SMT) system. Soft alignments, or attention, are also an important component in neural machine translation (NMT) systems. The classic generative model approach to word alignment is based on IBM models 1-5 (Brown et al., 1993) and the HMM model (Vogel et al., 1996; Och and Ney, 2000a). These traditional models use unsupervised algorithms to learn alignments, relying on a large amount of parallel training data without hand annotated alignments. Supervised algorithms for word alignment have become more widespread with the availability of manually annotated word-aligned data and have shown promising results (Taskar et al., 2005; Blunsom and Cohn, 2006; Moore et al., 2006; Liang et al., 2006). Manually word-aligned data are valuable resources for SMT research, but they are costly to create and are only available for a handful of language pairs. Semi-supervised methods for word alignment combine hand-annotated word alignment data with parallel data without explicit word alignments. Even small amounts of hand-annotated word alignment data has been shown to improve the alignment and translation quality (Callison-Burch et al., 2004). In this chap-

ter, we provide a novel semi-supervised word alignment model that adds alignment type information to word alignments.

Unsupervised or semi-supervised probabilistic word alignment models do not play a central role in neural machine translation (NMT) (Bahdanau et al., 2015; Sutskever et al., 2014; Luong et al., 2015; Chung et al., 2016). However, attention models, which are crucial for high-quality NMT, have been augmented with ideas from statistical word alignment (Luong et al., 2015; Cohn et al., 2016). Other than machine translation, word alignments are also important in the best performing models for NLP tasks. They play a central role in learning paraphrases in a source language by doing round-trips from source to target and back using word alignments (Ganitkevitch et al., 2013). Alignments have also been used for learning multi-lingual word embeddings (Faruqui and Dyer, 2014a; Lu et al., 2015) and in the projection of syntactic and semantic annotations from one language to another (Hwa et al., 2005; McDonald et al., 2011). Therefore, there is still a prominent role for word alignment in NLP; research into improvements in word alignment is a worthy goal.

Adding additional information such as part-of-speech tags and syntactic parse information has yielded some improvements in word alignment quality. Toutanova et al. (2002) incorporated the part-of-speech (POS) tags of the words in the sentence pair as a constraint on HMM-based word alignment. Additional constraints have also been injected into generative and discriminative models by designing linguistically-motivated features (Ittycheriah and Roukos, 2005; Blunsom and Cohn, 2006; Deng and Gao, 2007; Berg-Kirkpatrick et al., 2010; Dyer et al., 2011). These models provide evidence that additional constraints can help in modelling word alignments in a log-linear model where word based features can be augmented with morphological, syntactic or semantic features. For example, such a model might learn that function words in one language tend to be aligned to function words in the other language.

In this chapter, we propose a novel task which is the joint prediction of word alignment and alignment types for a given sentence pair in a parallel corpus. We present how to enhance the alignment model with alignment types. The primary contribution of this chapter is to demonstrate the success of the proposed joint model (alignment-type-enhanced model) to improve word alignment and translation quality. We apply our method on Chinese-English, because the annotated alignment type training data is provided in this language pair. However, the proposed method is potentially language-independent and can be applied to any language-pair as long as alignment type annotated data is created. The alignment types themselves may be language dependent and may vary in different language pairs.

4.2 The Data Set

The Linguistic Data Consortium (LDC) developed a linguistically-enriched word alignment data set: the GALE Chinese-English Word Alignment and Tagging Corpus. This human

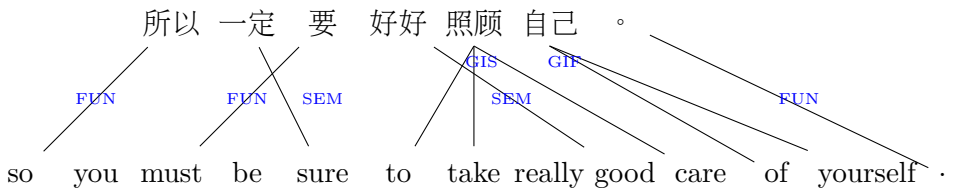


Figure 4.1: An alignment between a Chinese sentence and its translation in English which is enriched with alignment types . SEM (semantic), FUN (function), GIF (grammatically inferred function) and GIS (grammatically inferred semantic) are tags of the links.

annotated data set adds alignment type information to word alignments. The goal was to sub-categorize different types of alignment and draw a distinction between different types of alignment. For instance, it makes a distinction between aligned function words in both languages versus aligned content words. The goal was to improve word alignment and translation quality. Figure 4.1 shows an example of an enriched word alignment with alignment types extracted from the LDC data. Each link tag in the figure demonstrates the alignment type between its constituents.

The GALE Chinese-English Word Alignment and Tagging corpus contains 22,313 manually word aligned sentence pairs from which we extracted 20,357 sentences for training and we kept the rest as a test set. Table 4.1 shows the type and number of each alignment type in our training data.

ID	Alignment Type	Count
1	SEM	159,277
2	GIS	81,235
3	FUN	97,727
4	GIF	12,314
5	PDE	1,421
6	COI	3,256
7	CDE	1,608
8	TIN	1,116
9	MDE	4,615
10	NTR	34,090
11	MTA	84

Table 4.1: Number of each alignment type in the annotated training data

We briefly explain the existing alignment types in the GALE Chinese-English Word Alignment and Tagging Corpus. The SEM tag represents a semantic link between content words/phrases of source and translation, indicating a direct equivalence. Content words are typically nouns, verbs, adjectives and adverbs. FUN refers to a Function link which indicates that a word on either side of the link is a function word. Grammatically Inferred Function

(GIF) link is a type of link in which by stripping off extra words, we get a pure function link. In Grammatically Inferred Semantic (GIS) links, stripping off extra words results in pure semantic links. Alignment types PDE (DE-possesive), CDE (DE-Clause) and MDE (DE-modifier) are designed to handle the different features of the Chinese word 的(DE). In Contextually Inferred (COI) links, the extra words attached to one side of the link are required. Without these words, the grammatical structure might still be acceptable, but it is not semantically sensible. TIN (Translated Incorrectly) and NTR (Not translated) types are designed to handle the various errors that occur in the translation process, such as incorrect translation and no translation. MTA (Meta word) was designed to handle special characters that usually appear in the context of web pages.

Sub-categorizing different types of word alignments is likely to result in better word alignments. The alignment types provided by the LDC as annotations on each word alignment link have never been used (as far as we are aware) in order to improve word alignment. A subset of this data was used in (Wang et al., 2014) to refine word segmentation for machine translation but they ignore the alignment link types in their experiments.

4.3 Word Alignment

Given a source sentence $\mathbf{f} = \{f_1, f_2, \ldots, f_J\}$ and a target sentence $\mathbf{e} = \{e_1, e_2, \ldots, e_I\}$, the goal in SMT is to model the translation probability $Pr(\mathbf{f}|\mathbf{e})$. In alignment models, a hidden variable $\mathbf{a} = \{a_1, a_2, \ldots, a_J\}$ is introduced which describes a mapping between source and target words. Using this terminology, $a_j = i$ denotes that f_j is aligned to e_i. The translation probability can therefore be written as a marginal probability over all alignments:

$$Pr(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} Pr(\mathbf{f}, \mathbf{a}|\mathbf{e}) \tag{4.1}$$

In IBM Model 1, the alignment model is decomposed into the product of translation probabilities as follows:

$$Pr(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \frac{1}{(I+1)^J} \prod_{j=1}^{J} p(f_j|e_{a_j}) \tag{4.2}$$

In the Hidden Markov alignment model, we assume a first order dependence for the alignments a_j. The HMM-based model has the following form:

$$Pr(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \prod_{j=1}^{J} p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j}) \tag{4.3}$$

where $p(a_j|a_{j-1}, I)$ are the alignment probabilities (transition parameters) and $p(f_j|e_{a_j})$ are the translation probabilities (emission parameters). Vogel et al. (1996) make the alignment parameters $p(i|i', I)$ independent of the absolute word positions and assume that $p(i|i', I)$ depend only on the jump width $(i - i')$. Hence, the alignment probabilities are estimated

using a set of distortion parameters $c(i - i')$ as follows:

$$p(i|i', I) = \frac{c(i - i')}{\sum_{i''=1}^{I} c(i'' - i')} \tag{4.4}$$

where at each EM iteration $c(i - i')$ is the fractional count of transitions with jump width $i - i'$.

The HMM network is extended by I NULL words (Och and Ney, 2000a) with the following constraints on the transition probabilities ($i \leq I, i' \leq I$):

$$p(i + I|i', I) = p_0 \cdot \delta(i, i') \tag{4.5}$$

$$p(i + I|i' + I, I) = p_0 \cdot \delta(i, i') \tag{4.6}$$

$$p(i|i' + I, I) = p(i|i', I) \tag{4.7}$$

where δ is the Kronecker delta function. The parameter p_0 controls NULL insertion and is optimized on a held-out dataset.

4.4 Joint Model for IBM Model 1 and HMM

We consider two classic generative models, IBM Model 1 (Brown et al., 1993) and the HMM alignment model (Vogel et al., 1996) as our baselines and present how we can enhance these models with alignment types. In this section, we introduce two models (a generative and a discriminative model) for each baseline to jointly find the word alignments and the corresponding alignment types for a sentence pair.

4.4.1 Generative Models

IBM Model 1 with Alignment Types

We augment IBM Model 1 (Equation 4.2) with alignment type information. In addition to alignment function $a : j \rightarrow i$, our model has a tagging function: $h : j \rightarrow k$ which specifies the mapping for each alignment link (f_j, e_i) to an alignment type k. Alignment type k can be any tag in the set of all possible linguistic tags. The new generative model with alignment type, has the following form:

$$Pr(\mathbf{f}, \mathbf{a}, \mathbf{h}|\mathbf{e}) = \frac{1}{(I + 1)^J N^J} \prod_{j=1}^{J} p(f_j, h_j|e_{a_j}) \tag{4.8}$$

where N is the number of possible linguistic alignment types. Using the chain rule, we have the following enhanced IBM Model 1 which includes alignment-types:

$$Pr(\mathbf{f}, \mathbf{a}, \mathbf{h}|\mathbf{e}) = \frac{1}{(I+1)^J N^J} \prod_{j=1}^{J} p(f_j|e_{a_j}) \cdot p(h_j|f_j, e_{a_j})$$

In order to normalize the probability, we modify the fraction in Equation 4.2 by adding term N^J as there are N different alignment types for each alignment link from each source word.

EM algorithm

Similar to IBM Model 1, we use EM algorithm to estimate the parameters of our model. In the expectation step, we need to compute the posterior probability $Pr(\mathbf{a}, \mathbf{h}|\mathbf{f}, \mathbf{e})$ which is the probability of an alignment with its types given the sentence pair. Applying the chain rule gives:

$$Pr(\mathbf{a}, \mathbf{h}|\mathbf{f}, \mathbf{e}) = Pr(\mathbf{a}|\mathbf{f}, \mathbf{e}) \times Pr(\mathbf{h}|\mathbf{a}, \mathbf{f}, \mathbf{e}) \tag{4.9}$$

where $Pr(\mathbf{a}|\mathbf{f}, \mathbf{e})$ is the posterior probability of IBM Model 1:

$$Pr(\mathbf{a}|\mathbf{f}, \mathbf{e}) = \prod_{j=1}^{J} \frac{p(f_j|e_{a_j})}{\sum_{i=0}^{I} p(f_j|e_i)} \tag{4.10}$$

$Pr(\mathbf{h}|\mathbf{a}, \mathbf{f}, \mathbf{e})$ can be written as a product of alignment type parameters over the individual source and target words and their corresponding alignment type:

$$Pr(\mathbf{h}|\mathbf{a}, \mathbf{f}, \mathbf{e}) = \prod_{j=1}^{J} p(h_j|f_j, e_{a_j}) \tag{4.11}$$

Substituting Equations 4.10 and 4.11 in Equation 4.9 and simplifying it results in:

$$Pr(\mathbf{a}, \mathbf{h}|\mathbf{f}, \mathbf{e}) = \prod_{j=1}^{J} \frac{p(f_j|e_{a_j}) \times p(h_j|f_j, e_{a_j})}{\sum_{i=0}^{I} p(f_j|e_i)} \tag{4.12}$$

We collect the expected counts over all possible alignments and their alignment types, weighted by their probability. Suppose $c(f, h|e; \mathbf{f}, \mathbf{e})$ is the expected count for a word e generating a word f with an alignment type h in a sentence pair $(\mathbf{f}, \mathbf{e})$:

$$c(f, h|e; \mathbf{f}, \mathbf{e}) = \sum_{\mathbf{a},\mathbf{h}} [Pr(\mathbf{a}, \mathbf{h}|\mathbf{f}, \mathbf{e}) \sum_{j=1}^{J} \delta(f, f_j)\delta(e, e_{a_j})\delta(h, h_j)] \tag{4.13}$$

Plugging $Pr(\mathbf{a}, \mathbf{h}|\mathbf{f}, \mathbf{e})$ (Equation 4.12) in Equation 4.13, yields

$$c(f, h|e; \mathbf{f}, \mathbf{e}) = \frac{p(f|e) \times p(h|f, e)}{\sum_{i=0}^{I} p(f|e_i)} \sum_{j=1}^{J} \delta(f, f_j) \sum_{i=0}^{I} \delta(e, e_i)\delta(h, h_j) \tag{4.14}$$

The alignment type parameters are then estimated by Equation 4.15.

$$p(h|f, e) = \frac{\sum_{(\mathbf{f},\mathbf{e})} c(f, h|e; \mathbf{f}, \mathbf{e})}{\sum_{h} \sum_{(\mathbf{f},\mathbf{e})} c(f, h|e; \mathbf{f}, \mathbf{e})} \tag{4.15}$$

Translation probabilities are estimated similar to IBM Model 1. This model is called IBM1+Type+Gen in the experiments section.

