

File ID 351146
Filename 2: Machine translation and machine learning concepts

SOURCE (OR PART OF THE FOLLOWING SOURCE):

Type Dissertation
Title Learning the latent structure of translation
Author M. Mylonakis
Faculty Faculty of Science
Year 2012
Pages xv, 198
ISBN 978-90-5776-235-2

FULL BIBLIOGRAPHIC DETAILS:

<http://dare.uva.nl/record/407017>

Copyright

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use.

Chapter 2

Machine Translation and Machine Learning Concepts

Machine Translation (MT) on the one hand and Artificial Intelligence (AI) and Machine Learning (ML) on the other, have always followed crossing paths. MT has constantly figured as one of the most prominent fields of AI research, with the progress and disillusion on MT strongly affecting the appeal of AI, as highlighted during the period before and after the ALPAC report (Pierce et al., 1966).

The last two decades, MT has flourished as a result of the availability of more and cheaper computing power and the introduction of statistical models for Natural Language Processing (NLP). Most of the MT systems before this were based on translation lexica and fixed predefined rules which would ‘fire’ for a host of translation phenomena. In contrast, the statistical approach is centred around the formulation of stochastic translation models and training these models on corpora. The transition towards explaining the translation process through a statistical model crucially allowed tapping into the wealth of Machine Learning research in statistical estimation. Furthermore, it also contributed to the introduction of novel Machine Learning approaches motivated by the challenges posed by the MT models, such as the large number of parameters, the interplay between memorising and generalising, dealing with yet unseen events and others. This thesis follows this trend by contributing MT solutions through exploring and proposing novel approaches on fundamental learning problems.

The pioneering work on Statistical Machine Translation (SMT) in the IBM labs (Brown et al., 1990) introduced *word-based* statistical translation models. From there on, the major steps in the SMT literature involve models translating contiguous phrases together (Och et al., 1999; Koehn et al., 2003) and later recursive translation employing phrasal patterns with gaps (Chiang, 2005a). Recent developments focus on employing hierarchical structure for MT, often taking advantage of monolingual syntactic analyses, e.g. (Galley et al., 2004; Zollmann and Venugopal, 2006; Mylonakis and Sima’an, 2011).

As the MT models become more complex however, the stress on the associated

ML methods used to train them and translate with them is increased. Training the IBM models already relied on approximations for the more complicated models. The subsequent step towards models employing phrases, hierarchical or not, was also marked by a transition to heuristic estimation, making less clear how the estimates relate to the training corpus. This is not without reason: it is notoriously difficult to estimate such models. We believe however that moving away from relying on hand-crafted arbitrary heuristics and towards well-founded estimation is fundamental when progressing to even more complex models involving rich latent MT structure, in the same way that leaving behind hand-crafted translation rules was important in the early years of statistical MT. This has recently proved a very vibrant research direction and part of the work in this thesis is occupied with this topic.

In this chapter we will first present the basic concepts of the key frameworks and models for Machine Translation that play a role in this thesis. We begin by presenting the IBM word-based SMT models, and continue with a discussion of phrase-based and hierarchical SMT. In the second part, we will focus on the two Machine Learning methods which form the backbone of the learning approach contributed by this work. We will first examine the basics of the Expectation-Maximization (EM) algorithm (Dempster et al., 1977), a key and powerful statistical estimation algorithm that has already been widely and successfully employed for MT (Brown et al., 1993). Nevertheless, the application of EM on phrase-based models exposed their strong tendency to overfit the training data and generalise poorly. Keeping this in mind, we subsequently close the chapter by introducing Cross-Validation (CV), a well-understood method to estimate the generalisation error of model estimates. In the following chapter we will show how EM and CV can be combined towards MT model estimation specifically aiming towards strong generalisation over yet unseen data.

2.1 Modelling Machine Translation

The branch of Machine Translation where a high proportion of current MT research is directed and on which this work focuses is Statistical MT. Given a source language sentence \mathbf{f} , the fundamental problem in MT is to produce its target language translation \mathbf{e} by means of a computer program. Output \mathbf{e} must both sufficiently convey the *meaning* of the original sentence \mathbf{f} , as well as enjoy target language *fluency*. SMT aims to achieve this through the application of statistical models. By introducing a probability distribution $p(\mathbf{e}|\mathbf{f})$, assigning to every target sentence \mathbf{e} a probability of being the translation of source input \mathbf{f} , an SMT system outputs the target sentence $\hat{\mathbf{e}}$ with the highest conditional probability:

$$\hat{\mathbf{e}} = \arg \max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) \quad (2.1)$$

Building an SMT system can be mostly divided in three parts. Firstly, it involves *designing the model* $p(\mathbf{e}|\mathbf{f})$. Some of the questions here might be what kind of translation phenomena does it capture and how does it capture them, what are the parameters and which latent variables are assumed. Model design plays a crucial role in SMT, as it defines the rules of the game: what needs to be learnt from the training corpora and later applied to actually translate, according to the modellers view of translation. After the model is set, we need to *train* it, select the model instance which is best according to some learning objective, by employing training data possibly coupled with prior knowledge. This entails the usage of a statistical *estimator*. The final step, *decoding*, employs the trained model estimate to actually translate by selecting for every input \mathbf{f} the translation $\hat{\mathbf{e}}$ according to equation (2.1).

2.1.1 The Noisy Channel Approach

Shannon’s noisy channel (Shannon, 1948) has been an influential paradigm for SMT (Brown et al., 1990). Instead of directly modelling target sentences given source input, we consider the target sentence as a message which got corrupted while being transmitted through a translation communication channel, resulting in the source sentence. Our objective is to retrieve this original message. We use Bayes law to rewrite the search objective of (2.1) in the equivalent formulation below, separately modelling the language of the target sentences and the corruption of these sentences when translated from target to source.

$$\begin{aligned} \hat{\mathbf{e}} &= \arg \max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) \\ &= \arg \max_{\mathbf{e}} \overbrace{p(\mathbf{f}|\mathbf{e})}^{\text{translation model}} \overbrace{p(\mathbf{e})}^{\text{language model}} \end{aligned} \quad (2.2)$$

This crucially splits the modelling effort in a stochastic component focused on translation correspondence, the Translation Model (TM), and a component exclusively occupied with output well-formedness, the Language Model (LM). Each of these models is then occupied with one of the two key objectives of the translation system’s output outlined above: meaning correspondence and fluency. Considering these two notions apart avoids modelling all aspects of translation at once, letting the TM focus on the transformations that take place during translation while the LM attends to output fluency. In addition, it also allows employing different resources for training. While the translation model usually requires more expensive bilingual data to train, language model training only demands monolingual data which are cheaper to assemble in large quantities.

