Feature-Rich Unsupervised Word Alignment Models

Guido M. Linders
10527605

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dr. Wilker Ferreira Aziz

Institute for Logic, Language and Computation (ILLC)
Faculty of Science
University of Amsterdam
Science Park 107
1098 XG Amsterdam

June 24th, 2016

Abstract

Brown et al. (1993) introduced five unsupervised, word-based, generative and statistical models, popularized as IBM models, for translating a sentence into another. These models introduce alignments which maps each word in the source language to a word in the target language. In these models there is a crucial independence assumption that all lexical entries are seen independently of one another. We hypothesize that this independence assumption might be too strong, especially for languages with a large vocabulary, for example because of rich morphology. We investigate this independence assumption by implementing IBM models 1 and 2, the least complex IBM models, and also implementing a feature-rich version of these models. Through features, similarities between lexical entries in syntax and possibly even meaning can be captured. This feature-richness, however, requires a change in parameterization of the IBM model. We follow the approach of Berg-Kirkpatrick et al. (2010) and parameterize our IBM model with a log-linear parametric form. Finally, we compare the IBM models with their log-linear variants on word alignment. We evaluate our models on the quality of word alignments with two languages with a richer vocabulary than English. Our results do not fully support our hypothesis yet, but they are promising. We believe the hypothesis can be confirmed, however, there are still many technical challenges left before the log-linear variants can become competitive with the IBM models in terms of quality and speed.

Contents

1 Introduction

Machine translation (MT) is a sub-area of natural language processing (NLP) concerned with automated translations of text from a *source* language into a *target* language. MT dates back to 1950s during the cold war between the United States and the former USSR (Locke and Booth, 1955). For a historical survey of MT we refer the reader to Hutchins (2007).

Modern MT is mostly studied under the paradigm of statistical learning. *Statistical machine translation* (SMT) is a data-driven approach based on statistical estimation of models from examples of human-made translations. These examples constitute bilingual *parallel corpora*. The techniques relevant are many and there is more than 20 years of active research since the first statistical approaches to MT (Brown et al., 1993). Thus, in this work we will not survey SMT extensively, instead we refer the reader to Lopez (2008).

Brown et al. (1993) introduced five statistical models for translating a sentence into another language. These models are called word-based, since they perform translation word by word. They were originally introduced as fully fledged translation models, but due to their strong assumptions, particularly that translation can be performed word by word. Nowadays they are no longer used as translation models, but rather as *word alignment models*.

In a parallel corpus we say that sentence pairs are examples of sentence-level *translation equivalence*. That is because at the sentence-level our observations (sentence pairs) are examples of meaning equivalence expressed in two different codes (languages). Word alignment models break down translation equivalence to units smaller the are more amenable to statistical methods. Much of state-of-the-art MT research still relies heavily on good word alignment models as well as extensions that account for alignments of phrases and tree fragments. This includes: phrase-based SMT (Koehn et al., 2003), hierarchical phrase-based SMT (Chiang, 2005), syntax-based SMT (DeNeefe and Knight, 2009), graph-based SMT (Jones et al., 2012), and many other approaches. Word alignment models also contribute to research on other applications, such as statistical morphological analysis and parsing (Snyder and Barzilay, 2008; Das and Petrov, 2011; Kozhevnikov and Titov, 2013; Daiber and Sima'an, 2015), where automatically word-aligned parallel corpora are used to transfer resources from a resource-rich language (e.g. English) to a resource-poor language (e.g. Hindi, Urdu, etc.).

In this research we focus on two of the IBM models, namely, IBM models 1 and 2 (Brown et al., 1993). We choose these two, because beyond model 2 inference is intractable, requiring sophisticated approximation, and because for many language pairs IBM model 2, and its variants in particular (Liang et al., 2006; Mermer and Saraçlar, 2011; Dyer et al., 2013), still perform really well.

IBM models 1 and 2 are instances of direct graphical models (Koller and Friedman, 2009). In particular, they are *generative conditional models*. We will present them in great detail in Chapter 2. For now it suffices to say that they learn how to reconstruct one side of the parallel corpus (which might be expressed in French for example) given the other side (which might be expressed in English). They do so by imposing probability distributions over events such as co-occurrence of word pairs within sentence pairs in a parallel corpus.

IBM models 1 and 2 perform particularly poorly on languages whose vocabulary is very large. Large vocabularies are typically the result of productive morphological processes that yields many inflected variants of a basic word form. In general we call languages like that *morphologically rich*.[1] Lexical alignment models, such as IBM models 1 and 2, are heavily

[1] The notion of morphologically rich languages is not fully specified. In this thesis we mean languages that are morphologically marked beyond English. That is, they mark morpho-syntactic properties and roles through variation of basic word forms.

lexicalized, that is, they largely condition on lexical events on one language in order to predict lexical events in another language. Lexical events are words as they appear in the parallel corpus with no special pre-processing. Thus lexical entries such as *work* and *works* are treated as completely unrelated events, even though intuitively they have a lot in common. Rich morphology takes this problem to an extreme where thousands of words can be related to a common root, but are expressed with unique strings. Morphological variants are usually obtained with affixes attached to the root and/or by compounding. All these processes lead to large vocabularies of related language events, which IBM models 1 and 2 ignore by giving them a *categorical* treatment.

In light of this discussion, in this thesis, the following research question is addressed:

"How is the quality of word alignment models influenced by "loosening" the assumption that lexical entries are independent of each other."

The goal of this thesis is to investigate the validity of the assumption that lexical entries are independent of one other. Linguistic knowledge and also intuition say this assumption is too strong. In practice, however, statistical models may still benefit from such simplifying assumptions due to feasibility of statistical inference and estimation. Our take on this is to attempt at showing evidence that word alignment models may benefit from a training regime where different lexical entries share similarities through a feature-rich representation. For this reason in our evaluation we include German as an example of a morphologically rich language.[2]

To summarize, in this study will only focus on IBM models 1 and 2, these being the least complex IBM models. The other IBM models are too computationally expensive, therefore the estimation of the parameters needs to be approximated (Och and Ney, 2003). Furthermore, we only look at the quality of the alignments, and not at the translations made by these models. That is so because a full integration with translation models go beyond the scope of this thesis.

Chapter 2 explains IBM models 1 and 2 and gives an overview of existing approaches to circumvent sparse vocabularies in these models. In Chapter 3, the approach taken in this study is explained. This Chapter explains the models that have been implemented and the evaluation methods that were applied. Chapter 4 describes the results and Chapter 5 concludes this thesis.

2 Theoretical Framework

This thesis builds upon methodology from data-driven statistical learning, particularly, statistical machine translation (SMT). One of the most successful approaches in SMT is based on the information-theoretical noisy-channel formulation which we shortly revisit in the following. The IBM models are also based on this approach and we will explain IBM models 1 and 2 thereafter.

2.1 Noisy-Channel Approach

In SMT we translate a sentence, expressed in a source language, into an equivalent sentence, expressed in a target language. In order to keep our presentation consistent with literature on SMT, we will call the source language *French* and the target language *English*, but note that the methods discussed here are mostly language independent and directly applicable to a wide

[2]Even though there are languages which are arguably morphologically richer (e.g. Czech, Arabic, Turkish) we believe German is a good starting point to illustrate the discussion. Also, the choice of languages was limited by time constraints.

range of language pairs. Of course, some of the independence assumptions we will discuss may be more or less adequate depending on the choice of language pair. Therefore we will explicitly note in the text if that is the case.

Let V_F be the vocabulary of French words, and let a French sentence be denoted by $f = \langle f_1, \ldots, f_m \rangle$, where each $f_j \in V_F$ and m is the length of the sentence. Similarly, let V_E be the vocabulary of English words, and let an English sentence be denoted by $e = \langle e_1, \ldots, e_l \rangle$, where each $e_i \in V_E$ and l is the length of the sentence. In the noisy-channel metaphor, we see translation as the result of encoding a message and passing it through a noisy medium (the *channel*). In this metaphor, translating from French into English is equivalent to learning how the channel distorts English messages into French counterparts and reverting that process. In statistical terms, translation is modeled as shown in Equation (1c).

$$\begin{aligned} e^* &= \arg\max_e P(e|f) && \text{(1a)} \\ &= \arg\max_e \frac{P(e)P(f|e)}{P(f)} && \text{(1b)} \\ &= \arg\max_e P(e)P(f|e) && \text{(1c)} \end{aligned}$$

Equation (1b) is the result of the application of the Bayes theorem, and Equation (1c) is obtained by discarding the denominator, which is constant over all possible English translations of a fixed French sentence. In other words, $P(e|f)$ is proportional to $P(e)P(f|e)$ and, therefore, the English sentence e^* that maximizes either expression is the same.

Learning probability distributions for $P(e)$ and $P(f|e)$ is a task called *parameter estimation*, and performing translation for previously estimated distributions is a task known as *decoding*. In Equation (1c), $P(e)$ is called *language model*, specifically, it expresses a probabilistic belief on what sequences of words make good (fluent and meaningful) English sentences. $P(f|e)$ is called *translation model*, it is the part that realizes possible mappings between English and French, expressing a probabilistic belief on what makes good (meaning preserving) translations. In the noisy-channel metaphor, $P(e)$ encodes a distribution over English messages and $P(f|e)$ accounts for how the medium distorts the original message.

In this thesis, we focus on alignment models, particularly, a subset of the family of models introduced by IBM researches in the 90s (Brown et al., 1993). These IBM models deal with the estimation of the translation model $P(f|e)$ from a dataset of examples of translations (a *parallel corpus*). Thus, we will not discuss about language modeling any further.

2.2 IBM Models 1 and 2

This section revisits IBM models 1 and 2 as introduced by Brown et al. (1993). First, we give an overview of the model's generative story. We then present specific parametric forms of the model, after which we explain how parameters are estimated from data. We conclude this section by explaining how predictions are made using these models.

2.2.1 Generative story

IBM models are generative conditional models where the goal is to estimate $P(f|e)$ from examples of translations in a parallel corpus. Being a conditional model means that we are interested in explaining how French sentences come about, without explicitly modeling English sentences. That is, IBM models assume that English sentences are given and fixed. In probabilistic modeling, a *generative story* describes the sequence of conditional independence

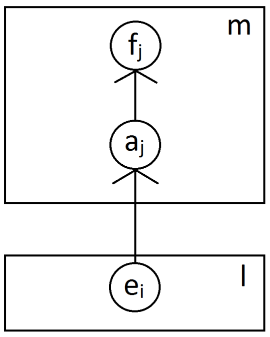


Figure 1: A graphical depiction of the generative story that describes the IBM models for a French sentence of length m and an English sentence of length l

assumptions that allows one to generate random variables (French sentences in this case) conditioned on other random variables (English sentences in this case). A central independence assumption made by IBM models 1 and 2 is that *each French word is independently generated by one English word*. In the following, this process is broken-down and explained step by step.

1. First we observe an English sentence $e = \langle e_1, \ldots, e_l \rangle$;
2. then, conditioned on l, we choose a length m for the French sentence;
3. then, for each French position $1 \leq j \leq m$,
   (a) we choose a position i in the English sentence which is aligned to j, here we introduce the concept of *word alignment*, which we denote by $a_j = i$;
   (b) conditioned on the alignment $a_j = i$ and the English word sitting on position i, we generate a French word f_j

In this generative story we introduced *alignment variables*, in particular, we introduced one such variable for each French word position, i.e. $a = \langle a_1, \ldots, a_m \rangle$. This makes learning an IBM model a case of learning from incomplete supervision. That is the case because given a set of English sentences, we can only observe their French translations in the parallel corpus. The alignment structure that maps each French word to its generating English word is *latent*. In probabilistic modeling, latent variables are introduced for modeling convenience. While we do expect these latent variables to correlate with a concept in the real world (for example, that of translation equivalence), they are introduced in order to enable simplifying conditional independence assumptions, which in turn enable tractable inference and estimation methods.