After training, we can jointly predict the best alignment and the best alignment types for each sentence pair:

$$\hat{\mathbf{a}}, \hat{\mathbf{h}} = \arg\max_{\mathbf{a},\mathbf{h}} \prod_{j=1}^{J} p(f_j|e_{a_j})p(h_j|f_j, e_{a_j}) \tag{4.16}$$

In this decoding method, for a given sentence pair, for each source word f_j, we have to go through all the target words e_{a_j} in the target sentence and all the possible alignment types and find the pair of target position and alignment type that maximizes $p(f_j|e_{a_j})p(h_j|f_j, e_{a_j})$.

HMM with Alignment Types

Our HMM with alignment types model has the factor $p(f_j, h_j|e_{a_j})$ in its formulation, which can be further decomposed to give:

$$Pr(\mathbf{f}, \mathbf{a}, \mathbf{h}|\mathbf{e}) = \prod_{j=1}^{J} p(a_j|a_{j-1}, I)p(f_j|e_{a_j})p(h_j|f_j, e_{a_j})$$

This model is called HMM+Type+Gen. We now explain how we can estimate the parameters of this model. A compact representation of this model is $\theta = \{p(i|i', I), p(f|e), p(h_j|f, e)\}$ where $p(i|i', I)$ are the transition probabilities, $p(f_j|e_i)$ are the emission probabilities and $p(h_j|f_j, e_i)$ are the alignment type probabilities.

Let $\gamma_i(j, h) = Pr(a_j = i, h_j = h|\mathbf{f}, \theta)$ be the posterior probabilities for the generative HMM (HMM+Type+Gen). Since $Pr(a_j = i, h_j = h|\mathbf{f}, \theta) = Pr(a_j = i|\mathbf{f}, \theta) \times Pr(h_j = h|a_j = i, \mathbf{f}, \theta)$, we have:

$$\gamma_i(j, h) = \gamma_i(j) \times p(h|f_j, e_i) \tag{4.17}$$

Equation 4.17 confirms that the posterior probability of the HMM+Type+Gen model is the HMM posterior multiplied by the alignment type probability factor $p(h|f_j, e_i)$ which is similar to the relationship between posterior probabilities in case of IBM Model 1 as shown in Equation 4.12.

Using this equation, we can compute the expected counts:

$$c(f, h|e; \mathbf{f}, \mathbf{e}) = \sum_{i,j} \gamma_i(j, h)\delta(f_j, f)\delta(e_i, e)\delta(h_j, h) \tag{4.18}$$

The expected counts are then normalized in the M-step to re-estimate the parameters. The transition and emission parameters are estimated as the standard HMM-based alignment model.

The EM algorithm for this model is similar to the Baum-Welch algorithm for the standard HMM-based word alignment model. The only change is that in the E-step, we need to collect the alignment type expected counts and in the M-step, alignment type parameters are re-estimated.

After training, Viterbi decoding is used to find the best word alignment and alignment types for new sentences. We define $V_i(j, h)$ to be the probability of the most probable alignment for $f_1 \dots f_j$ that f_j is aligned to e_i and the alignment type for this link is h. It can be computed recursively as follows:

$$V_i(j, h) = \max_{i',h'} \{V_{i'}(j-1, h')p(i|i', I)p(f_j|e_i)p(h|f_j, e_i)\} \tag{4.19}$$

4.4.2 Discriminative Models

Although we can use the generative models explained in Section 4.4.1 to estimate the alignment type probabilities $p(h|f, e)$, we can build a classifier to predict the alignment type given a pair of aligned words.

We have a set of 11 possible alignment types in the LDC data which are the possible classes in the classification problem. We use logistic regression to model the alignment type prediction problem. The rationale for using this model is that it can provide us with both the alignment type and the probability of being classified as this type.

Features

We used 22 different types of features in our logistic regression model as shown in Table 4.2. Lexical features are the heart of all lexical translation models; here, they are defined on pairs of Chinese and English words, shown by feature template (c_0, e_0) in Table 4.2.

Moreover, we include features taking the context into consideration. For example, feature (c_{-1}, c_0, e_0) uses the previous Chinese word apart from the pair of English and Chinese words. Part-of-speech (POS) tags are used to address the sparsity of the lexical features. For example, POS tags of the pair of Chinese and English words (c_{t_0}, e_{t_0}) are included. We also use the first five letters of the English word in a feature, to approximate the stem of an English word. An example used as a feature is $(c_0, [e_0]_5)$, where the pair of Chinese word and the prefix of English word is used as a feature.

word-based	(c_0, e_0), (c_{-1}, c_0, e_0), $(c_{-2}, c_{-1}, c_0, e_0)$, (c_0, c_1, c_2, e_0), (c_0, e_{-1}, e_0), (c_0, e_{-1}, e_0, e_1), (c_0, e_0, e_1)
part-of-speech tag-based	(c_0, e_{t_0}, e_0), $(c_{-1}, c_0, e_{-1}, e_{t_0})$, $(c_0, e_{t_{-1}}, e_{t_0}, e_0)$ $(c_0, e_{t_{-1}}, e_0, e_{t_1})$, $(c_0, e_{t_{-1}}, e_{t_0}, e_{t_1})$, (c_{t_0}, e_{t_0}), $(c_{t_0}, c_{t_1}, c_{t_2})$, $(c_{t_0}, c_{t_{-1}}, e_{t_0})$, $(c_{t_{-1}}, c_{t_0}, c_{t_{-2}}, e_{t_0})$, $(c_{t_0}, c_{t_1}, e_{t_0})$, $(c_{t_{-1}}, c_{t_0}, c_{t_1}, e_{t_0})$
substring-based	$(c_0, [e_0]_5)$, $(c_0, [e_{-1}]_5, [e_0]_5, [e_1]_5)$, $(c_{t_0}, e_{-1}, [e_0]_5, e_{t_0})$, $(c_0, e_{t_0}, e_{t_{-1}}, [e_0]_5)$

Table 4.2: Feature types used in our alignment type classifier.

EM for the Discriminative Models

We introduce discriminative variants of the generative models explained in Section 4.4.1. These discriminative models are referred to as IBM1+Type+Disc and HMM+Type+Disc. The main difference between these models and their generative counterparts is in the way they compute alignment type probabilities $p(h|f, e)$. Whereas the generative models estimate these probabilities using the EM algorithm, the discriminative models estimate these probabilities using the logistic regression classifier. For the discriminative models, we first train a logistic regression model on the LDC data (see Section 4.5). The model provides us with the alignment type probabilities which are used in the decoding stage. For IBM1+Type+Gen model, expected counts for alignment types are collected and alignment type parameters are updated in each iteration. In the EM algorithm for IBM1+Type+Disc, however, we do not need to collect the expected counts for alignment types since these parameter values are obtained from the pre-trained logistic regression classifier. However there is an important difference in the decoding step: Equation 4.16 is used to jointly find the best alignment and alignment types for each sentence pair. This joint decoding step makes this approach different from simply using a pipeline trained EM model followed by a discriminative classifier on the Viterbi output of the EM trained model. A comparison with the pipeline model is given in Section 4.5.2. Similarly, the EM training of HMM+Type+Disc is similar to the EM training of baseline HMM. For decoding a new sentence pair, Equation 4.19 is used.

4.5 Experiments

For the experiments, we have used two datasets. The first is the GALE Chinese-English Word Alignment and Tagging corpus which is released by LDC[1]. This dataset is annotated with gold alignment and alignment types (see Section 4.2 for more details). The second dataset is the Hong Kong parliament proceedings (HK Hansards) for which we do not have the gold alignment and alignment types.

[1]Catalog numbers: LDC2012T16, LDC2012T20, LDC2012T24, LDC2013T05, LDC2013T23 and LDC2014T25.

We used 1 million sentences of the HK Hansards in the experiments to augment the training data. In the following sections, we describe three experiments. First, we examine how effective the logistic regression classifier is for alignment type prediction. Second, we present our experiments for two tasks: word alignment and the joint prediction of word alignment and alignment types. Finally, we explain the machine translation experiment.[2]

4.5.1 Alignment Type Prediction Given Alignments

For the alignment type prediction task given an aligned word pair, we have examined three simple maximum likelihood classifiers, as well as the logistic regression classifier with the features shown in Table 4.2. We have trained all these classifiers on the parallel Chinese-English 20K LDC data which is annotated with gold alignment and alignment types. To obtain the word pairs, we have extracted the word pairs from the parallel sentences with the gold alignment. To get the part-of-speech tags, we annotated the 20K LDC data with the Stanford POS tagger (Toutanova et al., 2003). We ignored the gold alignment if the Chinese side of the gold alignment is not contiguous; i.e., it cannot form one Chinese word. This usually happens in the many-to-one and many-to-many alignments. There were only a small number of these discontiguous alignments (2% of all alignments) as mentioned in the LDC catalog entry for this data.

Maximum Likelihood Classifiers

We have examined three maximum likelihood (ML) classifiers. The first model is a word-based ML classifier that uses the maximum likelihood estimate of the alignment type parameters $p(h|f,e)$, computed from the training data, to predict the alignment type for a new given pair of aligned words in a sentence pair in the test data. If the aligned words were not seen in the training data, this model backs-off to SEM as it is the most probable alignment type.

The second model which is a tag-based ML classifier, uses the maximum likelihood estimate of $p(h|t_f,t_e)$ parameters of the model trained on the POS tagged data. t_f and t_e are the POS tags of the Chinese word f and the English word e, respectively. It backs-off to SEM for unseen pairs of POS tags. Finally, for a pair of word (f,e), the last classifier first uses the ML estimate of $p(h|f,e)$ parameter. For unseen pair of words, it backs-off to use the ML estimate of $p(h|t_f,t_e)$ and in case the pair of POS tags was not seen, it backs-off to SEM.

[2]All our codes for the baselines and the proposed models are available at `https://github.com/sfu-natlang/align-type-tacl2017-code`.

Model	accuracy
ML word-based	73.8
ML tag-based	72.3
ML word-tag	79.1
Logistic regression	**81.4**

Table 4.3: Accuracy of the alignment type classifiers given the alignment.

Logistic Regression Classifier

We evaluated the logistic regression classifier which makes use of the features shown in Table 4.2 and the combination of different sets of these features. We assessed the performance of our features using 10-fold cross-validation. The best average cross-validation accuracy of 81.5% was achieved by a classifier that combines all the 22 features, shown in Table 4.2. We have used this trained classifier for the discriminative models (IBM1+Type+Disc and HMM+Type+Disc) in the experiments reported in Section 4.5.2.

Results

Table 4.3 shows the accuracy of the classifiers on the 2K sentences used as held-out data. The logistic regression classifier achieved the best accuracy on the test data. Since the logistic regression classifier obtains 87.5% on training, and the cross-validation accuracy variance was small, we do not believe the classifier overfits on our training data.

4.5.2 Joint Word Alignment and Alignment Type Experiments

We measure the performance of our models using precision, recall, and F1-score. We also evaluated the performance of our models and the baseline models on two different tasks: (1) The traditional word alignment task and (2) The joint prediction of word alignment and alignment types task. The second task is harder as the model has to predict both word alignment and alignment types correctly. Moreover, as the baseline IBM Model 1 and the baseline HMM cannot predict the alignment types, we can only make a comparison between our generative and discriminative models for the second task.

We initialized the translation probabilities of Model 1 uniformly over the word pairs that occur together in the same sentence pair.

We built an HMM similar to the one proposed by Och and Ney (2003). This model is referred to as HMM in this chapter. HMM was initialized with uniform transition probabilities and Model 1 translation probabilities. Model 1 was trained for 5 iterations; it is followed by 5 iterations of HMM.

To handle unseen data when the model is applied to the test data, smoothing has been used. We smooth translation probability $p(f|e)$ by backing-off to a uniform probability $1/|V|$ where $|V|$ is the source vocabulary size.

For smoothing alignment type probabilities $p(h|f,e)$, we used the following linear interpolation:

$$p^*(h|f,e) = \lambda_1 p(h|f,e) + \lambda_2 p(h|t_f,t_e) + \lambda_3 p(h) \tag{4.20}$$

where $\lambda_1 \geq 0, \lambda_2 \geq 0$ and $\lambda_3 \geq 0$ are the smoothing parameters and $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

t_f and t_e are the POS tags of the Chinese word f and the English word e, respectively. To obtain $p(h|t_f,t_e)$, we labelled the parallel data with the Stanford POS tagger and then trained a model on these POS tags. $p(h)$ is the prior probability of alignment type h estimated over the gold training data using Table 4.1. Both $p(h|f,e)$ and $p(h|t_f,t_e)$ are smoothed with $p(h)$ using linear interpolation.

To learn the hyper-parameters, we split the 20K LDC training data into two sets: a train set of 18K sentences and a 2K validation set. To learn p_0 and NULL emission probability, we performed a two-dimensional grid search varying p_0 in the set $\{0.05, 0.1, 0.2, 0.3, 0.4\}$ and NULL emission probability in the set {1e-7, 5e-7, . . .,1e-2, 5e-2, 1e-1}. The tuned parameters that lead to the best result were achieved when $p_0 = 0.3$ and NULL emission probability was 5e-6. To tune the hyper-parameters λ_1, λ_2 and λ_3, we performed a two-dimensional grid search. The tuned parameters that lead to the best result was achieved when $\lambda_1 = 0.99$, and λ_3 =1e-15. Hence, $\lambda_2 = 1 - \lambda_1 - \lambda_3 = 9.99$e-11. We then used these learned parameters in the experiments.

Finally, for HMM-based models, we smooth transition parameters $p(i|i', I)$ by backing off to a uniform prior $1/I$.

Results on the LDC Alignment Type Data

Table 4.4 shows the models' performance for the word alignment task for all the baselines and the methods introduced in this thesis. In this table, MODEL+Type+Gen denotes the proposed generative variant of MODEL while MODEL+Type+Disc denotes the proposed discriminative variant of MODEL. We trained all the models on the 20K data and tested on the 2K sentences used as held-out data.