Early SMT work, such as the IBM models later discussed in this chapter, applied the Noisy Channel paradigm in a relatively literal fashion. However,

translation adequacy and fluency can in practice hardly be considered separate. Malformed target output cannot appropriately convey the meaning of the source sentence; an adequate translation would probably be expected to also be relatively well-formed. Subsequent SMT research deviated from a strict reading of the Noisy Channel approach, regarding the language model probability of the target sentence as just one of the elements considered to assess the overall translation probability, together with other, more bilingual in nature, translation features.

2.1.2 Generative and Discriminative Models

Generative translation models capture the stochastic joint generation of source and target sentence pairs. They can also straightforwardly be employed to select the translation \mathbf{e} with the highest probability given \mathbf{f} , as with \mathbf{f} fixed we have:

$$\hat{\mathbf{e}} = \arg \max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) = \arg \max_{\mathbf{e}} p(\mathbf{e}, \mathbf{f}) \quad (2.3)$$

These models are usually based on a generative process tracking the steps to emit the tuple $\langle \mathbf{e}, \mathbf{f} \rangle$. For example, we might begin by considering the generation of corresponding source and target word-pairs following the word order of the source language, subsequently reordering the target language words to form the target sentence. Each of the generative steps is modelled by a separate distribution conditioned on the previous steps, often under independence assumptions which simplify the modelling effort. Some conditional translation models $p(\mathbf{e}|\mathbf{f})$ are formulated in a similar fashion, emitting \mathbf{e} from \mathbf{f} under a generative process (Brown et al., 1993).

Generative models require extensive effort to consider all the steps and transformations that take place during translation, as well as to introduce independence assumptions taking into account the available training data (e.g. to avoid overfitting) or computational limitations etc. In contrast, *discriminative* modelling directly models the conditional distribution $p(\mathbf{e}|\mathbf{f})$, instead of putting effort towards formulating a full generative process emitting samples $\langle \mathbf{e}, \mathbf{f} \rangle$. For MT, this typically happens through employing *feature functions* $\phi_i(\mathbf{e}, \mathbf{f})$, each assigning a non-negative score examining the two sentences from a different perspective, e.g. word or phrase correspondence, output fluency (frequently the LM score), target word reordering and others. The modeller does not need to consider a coherent generative story but only what kind of features could be useful in discriminating between strong and weak translations. These scores are weighted together log-linearly with weights λ_i and normalised to obtain the conditional translation model (Och and Ney, 2004).

$$p(\mathbf{e}|\mathbf{f}) = \frac{1}{Z_{\lambda}(\mathbf{f})} \exp \left(\sum_i \lambda_i \phi_i(\mathbf{e}, \mathbf{f}) \right) \quad (2.4)$$

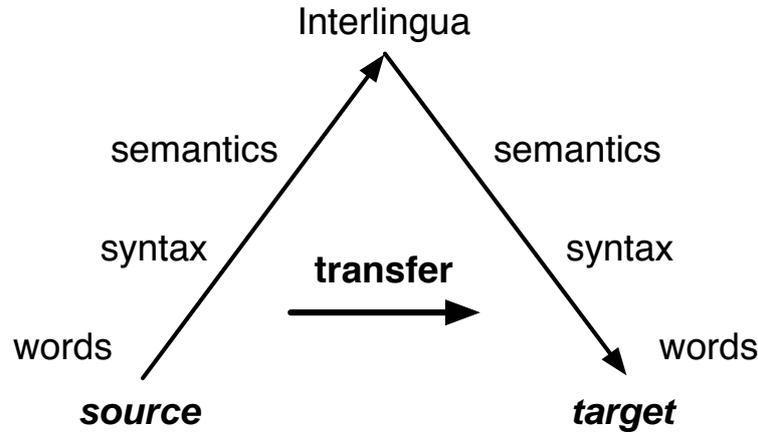


Figure 2.1: The Machine Translation pyramid

The step of estimating the feature weights is crucial. Possible training objectives are constrained entropy maximisation (Berger et al., 1996) and error rate minimisation according to a translation quality metric (Och, 2003).

Crucially, if we consider selecting the translation \mathbf{e} with the highest probability as a classification task, while for many machine learning tasks the class space is relatively constrained, in MT (and NLP in general) the class space is very large or even countably infinite. For this, frequently a generative process is still needed as part of a translation system based on a discriminative model, to supply the set of target sentence translations that will be scored by the model to select the most probable one, sometimes producing also a score embedded as a feature function of (2.4). The latter is the case for all phrase-based and hierarchical translation approaches later discussed in this chapter.

One disadvantage of discriminative MT models is that it is more difficult to introduce and train the parameters for *latent* variables in the model, such as latent structure which is not part of the observed training data. In this thesis we take a hybrid approach. We first train a generative model employing latent translation variables, which is afterwards included as both a feature function and a generative process backbone for a discriminative translation model.

2.1.3 Model Categorisation

Apart from the probabilistic formalisation approach that they follow as discussed above (e.g. generative, discriminative, hybrid), MT models can be also categorised according to the, often latent, *abstraction* from the lexical level that they employ. The familiar MT pyramid in Figure 2.1 (Vauquois, 1968) presents a view on the different levels of abstraction. At the most basic level, MT models operate directly on the lexical surface, translating and reordering based on lexical

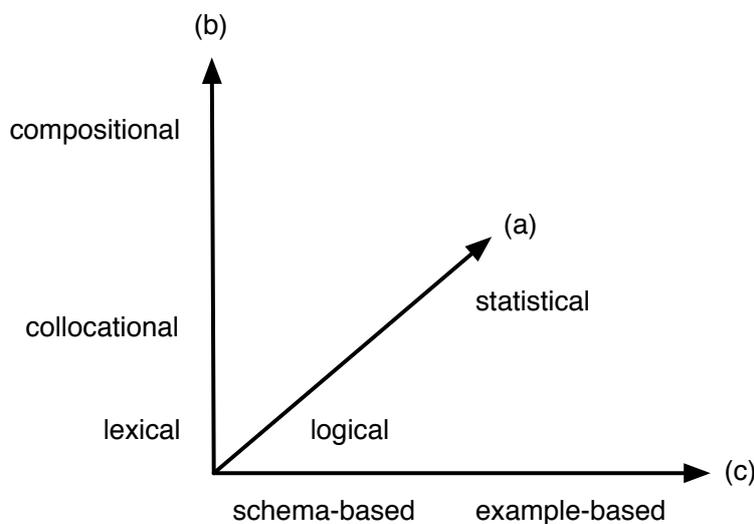


Figure 2.2: 3-axes categorisation of Machine Translation models

cues. Moving up the pyramid we find models which utilise syntactic and semantic categorisations and representations, with a transfer step modelling the transformation of this representation from source to target form. Finally, at the top of the pyramid stands the Interlingua approach, which is based on constructing an internal, natural language independent abstraction of the full meaning of the source sentence and subsequently building the target language from it.