Finally, in order to account for words in French that typically do not align to any English word (e.g. certain functional words), we augment English sentences with a dummy NULL token. This NULL token is introduced at the beginning of every English sentence observation at position zero and is denoted e_0. Therefore an English sentence of l words will consist of the words $e_0, \ldots, e_l$.

Figure 1 is a graphical depiction of the generative story that underpins IBM models 1 and 2. Figure 2 shows an example of a possible translation of the sentence “La maison bleu”. The variable names have been replaced by words and for the sake of clarity the position of the word in the sentence is given. Note that there is no French word that is aligned to the NULL word in the English sentence.

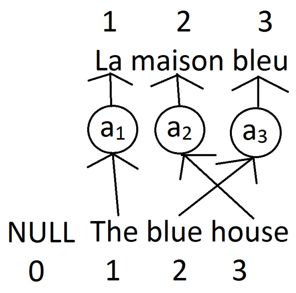


Figure 2: A directed graph that shows a possible alignment for the French sentence "La maison bleu" and the English sentence "The blue house", clarified with word positions.

As a function of latent assignments of the hidden word alignment variable, our translation model is now expressed as shown in Equation (2).

$$P(f|e) = \sum_{a \in \mathcal{A}} P(f, a|e) \tag{2}$$

In this equation, we denote by $\mathcal{A}$ the set of all possible alignments. In probability theory terminology, Equation (2) is a case of marginalization and we say we "marginalize over all possible latent alignments". As a modeler, our task is now to design the probability distribution $P(f, a|e)$ and for that we resort to the independence assumptions stated in the generative story.

Recall from the generative story, that alignment links are set independently of one another. That is, choosing an alignment link (e.g. $a_2 = 3$ in Figure 2) does not impact on the choices of any other alignment link (for example, $a_3 = 2$ in Figure 2). Moreover, generating a French word at position j only depends on the alignment link a_j, or in other words, French words are conditionally independent given alignment links. This crucial independence assumption implies that the probability $P(f, a|e)$ factors over individual alignment decisions for each $1 \leq j \leq m$. The joint probability assigned to a French sentence and a configuration of alignments given an English sentence is therefore expressed as shown in Equation (3c).

$$\begin{aligned} P(f, a|e) &= P(f_1, \ldots, f_m, a_1, \ldots, a_m | e_0, \ldots, e_l) && \text{(3a)} \\ &= P(m|l) \prod_{j=1}^{m} P(f_j, a_j | e, m, l) && \text{(3b)} \\ &= P(m|l) \prod_{j=1}^{m} P(f_j | e_{a_j}) \times P(a_j | j, m, l) && \text{(3c)} \end{aligned}$$

In Equation (3b), we take into account the independence assumption over French positions and their alignments links. The term $P(m|l)$ expresses the fact that we choose the length m of the French sentence conditioned on the length l of the English sentence. In Equation (3c), we take into account the conditional independence of French words given their respective alignment links. We denote by e_{a_j} the English word at position a_j. Hence, this is the English word which is aligned to the French word f_j.

In the next sections, we choose parametric families for the distributions in (3c) and explain parameter estimation.

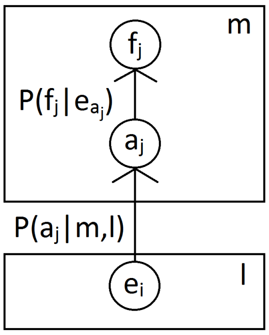


Figure 3: A graphical representation that shows the factorization of the IBM models for a French sentence of length m and an English sentence of length l, with added lexical and alignment distributions next to their corresponding arrow.

2.2.2 Parameterization

In this section we describe the choices of parametric distributions that specify IBM models 1 and 2. First, we note that in these models, $P(m|l)$ is assumed uniform and therefore is not directly estimated. For this reason we largely omit this term from the presentation.

The generative story of IBM models 1 and 2 implies that the joint probability distribution $P(f_j, a_j|e, m, l)$ is further decomposed as an alignment decision (e.g. a choice of a_j) followed by a lexical decision (e.g. a choice of f_j conditioned on e_{a_j}). In statistical terms, this is the product of two probability distributions as shown in Equation (4).[3]

$$P(f_j, a_j|e, m, l) = P(f_j|e_{a_j}) \times P(a_j|j, m, l) \tag{4}$$

We will refer to $P(f_j|e_{a_j})$ as a *lexical distribution* and to $P(a_j|j, m, l)$ as an *alignment distribution*. For each French position j, the alignment distribution picks a generating component, that is, a position a_j in the English sentence. Then the lexical distribution generates the French word that occupies position j, given the English word e_{a_j}, that is, the English word that occupies a_j.

This is visualized in Figure 3. Recall that the alignment distribution only looks at positions and not at the words at these positions. The difference between IBM models 1 and 2 is the choice of parameterization of the alignment distribution. The lexical distribution is parameterized in the same way in both models.

In order to introduce the parametric forms of the distributions in (4), we need first to introduce additional notation. Let c represent a conditioning context (e.g. the English word e_{a_j} sitting at position a_j), and let d represent a decision (e.g. the French word f_j sitting at position j). Then, a *categorical distribution* over the space of possible decisions $\mathcal{D}$ is defined as shown in Equation (5).

$$\text{Cat}(d; \boldsymbol{\theta}_c) = \theta_{c,d} \tag{5}$$

In this equation, c belongs to the space of possible contexts $\mathcal{C}$ (e.g. all possible English words). The vector $\boldsymbol{\theta}_c$ is called a vector of *parameters* and it is such that $0 \leq \theta_{c,d} \leq 1$ for every decision d and $\sum_{d \in \mathcal{D}} \theta_{c,d} = 1$. The parameters of a categorical distribution can be seen as specifying

[3]These distributions are also referred to as *generative components*. These components are in fact probability distributions. Because of that, a model such as IBM models 1 and 2 is also known as a *locally normalized model*. In probabilistic graphical modeling literature, such models are also known as *Bayesian networks* (Koller and Friedman, 2009).

conditional probabilities, that is, the probability of decision d given context c. A categorical distribution is therefore convenient for modeling conditional probability distributions such as the lexical distribution and the alignment distribution.

The lexical distribution is parameterized as a set of categorical distributions, one for each word in the English vocabulary, each defined over the entire French vocabulary.

$$\begin{aligned} P(\mathsf{f}|\mathsf{e}) &= \mathrm{Cat}(\mathsf{f}; \boldsymbol{\lambda}_{\mathsf{e}}) \\ &\forall (\mathsf{f}, \mathsf{e}) \in V_F \times V_E \end{aligned} \tag{6}$$

Equation (6) shows this parameterization, where $\boldsymbol{\lambda}$ is a vector of parameters. Note that we have one such vector for each English word and each vector contains as many parameters as there are French words. Thus, if v_E denotes the size of the English vocabulary, and v_F denotes the size of the French vocabulary, we need $v_E \times v_F$ parameters to fully specify the lexical distribution. Note that the parameters of a categorical distribution are completely independent of each other (provided they sum to 1). Thus, by this choice of parameterization, we implicitly make another independence assumption, namely, that all lexical entries are independent language events. This independence assumption might turn out too strong, particularly for languages whose words are highly inflected. We will revisit and loosen this independence assumption in Chapter 3.

For IBM model 1 the alignment distribution (Equation (7)) is *uniform* over the English sentence length (augmented with the NULL word).

$$P(i|j, l, m) = \frac{1}{l+1} \tag{7}$$

This means that each assignment $a_j = i$ for $0 \leq i \leq l$ is equally likely. The implication of this decision is that only the lexical parameters influences the translation as all English positions are equally likely to be linked to the French position.

Incorporating the independence assumptions in Equation (3c) and the choice of parameterization for IBM model 1, the joint probability of a French sentence and alignment is shown in Equation (8c).

$$P(f, a|e) = \prod_{j=1}^{m} \frac{1}{l+1} P(f_j|e_{a_j}) \tag{8a}$$

$$= \frac{1}{(l+1)^m} \prod_{j=1}^{m} P(f_j|e_{a_j}) \tag{8b}$$

$$= \frac{1}{(l+1)^m} \prod_{j=1}^{m} \lambda_{e_{a_j}, f_j} \tag{8c}$$

To obtain $P(f|e)$, we need to marginalize over all possible alignments as shown in Equation (9c).

$$P(f|e) = \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \frac{1}{(l+1)^m} \prod_{j=1}^{m} P(f_j|e_{a_j}) \tag{9a}$$

$$= \frac{1}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} P(f_j|e_i) \tag{9b}$$

$$= \frac{1}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} \lambda_{e_i, f_j} \tag{9c}$$

Before moving on to parameter estimation, we first present the parameterization of IBM model 2. Originally, the alignment distribution in IBM model 2 is defined as a set of categorical distributions, one for each (j, l, m) tuple. Each categorical distribution is defined over all observable values of i. Equation (10) shows the definition, where $\boldsymbol{\delta}_{j,l,m} = \langle \delta_0, \dots, \delta_L \rangle$ and L is the maximum possible length for an English sentence.

$$P(i|j, l, m) = \mathrm{Cat}(i; \boldsymbol{\delta}_{j,l,m}) \tag{10}$$

This choice of parameterization leads to very sparse distributions. If M is the maximum length of a French sentence, there would be as many as $M \times L \times M$ distributions each defined over as many as $L + 1$ values, thus as many as $L^2 \times M^2$ parameters to be estimated. To circumvent this sparsity problem, in this study, we opt to follow an alternative parameterization of the alignment distribution proposed by Vogel et al. (1996). This reparameterization is shown in Equation (11), where $\boldsymbol{\gamma} = \langle \gamma_{-L}, \dots, \gamma_L \rangle$ is a vector of parameters called "jump probabilities".

$$P(i|j, l, m) = \mathrm{Cat}(\mathrm{jump}(i, j, l, m); \boldsymbol{\gamma}) \tag{11}$$

A *jump* quantifies a notion of mismatch in linear order between French and English and is defined as in Equation (12).

$$\mathrm{jump}(i, j, l, m) = i - \left\lfloor \frac{j \cdot l}{m} \right\rfloor \tag{12}$$

The categorical distribution in (11) is defined for jumps ranging from $-L$ to L.[4] Compared to the original parameterization, Equation (11) leads to a very small number of parameters, namely, $2 \times L + 1$, as opposed to $L^2 \times M^2$.