We can see that our generative models consistently outperform their corresponding baselines. The best performing model, HMM+Type+Gen, achieves up to 13.9% improvement in F1-score over the baseline HMM. To compare our models against GIZA++, we add the test data to the training, and use Moses (Koehn et al., 2007) with its default parameters to obtain word alignments. We report its performance on the test data. Unlike the other

Word Alignment (WA) Task: Train(20K)+Test(2K)			
Model	Prec.	Rec.	F1-score
IBM1	50.9	40.5	45.1
IBM1+Type+Disc	51.7	41.2	45.8
IBM1+Type+Gen	59.0	47.0	52.3
HMM	68.0	48.7	56.7
HMM+Type+Disc	66.2	50.5	57.3
HMM+Type+Gen	**72.9**	**58.0**	**64.6**
GIZA++	61.4	47.7	53.8

Table 4.4: Word alignment task results of the models trained on 20K LDC data (22K LDC data for GIZA++) and tested on 2K LDC test data.

WA+Type Task: Train(20K) + Test(2K)			
Model	Prec.	Rec.	F1-score
IBM1+Type+Disc	44.0	37.5	40.5
IBM1+Type+Gen	47.8	40.8	44.0
HMM+Type+Disc	55.3	45.2	49.8
HMM+Type+Gen	**59.2**	**50.5**	**54.5**
IBM1→Disc	42.9	36.6	39.5
HMM→Disc	57.2	43.8	49.6
GIZA++→Disc	52.2	43.5	47.5

Table 4.5: Results of the models trained on 20K LDC data (22K LDC data for GIZA++) and tested on 2K LDC test data for (1) joint prediction of word alignment and alignment types task, and (2) word alignment models followed by the discriminative classifier to predict alignment types.

models in Table 4.4 which are trained on 20K data, GIZA++ model is trained on 22K data.[3]

Table 4.5 shows the results obtained for the joint prediction of word alignment and alignment types task. As mentioned previously, the basic IBM Model 1 and HMM are incapable of predicting the alignment types and hence are not included in this table. However, it is interesting to compute word alignments using our baselines and then apply the logistic regression classifier on the alignments to get the corresponding alignment types. In Table 4.5, MODEL→Disc denotes this pipelined version of MODEL. The only difference between MODEL+Type+Disc and MODEL→Disc is in the decoding step. The former jointly pre-

[3]GIZA++ does not allow the user to run it as a classifier (a model that is trained on the training data and can be tested on new data). Initially, we performed incremental training with inc-giza-pp (Levenberg et al., 2010). Since the performance was very poor, we used GIZA++ in our experiments by appending the test data to the training data (even though our models did not see the test data) and reported the result of Viterbi output from the trained GIZA++ model on the combined data.

WA Task: Train(20K+1M) + Test(2K)			
Model	Prec.	Rec.	F1-score
IBM1	49.7	39.6	44.1
IBM1+Type+Disc	50.5	40.2	44.8
IBM1+Type+Gen	59.5	47.4	52.8
HMM	67.7	48.8	56.7
HMM+Type+Disc	66.1	50.7	57.4
HMM+Type+Gen	**73.1**	**58.2**	**64.8**
GIZA++	60.0	47.0	52.7

Table 4.6: Word alignment task results for the augmented model.

dicts word alignment and alignment types while the latter performs word alignment and then applies the classifier on the output of word alignment to obtain the alignment types. We also computed word alignments using GIZA++ as explained for the previous experiment and then ran our logistic regression classifier on the alignments to get the corresponding alignment types. This model is denoted as GIZA++→Disc in Table 4.5. The results in this table show that the generative model outperforms its discriminative counterpart. Similar to the previous experiment, HMM+Type+Gen model achieved the best result.

Results with Augmented Model

We conducted another experiment to see whether we can improve the current results by augmenting the training data. We trained on the 20K LDC data with gold alignment and alignment types, and 1 million HK Hansards which has no alignment or alignment type annotations and tested on the 2K sentences used as held-out data. Although HK Hansards data is not annotated, it can augment our vocabulary. We built a model with the 20K LDC data; we call it LDC model. We then trained a model using the 20K LDC data and the 1 million HK Hansards data; we refer to this as the augmented model. The alignment type parameters of the augmented model are initialized, based on the maximum likelihood estimate of the 20K LDC data. Table 4.6 shows the results of the augmented model for the word alignment task. Table 4.7 shows the results of this model for the joint prediction of word alignment and alignment types tasks.

Results with Augmented Model and Back-off Smoothing

Purely using the augmented model was not effective in estimating the translation probabilities $p(f|e)$, and hence did not contribute to any improvement compared to the previous experiment. This is due to the fact that HK Hansards data is from a different domain compared to our test LDC data. Since the 2K test data is from the LDC data, we applied a back-off smoothing technique: we estimated $p(f|e)$ from the LDC model if the word pair (f, e) was seen by the LDC model, and we used the augmented model to compute $p(f|e)$ otherwise.

WA+Type Task: Train(20K+1M) + Test(2K)			
Model	Prec.	Rec.	F1-score
IBM1+Type+Disc	42.9	36.6	39.5
IBM1+Type+Gen	48.0	40.9	44.2
HMM+Type+Disc	55.2	45.4	49.8
HMM+Type+Gen	**59.2**	**50.4**	**54.5**
IBM1→Disc	41.9	35.7	38.6
HMM→Disc	56.7	43.7	49.4
GIZA++→Disc	51.0	42.8	46.5

Table 4.7: Results using the augmented model for (1) joint prediction of word alignment and alignment types task, and (2) word alignment models followed by the discriminative classifier to predict alignment types.

WA Task: Train(20K+1M) + Test(2K)			
Model	Prec.	Rec.	F1-score
IBM1	52.7	42.0	46.7
IBM1+Type+Disc	53.5	42.6	47.4
IBM1+Type+Gen	60.3	48.0	53.5
HMM	69.4	50.0	58.1
HMM+Type+Disc	67.1	51.9	58.5
HMM+Type+Gen	**74.5**	**59.2**	**66.0**
GIZA++	60.0	47.0	52.7

Table 4.8: Word alignment task results, back-off using the augmented model.

Table 4.8 shows the results of the augmented model after the smoothing step is done for the word alignment task. Compared to the results in Table 4.4, all the models performed better, with the HMM+Type+Gen outperforming all the other methods.

The results for the joint prediction task are shown in Table 4.9. This confirms our success in improving the performance of all the methods, compared to the results in Table 4.5.

Statistical significance tests were performed using the approximate randomization test (Yeh, 2000) with 10,000 iterations. The generative models significantly outperform their baseline and discriminative counterparts (p-value < 0.0001).

4.5.3 Machine Translation Experiment

To see whether the improvement in F1-score by our generative model also improves the BLEU score, we aligned the 20K LDC data and 1 million sentences of the HK Hansards data using the augmented model and tested on 919 sentences of MTC part 4 (LDC2006T04). We trained models in each translation direction and then symmetrized the produced alignments using the grow-diag-final heuristic (Och and Ney, 2003). We used Moses (Koehn et al., 2007)

WA+Type Task: Train(20K+1M) + Test(2K)			
Model	Prec.	Rec.	F1-score
IBM1+Type+Disc	45.3	38.6	41.7
IBM1+Type+Gen	48.6	41.5	44.8
HMM+Type+Disc	55.9	46.2	50.6
HMM+Type+Gen	**60.3**	**51.3**	**55.4**
IBM1→Disc	44.3	37.8	40.8
HMM→Disc	58.2	44.9	50.7
GIZA++→Disc	51.0	42.8	46.5

Table 4.9: Results with back-off using the augmented model for (1) joint prediction of word alignment and alignment types task, and (2) word alignment models followed by the discriminative classifier to predict alignment types.

Model	BLEU	TER
GIZA++ HMM	23.4	70.4
GIZA++ (Moses)	23.2	69.1
HMM	23.5	68.3
HMM+Type+Gen	**24.4**	**67.8**

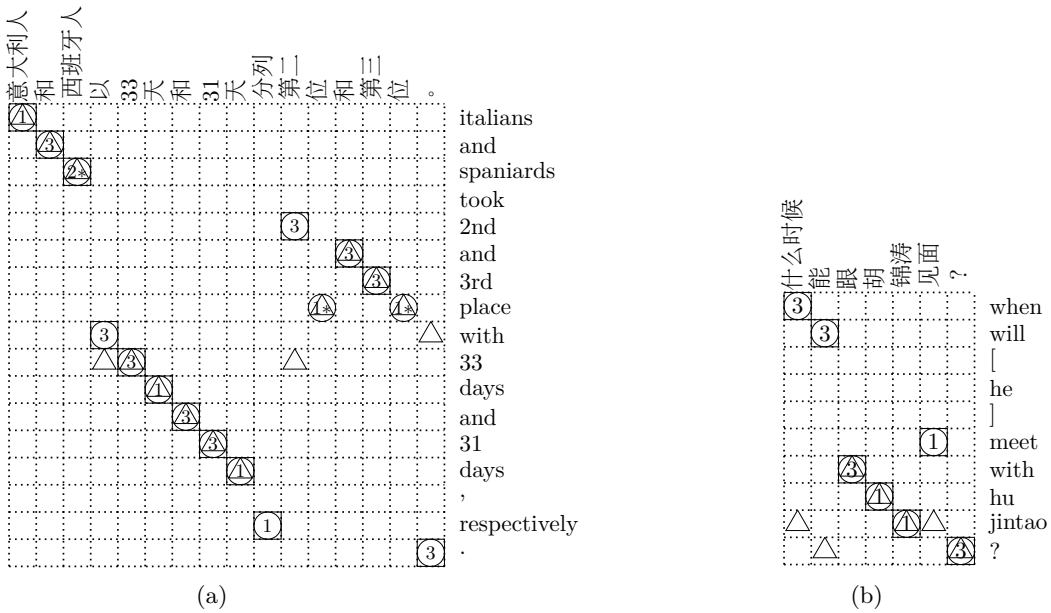
Table 4.10: Comparison of the BLEU and TER scores. HMM is our baseline HMM (cf. footnote 2). GIZA++ (Moses) is the version used in the Moses MT system.

with standard features, and tuned the weights with MERT (Och, 2003). An English 5-gram language model is trained using KenLM (Heafield, 2011) on the Gigaword corpus (Parker et al., 2011). We give a comparison between HMM+Type+Gen model, our baseline HMM, GIZA++ HMM and standard GIZA++ (as used by Moses) in Table 4.10. We report the BLEU scores and TER computed using MultEval (Clark et al., 2011).

The generative model improves over GIZA++ HMM by 1.0 BLEU points. It also improves over the standard GIZA++ by 1.2 BLEU points. HMM+Type+Gen significantly outperforms GIZA++ HMM (p-value=0.00036) and GIZA++ IBM4 (p-value=0.0004) evaluated by MultEval.

4.6 Discussion

Figure 4.2 shows the performance of baseline HMM and HMM+Type+Gen model for two word alignment examples extracted from the test data, where squares indicate the gold standard alignments. Numbers in the circles show the IDs of the predicted tags by the HMM+Type+Gen model, where ID of each tag is defined in Table 4.1. The incorrectly predicted tags are shown with the $*$ symbol.



(a) (b)

Figure 4.2: Word alignment performance of baseline HMM and HMM+Type+Gen model for two test sentences; ○: HMM+Type+Gen, △: HMM and □: gold alignment. Numbers in the circles show the IDs of the predicted alignment types by the HMM+Type+Gen model, where ids are given in Table 4.1. The incorrectly predicted alignment types are shown with the $*$ symbol.

In both examples, the HMM+Type+Gen model identifies difficult alignments over long distances better than the baseline HMM. For example, Figure 4.2(a) illustrates how our baseline HMM makes a mistake by aligning the Chinese word "。" to "with" possibly because the transition probabilities were dominant in the baseline HMM. HMM+Type+Gen model however avoids this mistake by making use of the alignment type information. The model takes into account the fact that "。" and "." are function words and should be aligned to each other with a FUN tag. Figure 4.2(b) shows that the HMM+Type+Gen model favors aligning 见面 (meet) to "meet", whereas the baseline HMM incorrectly aligns 见面 (meet) to "jintao". We hypothesize that this occurs because p(SEM| 见面, meet) has a high value.