Historically, MT research has followed an interesting pattern exploring the MT pyramid. The early approaches on MT, starting already from the IBM-Georgetown demonstration, emphasised the employment of grammatical abstractions and rule-based transfer steps between source and target language, while at the same period the Interlingua approach was quite influential. Since the advent of Statistical MT from the late 80's and onwards, most state-of-the-art MT systems (e.g. (Brown et al., 1993; Och et al., 1999; Koehn et al., 2003)) directly modelled translation on the lexical level, with this trend lasting for almost two decades. Recently, there have been considerable research efforts of increasing sophistication on syntactical approaches on SMT (e.g. (Chiang, 2005a; Zollmann and Venugopal, 2006)), finally delivering state-of-the-art performance, particularly for language pairs with heavy reordering such as English-Chinese.

Wu (2005) introduces a 3-axes system to categorise MT models, presented in Figure 2.2. Axis (a) examines if the model is mostly based on mathematical logic, or if it makes substantial use of statistics and probabilities. All the models we examine in this thesis are statistical MT models. The second axis (b) relates to the degree of recursion in the model. At the bottom stand lexical-based models like the IBM models, while moving up the axis we find collocational models such as those employed in phrase-based SMT and finally fully compositional models

such as those backed by Synchronous Context Free Grammars. The models contributed in later chapters of this work fall in these last two categories along axis (b).

Axis (c) considers if the model is based on abstracting versus memorising the training data. In the first case, an abstraction such as a generative translation model is built during training, which is later employed during test time to translate. In contrast, example-based MT (Nagao, 1984) relies on memorising the training data (examples) and reusing these to translate by breaking them down, adapting and recombining them according to the input source sentence. Recent SMT systems blur the line between statistical model estimation (schema-based MT) and memorisation (example-based MT). For example, while Data Oriented Translation (Poutsma, 2000) memorises aligned fragments of syntactic trees of source and target sentences, it also learns a probabilistic model that describes how to combine them together to derive sentence-pairs. Instead of training a hierarchical translation model prior to test time, (Lopez, 2007) memorises the training parallel corpus and extracts and scores recursive translation rules separately for every input sentence during test time. Furthermore, phrase-based models, whether they are recursive in nature or not, memorise parts of the training data. In this thesis, inspired by Data Oriented Processing (Bod and Scha, 1996), we take this further by opting for an *all phrase-pairs* approach, extracting and memorising all corresponding phrases of the training parallel corpus, up to the whole sentence-pair.

2.2 Word-Based Translation

Statistical MT was introduced by researchers from the IBM T.J. Watson research centre in the late 80's (Brown et al., 1990). This first attempt at SMT was further refined in the formulation of a succession of word-based translation models of increasing complexity, the IBM translation models (Brown et al., 1993). The models are founded on a Noisy-Channel approach to translation, as discussed in section 2.1.1 above. The translation process that is being modelled is inverted, so that we introduce a target-to-source translation model $p(\mathbf{f}|\mathbf{e})$, as well as a language model component $p(\mathbf{e})$ over the target language. This relieves the translation model from the task of concentrating probability mass on well-formed output sentences, as would be required by a direct translation model $p(\mathbf{e}|\mathbf{f})$. This task is assigned to the language model instead.

The authors recognise the three foundational problems in SMT: (a) estimating the *language model* probabilities $p(\mathbf{e})$ over target language sentences \mathbf{e} , (b) estimating the *translation model* probabilities $p(\mathbf{f}|\mathbf{e})$ from target to source and (c) *decoding*, i.e. searching for $\hat{\mathbf{e}}$ which maximises the product of the two as in equation (2.2). SMT took advantage of the existing research on language mod-

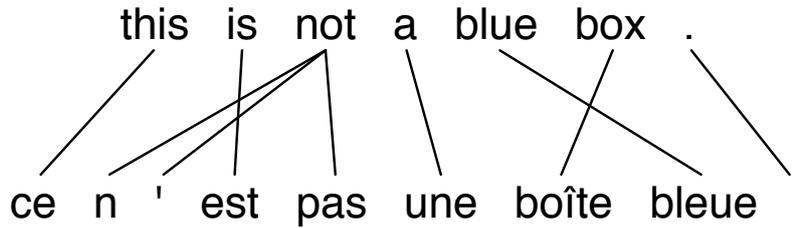


Figure 2.3: An alignment between English and French

elling by the speech processing community¹. Hence, attention was drawn to the translation model, while the highly non-trivial task of decoding efficiently forms a research direction of its own which we do not treat extensively in this thesis.

2.2.1 Alignments

The key concept introduced by the IBM models are word *alignments*, links between words which are considered translations of each other. For the parallel sentence pair $\langle \mathbf{e}, \mathbf{f} \rangle$, $\mathcal{A}(\mathbf{e}, \mathbf{f})$ is the set of word-position pairs of aligned words between the two sentences. Using French to English examples, these links can in general connect a single source and target word (*the - le*), a single source to multiple target words (*pick up - ramasser*), a single target to multiple source words (*implemented - mis en application*); or, considering every source word aligned to every target word, multiple source to multiple target words (e.g. idioms such as *take the trouble - prendre la peine*). The commonly aligned words need not be contiguous (e.g. *not - ne ___ pas*). An example of an aligned sentence pair can be seen in Figure 2.3.

To simplify matters, the IBM models considered only alignments where every source word f is at most aligned to a single target word e . As in Figure 2.3, this can nevertheless result in multiple target words being aligned to the same source word. For a source sentence $\mathbf{f} = f_1^m$ of length m and a target sentence $\mathbf{e} = e_1^l$ of length l , the alignment variable is then defined as the vector $\mathbf{a} := a_1^m$, with a_j the position of the target word that f_j is aligned to. A source word is also allowed to remain unaligned, in which case we consider it aligned with an additional empty token at target word-position zero.