Equation (13b) shows the joint distribution for IBM model 2 under our choice of parameterization.

$$\begin{aligned} P(f, a|e) &= \prod_{j=1}^{m} P(f_j|e_{a_j}) P(a_j|j, l, m) && \text{(13a)} \\ &= \prod_{j=1}^{m} \lambda_{e_{a_j}, f_j} \times \gamma_{\mathrm{jump}(a_j, j, l, m)} && \text{(13b)} \end{aligned}$$

By marginalizing over all alignments we obtain $P(f|e)$ for IBM model 2 as shown in Equation (14c):

$$\begin{aligned} P(f|e) &= \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} P(f_j|e_{a_j}) P(a_j = i|j, l, m) && \text{(14a)} \\ &= \prod_{j=1}^{m} \sum_{i=0}^{l} P(f_j|e_i) P(a_j = i|j, l, m) && \text{(14b)} \\ &= \prod_{j=1}^{m} \sum_{i=0}^{l} \lambda_{e_i, f_j} \times \gamma_{\mathrm{jump}(i, j, l, m)} && \text{(14c)} \end{aligned}$$

This concludes the presentation of the design choices behind IBM model 1 and 2, we now turn to parameter estimation.

[4] If $m = M$, l equal to the shortest length of the English sentence and j the last word in the French sentence, $\frac{j \cdot l}{m}$ approximates 0, the floor of this is then 0 and if i is the last word in the English sentence, this is the maximum English sentence length in the corpus L and thus the maximum value for the jump. The other way around, if j and m are equal and $l = L$, l is approximated, which should be subtracted from i. If $i = 0$, the smallest value for the jump is $-L$.

2.2.3 Parameter Estimation

Parameters are estimated from data by the principle of *maximum likelihood.* That is, we choose parameters as to maximize the probability the model assigns to a set of observations. This principle is generally known as maximum likelihood estimation (MLE). In our case, observations are sentence pairs, naturally occurring in an English-French parallel corpus. Let $\boldsymbol{\theta}$ collectively refer to all parameters in our model (i.e. lexical parameters and jump parameters), and let $D = \langle (f^{(1)}, e^{(1)}), \ldots, (f^{(N)}, e^{(N)}) \rangle$ represent a parallel corpus of N independent and identically distributed (i.i.d.) sentence pairs. The maximum likelihood estimate $\boldsymbol{\theta}^{\text{MLE}}$ is the solution to Equation (15d), where $\boldsymbol{\Theta}$ is the space of all possible parameter configurations.

$$\begin{aligned}
\boldsymbol{\theta}^{\text{MLE}} &= \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\max} \prod_{s=1}^{N} P(f^{(s)}|e^{(s)}) && \text{(15a)} \\
&= \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\max} \prod_{s=1}^{N} \prod_{j=1}^{m^{(s)}} P(f_j^{(s)}|e_{a_j}^{(s)}) && \text{(15b)} \\
&= \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\max} \prod_{s=1}^{N} \prod_{j=1}^{m^{(s)}} \sum_{a \in \mathcal{A}^{(s)}} P(f^{(s)}, a|e^{(s)}, l^{(s)}, m^{(s)}) && \text{(15c)} \\
&= \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\max} \prod_{s=1}^{N} \prod_{j=1}^{m^{(s)}} \sum_{i=0}^{l^{(s)}} P(f_j^{(s)}|e_i^{(s)}) \times P(i|j, l^{(s)}, m^{(s)}) && \text{(15d)}
\end{aligned}$$

Recall that he objective of the model is to maximize the probability assigned to the set of observations. For this reason we take the product over all observations and maximize this probability as is shown in Equation (15a). Because of the assumption that all French words are independently generated from each other, given the alignment links, we can take the product over the probabilities of each French word, given the English word it is aligned. This is shown in Equation (15b). In Equation (15c) introduce the alignment variables. We are marginalizing over all possible alignments. Finally, in Equation (15d) we substitute in the probability distributions as in Equation (4).

To simplify the optimization problem, we typically maximize the log-likelihood, rather than the likelihood directly. Note that because logarithm is a monotone function, the MLE solution remains unchanged. In this case the objective is shown in Equation (16). Further note that we can take the product out of the logarithm as shown in Equation (16b). In that case the product turns into a sum.

$$\begin{aligned}
\boldsymbol{\theta}^{\text{MLE}} &= \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\max} \log \prod_{s=1}^{N} \prod_{j=1}^{m^{(s)}} \sum_{i=0}^{l^{(s)}} P(f_j^{(s)}|e_i^{(s)}) \times P(i|j, l^{(s)}, m^{(s)}) && \text{(16a)} \\
&= \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\max} \sum_{s=1}^{N} \sum_{j=1}^{m^{(s)}} \log \sum_{i=0}^{l^{(s)}} P(f_j^{(s)}|e_i^{(s)}) \times P(i|j, l^{(s)}, m^{(s)}) && \text{(16b)}
\end{aligned}$$

At this point the fact that both the lexical distribution and the alignment distribution are parameterized by categorical distributions will become crucial. Suppose for a moment that we have complete data, that is, pretend for a moment that we can observe for each sentence pair the ideal assignment of the latent alignments. For fully observed data, the categorical distribution has a closed-form MLE solution, given in Equation (17).

$$\theta^{\text{MLE}}_{t,c,d} = \frac{n_a(t,c,d)}{\sum_{d' \in \mathcal{D}} n_a(t,c,d')} \tag{17}$$

In this equation, t selects a distribution type (for example, *lexical* or *alignment*), then c is a context for that type of distribution (for example, c is an English word in a lexical distribution), d is a decision and $\mathcal{D}$ is the space of possible decisions associated with t and c (for example, a word in the French vocabulary). The function $n_a(\text{event})$ counts the number of occurrences of a certain *event* in an observed alignment a (for example, how often the lexical entries *house* and *maison* are aligned).

Equation (18) formally defines n_a, where $\mathbb{I}[(a_j = i) \rightarrow \text{event}]$ is an indicator function which returns 1 if the alignment $a_j = i$ entails the event (t, c, d), and 0 otherwise. For example, in Figure 2, alignment link $a_3 = 2$ entails the event $(\text{lexical}, \text{blue}, \text{bleu})$, that is, the *lexical* entries *blue* (English) and *bleu* (French) aligned once.

$$n_a(t,c,d) = \sum_{a_j=i:j=1}^{m} \mathbb{I}[(a_j = i) \rightarrow (t,c,d)] \tag{18}$$

The MLE solution for complete data is thus as simple as counting the occurrences of each event, a (d, c, t) tuple, and re-normalizing these counts by the sum over all related events. Note that, because of the independence assumptions in the model, we can take the MLE for each distribution type independently. Moreover, because of the independence between parameters in a categorical distributions, we can compute the MLE solution for each parameter individually.

As it turns out, we do not actually have fully observed data, particularly, we are missing the alignments. Hence, we use our own model to hypothesize a distribution over completions of the data. This is an instance of the expectation-maximization (EM) optimization algorithm (Dempster et al., 1977). The EM solution to the MLE problem is shown in Equation (19). In this case, we are computing *expected counts* with respect to the *posterior distribution* over alignments $P(a|f, e; \boldsymbol{\theta})$ for a given configuration of the parameters $\boldsymbol{\theta}$.

$$\theta^{\text{MLE}}_{t,c,d} = \frac{\langle n_a(t,c,d)\rangle_{P(a|f,e;\boldsymbol{\theta})}}{\sum_{d' \in \mathcal{D}} \langle n_a(t,c,d')\rangle_{P(a|f,e;\boldsymbol{\theta})}} \tag{19}$$

Note that this equation is "cyclic", that is, we need an assignment of the parameters in order to compute posterior probabilities, which are then used to estimate new assignments. This is in fact an iterative procedure that converges to a local optimum of the likelihood function (Brown et al., 1993).

Equation (20) shows the posterior probability of an alignment configuration for a given sentence pair.

$$\begin{aligned} P(a|f,e) &= \frac{P(f,a|e)}{P(f|e)} && \text{(20a)} \\ &= \frac{\prod_{j=1}^{m} P(f_j|e_{a_j}) \times P(a_j|j,l,m)}{\prod_{j=1}^{m} \sum_{i=0}^{l} P(f_j|e_i) \times P(i|j,l,m)} && \text{(20b)} \\ &= \prod_{j=1}^{m} \frac{P(f_j|e_{a_j}) \times P(a_j|j,l,m)}{\sum_{i=0}^{l} P(f_j|e_i) \times P(i|j,l,m)} && \text{(20c)} \end{aligned}$$

From (20), we also note that the posterior factors as m independently normalized terms, that is, $P(a|f,e) = \prod_{j=1}^{m} P(a_j|f_j,e)$ where $P(a_j|f_j,e)$ is given in Equation (21).

$$P(a_j|f_j,e) = \frac{P(f_j,a_j|e)}{P(f_j|e)} \tag{21a}$$

$$= \frac{P(f_j|e_{a_j}) \times P(a_j|j,l,m)}{\sum_{i=0}^{l} P(f_j|e_i) \times P(i|j,l,m)} \tag{21b}$$

In computing maximum likelihood estimates via EM, we need to compute expected counts. These counts are a generalization of observed counts where each occurrence of an event in a hypothetical alignment a is weighted by the posterior probability $P(a|f,e)$ of such alignment. Equation (22) shows how expected counts are computed. Equation (22a) follows directly from the definition of expectation. In Equation (22b), we replace the counting function n_a(event) by its definition (18). In Equation (22c), we leverage the independence over alignment links to obtain a simpler expression.

$$\langle n(t,c,d)\rangle_{P(a|f,e;\boldsymbol{\theta})} = \sum_{a\in\mathcal{A}} n((t,c,d)\in a)P(a|f,e;\boldsymbol{\theta}) \tag{22a}$$

$$= \sum_{a\in\mathcal{A}} P(a|f,e;\boldsymbol{\theta}) \sum_{i=0}^{l}\sum_{j=1}^{m} \mathbb{I}[(a_j=i)\rightarrow(t,c,d)] \tag{22b}$$

$$= \sum_{i=0}^{l}\sum_{j=1}^{m} \mathbb{I}[(a_j=i)\rightarrow(t,c,d)]P(i|f_j,e;\boldsymbol{\theta}) \tag{22c}$$

We now turn to an algorithmic view of EM which is the basis of our own implementation of IBM models 1 and 2.

2.2.4 Expectation-Maximization Algorithm for IBM Models 1 and 2

The EM algorithm consists of two steps, which are iteratively alternated: an expectation (E) step and a maximization (M) step. In the E step we compute expected counts for lexical and alignment events according to Equation (22). In the M step new parameters are computed by re-normalizing the expected counts from the E step with Equation (19).

Before optimizing the parameters using EM, each distribution type should be initialized. IBM model 1 is convex, meaning that it has a only one optimum. This implies that any parameter initialization for IBM model 1 will eventually lead to the same global optimum. Because we do not have any heuristics on which word pairs occur frequently in the data, we initialize our lexical parameters uniformly.[5] As we want our lexical distribution to define a proper probability distribution, each categorical distribution over French words should sum to 1. Thus for all $\mathsf{f} \in V_F$ and all $\mathsf{e} \in V_E$, the parameter initialization for IBM model 1 is shown in Equation (23).

$$\lambda_{\mathsf{e},\mathsf{f}} = \frac{1}{v_F} \tag{23}$$

Recall that for IBM model 1 the alignment distribution is uniform. Thus, we fix it to $\frac{1}{l+1}$ for each English sentence length l. Note that the alignment distribution of IBM model 1 does not have parameters to be re-estimated with EM, see Equation (7).

[5]This is typically not a problem since IBM model 1 is rather simple and converges in a few iterations.