To give a detailed analysis of the precision of the generative model in alignment type prediction, we present a confusion matrix on the test data in Table 4.11 where the vertical axis represents the actual alignment type and the horizontal axis represents the predicted alignment type. From the confusion matrix, we found that our model works well in predicting SEM, FUN, GIS, GIF, MDE and CDE alignment types since the numbers on the diagonal are the largest in the row. PDE is hard to be distinguished from MDE. COI and TIN can be easily mis-predicted by the model. NTR and MTA are omitted from this table as all the predictions for these alignment types are zero. For MTA, it is probably because this type rarely occurs in our training data. An alignment type is NTR if either Chinese or English list of tokens for that alignment is empty. In other words, the NTR alignment type is used

	SEM	FUN	GIS	GIF	MDE	PDE	CDE	COI	TIN
SEM	11374	136	2002	10	0	0	0	14	3
FUN	196	8172	21	16	2	0	0	1	1
GIS	2790	31	3312	18	2	1	2	8	0
GIF	16	118	26	772	0	0	0	0	2
MDE	0	0	1	0	293	26	2	0	0
PDE	1	0	1	2	40	55	0	0	0
CDE	0	5	0	0	0	0	79	0	0
COI	91	2	48	0	0	0	0	38	0
TIN	22	12	11	3	0	0	0	0	5

Table 4.11: Confusion matrix of the HMM+Type+Gen model on the LDC test data. The vertical axis represents the actual alignment type and the horizontal axis represents the predicted alignment type.

when some words are dropped during the translation process. We could predict NTR for the Chinese words that are aligned to NULLs. However, predicting NTR for such cases worsened the F-score of the generative model (2.0 points drop for HMM+Type+Gen model). Hence, we do not predict the NTR alignment type for any Chinese words. In total, just for the confusion matrix, 10,216 alignments (or 25.54% of all alignments) are not included in Table 4.11 which shows the alignment type predictions for the word pairs that were correctly aligned by HMM+Type+Gen.[4]

4.7 Related Work

There has been several studies on semi-supervised word alignment models. Callison-Burch et al. (2004) improve alignment and translation quality by interpolating hand-annotated, word-aligned data and automatic sentence-aligned data. They showed that a much higher weight should be assigned to the model trained on word-aligned data. Fraser and Marcu (2006) propose a semi-supervised training approach to word alignment, based on IBM Model 4, that alternates the EM step which is applied on a large training corpus with a discriminative error training step on a small hand-annotated sub-corpus. The alignment problem is viewed as a search problem over a log-linear space with features (sub-models) coming from the IBM Model 4. In the proposed algorithm, discriminative training controls the contribution of sub-models while an EM-like procedure is used to estimate the sub-model parameters. Unlike previous approaches (Och and Ney, 2003; Fraser and Marcu, 2006, 2007a) that use discriminative methods to tune the weights of generative models, Gao et al. (2010b) proposes a semi-supervised word alignment technique that integrates discriminative and generative methods. They propose to use a discriminative word aligner to produce high precision partial alignments that can serve as constraints for the EM algorithm. The dis-

[4]We should note that these incorrectly predicted alignments are only kept out of the confusion matrix. All alignments, correct or incorrect, are included in all the results we show in the other tables.

criminative word aligner uses the generative aligner's output as features. This feedback loop iteratively improves the quality of both aligners. Niehues and Vogel (2008) propose a discriminative model that directly models the alignment matrix. Although the discriminative model provides the flexibility to use manually word-aligned data to tune its weights, it still relies on the model parameters of IBM models and alignment links from GIZA++ as features. Gao et al. (2010a) present a semi-supervised algorithm that extends IBM Model 4 by using partial manual alignments. Partial alignments are fixed and treated as constraints into the EM training.

DeNero and Klein (2010) present a supervised model for extracting phrase pairs under a discriminative model by using word alignments. They consider two types of alignment links, *sure* and *possible*, that are extracted from the manually word-aligned data. *Possible* alignment links dictate which phrase pairs can be extracted from a sentence pair.

Among the unsupervised methods, (Toutanova et al., 2002) utilizes additional source of information apart from the parallel sentences. Part-of-speech tags of the words in the sentence pair are incorporated as a linguistic constraint on the HMM-based word alignment. The part-of-speech tag translation probabilities in this model are then learned along with other probabilities using the EM algorithm. POS tags as used in Toutanova et al. (2002) were also utilized to act similarly to word classes in (Och and Ney, 2000a,b); however, the improvements provided by the HMM with POS tag model over HMM alignment model of Och and Ney (2000b) was for small training data sizes (<50K parallel corpus).

Neural approaches to word alignment have been explored in the literature. Yang et al. (2013) introduce feed-forward neural network based models for translation and alignment probabilities. Their model was trained on GIZA++ alignments. Tamura et al. (2014) propose an RNN-based alignment model. They apply noise-contrastive estimation (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012) to generate negative samples for unsupervised training of their RNN-based model. Legrand et al. (2016) presented a neural network alignment model which computes alignment scores as dot products between representations of windows around source and target words. Garg et al. (2019) present an approach to train a Transformer model to extract alignments and translations in a multitask setup. Zenkel et al. (2019) extend the Transformer with an alignment layer on top of the decoder sub-network and directly optimize its activations towards predicting the given target word.

All previous studies on word alignment have assumed that word alignments are untyped. To our knowledge, the alignment types for word alignment provided by the LDC as annotations on word alignment links, have never been used to improve word alignment. Our work differs from the previous works as it proposes a new task of jointly predicting word alignment and alignment types. A semi-supervised learning algorithm is presented to solve this task. Our method is semi-supervised as it combines LDC data, which is annotated with alignment and alignment types, with sentence aligned (but not word aligned) data from the HK Hansards corpus. Our generative algorithm makes use of the gold alignment and

alignment types data to initialize the alignment type parameters. The EM training is then used to re-estimate the parameters of the model in an unsupervised manner. We also use POS tags to smooth the alignment type parameters, unlike the approach in Toutanova et al. (2002).

4.8 Conclusion

In this chapter, we introduced new probabilistic models for augmenting word alignments with linguistically motivated alignment types. Our proposed HMM-based aligners with alignment types achieved up to 13.9% improvement in the alignment F1-score over the baseline. The BLEU score improved by 1.2 points over the standard GIZA++ aligner. The proposed method can also be used for the more recent release of LDC i.e. GALE Arabic-English word alignment dataset. In the future, we plan to use alignment type information as a feature function for feature rich word alignment models and explore how alignments types can improve attention models for neural MT models. The alignment types we predict can also be used for other tasks such as projecting part-of-speech tags and dependency trees from a resource-rich language to a resource-poor language.

Chapter 5

Unsupervised Neural Hidden Markov Model for Word Alignment

Despite the rapid rise of neural approaches in different areas of NLP, neural word alignment approaches have not matured enough, and traditional unsupervised statistical models remain the most widely used approaches for word alignment. We present an unsupervised neural Hidden Markov Model for word alignment, where emission and transition probabilities are modeled using neural networks. We investigate the effect of incorporating BERT representation into our model in order to include the full target context for the emission model. In the experiments, we demonstrate improvements over GIZA++ IBM4 model, which is still a strong baseline, on Romanian-English and Chinese-English datasets.

5.1 Introduction

Word alignment is the task of discovering a word-to-word correspondence in a pair of sentences that are translations of each other. Even though neural machine translation (NMT) does not use explicit alignments, word alignment is still essential for analysis of translation errors (Ding et al., 2017), guiding NMT models during training (Chen et al., 2016; Liu et al., 2016; Alkhouli and Ney, 2017; Alkhouli et al., 2018) and improving NMT decoding (Alkhouli et al., 2016). Apart from machine translation, word alignment is essential to many NLP tasks including projecting linguistic annotations (Yarowsky and Ngai, 2001; Hwa et al., 2005) and creating multilingual embeddings (Faruqui and Dyer, 2014b; Guo et al., 2016; Dufter et al., 2018).

Generative word alignment models, IBM models 1-5 (Brown et al., 1993) and HMM (Vogel et al., 1996) are the most widely used word alignment approaches. Neural network-based alignment models have been explored in the literature, including the early works (Yang et al., 2013; Tamura et al., 2014; Legrand et al., 2016) and the more recent ones (Zenkel et al., 2019; Garg et al., 2019).

In this work, we show how we can neuralize the HMM-based word alignment model (Vogel et al., 1996). The emission and transition probabilities in the HMM are modeled using feed-forward neural networks. We then extend this basic neural HMM, such that the emission model take the full target sentence into account. A combination of forward-backward algorithm and backpropagation is used for training. This framework can be used in an end-to-end neural network unlike statistical models (e.g. those in GIZA++) that can only be used as a point estimate (using Viterbi).

We apply our neural HMM aligners on three different language pairs. We provide a comparison of our word alignment models with GIZA++ IBM4 and attention-based NMT. Our results demonstrate improvements over GIZA++ IBM4 model in terms of F1-score, for two language pairs.

5.2 Word Alignment

Given a source sentence $f_1^J = f_1, f_2, \ldots, f_J$, and a target sentence $e_1^I = e_1, e_2, \ldots, e_I$, the alignment sequence $a_1^J = a_1, a_2, \ldots, a_J$ describes a mapping between source and target words, where $a_j = i$ denotes that f_j is aligned to e_i. Without loss of generality, $Pr(f_1^J|e_1^I)$ can be decomposed into a product of alignment probabilities and translation probabilities:

$$Pr(f_1^J|e_1^I) = \sum_{a_1^J} \prod_{j=1}^{J} Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) Pr(f_j|f_1^{j-1}, a_1^j, e_1^I) \tag{5.1}$$

In the Hidden Markov alignment model, we assume a first order dependence for the alignments a_j and that the translation probability depends only on the aligned target word. The HMM-based model has the following form:

$$Pr(f_1^J, a_1^J|e_1^I) = \prod_{j=1}^{J} p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j}) \tag{5.2}$$

where $p(a_j|a_{j-1}, I)$ are the alignment probabilities (transition probabilities) and $p(f_j|e_{a_j})$ are the translation probabilities (emission probabilities).

5.3 Neural Hidden Markov Model for Word Alignment

We present a Neural Hidden Markov alignment model, where the emission and transition probabilities are modeled using feed-forward neural networks. We extend the neural HMM for unsupervised POS tag induction (Tran et al., 2016) to the task of unsupervised word alignment, a non-trivial extension just as extending an HMM (Rabiner, 1989) to handle word alignment (Vogel et al., 1996) was non-trivial.

5.3.1 Emission Architecture

For the emission model, we assume a similar dependence as in the original HMM-based model:

$$Pr(f_j|f_1^{j-1}, a_1^j, e_1^I) = p(f_j|e_i) \tag{5.3}$$

For an input target word e_i, the emission model gives a probability distribution over possible source words f_j. This can be implemented by a simple feed-forward neural network with an embedding matrix of $V_e \times D$ following by a non-linear activation (tanh). We then apply a hidden layer of size D followed by a tanh. Finally, an output linear layer of size V_f will be applied. The softmax function will be applied to provide $p(f_j|e_i)$ probabilities:

$$p(f_j|e_i) = \frac{\exp(\boldsymbol{v}_i^T \bar{\boldsymbol{v}_j})}{\sum_{k=1}^{V_f} \exp(\boldsymbol{v}_i^T \bar{\boldsymbol{v}_k})} \tag{5.4}$$

where $\boldsymbol{v}_i \in \mathbb{R}^D$ is the vector embedding of word e_i, and $\bar{\boldsymbol{v}_j}$ is the weight of unit j in the output layer. V_f and V_e are the vocabulary sizes of source and target languages, respectively.

5.3.2 Transition Architecture

Our feed-forward transition model has the same conditional dependence as the one in the original HMM-based model:

$$Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) = p(a_j|a_{j-1}) \tag{5.5}$$

We employ a Multi-Layer Perceptron (MLP) for the transition model. Given a hidden layer vector $\boldsymbol{h} \in \mathbb{R}^D$ and a weight matrix $\boldsymbol{W} \in \mathbb{R}^{{I_{max}}^2 \times D}$, where I_{max} is the maximum target sentence length, we compute the un-normalized transition matrix as follows:

$$\boldsymbol{T} = \boldsymbol{W}\boldsymbol{h} \tag{5.6}$$

We reshape $\boldsymbol{T}$ to an $I_{max} \times I_{max}$ matrix. A softmax layer per row is then applied to produce transition probabilities $p(a_j|a_{j-1})$.

5.4 Contextual Emission Model

The model explained in the previous section, was a baseline neural HMM for word alignment, which closely follows the conditional assumptions of (Vogel et al., 1996)'s. Using neural networks, however, allows for seamless integration of additional context. We augment the emission model with the target word context:

$$Pr(f_j|f_1^{j-1}, a_1^j, e_1^I) = p(f_j|e_1^I) \tag{5.7}$$

We use BERT (Devlin et al., 2018) to obtain contextual representations of the subwords in the target sentence which can be fed into our network when producing emission probabilities. The emission model has the same architecture as the one explained in section 5.3.1 except that the embedding layer is replaced with BERT.

We leverage the pre-trained $\text{BERT}_{\text{BASE}}$ cased model which has 12 layers. To pool across these layers, we use the scalar mixing technique introduced by the ELMo model (Peters et al., 2018) where we learn a weighted sum of all the layers.

5.5 Training Neural HMM

Similar to the training procedure described in Berg-Kirkpatrick et al. (2010); Tran et al. (2016), we can apply the EM algorithm where the E-step is similar to a standard HMM, but the M-step involves a gradient-based training. For the baseline neural HMM, the gradient update rule is as follows:

$$\begin{aligned} J(\theta) &= \sum_{a_1^J} p(a_1^J | f_1^J, e_1^I) \frac{\partial \ln p(f_1^J, a_1^J | e_1^I, \theta)}{\partial \theta} \\ &= \sum_j \sum_i p(i | f_1^J, e_1^I) \frac{\partial \ln p(f_j | e_i, \alpha)}{\partial \alpha} + \\ &\quad \sum_j \sum_{i'} \sum_i p(i', i | f_1^J, e_1^I) \frac{\partial \ln p(i | i', \beta)}{\partial \beta} \end{aligned}$$

where $p(i|f_1^J, e_1^I)$ and $p(i', i|f_1^J, e_1^I)$ are the posterior probabilities that can be computed using the Baum-Welch algorithm; α and β are the parameters of the emission and transition models, respectively. The gradient terms come from the backpropagation algorithm. To enhance the computational efficiency, we have applied the Direct Marginal Likelihood (Salakhutdinov et al., 2003) optimization.

During training, for a given batch, we compute the posterior probabilities using the neural-network based emission and transition models, perform backpropagation, and update the parameters of the networks. This procedure is repeated for a fixed number of epochs.

5.6 Experiments

We conducted our experiments on three language pairs, French-English (Fr-En), Romanian-English (Ro-En) and Chinese-English (Cn-En). These languages represent a range of alignment difficulties. For example, Romanian introduces a significant morphological complexity.

For Fr-En and Ro-En, we have used the trial (dev) sets and test sets provided in the 2003 NAACL shared task (Mihalcea and Pedersen, 2003).[1] For Fr-En, we have trained our models on 100K parallel sentences from Europarl, validated on 37 sentence trial set, and tested on 447 sentence test set. For Ro-En, our models are trained on WMT'16 data with 612K sentence pairs.[2] The validation and test is performed on a 17 sentence trial set and a 248 test set, respectively. For Cn-En, we used 1M sentences of the Hong Kong parliament proceedings for training, 1K dev-set and a 2K test set, both taken from the GALE Chinese-English Word Alignment and Tagging corpus.[3]

We measure the performance of our models using precision, recall, and F1-score, as suggested by Fraser and Marcu (2007b). To evaluate the models with subword units, we need to make the produced alignment and reference alignment comparable. To convert subword alignment to word alignment, we align a source word to a target word if any of their subwords were aligned together.