The alignment \mathbf{a} between $\langle \mathbf{e}, \mathbf{f} \rangle$ is a *latent* variable, given that such information is not normally part of a parallel training corpus. The translation probability must thus sum over all values of \mathbf{a} .

$$p(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e}) \quad (2.5)$$

¹See (Chen and Goodman, 1998) for an overview of most models employed up to this day.

2.2.2 Translation Models

With the help of the alignment variable and without loss of generality, we can write employing the chain rule:

$$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \overbrace{p(m|\mathbf{e})}^{\text{source length}} \prod_{j=1}^m \overbrace{p(a_j|a_1^{j-1}, f_1^{j-1}, m, \mathbf{e})}^{\text{current alignment point}} \overbrace{p(f_j|a_1^j, f_1^{j-1}, m, \mathbf{e})}^{\text{current lexical choice}} \quad (2.6)$$

In the generative process of the equation above, first the length of the \mathbf{f} sentence is sampled from $p(m|\mathbf{e})$. Subsequently, in series from left to right in the order of \mathbf{f} , the alignment point for the current source position is established given the previous alignments and source words, as well as m and the \mathbf{e} sentence. Finally, the current source word is generated given a similar conditioning variable set with the addition of the current alignment point established in the previous step.

Working with such detailed conditioning contexts as in (2.6) leads to computational difficulties in estimation and can be prone to overfitting. The succession of models in (Brown et al., 1993) are the result of applying different sets of assumptions simplifying equation (2.6).

IBM Model 1 In the simplest of the translation models, $p(m|\mathbf{e})$ is assumed to be independent of both m and \mathbf{e} and thus equal to a constant ϵ . For every f_j word, its alignment is sampled uniformly from the l word positions of \mathbf{e} plus the option of aligning to empty and is therefore $(l+1)^{-1}$. Finally, lexical selection takes place conditioning only on the e_{a_j} target word that f_j is aligned to. Given these assumptions, (2.6) can be rewritten as:

$$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^m p(f_j|e_{a_j}) \quad (2.7)$$

Equation (2.7) establishes a foundational parameter set in MT models, the conditional translation probabilities $p(f|e)$, as well as a key data structure of MT systems: the *translation table* holding these probabilities for every $\langle e, f \rangle$ pair.

IBM Model 2 Model 1 does not occupy itself with the word order in the two strings as dictated by the alignment points and any reordering of the f words is assigned an equal translation probability. Model 2 introduces non-uniform alignment probabilities $p(a_j|j, m, l)$, by conditioning each alignment point on the word position of the word being aligned. This allows preferring when translating from French to English similar word positions between the two languages, as is often the case in human translations. Under the assumptions of Model 2, (2.6) becomes:

$$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^m p(a_j|j, m, l) p(f_j|e_{a_j}) \quad (2.8)$$

Model 2 introduces a further prominent component of MT models, namely a non-trivial *reordering model* aiming at distinguishing between good and weak reorderings of translated, in this case lexical, sentence components.

IBM Models 3 to 5 Even though the IBM models are founded on a word-based view of translation, many translation phenomena involve more than single words in each of the two languages, with some examples already discussed in section 2.2.1. Models 3 to 5 venture to capture the translation of single target words e into multiple source words. The tendency of certain e words to align to more than a single source word is modelled through target word *fertility* distributions, providing the probability of a given number of alignments leading to e . The reordering models then allow capturing a preference for these commonly aligned f words to be clustered together in the word order of sentence \mathbf{f} .

HMM Alignment Model A different approach to the same problem is found in (Vogel et al., 1996). The Hidden Markov Model (HMM) for word alignment treats the clustering of translated words by modelling alignment as a Hidden Markov process.

$$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \prod_{j=1}^m p(a_j|a_{j-1}, l) p(f_j|e_{a_j}) \quad (2.9)$$

The latent alignment variable is modelled in a Markovian fashion, with every alignment point conditioned on that of the previous source word under distribution $p(a_j|a_{j-1}, l)$. Target words are then emitted conditioned on the source word that they are aligned to, as in the IBM models. This allows modelling in a simpler fashion the movement of clusters of translated target words in the source sentence than the IBM models.

While a detailed examination of these last models is beyond the scope of this thesis, it is important to note that the need to model phrasal translations was already recognised in word-based approaches, attempting to address this problem through word-based modelling steps. Phrase-based translation, discussed later in this chapter, capitalises on the advancements in word-based MT to directly model phrasal translation phenomena.

2.2.3 Estimation

The parameters θ for these word-based translation models are trained with Maximum-Likelihood Estimation (MLE) on a training parallel corpus \mathcal{X} of sentence pairs

$\langle \mathbf{e}, \mathbf{f} \rangle$. Maximising the likelihood \mathcal{L} of the corpus to retrieve MLE estimate $\hat{\theta}$, boils down to maximising the conditional translation probability of independently emitting source sentences given target sentences, as the language model probabilities are constants given the sentence pairs.

$$\mathcal{L}(\mathcal{X}; \theta) = \prod_{\langle \mathbf{e}, \mathbf{f} \rangle} p(\mathbf{e}, \mathbf{f}; \theta) = \prod_{\langle \mathbf{e}, \mathbf{f} \rangle} p(\mathbf{e})p(\mathbf{f}|\mathbf{e}; \theta) \quad (2.10)$$

$$\hat{\theta} = \arg \max_{\theta} \mathcal{L}(\mathcal{X}; \theta) = \arg \max_{\theta} \prod_{\langle \mathbf{e}, \mathbf{f} \rangle} p(\mathbf{e})p(\mathbf{f}|\mathbf{e}; \theta) = \arg \max_{\theta} \prod_{\langle \mathbf{e}, \mathbf{f} \rangle} p(\mathbf{f}|\mathbf{e}; \theta) \quad (2.11)$$

All the word-based translation models of this section interpret translation as the latent alignment variable \mathbf{a} . Using equation (2.5), we rewrite the search for the MLE estimate as:

$$\hat{\theta} = \arg \max_{\theta} \prod_{\langle \mathbf{e}, \mathbf{f} \rangle} \sum_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e}; \theta) \quad (2.12)$$

For all the models discussed, solving this optimisation problem cannot be addressed analytically. Instead, after initialising the parameter set, an instance of the Expectation Maximisation algorithm iteratively climbs the likelihood function returning a series of estimates, each increasing the training data likelihood until a local maximum is reached. The models presented here largely succeed each other by refining the parameter set. For this, they are usually trained in a pipeline with the more complicated models initialising some of their parameters using the estimates of simpler models.