In case of IBM model 2, besides the lexical distribution, we also need to initialize the alignment distribution. As we also do not have any heuristics on which jumps occur frequently in the data, we initialize this distribution uniformly. Recall that we have $2 \times L + 1$ parameters for the alignment distribution, see Equations (11) and (12), where L is the maximum length of an English sentence. Each parameter $\gamma_{\text{jump}(i,j,l,m)}$ is therefore initialized as in Equation (24).

$$\gamma_{\text{jump}(i,j,l,m)} = \frac{1}{2 \times L + 1} \tag{24}$$

IBM model 2 is non-convex (Brown et al., 1993). This means EM can become stuck in a local optimum of the log-likelihood function and never approach the global optimum. In this case, initialization heuristics may become important. A very simple and effective heuristic is to first optimize IBM model 1 to convergence, and then initialize the lexical distribution of IBM model 2 with the optimized lexical distribution of IBM model 1. The initialization of the alignment distribution remains uniform as we have no better heuristic that is general enough.

After initialization an E step is computed. In the E step we are computing expected counts $\langle n_a(t, c, d) \rangle_{P(a|f,e;\boldsymbol{\theta})}$ for each decision, context and distribution type. The expected counts of each word pair are initialized to 0 at the beginning of the E step. We compute expected counts by looping over all sentence pairs. For each possible combination of a French word (decision) with an English word (context) in a sentence pair and for each distribution type t (only 1 distribution in case of IBM model 1: lexical distribution) we are adding the posterior probability of an alignment configuration to the expected counts of the word pair and distribution type. Recall that this posterior probability is defined in Equation (21).

After we obtained expected counts for each decision, context and distribution type, the E step is finished. New parameters are created from the expected counts in the M step. This is done by re-normalizing the expected counts over the sum of all the expected counts for each English lexical entry, so that our categorical distributions again sum to 1 (see equation (19)).

Algorithm 1 illustrates the parameter estimation using IBM model 2. Note that IBM model 1 is a special case of IBM model 2 where the alignment parameters are uniform. After the M step the algorithm is repeated for a predefined number of iterations.

We now turn to how predictions are made using these models.

2.2.5 Prediction

As previously discussed, IBM models are mostly used to infer word alignments, as opposed to produce translations. In a *translation task*, we would be given French data and we would have to predict English translations. In a *word alignment task*, we are given parallel data and we have to infer an optimal assignment of the latent word alignment between French words and their generating English contexts.

Note that the objective discussed in Section 2.2.3, and presented in Equation (15) is a *learning criterion*. That is, it is used for *parameter estimation* which is the task of learning a good model. The goodness of a model is defined in terms of it assign maximum likelihood to observed string pairs. In parameter estimation, even though we do infer alignments as part of EM, we are not inferring alignments as an end, our goal is to infer sets of parameters for our distributions (i.e. lexical distribution, alignment distribution).

At prediction time, we have a fixed set of parameters produced by EM which we use to infer alignment links. This is a task which is different in nature and for which the criterion of Section (15) is not applicable. Instead, here we devise a different criterion shown in Equation (25). This criterion, or *decision rule*, is known as *maximum a posteriori* (MAP) because

Algorithm 1 IBM model 2

```
$N \leftarrow$ number of sentence pairs
$I \leftarrow$ number of iterations
$\boldsymbol{\lambda} \leftarrow$ lexical parameters
$\boldsymbol{\gamma} \leftarrow$ alignment parameters

**for** $i \in [1, \dots, I]$ **do**
    **E step:**
    $n(\lambda_{\mathsf{e},\mathsf{f}}) \leftarrow 0$            $\forall (\mathsf{e}, \mathsf{f}) \in V_E \times V_F$
    $n(\gamma_x) \leftarrow 0$              $\forall x \in [-L, L]$
    **for** $s \in [1, \dots, N]$ **do**
        **for** $j \in [1, \dots, m^{(s)}]$ **do**
            **for** $i \in [0, \dots, l^{(s)}]$ **do**
                $x \leftarrow \text{jump}(i, j, l^{(s)}, m^{(s)})$
                $n(\lambda_{e_i,f_j}) \leftarrow n(\lambda_{e_i,f_j}) + \frac{\lambda_{e_i,f_j} \times \gamma_x}{\sum_{k=0}^{l} \lambda_{e_k,f_j} \times \gamma_{\text{jump}(k,j,l^{(s)},m^{(s)})}}$
                $n(\gamma_x) \leftarrow n(\gamma_x) + \frac{\lambda_{e_i,f_j} \times \gamma_{\text{jump}(i,j,l,m)}}{\sum_{k=1}^{l} \lambda_{e_k,f_j} \times \gamma_{\text{jump}(k,j,l^{(s)},m^{(s)})}}$
            **end for**
        **end for**
    **end for**

    **M step:**
    $\lambda_{\mathsf{e},\mathsf{f}} \leftarrow \frac{n(\lambda_{\mathsf{e},\mathsf{f}})}{\sum_{\mathsf{f}' \in V_F} n(\lambda_{\mathsf{e},\mathsf{f}'})}$        $\forall (\mathsf{e}, \mathsf{f}) \in V_E \times V_F$
    $\gamma_x \leftarrow \frac{n(\gamma_x)}{\sum_{x' \in [-L,L]} n(\gamma_{x'})}$        $\forall x \in [-L, L]$
**end for**
```

we select the alignment assignment that maximizes (25a) the posterior probability over latent alignments after observing data point (e, f). Equation (25b) is a consequence of the posterior factorizing independently over alignment links, as per Equation (20).

$$a^* = \arg\max_{a \in \mathcal{A}} P(a|f, e) \tag{25a}$$
$$= \arg\max_{a \in \mathcal{A}} \prod_{j=1}^{m} P(a_j|f_j, e) \tag{25b}$$

Because of the independence assumptions in the model, this maximization problem is equivalent to m independent maximization problems, one per alignment link, as shown in Equation (26a). In Equation (26b) we use the definition of the posterior per link (21). Equation (26c) is a consequence of the denominator in (26b) being constant for all assignments of i. Therefore, predicting the optimum alignment configuration a^* for a given sentence pair (e, f) for a fixed set of parameter values is equivalent to solving Equation (26c) for each French position.

$$a_j^* = \arg\max_{0 \leq i \leq l} P(a_j = i|f_j, e) \tag{26a}$$
$$= \arg\max_{0 \leq i \leq l} \frac{P(f_j|e_i) \times P(i|j, l, m)}{\sum_{k=0}^{l} P(f_j|e_k) \times P(k|j, l, m)} \tag{26b}$$
$$= \arg\max_{0 \leq i \leq l} P(f_j|e_i) \times P(i|j, l, m) \tag{26c}$$

Finally, Equation (27) shows this decision rule for IBM model 1, where we substituted in lexical parameters and omitted the uniform alignment probability (for it being constant). And Equation (28) shows this decision rule for IBM model 2, where we substituted in both the lexical and the jump parameters.

$$a_j^* = \arg\max_{0 \leq i \leq l} \lambda_{e_i, f_j} \tag{27}$$

$$a_j^* = \arg\max_{0 \leq i \leq l} \lambda_{e_i, f_j} \times \gamma_{\text{jump}(i,j,l,m)} \tag{28}$$

2.3 Improvements of IBM Models 1 and 2

In this section we survey a few improvements to IBM models 1 and 2. These improvements mostly deal with sparsity and over-parameterization of these models.

2.3.1 Sparsity and Over-parameterization

The assumption that all lexical entries are independent of each other becomes problematic in the presence of very large vocabularies. Moore (2004) identifies a few problems with IBM models 1 and 2. One of them, the "garbage collection" effect, is particularly prominent when dealing with large vocabularies. This problem is the result of a form of over-parameterization. Frequently occurring lexical entries are often present in the same sentences as rare lexical entries. Therefore a small probability is transferred from the frequent lexical entries to the rare ones. Moore (2004) addresses this problem by smoothing rare lexical entries. The effects of smoothing will be limited when there are too many rare lexical entries in the corpus, which is typical in morphologically rich languages.

Often, particular lexical entries cannot be aligned to any word in the English sentence. These words are typically motivated by syntactic requirements of French and do not correspond to any surface realization in English. According to IBM models, they should be aligned to the NULL word, however, due to the garbage collection effect, they end up aligning to frequent words instead. Moore (2004) addresses this problem by adding extra weight to the NULL word.

Another case of over-parameterization is the alignment distribution of IBM model 2. One alternative, which we took in this thesis, is to use the reparameterization of Vogel et al. (1996). This reparameterization is still categorical, which makes the jumps completely unrelated events. Intuitively, jumps of a given size must correlate to jumps of similar sizes. To address this issue, Dyer et al. (2013) propose to use an exponential parametric form, instead of a categorical distribution. This change requires an adjusted version of the M-step. This approach capitalizes on the fact that distortion parameters are actually integers and therefore cannot be applied to lexical parameters, which are discrete events. This reparameterization is in principle very similar to the one we focus on in Chapter 3, however, limited to the alignment distribution.

Another way to address over-parameterization is by employing sparse priors. A sparse prior changes the objective (no longer maximum likelihood) and leads to somewhat more compact models. This in turn results in smoothed probabilities for rare lexical entries. Mermer and Saraçlar (2011) proposed a full Bayesian treatment of IBM models 1 and 2 making use of Dirichlet priors to address the sparsity problem of vocabularies for rare lexical entries.

2.3.2 Feature-Rich Models

Feature-rich models are a way to circumvent lexical sparsity by enhancing the models with features that capture the dependencies between different morphologically inflected word forms. The standard parameterization using categorical distributions is limited with respect to the features it can capture. In this form of parameterization we assign parameters to disjoint events that strictly correspond to our generative story. An alternative to such parameterization is a locally normalized log-linear model. Such model can capture many overlapping features in predicting a decision given a certain event. These overlapping features of the event being described allow information sharing between lexical entries.

In the original IBM models, the parameters are estimated for maximum likelihood via expectation-maximization (Dempster et al., 1977). However EM is not typically used with a log-linear parameterization of the generative components because the normalization in the maximization step (M-step) becomes intractable. This is so because, unlike the case of categorical distributions, there is no closed-form solution to the M-step of a log-linear distribution (Berg-Kirkpatrick et al., 2010).

A solution is to propose an alternative objective to maximum likelihood. Smith and Eisner (2005) proposes *contrastive estimation* (CE), a learning objective that approximates maximum likelihood by restricting the implicit set of all possible observations to a set of implicit negative examples (hypothesized observations) that are "close" to the positive examples (actual observations). This set is called a neighborhood of an actual observation. Smith and Eisner (2005) proposed different strategies to building these sets, namely, different neighborhood functions. Such function should provide examples that are close to the observation, but differ in some informative way. In this way the model can move probability mass from these "negative" examples towards the observed (correct) examples. They proposed to obtain these neighborhood functions by small perturbations of the observation. More specifically, they used functions that contained the original sentence with one word or a sequence of words deleted, a function where, in the original sentence, two adjacent words are swapped and a function that contained all possible sentences with a length until the length of the original sentence.

Dyer et al. (2011) have applied CE to word alignment models in the context of globally normalized models.[6] They defined a single neighborhood function which constrained the set of all possible observations to all hypothetical translations of the input up to the length of the actual observed translation. Such neighborhood function, although tractable, is very large, in fact, too large for most useful purposes. That is why they constrain this function further with different pruning strategies.

The foremost problem with this approach is to choose a neighborhood function. Using the neighborhood functions from Smith and Eisner (2005) for word alignment models is questionable. Swapping words can still lead to a correct sentence. More particularly for morphologically rich languages, morphology can compensate for word order. Thus through a richer morphology a language could have more freedom in word order, resulting in sentences that could still be correct when swapping words. Additionally, deleting a word could still lead to a correct sentence.