5.6.1 Experimental Setup

We closely follow the setting used in Tran et al. (2016). We considered the vocabularies to be the top 50K frequent words for both languages. Following Bahdanau et al. (2015), sentences of length 50 or lower were used for training. Hence, for the transition model, I_{max} is set to 50. The emission model of our baseline Neural HMM (NHMM) has an Embedding layer of size 512. Both emission and transition models have 512 hidden units, as it was the largest we could fit into memory.

NHMM+BERT uses pre-trained $\text{BERT}_{\text{BASE}}$ cased model for English (∼30K vocab size). This model works on subword level. For the source side, we used the cased multilingual BERT tokenizer which is based on the WordPiece model (∼120K vocab size). The BERT model's weights were frozen and the pre-trained representations were used in our model similar to classic feature-based approaches.

We used the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001. Our batch size is set to 64 due to GPU memory limitation. All the models were trained for 5 epochs, and the model with the best F1-score on the dev-set was picked to be evaluated on the test data.

[1] `http://web.eecs.umich.edu/~mihalcea/wpt/index.html`

[2] `https://www.statmt.org/wmt16/translation-task.html`

[3] Catalog number for training: LDC2000T50; catalog numbers for development and test sets: LDC2012T16, LDC2012T20, LDC2012T24, LDC2013T05, LDC2013T23 and LDC2014T25.

5.6.2 Results

We compare our models against two baselines: GIZA++ IBM Model 4, and attention-based NMT. For GIZA++, we add the test data to the training, and use Moses (Koehn et al., 2007) with its default parameters to obtain word alignments.[4] We report its performance on the test data. Unlike GIZA++, our models are not trained on the train and test data. For attention baseline, we used Open-NMT (Klein et al., 2017) to train our attention-based NMT systems. To obtain the F1-score of attention models, we follow Luong et al. (2015) to produce hard alignments from attentions by selecting the most attended source word for each target word.

Tables 5.1-5.3 show the models' performance for the GIZA++ baseline, attention-based NMT, and the proposed models for Fr-En, Ro-En and Cn-En language pairs, respectively. In these tables, NHMM refers to the proposed neural HMM baseline, while NHMM+BERT is the neural HMM with a contextual emission model.

In most cases, NHMM outperforms the GIZA++ baseline. NHMM outperforms the attention model in all language pairs. As expected, NHMM+BERT improves over the baseline NHMM, making good use of the additional context. NHMM+BERT achieves a notable gain of 8.2% F1-score for Cn-En. It is noted that during the tuning experiments, NHMM+BERT got 79.5 F1-score while GIZA++ got 77.2 F1-score on the dev-set. The superiority of GIZA++ for Fr-En, as seen from Table 5.1, is most likely because it uses the test set in the training.

As discussed in section 5.3.2 and unlike Wang et al. (2017); Ngo-Ho and Yvon (2019), our transition model depends on the absolute positions and not on the jump width. We have performed some experiments with the distance-based model where $Pr(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) = p(\Delta_j)$. For example, in the experiments on the dev-set, our model improves over the distance-based model F1-score from 75.9 to 76.5 for Fr-En and from 63.3 to 67.9 for Ro-En.

Our code is implemented in pytorch. When running on a single GPU of NVIDIA Quadro P6000, we achieve an average speed of 1212, 1543 and 650 target words per second for Fr-En, Ro-En and Cn-En, respectively. NHMM has 53.1M parameters while NHMM+BERT has 70.5M parameters.

5.7 Related Work

Neural approaches to word alignment have been explored in the literature. Yang et al. (2013) introduced feed-forward neural network based models for translation and alignment probabilities. Their model was trained on GIZA++ alignments. Tamura et al. (2014) proposed an RNN-based alignment model. They applied noise-contrastive estimation (Gutmann and

[4]Moses tokenizer was used for tokenization.

Model	Prec.	Rec.	AER	F1
GIZA++	75.2	90.7	19.3	82.3
Attention	64.6	62.1	36.3	63.3
NHMM	74.4	83.2	22.6	78.5
NHMM+BERT	75.5	86.2	21.0	80.5

Table 5.1: Word alignment results for Fr-En.

Model	Prec.	Rec.	AER	F1
GIZA++	65.0	55.7	40.0	60.0
Attention	59.0	54.1	43.6	56.4
NHMM	65.3	58.2	38.5	61.5
NHMM+BERT	63.1	63.6	36.6	63.4

Table 5.2: Word alignment results for Ro-En.

Model	Prec.	Rec.	AER	F1
GIZA++	54.9	43.4	51.5	48.5
Attention	42.4	44.1	56.8	43.2
NHMM	56.9	45.7	49.3	50.7
NHMM+BERT	58.3	55.2	43.3	56.7

Table 5.3: Word alignment results for Cn-En.

Hyvärinen, 2010; Mnih and Teh, 2012) to generate negative samples for unsupervised training of their RNN-based model.

Alkhouli et al. (2016) and Wang et al. (2017) applied the hidden Markov model decomposition using feed-forward translation and alignment models. Training of alignment-based NMT (Alkhouli et al., 2016) relies on GIZA++ alignments. The two models are trained separately and combined during decoding. Alkhouli and Ney (2017) used a modified attention model as a translation model and an RNN-based alignment model. Wang et al. (2017) employed an HMM with end-to-end training. The HMM is used for re-ranking n-best lists created by a phrase-based decoder. The training algorithm is a combination of forward-backward and backpropagation. This is similar to the training procedure of Tran et al. (2016) who proposed a general unsupervised neural HMM which can be used for models with latent variables and tractable inference. They demonstrated their method for part-of-speech induction task. We propose an unsupervised neural HMM for word alignment. Unlike in Tran et al. (2016), our emission model leverages the full target sentence context, using the pre-trained BERT model (Devlin et al., 2018). Ngo-Ho and Yvon (2019) present neural baselines for word alignment models, including IBM1 and HMM. They looked at various neuralizations. Their contextual translation model depends on a window context

of size 1 around the aligned target word, which is similar to the window context concept used in Alkhouli et al. (2016); Wang et al. (2017). Unlike the jump-based transition model of Wang et al. (2017); Ngo-Ho and Yvon (2019), our transition model is position-based. We leverage full sentence context using BERT embeddings unlike their work. Zenkel et al. (2019); Garg et al. (2019); Zenkel et al. (2020) extend the transformer architecture to obtain word alignments. Garg et al. (2019) present an approach to train a Transformer model to extract alignments and translations in a multitask setup. Zenkel et al. (2019) extend the Transformer with an alignment layer on top of the decoder sub-network and directly optimize its activations towards predicting the given target word. These methods do not use posterior probabilities in contrast to our approach.

5.8 Conclusion

We have presented a neural HMM for word alignment. Using pre-trained BERT embeddings, we have shown how to augment the emission model with the full target sentence context. Our results show improvements for Chinese-English and Romanian-English. Our NHMM+BERT achieved up to 8.2% improvement in the alignment F1-score over GIZA++ IBM4 baseline for Chinese-English. In the next chapter, we use our neural aligner for projecting part-of-speech tags from a resource-rich language to a resource-poor language.

Chapter 6

Cross-lingual Annotation Projection with Neural HMM for Low-resource Languages

We tackle the part-of-speech tagging task for the zero-resource scenario where no POS annotated training data is available. We present a cross-lingual projection approach where neural HMM aligners, discussed in Chapter 5, are used to obtain high quality word alignments between low-resource languages and high-resource language. Moreover, high quality neural POS taggers are used to provide annotations for the resource-rich language side of the parallel data, as well as to train a tagger on the projected data. Our experimental results on truly low-resource languages show that our methods outperform their corresponding baselines.

6.1 Introduction

Part-of-speech (POS) tagging is a crucial task in Natural Language Processing (NLP). Supervised POS taggers have shown state-of-the-art accuracies for many well-resourced languages which is close to the level of inter-annotator agreement. However, state-of-the-art approaches for POS tagging only scale to a small fraction of the world's 6900 languages. The main bottleneck is the lack of annotated data for the vast majority of these languages.

Although it is not feasible to acquire manually annotated data for the majority of low-resource languages, finding a parallel data between the low-resource language and the high-resource language is much easier. Previous studies on low-resource NLP has primarily focused on leveraging parallel corpora to project annotations from a resource-rich language to a resource-poor language (Yarowsky and Ngai, 2001; Guo et al., 2015; Buys and Botha, 2016). POS tags are projected via word alignments, and the projected POS data is then used to train a model in the low-resource language (Das and Petrov, 2011; Täckström et al., 2013; Fang and Cohn, 2016).

Majority of the previous works in the cross-lingual transfer learning for low-resource languages assume that linguistic resources are available for the low-resource languages. This assumption does not hold for truly low-resource languages. On the other hand, these approaches are biased toward Indo-European languages. Cross-lingual learning between these languages are easier due to the large volume of translated texts between these languages, easier tokenization and segmentation for these languages, and word order similarity which makes the word alignments more reliable.

We present an approach to learn POS taggers for truly low-resource languages, making minimum assumptions about the available linguistic resources. These assumptions do in fact hold for truly low-resource languages. Therefore, we consider zero-resource scenario for the target languages i.e. we assume no labeled data, tag dictionaries, typological information, etc. is available for these languages. We, however, need a parallel corpus for annotation projection. Therefore, we have to rely on low-resource languages for which we can find parallel corpora between those languages and a resource-rich language (preferably a related one). For this reason, we found two real low-resource languages from the Universal Dependencies (UD) treebanks (Nivre et al., 2016), Kazakh and Breton, which could fit into our assumption. We have also simulated the zero-resource scenario for French by not using any POS-annotated French training data.

Our approach follows Yarowsky and Ngai's original annotation projection approach. However, we used our proposed neural HMM aligners to obtain high quality word alignments between our low-resource languages and English. Moreover, we used a high quality BiLSTM-CRF tagger (Rei and Yannakoudakis, 2016; Rei et al., 2016) to tag the English side of the parallel data, and to train a tagger on the target language after projection. Our experimental results demonstrates the effectiveness of our approach for low-resource languages.

6.2 Part-of-Speech Tagger Induction

The lack of annotated data for the vast majority of languages is a major obstacle to the development of statistical part-of-speech taggers for these languages. Although hand-annotated data for low-resource languages are scarce, parallel data between a low-resource language and a high-resource language is easier to obtain. Yarowsky and Ngai (2001) pioneered the annotation projection approach where they use word-aligned data to project annotations from a resource-rich language to an arbitrary language. The resulting projected data can then serve as training data for developing applications in the arbitrary (probably low-resource) language. We closely follow Yarowsky and Ngai (2001) for cross-lingual POS tag projection. Their approach is explained for a cross-lingual transfer from English to French. However, the proposed method is language-independent and can be applied to any language-pair as long as we have a parallel data between the two languages and a high quality POS tagger

for the resource-rich language. Throughout this chapter, we assume that the source of the projection (resource-rich side of the parallel data) is English.

6.2.1 POS projection

We present our cross-lingual projection approach here. First, we tag the English side of the parallel data using a BiLSTM-CRF tagger (Section 6.5.3). We use our Neural HMM aligners (Chapter 5) to induce word alignments on the parallel data. Following the prior works (Lacroix et al., 2016), we ignore unaligned words as well as many-to-many alignments by using one directional alignments where one target token can never be aligned to multiple English tokens. Note that, we allow many-to-one projection from one English token to multiple target tokens. By doing this, annotation projection becomes straightforward. Although using all the alignments (i.e. many-to-many and one-to-many) result in more POS-tagged tokens, it also introduces considerable noise. The NHMM+BERT model presented in Chapter 5 produces many-to-many alignments. For the experiments in this chapter, we use the first subword representation as the input to our network when producing emission probabilities.

To reduce the noise in the projected data, following Yarowsky and Ngai (2001), we develop a re-estimation technique based on the assumption that words have a strong tendency to exhibit only one core POS tag and very rarely have more than two. Finally, We learn a BiLSTM-CRF tagger on the target side of the parallel data.

6.2.2 Yarowsky and Ngai (2001)'s Approach

The goal is to use a state-of-the-art POS tagger to annotate the English side of the parallel corpus, and then project the annotations to the second language using the word alignment, and generalize from this noisy projection in a robust way. Figure 6.1 shows the different scenarios in the projection of English POS tags via word alignments to French. For a one-to-one alignment, the projection can be done easily by copying the source side tag to the target side. However, for the many-to-one alignment cases such as the alignment between the English word *Croissants* and French phrase *Les croissants*, the challenge is to assign the English plural noun (NNS) tag to which of the French words. Yarowsky and Ngai (2001) suggested to assign tags to the words in the compound considering their relative position in the compound (a, b, c, etc.). For example, NNS_a which corresponds to the first many-to-one alignment position in a French compound will have a tendency to have a high probability of corresponding to a French determiner while the second position NNS_b tend to have a low probability of corresponding to a French determiner.

Direct projection introduces noisy and unreliable annotations. Supervised POS tagging algorithms perform poorly on the noisy projected annotations. Here, we will give the noise-robust techniques for dealing with the challenging raw projection data. To this end, we

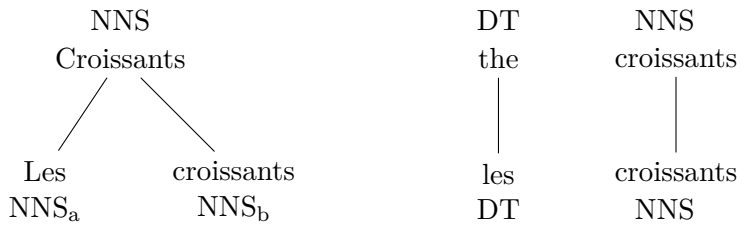


Figure 6.1: An alignment between a French sentence and its translation in English.

remove the sentences that are poorly aligned, train a bigram tagger by learning the lexical prior model $P(T)$ and tag-sequence model $P(W|T)$ separately.