Even though EM does not usually manage to find the global maximum of the likelihood function with respect to the model parameters, the word-based translation models were noted for employing a clear learning objective during training. This is a property not shared by the latter, phrase-based models discussed next.

2.2.4 The Word-Alignment Task

Overall, despite the fact that the word-based models are complete translation models, they were little used for translation proper. Instead, they were repurposed to *word-align* parallel corpora, i.e. retrieving for every sentence pair the alignment $\hat{\mathbf{a}}$ which maximises $p(\mathbf{f}, \mathbf{a}|\mathbf{e})$. Since all the models discussed are based on the assumption that every f aligns to a single e word, more complicated alignment patterns can be attained by aligning across both translation directions, computing $\hat{\mathbf{a}}_1$ for target to source and $\hat{\mathbf{a}}_2$ for source to target. A function of $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$ can then be employed to arrive at a hopefully more comprehensive alignment set, which needs not obey the constraints imposed on the alignment space by the IBM models (one alignment point per source word). Choices for this are the intersection of the

two alignment sets, their union, or heuristic functions to retrieve an alignment set between intersection and union (Och et al., 1999; Och and Ney, 2004).

The usage of word-based models for word-alignment was and still remains crucial for the development of improved translation models. All but one (the Joint Translation Model (Marcu and Wong, 2002)) of the MT models presented in the rest of this chapter utilise training methods which assume the word-alignments given, which in practice relates to alignments induced from the parallel corpus using word-based translation models.

2.3 Phrase-Based Translation

Many translation phenomena span across multiple words. Examples include idiomatic expressions, agreement between words, meaning differences according to the surrounding context and local reordering. Modelling such occurrences through word-based means often leads to awkward models that are difficult to train and decode with.

This motivated modelling phrasal translations directly, by means of memorising phrasal translation fragments and learning a model employing them to translate. In this section we will restrict ourselves to approaches based on *contiguous* phrases, while the next section treats *hierarchical* phrasal translation through a recursive process. We first discuss conditional probability phrase-based models, whose estimation is largely based on heuristics. We then subsequently present a method that has been proposed to estimate phrase translation probabilities with a clearer objective function: a joint probability phrase-based model.

2.3.1 Conditional Log-Linear Models

While phrase-based models were already introduced in earlier work such as (Wang and Waibel, 1998; Och and Weber, 1998), modern Phrase-Based SMT (PBSMT) traces its origins in the alignment template approach (Och et al., 1999; Och and Ney, 2004). This original formulation was also based on bilingual word classes (Och, 1999). As this feature is not widely adopted today as part of phrase-based models, we drop it from the presentation below for clarity reasons.

Alignment Templates Assuming an already word-aligned corpus, an Alignment Template (AT) is a triple $z = \langle \tilde{e}, \tilde{f}, \tilde{a} \rangle$, corresponding to a bilingual phrase-pair together with the alignment points between the phrases' words. In other words, an alignment template is memorising a contiguous bilingual phrase-pair together with the internal word-alignment between the two phrases. Some examples of ATs, covering a few of the phrasal translation phenomena mentioned earlier such as local reorderings and multi-word expressions, can be seen in the

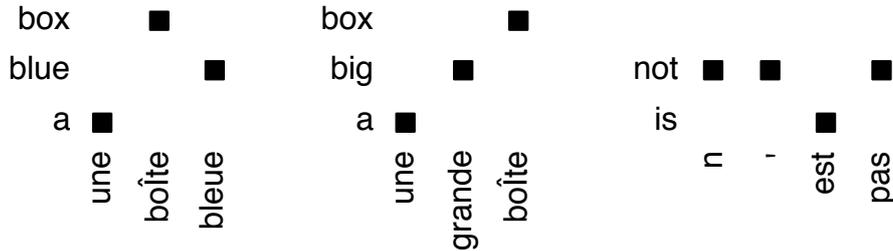


Figure 2.4: Examples of alignment templates

matrices of Figure 2.4, where blocks indicate an alignment point between the words on the two axes.

To arrive at translation \mathbf{e} given source \mathbf{f} employing ATs as building blocks, we need three latent variables, each governing source sentence segmentation, AT application and target sentence reordering, as can be seen in Figure 2.5. Firstly, the source sentence is split in K contiguous phrases according to the value of the *segmentation* variable $\boldsymbol{\sigma}$, choosing from the set of all possible segmentations $\Sigma(\mathbf{f})$. Then, for each phrase \tilde{f} of the segmented input, a sequence $\mathbf{z} = z_1^K$ of ATs is applied. Each AT $z_i = \langle \tilde{e}, \tilde{f}, \tilde{a} \rangle$ has \tilde{f}_i as its source phrase and leads to the phrase’s translation \tilde{e} as well as establishes the alignment points \tilde{a} between them. Finally, a reordering $\boldsymbol{\pi} = \pi_1^K$ of the ATs’ target sides positions with respect to the source determines the word order of the output. The conditional translation probability of an aligned sentence-pair within the AT model could then be written as a conditional generative process.

$$p(\mathbf{e}, \mathbf{a} | \mathbf{f}) = \sum_{\boldsymbol{\sigma} \in \Sigma(\mathbf{f})} \overbrace{p(\boldsymbol{\sigma} | \mathbf{f})}^{\text{segmentation}} \overbrace{p(\mathbf{z} | \boldsymbol{\sigma}, \mathbf{f})}^{\text{AT application}} \overbrace{p(\boldsymbol{\pi} | \mathbf{z}, \boldsymbol{\sigma}, \mathbf{f})}^{\text{reordering}} \quad (2.13)$$

If we assume that the AT corresponding to each of the phrases of the segmented \mathbf{f} sentence is applied independently, we have:

$$p(\mathbf{z} = z_1^K | \boldsymbol{\sigma}, \mathbf{f}) = \prod_{k=1}^K p(z_k = \langle \tilde{e}_k, \tilde{f}_k, \tilde{a}_k \rangle | \tilde{f}_k) \quad (2.14)$$

Heuristic Estimation An important point is that, in order to estimate the parameters of $p(z | \tilde{f})$ above and those of the distributions in equation (2.13) in general, the training corpus must either be already segmented in phrase-pairs or we must disambiguate between the possible segmentations of each sentence-pair. However, parallel corpora are normally not segmented and, as discussed in (DeNero et al., 2006; Mylonakis and Sima’an, 2010), estimation of the AT model’s distributions is prone to overfitting. For this, parameter estimation for the AT



Figure 2.5: The alignment template approach latent variables

model takes place heuristically, disregarding the latent segmentation variable σ .