Besides, using negative examples based on the morphemes in a word, for example by deleting or replacing an affix, is very difficult. First, deciding what an affix is in a word is difficult. Second, some affixes have a more significant impact on the meaning of a word than others. For example: The “-s” in “he works” only serves as agreement morphology, which is a form of inflection, compared to “I work”. On the other hand, consider the following word pairs: “kind”–“unkind” and “teach”–“teacher”. Both examples are instances of derivation and have a larger change on the meaning of the root than in the inflection example, by negating and by changing the syntactic category (from verb to noun) respectively. Lastly, morphological differences are significant across languages. What features are marked and the way how certain features are marked in a language can be very different.

Berg-Kirkpatrick et al. (2010) use a log-linear model and train it on unlabeled data as well. Instead of approximating the objective (as CE would do), they focus on locally normalized models and use a gradient-based optimization in the M step. Instead of learning all the generative components directly as conditional probability distributions (CPD’s), which are the categorical distributions discussed in Section 2.2.2, they parameterize each CPD as a log-linear combination of multiple overlapping features and only estimate the parameters of these log-linear models. This makes their model tractable. Through these multiple overlapping features, similarities across lexical entries can be captured. Thus the assumption that every lexical entry is independent of one another is dropped through the use of a feature-rich log-linear model. We will discuss this in greater detail in Chapter 3.

Berg-Kirkpatrick et al. (2010) implemented the log-linear model for both IBM model 1 and a first-order hidden Markov alignment model (Vogel et al., 1996). In this thesis we reproduce the work done by Berg-Kirkpatrick et al. (2010) for IBM model 1 and extend it to IBM model 2. We have not looked at the hidden Markov alignment model because inference for such models is more complex and goes beyond the scope of this thesis. Furthermore Berg-Kirkpatrick et al. (2010) evaluated their model on a Chinese-English dataset. Chinese is notable for its use of very little morphology. Compared to the original IBM model 1, their

[6]Globally normalized models differ from standard IBM models 1 and 2 with respect to the definition of the components. Unfortunately, discussing globally normalized models falls beyond the scope of this thesis.

log-linear version led to an improvement on alignment error rate (AER)[7] of approximately 6%.

3 Method

This chapter starts with introducing the feature-rich log-linear parameterization of IBM models 1 and 2, which we will refer to as log-linear IBM models. Next, the methods for evaluating these models are discussed.

3.1 Log-Linear IBM Models 1 and 2

In this section we introduce the feature-rich log-linear parameterization of the of IBM models 1 and 2, as described by Berg-Kirkpatrick et al. (2010). We will refer to the model as *log-linear IBM model*. First, we present the log-linear parametric form of the model. We then conclude this section by explaining how parameters are estimated from data.

3.1.1 Parameterization

In this section we explain the log-linear parametric form and how it is applied to the IBM model. A log-linear parametric form is enhanced with features. These possibly overlapping features can capture similarities between events in predicting a decision in a certain context. The features of each distribution type t, context c and decision d are combined in a feature vector. Each feature vector, associated with a (t, c, d) tuple, will be denoted by $\phi(t, c, d) = \langle \phi_1, \dots, \phi_g \rangle$, where g is the number of features in the feature vector.

Instead of being parameterized by a categorical distribution, the log-linear parametric form is parameterized by a weight vector with a length equal to the number of features. This weight vector assigns weights to each feature in the the feature vector. Through this weight vector the distributions are optimized as this weight vector can be optimized to assign higher probabilities to more informative features.

The log-linear parameterization is an exponentiated linear function of a parameter vector $w \in \mathbb{R}^g$. We show the definition in Equation(29), where $\mathcal{D}$ is the space of all possible decisions associated with t and c.

$$\theta_{t,c,d}(w) = \frac{\exp(w \cdot \phi(t, c, d)}{\sum_{d' \in \mathcal{D}} \exp(w \cdot \phi(t, c, d'))} \tag{29}$$

Note that we first compute a dot product between the weight vector and the feature vector, which is a linear operation. Then, we exponentiate this value obtaining a non-negative number, which we call a *local potential*. Finally, we compute a normalization constant in the denominator. This constant is the sum over all predictions $d' \in \mathcal{D}$ of their potentials. This yields a non-negative conditional probability distribution over decisions for each distribution type and context.

Originally, the log-linear IBM model is implemented for the lexical distribution only, but it is straightforward to extend it to the alignment distribution. In the presentation, we will focus on the lexical distribution for simplicity and clarity of exposition. In the case of the lexical distribution, features capture characteristics of the lexical entries in both languages and their co-occurrences. Through feature-rich representation events are related in the a g-dimensional

[7] The AER is a metric for comparing human annotated alignments to annotations made by using a model. See Section 3.2 for more information on the alignment error rate.

space which alleviates the strong independence assumptions imposed by categorical distributions. The full parameterization of the lexical distribution is shown in Equation (30), where $\mathsf{f} \in V_F$ is a French word (decision), $\mathsf{e} \in V_E$ is an English word (conditioning context), $w \in \mathbb{R}^g$ is the parameter vector, and $\phi : V_F \times V_E \to \mathbb{R}^g$ is a feature vector function.

$$P(\mathsf{f}|\mathsf{e}; w) = \frac{\exp(w \cdot \phi(\mathsf{e}, \mathsf{f}))}{\sum_{\mathsf{f}' \in V_F} \exp(w \cdot \phi(\mathsf{e}, \mathsf{f}'))} \tag{30}$$

Note that Equation (30) requires a normalization which runs in time proportional to $O(v_F \times g)$, that is, the size of the French vocabulary times the dimensionality of the feature space. Even though such log-linear parameterization is meant to address problems for which feature-rich representations are crucial. It might be the case that the task at hands, word alignment, will lead to inefficiency, particularly due to the fact that, by assumption, the French vocabulary is rather large. Also note that Equation (30) has to be solved as many as v_E times, since the conditioning context is a word in the English vocabulary. Altogether, solving (30) for all relevant events takes time proportional to $O(v_E \times v_F \times g)$. Whereas this gives us generative models with higher expressive power, the computational complexity may be too high compared to standard categorical-based models. These issues will lead to limitations in the experimental setup in Chapter 4.

We now turn to the estimation of the parameters w with EM.

3.1.2 Parameter Estimation

The estimation of the parameters consists in MLE via EM, as in the standard IBM models. That is, we are still optimizing our parameters using the maximum likelihood principle as shown in Equation (15d). We still have the hidden alignment variable, and therefore incomplete data. However, the log-linear parameterization in Equation (29) does not have a closed-form MLE solution (Berg-Kirkpatrick et al., 2010). Therefore we cannot use Equation (19) in the M step in order approach maximum likelihood estimates. Instead, we can rely on numerical optimization techniques.

To approach the MLE for these distributions, we follow Berg-Kirkpatrick et al. (2010) in adapting the M-step with a gradient-based update of the parameter vector w. In the E step, we still compute expected counts with respect to the posterior distribution over alignments ($P(a|f, e; \boldsymbol{\theta})$) as shown in Equation (22). This is mostly a direct application of previous techniques with very minor changes. The only large change is that instead of a lexical distribution with categorical parameters, we need to solve Equation (30) for each $(\mathsf{e}, \mathsf{f}) \in V_E \times V_F$. In the M step we rely on previously computed expectations and update w by taking the steepest direction of the gradient of the expected log-likelihood function. This optimization is reminiscent of logistic regression supervised classifiers with the difference that we are performing unsupervised learning and therefore rely on expected counts rather than observed counts.

We are maximizing the regularized *expected log-likelihood* with respect to a training set of partial observations as shown in Equation (31), where $\mu_{t,c,d} = \langle n_a(t, c, d) \rangle_{p(a|f,e)}$ is the expected number of occurrences of event (t, c, d). Recall that this expectation was defined in Equation (22). The last term in Equation (31) is a regularization term, meant to prevent the model from overfitting. In particular, we use the squared of L_2 norm as a regularization and κ is a hyperparameter called *regularization strength*.[8]

[8]Regularization is connected to a prior over the parameters w, a full treatment of the theory behind this is beyond the scope of this thesis.

$$l(w; \boldsymbol{\mu}) = \sum_{t,c,d} \mu_{t,c,d} \cdot \log \theta_{t,c,d}(w) - \kappa ||w||_2^2 \tag{31}$$

In gradient-based optimization we need to compute the first derivative of the objective function $l(w; \mu)$ with respect to parameters w. As w is a vector, we compute a gradient which is shown in Equation (32).

$$\nabla l(w; \boldsymbol{\mu}) = \sum_{t,c,d} \mu_{t,c,d} \cdot \left(\phi(t,c,d) - \sum_{d' \in \mathcal{D}} \theta_{t,c,d'}(w) \cdot \phi(t,c,d') \right) - 2\kappa \cdot w \tag{32}$$

Note that the first summation ranges over all possible (t, c, d) events. The term in brackets represent the difference between the feature vector $\phi(t, c, d)$ of an event and the expected feature vector $\sum_{d' \in \mathcal{D}} \theta_{t,c,d'}(w) \cdot \phi(t, c, d')$ for t, c under the current assignment of the parameters w. Whereas $\boldsymbol{\mu}$ is computed at the end of the E step and remains fixed throughout the M step, the inner term (expected feature vectors) has to be recomputed each time w is updated by a step of gradient-ascent.

We optimize the weight vector by iteratively computing the gradient of the expected log-likelihood and updating the weight vector in the steepest gradient direction. We use a gradient-based search algorithm called limited-memory BFGS (L-BFGS) (Liu and Nocedal, 1989). The explanation of this algorithm is beyond the scope of this thesis, it suffices to say that it is a standard gradient-based technique and it was also used by Berg-Kirkpatrick et al. (2010). After each update of the weight vector, we must recompute our parameters, i.e. Equation (30), before we can compute the gradient, i.e. Equation (32), as all these Equations depend on the specific assignment of the weight vector. This iterative process is repeated for predefined number of iterations. After this process is finished the M step is finished and we may continue iterating the main EM computation.

We know turn to algorithmic details of this featurized EM procedure.

3.1.3 Expectation-Maximization Algorithm for Log-Linear IBM Models 1 and 2

This section explains the EM algorithm for the log-linear parameterization of the IBM Models. Instead of initializing the distribution directly, we are initializing the weight parameter w. In case of IBM model 1 we initialize the weight parameter, uniformly as we do not have any good heuristics on which features will have a relatively high frequency of occurrence in the data. As the length of the weight parameter is equal to the number of features, we initialize our weight vector as defined in Equation (33). Recall that the g denotes the length of the feature vector.

$$w = \frac{1}{g} \tag{33}$$

In case of IBM model 2 we use the optimized weight distribution from IBM model 1 as initialization of the weight vector.

Like in the original parameterization, we loop over each sentence pair aggregating expected counts. For each possible word pair $(e_i = \mathsf{e}, f_j = \mathsf{f})$ in a sentence pair, the posterior $p(a_j = i|f_j, e)$ is added to the expected counts of lexical events $\mu_{\mathsf{lex,e,f}}$.