Lexical Prior Estimation

Given a sequence of words W with length n, let T be the corresponding sequence of tags of W. A bigram tagger finds the best tag sequence:

$$\arg\max_T P(T|W) = P(W|T)P(T) \tag{6.1}$$

where $P(T)$ can be approximated using chain rule and Markov assumption:

$$P(T) = P(t_1 \dots t_n) \approx P(t_1)P(t_2|t_1) \dots P(t_n|t_{n-1}) \tag{6.2}$$

and $P(W|T)$ can be estimated using independence assumptions:

$$P(W|T) = P(w_1 \dots w_n|t_1 \dots t_n) \approx \prod_{i=1}^{n} P(w_i|t_i) \tag{6.3}$$

To estimate $p(w_i|t_i)$, using Bayes rule we have:

$$P(w_i|t_i) = \frac{P(t_i|w_i)P(w_i)}{\sum_j P(t_i|w_j)P(w_j)} \tag{6.4}$$

where $P(w_i)$ can be estimated from the French data, and $P(t_i|w_i)$ is estimated based on their frequency in the corpus:

$$P(t_i|w_i) = \frac{c(t_i, w_i)}{c(w_i)} \tag{6.5}$$

However, the distributions of tags per word in the manually tagged corpora in French and English show that words in French and English usually have a unique POS tag, and very rarely have more than 2. We therefore apply an aggressive re-estimation in favor of this bias, amplifying the model probability of the most frequent POS tag, and reducing the model

probability of the 2nd most frequent core tags and zeroing the probability for the rest of the tags. Let $t_{(i)}$ be the i^{th} most frequent tag for w, we smooth the lexical prior model as follows:

$$\begin{aligned}
&\hat{P}(t_{(2)}|w) = \lambda_1 P(t_{(2)}|w) && \text{where } \lambda_1 < 1.0 \\
&\hat{P}(t_{(1)}|w) = 1 - \hat{P}(t_{(2)}|w) \\
&\hat{P}(t_{(c)}|w) = 0 && \text{for all c} > 2
\end{aligned} \tag{6.6}$$

It is worth noting that for many-to-one alignment cases as shown in Figure 6.1 even though it might seem that function words get substantial probability mass through such alignments (eg. Croissants/NNS $\rightarrow$ Les/NNS_a croissants/NNS_b), because of the frequent correct one-to-one alignments (eg. The/DT $\rightarrow$ Les/DT) and the aggressive smoothing toward the most frequent POS tag, the model does not suffer from an incorrect assignment for function words. Yarowsky and Ngai (2001) also performs a linear interpolation of the tags distributions from one-to-one alignments, and from the many-to-one alignments as follows:

$$P(t|w) = \lambda_2 P_{\text{1-to-1}}(t|w) + (1 - \lambda_2) P_{\text{n-to-1}}(t|w) \tag{6.7}$$

Tag Sequence Model Estimation

The tag sequence bigram model is estimated over a chosen filtered set of aligned sentences in the corpus. The sentences are chosen based on two measures: 1) The alignment score of the sentence which indicates the overall alignment confidence and 2) The tag sequence model score which is computed after the lexical prior models have been trained. A sentences of length k is scored by a pseudo-divergence weighting as follows:

$$\frac{1}{k} \sum_{i=1}^{k} log\hat{P}(\textit{projected-tag}_i|w_i) \tag{6.8}$$

where a sentence is penalized for those words whose projected tags do not match the most frequent tag. Sentences are sorted and filtered based on their score, and the tag model is estimated over a confident subset of the data. Note that since there is a high probability for function words in a many-to-one alignment to get an incorrect tag from raw projections, we correct the tags before training the tag sequence model by simply replacing their raw projection tag with the most frequent tag for the word.

6.3 Universal Tagset

The projection method requires a common tagset between the source and target languages. This way, projected tags can be used as target labels. However, any two languages exhibit mismatch in their POS tagset. It is uncommon for languages to be annotated with the same

tagset. The universal POS tagset (Petrov et al., 2012) has been widely used as a solution to this issue. This tagset consists of 12 common part-of-speech tags. Petrov et al. (2012) also provide a mapping from 25 different treebank tagsets to this universal tagset.

6.3.1 Universal Dependencies

Universal Dependencies (UD) is another effort to create cross-linguistically consistent treebank annotation for many languages. The annotation scheme is a combination of the Stanford dependencies (De Marneffe et al., 2006; De Marneffe and Manning, 2008; De Marneffe et al., 2014) and the Google universal part-of-speech tags (Petrov et al., 2012). Table 6.1 shows the 17 universal core POS tags which are divided into open class words, closed class words, and other symbols.

Open class words		Closed class words		Other	
ADJ	adjective	ADP	preposition/postposition	PUNCT	punctuation
ADV	adv	AUX	auxilary	SYM	symbol
INTJ	interjection	CCONJ	coordinating conjunction	X	unspecified POS
NOUN	noun	DET	determiner		
PROPN	proper noun	NUM	numerals		
VERB	verb	PART	particle		
		PRON	pronoun		
		SCONJ	subordinating conjunction		

Table 6.1: Universal Dependencies part-of-speech tags (Nivre et al., 2016).

6.4 Neural Part-of-Speech Tagger

To perform the annotation projection, we use a high quality supervised neural POS tagger (Rei and Yannakoudakis, 2016; Rei et al., 2016) to tag the English side of the bitext, and to train a tagger on the projected data. The details of this model will be given in the following sections.

6.4.1 Bidirectional LSTM for Sequence Tagging

Given a sequence of words $(w_1, \ldots, w_T)$ as input, the goal of a neural sequence tagger is to predict a tag for each word in the input sequence. The words are first mapped to a sequence of word embeddings $(x_1, \ldots, x_T)$. The embeddings are then fed into a bidirectional LSTM to create context-specific representations for each word:

$$\overrightarrow{h_t} = LSTM(\overrightarrow{h}_{t-1}, \boldsymbol{x}_t)$$
$$\overleftarrow{h_t} = LSTM(\overleftarrow{h}_{t+1}, \boldsymbol{x}_t)$$
$$h_t = [\overrightarrow{h_t}^T; \overleftarrow{h_t}^T]^T \tag{6.9}$$

where a representation for each source word w_t is obtained by concatenating the forward hidden state $\overrightarrow{h_t}$ (forward representation) and the backward hidden state $\overleftarrow{h_t}$ (backward representation).

Rei and Yannakoudakis (2016) use an extra hidden layer on top of the LSTM to detect higher-level feature combinations:

$$d_t = \tanh(W_d h_t) \tag{6.10}$$

where W_d is the weight matrix. The softmax layer is applied to produce a probability distribution over all possible tags for each word:

$$P(y_t = k|d_t) = \frac{e^{W_{o,k} d_t}}{\sum_{\tilde{k} \in K} e^{W_{o,\tilde{k}} d_t}} \tag{6.11}$$

where $P(y_t = k|d_t)$ is the probability of y_t being tagged as k, and K is the set of all possible tags. $W_{o,k}$ is the k^{th} row of the weight matrix W_o between the hidden layer and the output layer. During training, we minimize the negative log-probability of the correct tags:

$$E = -\sum_{t=1}^{T} log(P(y_t|d_t)) \tag{6.12}$$

Huang et al. (2015) proposed a BiLSTM with a Conditional Random Field (CRF) layer, denoted as BiLSTM-CRF, for the sequence tagging task. In BiLSTM-CRF, given a sequence of output tag predictions y = $(y_1, y_2, \ldots, y_n)$, the CRF score of this output sequence can be calculated as

$$s(y) = \sum_{t=1}^{T} A_{t,y_t} + \sum_{t=0}^{T} B_{y_t, y_{t+1}} \tag{6.13}$$

$$A_{t,y_t} = W_{o,y_t} d_t \tag{6.14}$$

where A_{t,y_t} is the probability of y_t being the tag of t and $B_{y_t,y_{t+1}}$ is the probability of transitioning from tag y_t to tag y_{t+1}; these values are optimized during training. In the decoding, the tag sequence with the largest $s(y)$ score is computed using the Viterbi algorithm. To optimize the CRF model, the loss function maximizes the score for the correct tag sequence,

while minimizing the scores for all the other sequences:

$$E = -s(y) + log \sum_{\tilde{y} \in \tilde{Y}} e^{s(\tilde{y})} \tag{6.15}$$

where $\tilde{Y}$ is the set of all possible tag sequences.

6.4.2 Character-level Sequence Tagging

Word embeddings treat words as atomic units and cannot capture the morphological and orthographic similarity between different words. By incorporating the character-level information, we can benefit from these regularities. This is especially helpful in handling unseen words. Rei et al. (2016) proposed two methods for incorporating the character-based representations into their sequence tagger. In the first method, each word is broken down into its characters. These characters are mapped to a sequence of character embeddings $(c_1, \ldots, c_R)$ which are then fed into a BiLSTM. The last hidden states of the forward and the backward LSTMs are concatenated and then passed to a non-linear layer to form a character-based representation of the word. This representation can be concatenated with the word-level representation x to form a new word-level representation.

6.4.3 Attention over Characters

We now briefly explain the second approach of Rei et al. (2016) for incorporating the character-level information into the sequence tagger model. In this approach, given an input word, a BiLSTM is applied over its characters and the last hidden states are used to form a vector m for the word. Instead of concatenating m with x (as in section 6.4.2), these two vectors are combined using a weighted sum:

$$z = \sigma(W_z^{(3)} \tanh(W_z^{(1)} x + W_z^{(2)} m)) \tag{6.16}$$

where

$$\tilde{x} = z \odot x + (1 - z) \odot m \tag{6.17}$$

where σ and $\odot$ are the logistic sigmoid function and the element-wise multiplication function, respectively. $W_z^{(1)}, W_z^{(2)}$ and $W_z^{(3)}$ are the weight matrices. The vector z controls how much information to use from the word embedding or the character-level component. This modelling allows unknown words to benefit from the character-level regularities whenever possible.

6.5 Experiments

We have performed our experiments for three languages. A simulation experiment for French and two real-world low-resource languages, Kazakh and Breton. For all the experiments, we used English as the source language and each of the three languages as the target language.

6.5.1 Data

We used Universal Dependencies (UD) v2.3 dataset (Nivre et al., 2018)[1] which covers 76 languages. The annotated data that we used to train the English POS tagger is taken from UD English Web Treebank (UD-EWT) corpus. Our test data for all three languages also comes from UD. It consists of 416, 1047 and 888 test sentences for French, Kazakh and Breton, respectively. For French, we used the test data of the UD_French-GSD treebank.

Parallel Corpora: For French-English, we have trained our word alignment models on 100K parallel sentences from Europarl (Koehn, 2005).[2] For Kazakh-English, our models are trained on 70K parallel Kazakh-English data provided by (Zhumanov et al., 2017). For Breton-English, OpenSubtitles2018 (Lison et al., 2019), an 18K sentence-aligned parallel data from the OPUS corpus (Tiedemann, 2012) [3] was used. We have tokenized our parallel training data with UDPipe 1.2 tokenizer (Straka and Straková, 2017).

6.5.2 Word Alignment Experimental Setup

We used both of our proposed neural HMM aligners, NHMM and NHMM+BERT, with the same setting that was given in Section 5.6.1. Our code is implemented in pytorch. When running on a single GPU of NVIDIA Quadro P6000, we achieve an average speed of 1891, 1624 and 2510 target words per second for French-English, Kazakh-English and Breton-English, respectively.

6.5.3 BiLSTM Tagger

We used a BiLSTM-CRF (with character level component) sequence tagger provided by (Rei and Yannakoudakis, 2016; Rei et al., 2016) [4] to tag the English side of the parallel data; see Section 6.4 for the details of the model. Because of the high accuracy of the tagger we can treat these tags as gold POS tags, and project the labels using the robust techniques (discussed in section 6.2) to the target side of the parallel data. Once we obtained

[1] `http://hdl.handle.net/11234/1-2895`

[2] `https://www.statmt.org/europarl`

[3] `http://opus.nlpl.eu/`

[4] https://github.com/marekrei/sequence-labeler

the projected annotations on the target side, we trained our BiLSTM-CRF tagger on each target dataset, and finally evaluated the accuracy on the corresponding test treebank.

Experimental Setup. The word embeddings were initialized with the gloves 300-dimensional word embeddings and then fine-tuned during the training. The size of character embeddings was set to 50. The sizes of the word-level LSTM and the character-level LSTM were 300 and 100, respectively. The hidden layer size was set to 50. Parameters were optimized using AdaDelta (Zeiler, 2012) with the default learning rate of 1.0. The batch size of 32 was used. For the English tagger, performance on the development set was measured after each epoch and training was stopped if the performance had not improved for 7 epochs. The model with the best performance on the development set was used for evaluation on the test set.

Result. Using the UD_English-EWT train/dev/test split, the neural part-of-speech tagger achieved an accuracy of 0.956 on the test set.

6.5.4 Baselines

We compare our models with several baselines:

Unsupervised: the unsupervised neural HMM POS tagger (Tran et al., 2016) discussed in Chapter 3. We used 17 tag clusters, the number of POS tags that appear in the UD treebanks. We evaluated the performance of POS tagging with One-to-One (1-1), Many-to-One (M-1) accuracy (Johnson, 2007) and V-Measure (VM) (Rosenberg and Hirschberg, 2007). However, we have reported the 1-1 accuracy in order to provide a fair comparison to the accuracy of our projection methods. Our baselines are trained on the target side of the parallel data using NHMM+Conv+LSTM model, the best performing model presented in (Tran et al., 2016), which is the model with convolutional neural network-based emission, and LSTM-based transition model (Section 3.6). We used a three layer LSTM as it worked well in their experiments. We set the hidden layer size to be 512 as this was the largest we could fit into the memory.