A multiset of ATs is constructed from the word-aligned parallel corpus, by *extracting* ATs under the following heuristic rule: an alignment template is extracted once for every pair of $\langle \tilde{e}, \tilde{f} \rangle$ phrases with (a) at least one alignment point between words of the phrases and (b) all alignment points are contained within the phrase-pair, i.e. there are no alignment points from words of the phrase-pair leading to words outside the phrase pair. With $C(z)$ counting the number of times a particular AT was extracted, the conditional probability of the template given its source phrase is defined as:

$$p(z = \langle \tilde{e}, \tilde{f}, \tilde{a} \rangle | \tilde{f}) = \frac{C(z)}{\sum_{\tilde{e}', \tilde{a}'} C(\langle \tilde{e}', \tilde{f}, \tilde{a}' \rangle)} \quad (2.15)$$

Crucially, the above is *not* the MLE estimate when training on the parallel corpus, but one that uses the counts in the heuristically extracted ATs multiset. As a consequence of its heuristic nature, this estimate is not known to optimise any meaningful function of the training parallel corpus itself. As these estimates have little to do with the formulation of equation (2.13), they are instead employed as features in a log-linear conditional translation model

Log-Linear Model Formulating the AT model as a log-linear, feature-based model allows for the easier integration of additional translation features examining translation quality from different perspectives. Each feature $\phi(\mathbf{e}, \mathbf{f}, \mathbf{z}, \boldsymbol{\pi})$ assigns a non-negative score to the construction of \mathbf{e} from \mathbf{f} , using ATs \mathbf{z} under reordering $\boldsymbol{\pi}$. The features are weighted together log-linearly under weights λ , arriving at the following conditional translation model, with $Z(\mathbf{f})$ as the normalisation constant:

$$p(\mathbf{e}, \mathbf{z}, \boldsymbol{\pi} | \mathbf{f}) = \frac{1}{Z(\mathbf{f})} \exp \sum_i \lambda_i \phi_i(\mathbf{e}, \mathbf{f}, \mathbf{z}, \boldsymbol{\pi}) \quad (2.16)$$

While in principle marginalising out the latent variables $\mathbf{z}, \boldsymbol{\pi}$ is needed to arrive at the conditional translation probability $p(\mathbf{e} | \mathbf{f})$, this is in practice computationally prohibitive. Decoding is instead recast as a Viterbi search for the AT application and reordering which maximises the probability of (2.16):

$$\langle \hat{\mathbf{e}}, \hat{\mathbf{z}}, \hat{\boldsymbol{\pi}} \rangle = \arg \max_{\mathbf{e}, \mathbf{z}, \boldsymbol{\pi}} \sum_i \lambda_i \phi_i(\mathbf{e}, \mathbf{f}, \mathbf{z}, \boldsymbol{\pi}) \quad (2.17)$$

A key feature employed in the AT model (Och and Ney, 2004) is the logarithm of the conditional probability of independently selecting each alignment template as in equation (2.14), according to the heuristic scores of (2.15). This is complemented by features examining word correspondence within every template through the template’s alignment pattern, as well as a word penalty feature counting the number of target words produced, regulating target output length. A simple target phrase-reordering model based on word-position movement of target phrases provides the means to prefer mostly monotonic phrase alignments, which largely preserve the order of the source sentence as is the case for many European language pairs. Finally, the language model score of the target output \mathbf{e} is added as an additional feature focusing on target sentence well-formedness.

Phrase-based SMT The original formulation of the alignment template models in (Och et al., 1999; Och and Ney, 2004) put an emphasis on the alignment between phrase-pairs being an integral part of the memorised fragments extracted from the word-aligned training corpus. Zens et al. (2002) and (Koehn et al., 2003) depart from this view, to formulate a phrase-based SMT model. Founded on a simplified version of the assumptions of the alignment template approach, it extracts only *phrase-pairs* $\langle \tilde{e}, \tilde{f} \rangle$ where the AT approach would extract full phrasal alignment templates. The conditional phrase translation probabilities $p(\tilde{e} | \tilde{f})$ as well as $p(\tilde{f} | \tilde{e})$ are trained under a similar extraction heuristic as in (2.15).

$$p(\tilde{e} | \tilde{f}) = \frac{C(\langle \tilde{e}, \tilde{f} \rangle)}{\sum_{\tilde{e}'} C(\langle \tilde{e}', \tilde{f} \rangle)} \quad p(\tilde{f} | \tilde{e}) = \frac{C(\langle \tilde{e}, \tilde{f} \rangle)}{\sum_{\tilde{f}'} C(\langle \tilde{e}, \tilde{f}' \rangle)} \quad (2.18)$$

This leads again to a log-linear translation model as in (2.16), this time employing features $\phi(\mathbf{e}, \mathbf{f}, \tilde{e}_1^K, \tilde{f}_1^K, \boldsymbol{\pi})$. Translating \mathbf{f} under the model is performed through a Viterbi search on the space of all constructions of target sentences by applying and reordering phrase-pairs to a segmentation of \mathbf{f} , similarly to (2.17).

$$\langle \hat{\mathbf{e}}, \hat{e}_1^K, \hat{f}_1^K, \hat{\boldsymbol{\pi}} \rangle = \arg \max_{\mathbf{e}, \tilde{e}_1^K, \tilde{f}_1^K, \boldsymbol{\pi}} \sum_i \lambda_i \phi_i(\mathbf{e}, \mathbf{f}, \tilde{e}_1^K, \tilde{f}_1^K, \boldsymbol{\pi}) \quad (2.19)$$

Model Features While equations (2.16), (2.17) and (2.19) allow integrating an arbitrary set of translation features in the model, the following is a list of basic features often included in PBSMT systems.