The algorithm for the log-linear parameterization of IBM model 2 is shown in Algorithm 2. In principle the algorithm is the same as the original parameterization of IBM model 2 until the M step. In the M step the re-normalization of the alignment parameters is not changed. The maximization of the log-linear parameterization of the lexical distribution, however, requires an iterative process of optimizing the weight parameters and recomputing $P(\mathsf{f}|\mathsf{e}; w)$ for every $(\mathsf{e}, \mathsf{f}) \in V_E \times V_F$. Alternating an E step and an M step is repeated for a predefined number of iterations.

Algorithm 2 M step of log-linear parameterization of IBM models 1 and 2

```
1:  $I \leftarrow$ number of iterations
2:  $K \leftarrow$ number of M steps
3:  $w \leftarrow$ weight vector
4:  $\boldsymbol{\lambda}(w) \leftarrow$ lexical parameters
5:  $\boldsymbol{\gamma} \leftarrow$ alignment parameters
6:
7:  for $i \in [1, \ldots, I]$ do
8:      E step:
9:      $n(\boldsymbol{\lambda}(w)) \leftarrow$ "compute expected counts"
10:     $n(\boldsymbol{\gamma}) \leftarrow$ "compute expected counts"
11:
12:     M step:
13:     for $k \in [1, \ldots, K]$ do
14:         $\nabla l(w; n(\boldsymbol{\lambda}(w))) \leftarrow$ "compute Equation (32)"
15:         $w \leftarrow$ "update using L-BFGS with $\nabla l(w; n(\boldsymbol{\lambda}(w)))$"
16:         $\lambda_{\mathsf{e},\mathsf{f}}(w) \leftarrow$ "compute Equation (30)"    $\forall(\mathsf{e}, \mathsf{f}) \in V_E \times V_F$
17:     end for
18:     $\gamma_x \leftarrow \frac{n(\gamma_x)}{\sum_{x' \in [-L,L]} n(\gamma_{x'})}$    $\forall x \in [-L, L]$
19: end for
```

3.1.4 Feature Extraction

We have implemented the log-linear parameterization both for the lexical and for the alignment distribution, however, there was only enough time to conduct initial experiments for lexical distribution. The features extracted from our observations therefore all capture lexical properties of the English and French lexical entries. Each feature vector contains features that describe the current word pair. Arbitrary features of each word pair can be used to describe the French or the English lexical entry. Features can be extracted independently for each word or we can also pair them. Thus we could have one feature that pairs the description of the French lexical entry with the description of the English lexical entry. A feature that describes the French lexical entry will be denoted by "f". Similarly, a feature that describes the English lexical entry will be denoted by "e". A paired feature will be denoted by "e-f".

In this study we extract three different types of features. The first feature type extracts the whole lexical entry as feature. The second feature type extracts suffixes and prefixes of a pre-specified length. The third feature type only checks if the lexical entry contains a digit. The features have been summarized in Table 1. The first column is the name where we will refer to. The second column gives the description of the feature and the third column the options are possible. Admittedly, there many interesting features that we do not list in Table 1, such as linguistic features that can be obtained with automatic analyzers and parsers, those are left for future work.

Feature name	Description	Options
word	Whole lexical entry	e, f, e-f
prefix	Prefix of specified length	e, f, e-f, length
suffix	Suffix of specified length	e, f, e-f, length
category	Boolean: checks if lexical entry contains digit(s)	e, f, e-f

Table 1: Summarization of the features extracted from the lexical entries in this thesis.

Language	Corpus
French	Canadian Hansard's parliament proceedings
German	Europarl European parliament proceedings

Table 2: Languages on which the models are evaluated.

3.2 Evaluation Methods

In this section we discuss the evaluation setup.

3.2.1 Morphologically Rich Languages

The evaluation concentrates on languages whose morphological processes are richer than English. These languages generally have a larger set of lexical entries, compared to languages with little morphology. That is because the marking of morphology on the root results in more variation in word forms and hence results in more lexical entries.

Table 2 shows the morphologically rich languages that are used for the evaluation. The evaluation is always done using a translation from a foreign language to English. Recall that because of the noisy-channel metaphor, and the application of Bayes rule, this means the alignment model generates foreign text observations by conditioning on English observations. The French data is taken from the Canadian Hansard's corpus and consists of proceedings of the Canadian parliament (Mihalcea and Pedersen, 2003). For the evaluation of German, the Europarl parallel corpus is used (Koehn, 2005). This corpus consists of several European languages, including German, and contains data from proceedings of the European parliament. We will refer to the Canadian Hansards corpus as the English-French corpus. Similarly, we will refer to the German data in the Europarl corpus as English-German corpus.

Now we turn to evaluation metrics.

3.2.2 Perplexity

Perplexity is a metric that measures how well a model fits a dataset. In this study, perplexity is measured in terms of cross entropy. The cross entropy is defined as:

$$H(P^*, P) = -\sum_{s=1}^{N} \frac{1}{S} \log P(f^{(s)}|e^{(s)}) \tag{34}$$

In this equation P' is the unknown true probability distribution and P is the probability distribution given by our model. N is the total number of sentence pairs in the parallel corpus. Then $P(f^s|e^{(s)})$ is the probability of the sth observation (that is, a French sentence paired with its English equivalent).

A higher probability of a French sentence given an English sentence means that the model is better able to fit the observed data. Note that the whole expression is negated, thus a lower cross entropy means a better performance. We can use cross-entropy to show how an optimization procedure, such as EM, converges to a local optimum of the likelihood function. If the cross-entropy does not decrease significantly after a certain number iterations, we can be confident that the model has converged.

3.2.3 Alignment Error Rate

For each language pair we have a small test set annotated with word alignments. This means that we can use such data to evaluate to predictions made by our models. With annotated data, we evaluate our model's prediction in terms of *alignment error rate* (AER). The annotations are done by human annotators and consists of alignments marked as "sure" in case of an unambiguous alignment (as evaluated by the annotator) and marked as "possible" in case of an ambiguous[9] alignment. The alignments are annotated in NAACL-format (Mihalcea and Pedersen, 2003). The manual annotation of the English-French test set has been constructed by Mihalcea and Pedersen (2003) and the manual annotation of the English-German test set by Padó and Lapata (2005).

To be able to evaluate our models we need our models to generate actual alignments. We generate the alignments using the MLE principle. Thus, we choose the alignments which has the highest probability $P(f, a|e)$ in our model. These alignments are called Viterbi alignments and are the alignments we evaluate against.

The results are evaluated using the AER (Och and Ney, 2000). This is a standardized metric that specifically evaluates the quality of the alignments. It requires annotated alignments to evaluate on and is based on precision and recall. If A is the set of actual alignments learned by the model, S the set of "sure" alignments and P the set of "possible" alignments, the AER formula for this evaluation is given in Equation (35).

$$\text{AER}(A; S, P) = \frac{|A \cap S| + |A \cap P|}{|S| + |A|} \tag{35}$$

4 Results

In this chapter the results of experiments using our models are presented. We take the log-linear IBM model with only the paired "word" feature ("word e-f") as our baseline version of the log-linear IBM model. That is because, using only this feature, the log-linear IBM model is most similar to the standard IBM model. From this baseline we will show the effects of adding different features. This chapter will start with showing statistics about the English-French and German-French corpus, after which we show the results for the IBM models. We then turn to the log-linear IBM models. We show how these models converge and what the effects are of adding different features. Lastly, we will compare the performance of the IBM models with the baseline variants of the log-linear IBM models as well as their optimized log-linear variants.

4.1 Corpus Statistics

Table 3 shows the number of words in the English-French corpus for different data sizes. Additionally it shows the vocabulary sizes (number of lexical entries) at each different data size. Table 4 shows the same statistics for the English-German corpus for different data sizes. For

[9]In this case there are more possible alignments that are seen as correct by the annotator.

Language	Corpus size	#Words	#Lexical entries
French	1 000	18 826	3 441
	10 000	219 731	12 234
	50 000	1 055 897	24 792
	100 000	2 107 667	33 567
	200 000	3 904 285	42 775
	Test set: 447	7 761	1 943
English	1 000	15 712	2 907
	10 000	186 366	9 659
	50 000	894 969	19 134
	100 000	1 783 743	25 946
	200 000	3 317 881	33 556
	Test set: 447	7 020	1 732

Table 3: Statistics for the English-French corpus for different data sizes.

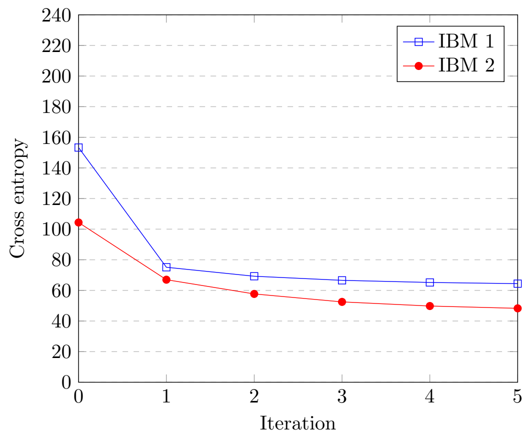
Language	Corpus size	#Words	#Lexical entries
German	1 000	23 682	4 440
	10 000	236 981	18 981
	50 000	1 194 611	48 839
	100 000	2 403 258	72 379
	200 000	4 794 356	105 635
	Test set: 987	22 777	4 820
English	1 000	26 321	3 399
	10 000	254 546	11 531
	50 000	1 269 802	24 724
	100 000	2 558 368	34 025
	200 000	5 085 199	47 299
	Test set: 987	23 336	3 852

Table 4: Statistics for the English-German corpus for different data sizes.

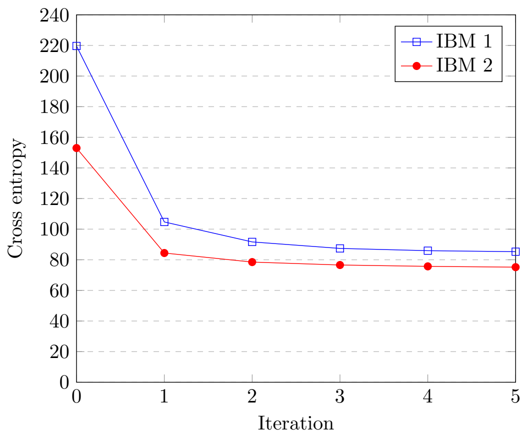
the evaluation of the AER we use small manually annotated test sets as described in Section 3.2.3. The statistics of these datasets are also shown in Figures 3 and 4. What can be observed is that the English-French corpus contains shorter sentences as the number of words generally is lower, when comparing to the English-German corpus and looking at the same data sizes. Furthermore, both French and German have a richer vocabulary, compared to the English translation, as their data contains more lexical entries. This could be an indication that both French and German have a richer morphology than English.

4.2 Standard IBM Models 1 and 2

Figure 4 shows the cross entropy for the English-French corpus at different iterations of IBM models 1 and 2. For the evaluation a set of 1000 sentence pairs (4a) and a set of 100 000 sentence pairs (4b) has been used. Figure 5 shows the graphs for the English-German corpus. What can be observed from both figures is that IBM models 1 and 2 both have converged only after a few iterations. To avoid overfitting of the parameters, we will only compute 5 iterations of IBM models 1 and 2 in further experiments.