Supervised: A supervised model trained on a small amount of annotated data. For this baseline, we trained a BiLSTM-CRF tagger (Rei and Yannakoudakis, 2016) on a small (500 tokens) annotated data from the resource-poor language. This baseline gives a comparison between the low-resource and zero-resource scenario. Note that this baseline can only be evaluated for French and Kazakh. For Breton, there is no training or development data available in UD.

Cross-lingual Transfer with GIZA++ Word Alignments: This baseline gives a comparison between the performance of our neural HMM aligners against GIZA++ IBM4 in the POS projection task. For GIZA++, we used Moses (Koehn et al., 2007) with its default parameters to obtain word alignments. The unaligned words are tagged with NOUN which is the most frequent tag in the corpora.

Model	French	Kazakh	Breton
Unsupervised (Tran et al., 2016)	43.5	32.7	51.9
GIZA++ +DP	77.5	59.0	42.9
GIZA++ +NR	81.9	59.5	48.9
Attention+DP	73.9	56.1	31.9
Attention+NR	**84.2**	61.6	37.5
Supervised (500 tokens)	74.9	59.8	-
NHMM+DP	75.6	56.2	57.8
NHMM+NR	81.2	53.4	56.9
NHMM+BERT+DP	75.5	61.7	56.9
NHMM+BERT+NR	82.0	**62.4**	**57.8**

Table 6.2: POS tagging accuracy on French, Kazakh and Breton.

Cross-lingual Transfer with Word Alignments Extracted from Attention: Similar to the previous baseline, this baseline is also based on the annotation projection approach, but with a different alignment model. It relies on the alignments that are extracted from attention-based NMT systems. We used Open-NMT (Klein et al., 2017) to train our attention-based NMT systems. We follow Luong et al. (2015) to produce hard alignments from attentions by selecting the most attended source word for each target word. To be consistent with our aligners (in producing many-to-one alignments), we trained English-French, English-Kazakh and English-Breton NMT models, keeping 1k from the data as the dev set and the rest for training. We tested on the whole training data to obtain word alignments.

6.6 Results

Table 6.2 presents the results for all our proposed methods and their corresponding baselines for all the languages. We experimented with two different transfer methods, i.e. direct projection (DP) and noise robust (NR) projection. In Table 6.2, MODEL+DP denotes the direct projection method that uses MODEL word alignment and BiLSTM-CRF tagger on the English and the target side while MODEL+NR denotes the noise robust transfer method with MODEL word alignment and BiLSTM-CRF tagger.

From the table, we observe that our model generally works best for truly low-resource languages. Our best performing model, NHMM+BERT+NR, makes good use of the additional context by using NHMM+BERT aligner. NHMM+BERT performs better than the GIZA++ baseline. The difference is especially substantial for Breton for which we have a smaller parallel data. The unsupervised NHMM POS tagger is the weakest baseline for French and Kazakh, but the strongest for Breton. Noise reduction techniques are most effective when used for French. The limited effectiveness of these techniques for Breton and Kazakh are possibly due to the dissimilarity between these languages and English, and also

	NOUN	VERB	ADJ	ADV	ADP	AUX	PRON	PROPN	INTJ	DET	CCONJ	NUM	PART	SCONJ	PUNCT	SYM	X
NOUN	1617	59	96	22	9	4	7	9	0	1	0	27	1	0	0	0	0
VERB	64	628	14	1	1	115	3	2	0	0	0	0	0	0	0	0	0
ADJ	218	23	339	6	0	0	0	3	0	3	0	0	0	0	0	0	3
ADV	11	11	20	288	13	76	6	0	1	0	6	0	54	11	0	0	0
ADP	1	11	0	7	1442	3	1	0	0	0	0	0	6	13	0	0	0
AUX	2	15	0	0	0	338	0	0	0	0	0	0	0	0	0	0	0
PRON	7	31	7	14	7	73	368	0	0	24	0	3	1	12	0	0	0
PROPN	365	10	50	1	0	3	14	60	0	0	0	11	0	0	0	0	2
INTJ	2	1	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0
DET	0	1	4	1	48	0	73	0	0	1369	0	0	0	0	0	0	0
CCONJ	0	0	0	0	0	0	0	0	0	0	238	0	0	7	0	0	0
NUM	8	0	1	0	1	0	1	0	0	4	0	214	0	0	0	0	0
PART	0	1	0	0	0	0	1	0	0	0	0	3	0	0	0	0	0
SCONJ	0	0	0	7	0	4	0	0	0	0	0	0	0	120	0	0	0
PUNCT	13	1	0	0	0	1	1	0	0	2	0	3	0	0	1174	0	0
SYM	2	0	1	2	0	0	2	0	0	0	0	3	0	0	1	19	0
X	5	0	0	0	0	0	0	0	0	0	0	2	0	0	1	0	1

Table 6.3: Confusion matrix on the French test data. The vertical axis represents the actual POS tag and the horizontal axis represents the predicted POS tag.

	NOUN	VERB	ADJ	ADV	ADP	AUX	PRON	PROPN	INTJ	DET	CCONJ	NUM	PART	SCONJ	PUNCT	SYM	X
NOUN	2110	153	127	33	16	8	35	319	0	89	0	51	0	1	15	4	0
VERB	376	826	27	13	93	25	7	9	0	24	4	0	0	0	106	0	0
ADJ	204	33	372	12	9	43	5	25	0	34	0	2	0	0	48	0	0
ADV	56	19	20	83	7	12	14	4	0	14	3	0	0	0	59	0	0
ADP	42	33	5	23	41	0	0	1	0	3	0	0	0	0	14	0	0
AUX	31	164	3	0	45	11	2	2	0	8	11	0	0	0	115	0	0
PRON	95	21	4	33	0	1	209	7	0	83	18	0	0	0	3	0	0
PROPN	260	20	13	7	0	0	8	183	0	26	0	1	0	0	1	0	0
INTJ	3	10	0	0	0	0	2	2	0	1	1	0	0	0	4	0	0
DET	58	0	15	11	0	1	40	3	0	80	7	0	0	0	0	0	0
CCONJ	11	1	1	1	0	0	11	1	0	2	132	0	0	0	13	0	0
NUM	22	1	6	2	3	0	2	4	0	35	0	266	0	0	3	0	11
PART	3	3	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0
SCONJ	7	1	0	3	0	0	2	3	0	0	0	0	0	2	2	0	0
PUNCT	13	0	2	0	0	0	0	15	0	14	0	2	0	0	1936	0	0
SYM	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
X	52	33	4	3	4	0	2	2	0	6	1	0	0	0	6	0	0

Table 6.4: Confusion matrix on the Kazakh test data. The vertical axis represents the actual alignment type and the horizontal axis represents the predicted alignment type.

their lower alignment quality. Attention-based methods provide surprisingly competitive results for French and Kazakh while their tagging accuracies are very poor for Breton.

To give a detailed analysis of the performance of our best model (NHMM+BERT+NR) in POS prediction, we present confusion matrices on the test data for all three languages in Tables 6.3, 6.4 and 6.5. In these tables, the vertical axis represents the actual POS tag while the horizontal axis represents the predicted POS tag. We found that our model works well in predicting most of the POS tags since the numbers on the diagonal are the largest in their corresponding rows. Our model, in most cases, confuses PROPN with NOUN for French and Kazakh, which is a common error to occur. Our model fails to capture X and PART POS tags. INTJ is hard to predict probably because there is very few examples in the projected data to learn this POS tag.

	NOUN	VERB	ADJ	ADV	ADP	AUX	PRON	PROPN	INTJ	DET	CCONJ	NUM	PART	SCONJ	PUNCT	SYM	X
NOUN	1487	192	41	9	24	24	88	125	0	0	0	0	0	0	0	0	0
VERB	205	861	1	9	28	463	245	21	0	0	0	0	0	0	0	0	0
ADJ	235	43	58	3	5	4	70	17	0	0	0	0	0	0	0	0	0
ADV	178	26	2	110	11	9	131	5	1	1	0	0	76	0	0	0	0
ADP	41	71	1	1	523	62	390	3	0	8	0	0	1	0	1	0	0
AUX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PRON	81	17	0	3	1	27	112	0	0	0	0	0	0	0	0	0	0
PROPN	25	38	0	3	0	0	10	232	0	0	0	0	0	0	0	0	0
INTJ	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
DET	72	24	8	2	2	14	233	0	0	849	0	0	0	0	1	0	0
CCONJ	0	1	1	0	0	0	4	0	0	0	200	0	0	0	0	0	0
NUM	29	9	15	1	1	5	6	5	0	1	0	155	0	0	5	0	0
PART	2	64	0	0	0	311	215	2	0	9	0	0	0	0	0	0	0
SCONJ	0	0	0	10	1	0	26	0	0	0	2	0	0	0	0	0	0
PUNCT	27	1	0	0	2	2	6	0	0	0	0	0	0	0	1113	0	0
SYM	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X	31	13	7	4	34	1	28	3	0	22	2	7	1	0	0	0	0

Table 6.5: Confusion matrix on the Breton test data. The vertical axis represents the actual alignment type and the horizontal axis represents the predicted alignment type.

6.7 Related Work

Yarowsky and Ngai (2001) pioneered the annotation projection approach where they use word-aligned data to project annotations from a resource-rich language to an arbitrary language. They used the projected partial target annotation to estimate the parameters of an HMM. However, Because of the nature of the word alignments and the POS tags, direct tag projection introduces a considerable noise. To conquer this noise, they used aggressive smoothing techniques when training their HMM.

Fossum and Abney (2005) extended the work of Yarowsky and Ngai (2001) by projecting from multiple source languages into a target language to filter out the systematic transfer errors that arise from differing source languages. Agić et al. (2015) also combined annotation projection with multi-source transfer to learn POS taggers for 100 languages.

Using a tag dictionary has proven to be helpful for learning POS taggers (Smith and Eisner, 2005; Goldberg et al., 2008; Ravi and Knight, 2009). This is because words have a limited number of possible tags. A dictionary that constrains the allowable tags for a word helps restrict the search space. However, these approaches rely on tag dictionaries which are extracted directly from the underlying treebank data. Such dictionaries has a broad coverage of the test domain which is difficult to obtain for resource-poor languages.

Das and Petrov (2011) used graph-based label propagation to project syntactic information from one language to another. In their approach, the graph has two kinds of vertices; a trigram type on the target language side, and individual word types on the English side. The edge weights are the similarity between the middle words of the trigram types. In the first stage, some vertices received a label from direct label projection, and then labels were propagated to the rest of the graph. The projected labels were treated as features in an unsupervised model (Berg-Kirkpatrick et al., 2010). Tagging dictionaries were extracted from the graph and were used as the constraints for a feature-based HMM tagger (Berg-Kirkpatrick et al., 2010).

Täckström et al. (2013) combined token constraints (from direct projected data) and type constraints (from Wiktionary's dictionary) to induce multilingual POS taggers for resource-poor languages. For each sentence in the training data, they constructed a lattice and used these token and type constraints to prune it. The best tagging accuracies were obtained with a partially observed CRF model that integrates these complementary constraints.

Li et al. (2012) employed tag dictionaries from the Wiktionary. They incorporated type constraints from Wiktionary into the feature-based HMM of (Berg-Kirkpatrick et al., 2010).

An alternative approach for tagging low-resource languages is to assume a small corpus of manually annotated data is available. Xi and Hwa (2005) proposed an approach inspired by backoff language modeling techniques in which the parameters of their two POS tagging models (one trained on manually annotated data and the other one trained on projected data) are combined to achieve a higher quality final model. Their best results were obtained when 3000 manually annotated tokens (approximately 100 sentences) were used.

Duong et al. (2014) proposed an approach that takes advantage of parallel data to do the cross-lingual projection and a small amount of annotated data, (1000 tokens) in the target language, to learn to correct the errors from the projected approach such as tagset mismatch between the two languages. Their model was trained in two stages: First, a maximum entropy classifier T was trained on the noisy projected data. A supervised classifier P was then trained on the small annotated data, using a minimum divergence technique to incorporate the first model (T).

Fang and Cohn (2016) presented a model that learns when and how much to trust the distant supervision in the cross-lingual projection. They used parallel data to obtain projected tags as distant annotations. They proposed a joint BiLSTM model trained on the distant data and 1000 gold annotated data. Their model learns to correct the errors from cross-lingual projection using an explicit debiasing layer.

Our approach follows Yarowsky and Ngai's (2001) original annotation projection approach. However, we used our proposed neural HMM aligners to perform word alignment. Moreover, we leverage a neural sequence tagging model (Rei and Yannakoudakis, 2016; Rei et al., 2016) to tag the English side, and to train POS taggers on the projected data. We learn POS taggers for truly low-resource languages for which no training data is available in the universal dependencies treebanks.

6.8 Conclusion

We have presented a method for building part-of-speech taggers for resource-poor languages. We considered the zero-resource scenario and did not use any additional information. The only assumption is the availability of a parallel data between the resource-poor language under study and a resource-rich language. Our approach performs cross-lingual annota-

tion projection using our proposed neural HMM word aligner. We used high quality neural sequence taggers for tagging the source side of the parallel. Our results show the overall superiority of our approach over unsupervised neural HMM POS taggers baseline, supervised BiLSTM-CRF tagger trained on a small amount of annotated data, cross-lingual projection methods that use GIZA++ alignments and cross-lingual projection methods that use alignments extracted from attention-based NMT, for low-resource languages. The best results were obtained with the model that used NHMM+BERT aligner and applied noise robust techniques.

Chapter 7

Conclusion

In this thesis, we have investigated new models for the unsupervised word alignment. We introduced new probabilistic models for augmenting word alignments with linguistically motivated alignment types. We proposed a novel task of joint prediction of word alignment and alignment types. we proposed two models, a generative and a discriminative one, to solve the joint prediction task. We augmented the classic generative IBM Model1 and HMM-based word alignment model with alignment types. Our generative models outperform their corresponding baseline. We showed that the proposed joint model (alignment-type-enhanced model) improved word alignment and translation quality.