- **Conditional phrase translation probabilities:** A score based on the logarithm of the conditional probability of independently translating each phrase, according to the heuristic scores of (2.18). Interestingly, two scores are employed, $\phi_{\text{PHR}}^{\mathbf{e}|\mathbf{f}}$ examining conditional translation of phrases from target to source and $\phi_{\text{PHR}}^{\mathbf{f}|\mathbf{e}}$ treating the opposite translation direction.

$$\phi_{\text{PHR}}^{\mathbf{e}|\mathbf{f}} = \log \prod_{k=1}^K p(\tilde{e}_k | \tilde{f}_k) \quad \phi_{\text{PHR}}^{\mathbf{f}|\mathbf{e}} = \log \prod_{k=1}^K p(\tilde{f}_k | \tilde{e}_k) \quad (2.20)$$

- **Lexical smoothing features:** These features consider the quality of phrasal translations on the lexical level. They serve as a smoothing conditional phrase translation probability value, which is particularly helpful for phrase-pairs extracted only a few times from the training corpus and for which the phrase translation probability values above are set using sparse evidence. They are based on a model similar to IBM Model 1 of equation (2.7), employing lexical translation probabilities $w(e|f)$, $w(f|e)$ estimated under relative frequency from the word-aligned training parallel corpus, with the unaligned words treated as being aligned to an additional EMPTY token. The contribution of multiply aligned words is averaged and if a phrase-pair appears with multiple alignment patterns, the maximum alignment score is used. One feature per translation direction ($\phi_{\text{LEX}}^{\mathbf{e}|\mathbf{f}}$, $\phi_{\text{LEX}}^{\mathbf{f}|\mathbf{e}}$) is also employed for this feature category.

The equations arriving at $\phi_{\text{LEX}}^{\mathbf{e}|\mathbf{f}}$ are shown below, while the values of $\phi_{\text{LEX}}^{\mathbf{f}|\mathbf{e}}$ are similarly computed based on $w(f|e)$ instead. For the phrase-pair of target phrase \tilde{e} of length \tilde{l} and source phrase \tilde{f} of length \tilde{m} , with alignment points $\langle i, j \rangle \in \tilde{a}$ between them, we have:

$$w(e|f) = \frac{C(e, f)}{\sum_{e'} C(e', f)} \quad (2.21)$$

$$p_w(\tilde{e}|\tilde{f}, \tilde{a}) = \prod_{i=1}^{\tilde{l}} \frac{1}{|\{j|\langle i, j \rangle \in \tilde{a}\}|} \sum_{\langle i, j \rangle \in \tilde{a}} w(e_i|f_j) \quad (2.22)$$

$$\widehat{p}_w(\tilde{e}|\tilde{f}) = \max_{\tilde{a}} p_w(\tilde{e}|\tilde{f}, \tilde{a}) \quad (2.23)$$

$$\phi_{\text{LEX}}^{\text{elf}} = \log \prod_i \widehat{p}_w(\tilde{e}|\tilde{f}) \quad (2.24)$$

- **Phrase reordering feature:** This feature ϕ_{RE} examines the reordering pattern of the phrasal translations. A choice employed in earlier PBSMT systems was based on distance-based scores. For example (Koehn et al., 2003) captured the movement of phrasal translations in the target sentence by providing a score proportional to the sum of the distances of each phrase’s first word to the previous phrase’s last word.

A different approach was followed by (Tillman, 2004; Koehn et al., 2005). For each phrase-pair in the training corpus, a *monotone*, *swapping* or *discontinuous* reordering event was recorded based on the target phrase’s relative position in regard to the previous source phrase’s translation. A simple model built around relative frequency estimates from these heuristic counts is used to compute the reordering feature score. This lexicalised model allows learning for example that between French and English, *grande* usually translates monotonically, while *bleue* frequently swaps in relation to the noun before it.

- **Word and phrase penalties:** A feature counting how many target words are produced helps tune output length. An additional feature counting how many phrase-pairs were used in the derivation of a translation can be employed to prefer less (i.e. larger) or more (i.e. smaller) phrase-pairs.
- **Language model:** Finally, a language model feature ϕ_{LM} is one of the most crucial features of PBSMT models, examining output well-formedness. This is typically the logarithm of the probability $p(\mathbf{e})$ assigned to the target sentence by an already trained language model.

The state-of-the-art Markovian language models employed for MT consider the target output across phrase-pair boundaries, often providing essential input to the model towards preferring overall well-formed target sentences. However, for the same reason the LM feature is also one of the most difficult to integrate in decoder implementations. A comprehensive presentation of LMs frequently used for MT can be found in (Chen and Goodman, 1998).

Feature Weights Training Log-linear models combine a multitude of feature functions together. Each of these features focuses on a different aspect of translation, while their typical value ranges often differ. A crucial part of training such a model is setting the feature weights λ , to optimally combine all the feature outputs together so that strong translations score higher. There is a growing literature of approaches and related learning objectives to perform this, including constrained entropy maximisation (Berger et al., 1996) and the Margin Infused Relaxed Algorithm (MIRA) (Crammer et al., 2006), that have been proven appealing also for SMT. Nevertheless, for the majority of phrase-based model applications including the relevant parts of this thesis, the feature weights are trained employing Minimum Error Rate Training (MERT) (Och, 2003).

Relying on optimising some information theoretical value such as test data likelihood or perplexity under the model often assumes a zero-one loss function. This ignores that apart from exact matches, there exists an ordering of alternative translations according to their appeal for humans. MT evaluation metrics such as BLEU (Papineni et al., 2002), TER (Snover et al., 2006) and METEOR (Banerjee and Lavie, 2005) try to address this deficiency by providing a numeric assessment of MT output quality which aims to correlate with human judgement. MERT focuses on optimising model parameters with respect to the quality of the model's output as measured by an MT evaluation metric, most frequently BLEU.

Let us assume a *development* set of source sentences \mathbf{f}_1^S and target reference translations \mathbf{r}_1^S . We further assume a set $\mathbf{C}_s = \{\mathbf{e}_{s,1}, \dots, \mathbf{e}_{s,K}\}$ of K candidate translations for each \mathbf{f}_s input sentence. Let the number of errors of a translation \mathbf{e} in relation to reference \mathbf{r} according to the metric in use be $E(\mathbf{r}, \mathbf{e})$ and assume the total errors over the development corpus is the sum of the errors of the individual sentences, i.e. $E(\mathbf{r}_1^S, \mathbf{e}_1^S) = \sum_{s=1}^S E(\mathbf{r}_s, \mathbf{e}_s)$. MERT is centred around optimising the feature weights λ_1^M by minimising the error of the best candidate translations $\hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M)$ between every set \mathbf{C}_s according to the model.

$$\hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M) = \arg \max_{\mathbf{e} \in \mathbf{C}_s} \sum_{m=1}^M \lambda_m \phi_m(\mathbf{e} | \mathbf{f}_s) \quad (2.25)$$

$$\hat{\lambda}_1^M = \arg \min_{\lambda_1^M} \sum_{s=1}^S E(\mathbf{r}_s, \hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M)) \quad (2.26)$$

As the criterion of (2.26) is difficult to solve analytically, (Och, 2003) proposes an approximation of it and an optimisation algorithm operating on this approximation. This is used in practice as follows. The log-linear model is used to produce an N-best list of candidate translations for each development source sentence. The optimisation algorithm is run to arrive at the feature weights which promote (i.e. assign greater probability to) the best candidate translations. However, as the new model parameters might change not only the ordering but also

the sentences present in the N-best list, a new N-best list is generated with the current best weights. These new candidate translations are added to the existing set produced during the previous steps and the optimisation algorithm is run on this expanded set of translation candidates. The process iterates until there are no novel candidate translations produced with the latest feature weights setting.