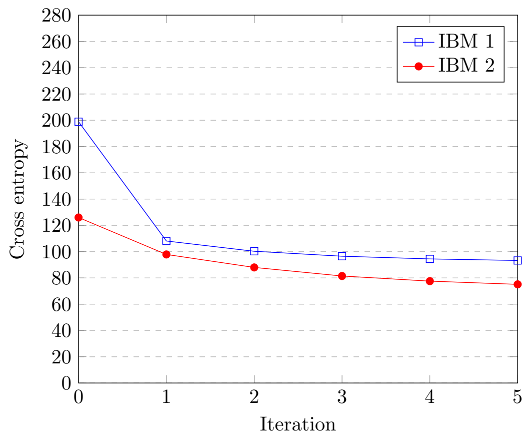


(a) Corpus size of 1000 sentence pairs.

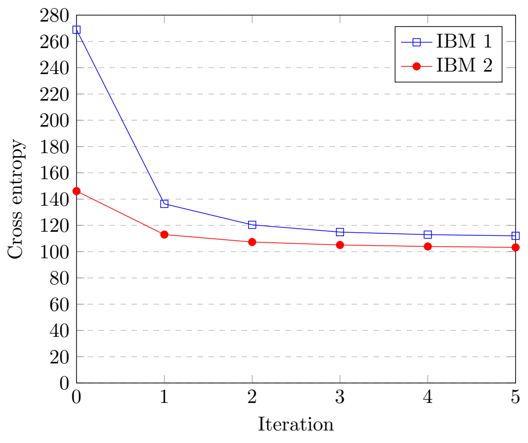


(b) Corpus size of 100 000 sentence pairs

Figure 4: Cross entropy of IBM model 1 and IBM model 2 on different corpus sizes with English-French sentence pairs.



(a) Corpus size of 1000 sentence pairs.

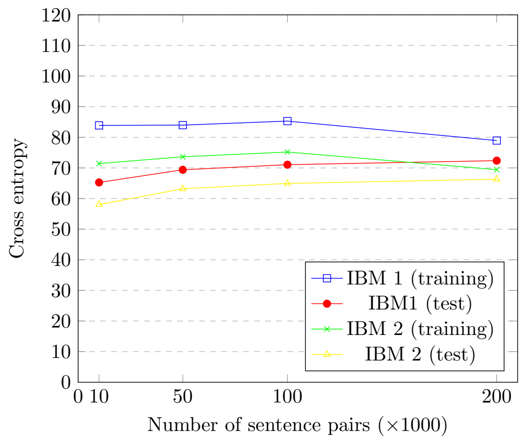


(b) Corpus size of 100 000 sentence pairs

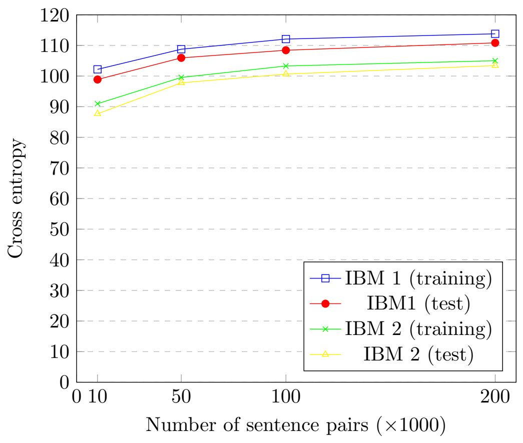
Figure 5: Cross entropy of IBM model 1 and IBM model 2 on different corpus sizes with English-German sentence pairs.

Note that IBM model 2 has a lower entropy at the zeroth iteration, compared to IBM model 1. This can be explained by the fact that we initialized the lexical distribution of IBM model 2 with the optimized lexical distribution of IBM model 1, but initialized the alignment distribution uniformly. Furthermore can be observed that the cross entropy of IBM model 2 at every iteration is lower than the cross entropy of IBM model 1. There are two reasons for this. Firstly, we initialize the lexical distribution of IBM model 2 with the optimized lexical distribution of IBM model 1. Secondly, IBM model 2 has more parameters, more specifically, an alignment distribution that is not uniform, as is the case in IBM model 1.

In Figure 6 the cross entropy on different corpus sizes is shown. As in the comparison of Figures 4 and 5, we can observe that the cross entropy is higher on the English-German corpus than on the English-French corpus. In addition, the cross entropy of IBM model 2 is lower for both the training and test set when comparing to the cross entropy of the training and test set of IBM model 1. This holds for both the English-French as the English-German



(a) English-French sentence pairs.



(b) English-German sentence pairs.

Figure 6: Cross entropy of IBM model 1 and IBM model 2 on a training set with different number of sentence pairs and a fixed test set after 5 iterations.

Corpus size	IBM 1	IBM 2
1 000	52.15	34.75
10 000	39.72	27.78
50 000	39.63	24.78
100 000	38.19	23.29
200 000	32.68	22.39

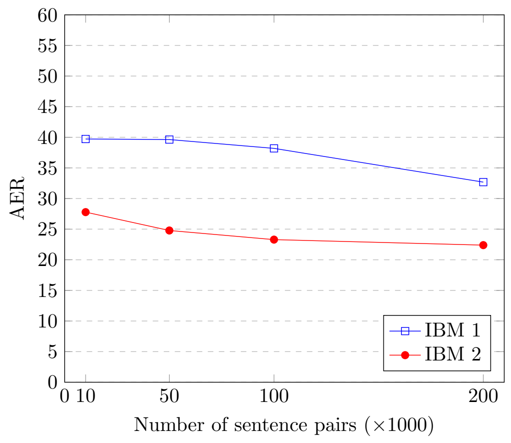
Table 5: AER on different corpus sizes with English-French sentence pairs.

corpus. We can further note that generally the cross entropy is slightly increasing with more sentence pairs. The only exceptions to this are the training sets of both IBM models 1 and 2 in the English-French corpus, where the cross entropy decreases when we increase the number of sentence pairs from 100 000 to 200 000.

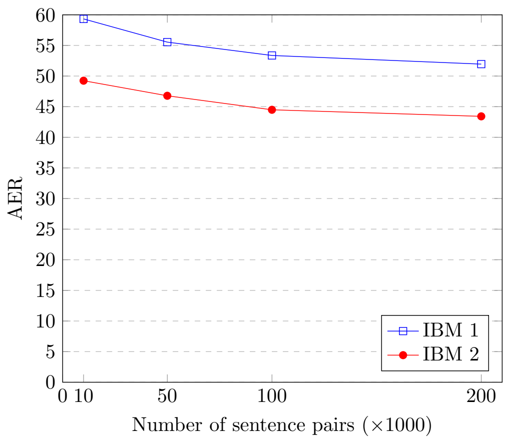
The results of the AER on different numbers of English-French sentence pairs have been summarized in Table 5 and on different numbers of English-German sentence pairs in Table 6. The AER of IBM model 1 on 200 000 English-French sentence pairs is on par with previous work (Och and Ney, 2000). We can observe that the AER is decreasing with larger corpus sizes. Furthermore, IBM model 2 has a lower AER on both the English-French and English-German corpus. We can also observe that the AER is lower on the English-French corpus than on the English-German corpus. One of the causes could be that German might have a richer vocabulary than French.

Data size	IBM 1	IBM 2
1 000	60.70	52.87
10 000	59.33	49.24
50 000	55.54	46.78
100 000	53.36	44.49
200 000	51.95	43.42

Table 6: AER on different corpus sizes with English-German sentence pairs.



(a) English-French sentence pairs.



(b) English-German sentence pairs.

Figure 7: AER for different data sizes after 5 iterations for IBM models 1 and 2.

4.3 Log-Linear IBM Models 1 and 2

In all evaluations of the log-linear IBM models we have omitted the use of regularization because of time constraints, adjusting the regularization strength is a time-consuming task. This means that $\kappa = 0$ in all our experiments with the log-linear IBM model.

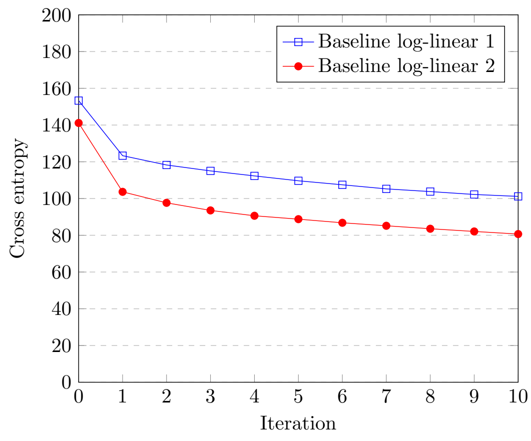
In Figure 8 the cross entropy for baseline version of log-linear IBM models 1 and 2 is displayed for a corpus size of 1000 sentence pairs. The results for the English-French corpus are shown in Figure 8a and the results for the English-German corpus in Figure 8b. We can observe that for the log-linear IBM models, it takes longer before they converge. It can be perfectly well explained that log-linear models need more iterations to converge as the log-linear object is harder to optimize. The log-linear models, as we described in Section 3.1.2, require iterative gradient-based updates, while categorical distributions have a closed-form MLE solution.

Also for this reason, running log-linear IBM model 1 and thereafter log-linear IBM model 2 takes much longer, compared to the categorical IBM models. While running 5 iterations of IBM models 1 takes only 14 seconds on 1000 English-French sentence pairs, it takes 39 minutes and 58 seconds for 10 iterations of the baseline log-linear IBM model 1 on the same English-French sentence pairs. Similarly, running 5 iterations of IBM model 2 takes 34 seconds on 1000 English-German sentence pairs and takes the baseline log-linear IBM model 2 on the same English-German sentence pairs 71 minutes and 19 seconds. With larger data sets we have larger event spaces, therefore more normalizations to compute in the E step and in the computation of gradients in the M step.

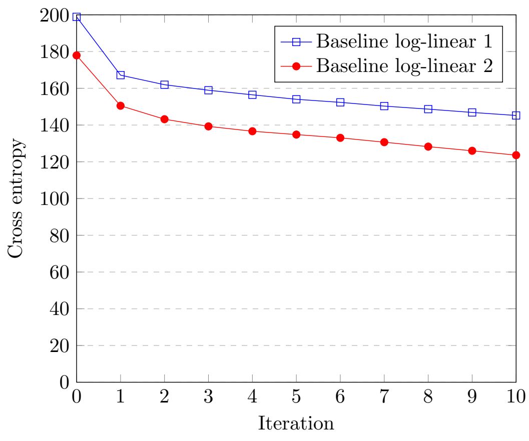
It would take even longer if more features were to be added. Because it takes this long to run the log-linear IBM models we are only using small of 1000 sentence to run the log-linear IBM models on.[10] Furthermore, we will use 10 iterations of the log-linear IBM models in all further experiments.

Again note that, because we initialize the lexical distribution of the baseline log-linear IBM model 2 with the lexical distribution of the baseline log-linear IBM model 1, the cross entropy of the baseline log-linear IBM model 2 is lower than the cross entropy of the baseline

[10]This is obviously not ideal, but it seems like speeding up log-linear models might not be so trivial. It turns out Berg-Kirkpatrick et al. (2010) also used a very small dataset, namely, 1000 sentences of English-Chinese data.



(a) English-French corpus.



(b) English-German corpus.

Figure 8: Cross entropy of the baseline versions of log-linear IBM models 1 and 2 on 1000 sentence pairs for English-French (8a) and English-German (8b).

log-linear IBM model 1 at the zeroth iteration.

Table 7 shows step by step how we have optimized log-linear IBM model 1. Because the models are very slow, we are restricting the optimization to only log-linear IBM model 1 and to only the English-French dataset. Furthermore, all experiments have been carried out on the same dataset of 1000 sentence pairs. Recall that our baseline log-linear model only contains the feature “word e-f”.