We presented an unsupervised neural HMM for word alignment. Our proposed neural HMM consists of neural-network based emission and transition models which are trained jointly. This modelling allows us to introduce additional dependencies into each of these models. We investigated the effect of incorporating BERT representation into our model in order to include the full target context for the emission model. We demonstrated improvements over GIZA++ IBM4 model, which is still a strong baseline, on Romanian-English and Chinese-English datasets.

We proposed a method that induces part-of-speech taggers for low-resource languages using cross-lingual tag projection. The proposed method relies on our proposed neuralHMM aligners to obtain high quality word alignments between low-resource languages and the high-resource language. Moreover, high quality neural POS taggers were used to provide annotations for the resource-rich language side of the parallel data, as well as to train a tagger on the projected data. Our experimental results on truly low-resource languages showed that our methods consistently outperformed their corresponding baselines.

7.1 Future Work

There are several research directions that could be explored in the future:

- **Latent alignment types:** For the joint prediction of word alignment with alignment types, we would like to further extend this work to a case where alignment types are latent i.e. there is no training data for alignment types.

- **Improving attention for RNN-based NMT**: Word alignment is still essential for the analysis of translation errors in the attention-based neural machine translation systems. It has been shown to be helpful in guiding NMT models during training and improving NMT decoding. Our proposed Neural HMM aligner can be used in the attention-based NMT to improve attention.

- **Improving transformer-based attention:** Alignment-enhanced transformer methods have been proposed recently. These methods use FastAlign or GIZA++ to obtain word alignments which can be used for the supervision during the training. We would like to leverage our proposed aligner in the transformer-based NMT instead. As better alignment leads to better constrained translation, we would like to investigate whether our aligner can be helpful in improving the transformer-based NMT.

- **Projection-based transfer approach for tasks other than POS tagging:** We have used our neural HMM aligner for building POS taggers for low-resource languages. Cross-lingual dataset creation using our aligner can be explored for a variety of tasks in NLP beyond part-of-speech tagging, including named-entity recognition (NER), parsing, information extraction (IE) and semantic role labeling.

Bibliography

Željko Agić, Dirk Hovy, and Anders Søgaard. If all you have is a bit of the bible: Learning pos taggers for truly low-resource languages. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 268–272, 2015.

Tamer Alkhouli and Hermann Ney. Biasing attention-based recurrent neural networks using external alignment information. In *Proceedings of the Second Conference on Machine Translation*, pages 108–117, 2017.

Tamer Alkhouli, Gabriel Bretschner, Jan-Thorsten Peter, Mohammed Hethnawi, Andreas Guta, and Hermann Ney. Alignment-based neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 54–65, 2016.

Tamer Alkhouli, Gabriel Bretschner, and Hermann Ney. On the alignment problem in multi-head attention-based neural machine translation. *arXiv preprint arXiv:1809.03985*, 2018.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Leonard E Baum. An equality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3:1–8, 1972.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 582–590, 2010.

Phil Blunsom and Trevor Cohn. Discriminative word alignment with conditional random fields. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (ACL)*, pages 65–72, 2006.

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.

Jan Buys and Jan A Botha. Cross-lingual morphological tagging for low-resource languages. *arXiv preprint arXiv:1606.04279*, 2016.

Chris Callison-Burch, David Talbot, and Miles Osborne. Statistical machine translation with word-and sentence-aligned parallel corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL)*, page 175, 2004.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. Guided alignment training for topic-aware neural machine translation. *arXiv preprint arXiv:1607.01628*, 2016.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL `https://www.aclweb.org/anthology/D14-1179`.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584, Cambridge, MA, October 2010. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D10-1056`.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 1693–1703, 2016.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers (ACL-HLT)*, pages 176–181, 2011.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technology (NAACL-HLT)*, pages 876–885, 2016.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2011.

Dipanjan Das and Slav Petrov. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P11-1061`.

Marie-Catherine De Marneffe and Christopher D Manning. The stanford typed dependencies representation. In *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*, pages 1–8, 2008.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. Generating typed dependency parses from phrase structure parses. In *Lrec*, volume 6, pages 449–454, 2006.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. Universal stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14, pages 4585–4592, 2014.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

John DeNero and Dan Klein. Discriminative modeling of extraction sets for machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1453–1463, 2010.

Yonggang Deng and William Byrne. Hmm word and phrase alignment for statistical machine translation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(3):494–507, 2008.

Yonggang Deng and Yuqing Gao. Guiding statistical word alignment models with prior knowledge. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 1–8, 2007.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1159, 2017.

Philipp Dufter, Mengjie Zhao, Martin Schmitt, Alexander Fraser, and Hinrich Schütze. Embedding learning through multilingual concept induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1520–1530, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1141. URL `https://www.aclweb.org/anthology/P18-1141`.

Long Duong, Trevor Cohn, Karin Verspoor, Steven Bird, and Paul Cook. What can we get from 1000 tokens? a case study of multilingual POS tagging for resource-poor languages. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 886–897, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1096. URL `https://www.aclweb.org/anthology/D14-1096`.

Chris Dyer, Jonathan Clark, Alon Lavie, and Noah A. Smith. Unsupervised word alignment with arbitrary features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 409–419, 2011.

Meng Fang and Trevor Cohn. Learning when to trust distant supervision: An application to low-resource POS tagging using cross-lingual projection. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 178–186, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1018. URL https://www.aclweb.org/anthology/K16-1018.

Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 462–471. The Association for Computer Linguistics, 2014a.

Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, 2014b.

Victoria Fossum and Steven Abney. Automatically inducing a part-of-speech tagger by projecting from multiple source languages across aligned corpora. In *Second International Joint Conference on Natural Language Processing: Full Papers*, 2005. doi: 10.1007/11562214_75. URL https://www.aclweb.org/anthology/I05-1075.

Alexander Fraser and Daniel Marcu. Semi-supervised training for statistical word alignment. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 769–776, 2006.

Alexander Fraser and Daniel Marcu. Getting the structure right for word alignment: LEAF. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 51–60. Association for Computational Linguistics, 2007a.

Alexander Fraser and Daniel Marcu. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303, 2007b.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N13-1092.

Qin Gao, Nguyen Bach, and Stephan Vogel. A semi-supervised word alignment algorithm with partial manual alignments. In *Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR*, pages 1–10, 2010a.

Qin Gao, Francisco Guzman, and Stephan Vogel. EMDC: a semi-supervised approach for word alignment. In *Proceedings of the 23rd International Conference on Computational Linguistics (ACL)*, pages 349–357, 2010b.

Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy, and Matthias Paulik. Jointly learning to align and translate with transformer models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, page 4453–4462, Hong Kong, China, November 2019. Association for Computational Linguistics.

Jean-Luc Gauvain and Chin-Hui Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *Speech and audio processing, ieee transactions on*, 2(2):291–298, 1994.

Yoav Goldberg, Meni Adler, and Michael Elhadad. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of ACL-08: HLT*, pages 746–754, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P08-1085`.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244, 2015.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. A representation learning framework for multi-source transfer parsing. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

Xiaodong He. Using word dependent transition models in hmm based word alignment for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 80–87. Association for Computational Linguistics, 2007.

Kenneth Heafield. KenLM: Faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation (WMT)*, pages 187–197, 2011.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015. URL `http://arxiv.org/abs/1508.01991`.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325, 2005.

Abraham Ittycheriah and Salim Roukos. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 89–96, 2005.

Mark Johnson. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D07-1031`.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D13-1176`.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In *AAAI*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, 2015. URL `http://arxiv.org/abs/1412.6980`.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P17-4012`.

Reinhard Kneser and Hermann Ney. Forming word classes by statistical clustering for statistical language modelling. In *Contributions to Quantitative Linguistics*, pages 221–226. Springer, 1993.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer, 2005.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions (ACL)*, pages 177–180, 2007.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1121. URL `https://www.aclweb.org/anthology/N16-1121`.

Joël Legrand, Michael Auli, and Ronan Collobert. Neural network-based word alignment through score aggregation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 66–73, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2207. URL `https://www.aclweb.org/anthology/W16-2207`.

Abby Levenberg, Chris Callison-Burch, and Miles Osborne. Stream-based translation models for statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 394–402, 2010.

Shen Li, João Graça, and Ben Taskar. Wiki-ly supervised part-of-speech tagging. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1389–1398, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D12-1127.

Xintong Li, Guanlin Li, Lemao Liu, Max Meng, and Shuming Shi. On the word alignment from neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1293–1303, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1124. URL https://www.aclweb.org/anthology/P19-1124.

Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pages 104–111, 2006.

Pierre Lison, Jörg Tiedemann, Milen Kouylekov, et al. Open subtitles 2018: Statistical rescoring of sentence alignments in large, noisy parallel corpora. In *LREC 2018, Eleventh International Conference on Language Resources and Evaluation*. European Language Resources Association (ELRA), 2019.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. Neural machine translation with supervised attention. *arXiv preprint arXiv:1609.04186*, 2016.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 250–256, 2015.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL https://www.aclweb.org/anthology/D15-1166.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: annotating predicate argument structure. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

Ryan McDonald, Slav Petrov, and Keith Hall. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 62–72, 2011.

Rada Mihalcea and Ted Pedersen. An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10, 2003. URL https://www.aclweb.org/anthology/W03-0301.

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758, 2012. URL `http://dblp.uni-trier.de/db/conf/icml/icml2012.html#MnihT12`.

Robert C. Moore, Wen-tau Yih, and Andreas Bode. Improved discriminative bilingual word alignment. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 513–520, 2006.

Ahn-Khoa Ngo-Ho and François Yvon. Neural baselines for word alignment. In *International Workshop on Spoken Language Translation*, Hong Kong, China, November 2019.

Jan Niehues and Stephan Vogel. Discriminative word alignment via alignment matrix modeling. In *Proceedings of the 3rd ACL Workshop on Statistical Machine Translation*, pages 18–25, 2008.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, 2016.

Joakim Nivre, Mitchell Abrams, and Željko Agić et al. Universal dependencies 2.3, 2018. URL `http://hdl.handle.net/11234/1-2895`. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*, pages 160–167, 2003.

Franz Josef Och and Hermann Ney. A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 1086–1090. Association for Computational Linguistics, 2000a.

Franz Josef Och and Hermann Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 440–447. Association for Computational Linguistics, 2000b.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51, 2003.

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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://www.aclweb.org/anthology/P02-1040`.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English Gigaword Fifth Edition, Linguistic Data Consortium. Technical report, Linguistic Data Consortium, 2011.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/274_Paper.pdf.

Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Sujith Ravi and Kevin Knight. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 504–512, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P09-1057.

Marek Rei and Helen Yannakoudakis. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1181–1191, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1112. URL https://www.aclweb.org/anthology/P16-1112.

Marek Rei, Gamal Crichton, and Sampo Pyysalo. Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 309–318, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1030.

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D07-1043.

Ruslan Salakhutdinov, Sam T Roweis, and Zoubin Ghahramani. Optimization with em and expectation-conjugate-gradient. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 672–679, 2003.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997. ISSN 1053-587X. doi: 10.1109/78.650093. URL http://dx.doi.org/10.1109/78.650093.

Noah A Smith and Jason Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362, 2005.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200. Cambridge, MA, 2006.

Milan Straka and Jana Straková. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, 2017.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112, 2014.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12, 2013. doi: 10.1162/tacl_a_00205. URL `https://www.aclweb.org/anthology/Q13-1001`.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. Recurrent neural networks for word alignment model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1470–1480, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1138. URL `https://www.aclweb.org/anthology/P14-1138`.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. A discriminative matching approach to word alignment. In *Proceedings of the 2005 Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 73–80, 2005.

Jörg Tiedemann. Parallel data, tools and interfaces in opus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Kristina Toutanova, H Tolga Ilhan, and Christopher D Manning. Extensions to hmm-based statistical word alignment models. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 87–94. Association for Computational Linguistics, 2002.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)*, pages 173–180, 2003.

Ke Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. Unsupervised neural hidden markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71, 2016.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International*

Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1, pages 477–485. Association for Computational Linguistics, 2009.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1008. URL `https://www.aclweb.org/anthology/P16-1008`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Andrew J Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, 1967.

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*, pages 836–841. Association for Computational Linguistics, 1996.

Weiyue Wang, Tamer Alkhouli, Derui Zhu, and Hermann Ney. Hybrid neural network alignment and lexicon model in direct HMM for statistical machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 125–131, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2020. URL `https://www.aclweb.org/anthology/P17-2020`.

Xiaolin Wang, Masao Utiyama, Andrew M. Finch, and Eiichiro Sumita. Refining word segmentation using a manually aligned corpus for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1654–1664, 2014.

Chenhai Xi and Rebecca Hwa. A backoff model for bootstrapping resources for non-English languages. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 851–858, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/H05-1107`.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. Word alignment modeling with context dependent deep neural network. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 166–175, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P13-1017`.

David Yarowsky and Grace Ngai. Inducing multilingual pos-taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies, NAACL '01*, pages 1–8. Association for Computational Linguistics, 2001.

Alexander Yeh. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational linguistics (COLING)*, pages 947–953, 2000.

Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL `http://dblp.uni-trier.de/db/journals/corr/corr1212.html#abs-1212-5701`.

Thomas Zenkel, Joern Wuebker, and John DeNero. Adding interpretable attention to neural translation models improves word alignment. *arXiv preprint arXiv:1901.11359*, 2019.

Thomas Zenkel, Joern Wuebker, and John DeNero. End-to-end neural word alignment outperforms giza++. *arXiv preprint arXiv:2004.14675*, 2020.

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*, pages 596–605. Association for Computational Linguistics, 2010.

Zhandos Zhumanov, Aigerim Madiyeva, and Diana Rakhimova. New kazakh parallel text corpora with on-line access. In *International Conference on Computational Collective Intelligence*, pages 501–508. Springer, 2017.