The first column shows which features are used. Each feature definition starts with the feature type, immediately followed by which language they describe. Separated by a colon, prefixes and suffixes are specified with lengths that have been used. A “+” (“plus”) at the start of a row means that, in addition to the features specified in the current row, the features of the previous row are also used for the current evaluation. The second column shows the number of (lexical) features that were extracted during the evaluation. The third and fourth columns show the cross entropy on the training and test set respectively. Finally, the last column shows the AER

What we can observe from Table 7 is that by adding more features the cross entropy on both the training as the test set generally decreases. The downside is that our feature space gets very large, therefore the log-linear models become very slow. Even though the cross entropy is lower when we add more feature, the AER is not in most evaluations. This means that we have a case of overfitting. Our log-linear model is fitting the data better, but is losing correlation with the task. This could happen for two reasons. Firstly, our independence assumptions may be too strong to properly model the task. Secondly, the parameter estimation is becoming stuck in a local optimum. Regularization could help alleviate this problem. However, in this thesis we have not used regularization.

The final results in terms of cross entropy and AER for the IBM models, the baseline log-linear IBM models and the optimized log-linear IBM models have been summarized in Table 8 for the English-French corpus with 1000 sentence pairs and in Table 9 for the English-German corpus with 1000 sentence pairs. For the final evaluation the standard IBM models were run with 5 iterations and the log-linear models with 10 iterations. Furthermore have we taken the log-linear model with the following features as our optimized version of the baseline log-linear model: “word e-f, suffix e-f: 2,3, prefix e-f: 2, 3”.

Features used	#Lexical features	Cross entropy training set	Cross entropy test set	AER
word e-f	250 891	104.38	100.68	51.66
+ word e, f, e-f	259 147	90.94	86.17	58.52
+ suffix e-f: 2,3	385 597	83.52	79.49	51.88
+ prefix e-f: 2,3	574 445	80.33	76.75	45.11
+ category e, f, e-f	574 453	72.08	66.12	58.44
word e-f, suffix e-f: 2,3	377 341	81.31	78.10	53.98
+ prefix e-f: 2,3	566 189	79.84	76.66	53.83

Table 7: Results of the optimization of the log-linear IBM model 1 on a English-French corpus with 1000 sentence pairs.

Model	Cross entropy training set	Cross entropy test set	AER
IBM model 1	65.11	59.77	52.15
IBM model 2	49.83	47.89	34.75
Baseline log-linear 1	104.38	100.68	51.66
Baseline log-linear 2	85.94	86.61	33.12
Optimized log-linear 1	79.84	76.66	53.83
Optimized log-linear 2	67.18	66.48	31.13

Table 8: Combined results of cross entropy on a training set of 1000 sentence pairs, a test set of 447 sentence pairs and AER on the English-French corpus.

We can observe that both baseline log-linear IBM model 1 as the optimized log-linear IBM model do not have a lower cross entropy than the standard IBM model 1 on both the training as the test set. Similarly, the baseline log-linear IBM model 2 and the optimized log-linear IBM model 2 do not perform better than the standard IBM model 2. Both the baseline as the optimized log-linear IBM models 2 perform better in terms of cross entropy than their log-linear IBM model 1 variants.

In addition the results of the evaluation on the AER show mixed results. On the English-French corpus we can observe that the baseline log-linear IBM model 1 has a lower AER, compared to the standard IBM model 1, while our optimized version has a higher AER than both the standard IBM model 1 as well as the baseline log-linear IBM model 1. On the other side, the optimized log-linear IBM model 2 has a lower AER than the baseline log-linear IBM model 2, which in turn has a lower AER than the standard IBM model 2. On the English-German corpus the log-linear IBM models all have a lower AER, compared to their standard variant. In addition, even the optimized log-linear IBM models perform worse than their baseline variants.

5 Conclusion

In this thesis we argued that loosening the assumption that all lexical entries are treated independently from each other is a natural and appealing idea. To test this hypothesis, we implemented IBM models 1 and 2 as well as a feature-rich log-linear parameterization of these models. Our results do not support this hypothesis yet, but they are promising. We believe the hypothesis can be confirmed, however, there are still many technical challenges left before our

Model	Cross entropy training set	Cross entropy test set	AER
IBM model 1	96.24	90.63	60.70
IBM model 2	79.91	74.19	52.87
Baseline log-linear 1	155.18	150.86	61.06
Baseline log-linear 2	140.00	134.05	53.42
Optimized log-linear 1	122.96	118.81	71.70
Optimized log-linear 2	112.16	105.87	56.79

Table 9: Combined results of cross entropy on a training set of 1000 sentence pairs, a test set of 987 sentence pairs and AER on the English-German corpus.

models can become competitive with respect to quality and speed. Even though the baseline versions of our log-linear IBM models do not perform better in terms of cross entropy and AER, the optimized versions do in terms of AER when the right set of features is used.

A foremost problem with the log-linear IBM models is that these models are very slow. Therefore it is not very practical to do experiments on large corpus sizes. This is limiting us in validating our hypothesis. We can partly explain this by the fact that we are loosening the independence assumption that all lexical entries are independent of one another, which requires us to do more computations. However, also the log-linear IBM models need to be optimized further. In particular, the computation of the gradient of the expected log-likelihood in the M step is slow and needs to be optimized.

In this thesis, due to time constraints, we have not used the regularization term in the M step of the log-linear IBM models. As our models overfit on the datasets, we can expect the optimized log-linear models to perform better when we do you use the regularization term. The regularization term would help alleviate the overfitting.

Furthermore, only a limited number of features has been used. The effect of loosening the independence assumption is hypothesized to be stronger by adding more features. The features that have been used in this thesis were also limited in terms of linguistic intuition. More abstract features are also more information-rich, that is, they relate to more events and are less sparse. We expect such features to lead to better and faster results. These kind of features might use part-of-speech tags or a stemmer like the Porter Stemmer (Porter, 1980).

There could also be looked at using feature induction. In feature induction, as opposed to feature engineering, we do not design our features but induce them. Inducing a dense feature representation would get away with large sparse feature space, while still maintaining feature richness. Chen and Manning (2014) predict features from raw text using a feed-forward neural network.

Furthermore, in this thesis we only looked at two languages: French and German. These languages are both Indo-European languages and share similarities between them. Therefore, if our hypothesis turns out to be true, to be able to generalize this claim we need to compare the IBM models with the log-linear IBM models for more and more unrelated languages. Besides, we are translating to English. English is also very related to both French and German.

References

Berg-Kirkpatrick, T., Bouchard-Côté, A., DeNero, J. and Klein, D. (2010). Painless Unsupervised Learning with Features. In Proceedings of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies pp. 582–590, Association for Computational Linguistics, Los Angeles, CA, USA.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D. and Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation: Parameter estimation. Computational linguistics *19*, 263–311.

Chen, D. and Manning, C. D. (2014). A Fast and Accurate Dependency Parser using Neural Networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing pp. 740–750, Association for Computational Linguistics, Doha, Qatar.

Chiang, D. (2005). A Hierarchical Phrase-Based Model for Statistical Machine Translation. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics pp. 263–270, Association for Computational Linguistics, Ann Arbor, MI, USA.

Daiber, J. and Sima'an, K. (2015). Machine Translation with Source-Predicted Target Morphology. In Proceedings of MT Summit XV.

Das, D. and Petrov, S. (2011). 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 - Volume 1 pp. 600–609, Association for Computational Linguistics, Stroudsburg, PA, USA.

Dempster, A., Laird, N. and Rubin, D. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society. Series B (Methodological) *39*, 1–38.

DeNeefe, S. and Knight, K. (2009). Synchronous Tree Adjoining Machine Translation. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 pp. 727–736, Association for Computational Linguistics, Stroudsburg, PA, USA.

Dyer, C., Chahuneau, V. and Smith, N. A. (2013). A Simple, Fast, and Effective Reparameterization of IBM Model 2. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies pp. 644–648, Association for Computational Linguistics, Atlanta, GA, USA.

Dyer, C., Clark, J. H., Lavie, A. and Smith, N. A. (2011). Unsupervised Word Alignment with Arbitrary Features. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies pp. 409–419, Association for Computational Linguistics, Portland, OR, USA.

Hutchins, J. (2007). Machine Translation: A Concise History. In Computer Aided Translation: Theory and Practice, C. S. Wai, Ed. Chinese University of Hong Kong.

Jones, B., Andreas, J., Bauer, D., Hermann, K. M. and Knight, K. (2012). Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In Proceedings of COLING 2012 pp. 1359–1376, COLING 2012, Mumbai, India.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. In MT summit vol. 5, pp. 79–86,, Phuket, Thailand.

Koehn, P., Och, F. J. and Marcu, D. (2003). Statistical Phrase-Based Translation. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1 pp. 48–54, Association for Computational Linguistics, Stroudsburg, PA, USA.

Koller, D. and Friedman, N. (2009). Probabilistic Graphical Models: Principles and Techniques. Adaptive computation and machine learning, MIT Press.

Kozhevnikov, M. and Titov, I. (2013). Cross-lingual Transfer of Semantic Role Labeling Models. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics - Volume 1: Long Papers pp. 1190–1200, Association for Computational Linguistics, Sofia, Bulgaria.

Liang, P., Taskar, B. and Klein, D. (2006). Alignment by Agreement. In Proceedings of the Human Language Technology Conference of the NAACL, Main Conference pp. 104–111, Association for Computational Linguistics, New York City, NY, USA.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. Mathematical programming *45*, 503–528.

Locke, W. and Booth, A. (1955). Machine translation of languages: fourteen essays. Published jointly by MIT Press and Wiley, New York City, NY USA.

Lopez, A. (2008). Statistical Machine Translation. ACM Computing Surveys *40*, 8:1–8:49.

Mermer, C. and Saraçlar, M. (2011). Bayesian Word Alignment for Statistical Machine Translation. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies pp. 182–187, Association for Computational Linguistics, Portland, OR, USA.

Mihalcea, R. and Pedersen, T. (2003). 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 pp. 1–10, Association for Computational Linguistics.

Moore, R. C. (2004). Improving IBM Word Alignment Model 1. In Proceedings of the 42nd Meeting of the Association for Computational Linguistics pp. 518–525, Association for Computational Linguistics, Barcelona, Spain.

Och, F. J. and Ney, H. (2000). Improved statistical alignment models. In Proceedings of the 38th Annual Meeting on Association for Computational Linguistics pp. 440–447, Association for Computational Linguistics.

Och, F. J. and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. Computational linguistics *29*, 19–51.

Padó, S. and Lapata, M. (2005). Cross-linguistic Projection of Role-Semantic Information. In Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing pp. 859–866,.

Porter, M. F. (1980). An algorithm for suffix stripping. Program *14*, 130–137.

Smith, N. A. and Eisner, J. (2005). Contrastive Estimation: Training Log-Linear Models on Unlabeled Data. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics pp. 354–362, Association for Computational Linguistics, Ann Arbor, MI, USA.

Snyder, B. and Barzilay, R. (2008). Unsupervised Multilingual Learning for Morphological Segmentation. In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies pp. 737–745, Association for Computational Linguistics, Columbus, OH, USA.

Vogel, S., Ney, H. and Tillmann, C. (1996). HMM-based Word Alignment in Statistical Translation. In Proceedings of the 16th conference on Computational linguistics - Volume 2 pp. 836–841, Association for Computational Linguistics.