Impact of Phrase-Based models The introduction and adoption by the SMT community of the phrase-based and log-linear modelling approach had a profound impact on modern SMT, even though many of the implications did not become initially apparent. One of the most visible novelties was the generalisation of the word translation table to the new central data-structure of PBSMT systems: the phrase-table. While it seems this might have been initially conceived as merely increasing the minimal translation units set to include phrase-pairs in addition to word-pairs, it gradually became evident that this memorisation of training corpus fragments significantly changes the nature of these models and brings with it fundamental challenges in training and applying them. This crucial issue will be further discussed in Chapter 3.

PBSMT modelling also takes a distance from the noisy-channel, generative process modelling approach of the IBM models, opting instead for discriminative, feature based models. Probabilistic conditional models that examine candidate translations across *both* translation directions are combined together, as part of an array of different translation features. However, while these conditional translation probability models form the backbone of the log-linear PBSMT models, they are still estimated heuristically, disregarding the latent segmentation of sentence-pairs into phrase-pairs. This issue is touched upon by the Joint Translation Model discussed below.

Finally, training the PBSMT log-linear model parameters mostly abandons directly fitting the training or development data. Instead, most PBSMT implementations opt for translation metric (e.g. BLEU) score optimisation, as evidenced by the prevalence of the MERT method for tuning feature weights.

2.3.2 Joint Probability Model

While both the Alignment Template approach and its later development into Phrase-Based SMT directly model phrase-pairs, their training is largely based on an already word-aligned parallel corpus. Moreover, to estimate a phrasal translation model they rely on various heuristics. These range from establishing what constitutes a phrase-pair given a word-aligned sentence-pair to estimation based on normalising heuristic phrase-pair extraction counts.

Marcu and Wong (2002) propose a joint, purely phrase-based SMT model, which directly models the generation of sentence-pairs from phrase-pairs, without assuming a word-alignment variable. Instead, they generalise word-alignments in phrasal alignments, which also include phrases of length one. They then draw

from the estimation literature related to word-alignment models (such as the IBM models) to estimate the model parameters by maximising the likelihood of the training set.

In more detail, the Joint Probability Translation Model (JPTM) generative process is based on drawing a bag of phrase-pairs $\langle \tilde{e}, \tilde{f} \rangle$ from a joint distribution with probability $p(\tilde{e}, \tilde{f})$. Subsequently, the phrases of both source and target are ordered according to a position-based distortion distribution $d(pos(\tilde{e}_1^K), pos(\tilde{f}_1^K))$, employing the word positions pos of the phrases for a particular phrase ordering. With C the bag of phrase-pairs $c_i = \langle \tilde{e}, \tilde{f} \rangle$, $L(\mathbf{e}, \mathbf{f})$ the set of all such bags of phrase-pairs from whose reordering and concatenation the sentence pair $\langle \mathbf{e}, \mathbf{f} \rangle$ can be formed and assuming that each phrase-pair is independently sampled from the joint distribution, we have:

$$p(\mathbf{e}, \mathbf{f}) = \sum_{C \in L(\mathbf{e}, \mathbf{f})} \left\{ \prod_{c_i \in C} p(\tilde{e}, \tilde{f}) \right\} \times d(pos(\tilde{e}_1^K), pos(\tilde{f}_1^K)) \quad (2.27)$$

Distortion models Two models were derived from equation (2.27), employing different distortion distributions. *Model 1* assumes a uniform distortion distribution, effectively modelling jointly phrases under similar assumptions as those used by IBM Model 1 to model words conditionally. While Model 1 is shown to be able to induce reasonable phrasal alignments, it can hardly be used to translate with, given that it imposes no constraints on phrase positioning. For this reason, *Model 2* introduces a distortion distribution based on absolute word positions in a manner reminiscent of IBM model 2.

Parameter Estimation The JPTM follows the estimation principles established with the IBM models and estimates model parameters using Maximum Likelihood Estimation under the EM algorithm. However, this poses significant computational challenges. A corpus of length N with sentence pairs of length n each contains $O(Nn^4)$ phrase-pairs. For typical values of $N = 1\text{M}$, $n = 40$ this amounts to a number of phrase-pairs in the order of trillions. Moreover, each such sentence pair has $\sum_{k=1}^n k! S^2(n, k)$ different phrase-alignment patterns where $S(n, k)$ is the Stirling number of the second kind, amounting to a number in the order of 10^{83} for $n = 40$. Even though this number represents an overestimation as it includes pairs of non-contiguous phrases not covered by the JPTM, it serves as a gross indication of the computational challenges involved.

These computational challenges are addressed as follows. Firstly, only the probability for phrase-pairs that appear at least 5 times in the training corpus and are of at most length 6 is tracked, greatly reducing the size of the phrase-table. A formula employing the Stirling number estimate of phrase-pair segmentations discussed above is employed to arrive at an estimate of the expected counts of phrase-pairs given the corpus and an initial uniform joint distribution $p(\tilde{e}, \tilde{f})$.

The joint distribution is initialised using these expected counts, approximating a single step of the EM algorithm initialised with uniform $p(\tilde{e}, \tilde{f})$. These choices both restrict the number of phrase-pairs considered and provide a reasonable initial estimate for their probability.

In addition, as it is not feasible during the expectation step of the EM algorithm² to consider all phrasal segmentations and alignments, the most probable Viterbi alignment is found and fractional expected counts are computed by exploring neighbouring phrasal alignments. This approximation of the expectation step had already been used successfully in the training of the IBM Models 3 to 5. Later work such as (Cherry and Lin, 2007) tries to estimate the parameters of the JPTM by limiting the phrase-alignment search space, using the Inversion Transduction Grammar and the assumptions behind it as a modelling vehicle.

Under these constraints and approximations, the parameters for the joint phrase-pair distribution $p(\tilde{e}, \tilde{f})$ and the distortion distribution $d(pos(\tilde{e}_1^K), pos(\tilde{f}_1^K))$ are estimated. Both are employed to derive conditional distributions $p(\tilde{f}|\tilde{e})$ and $d(pos(\tilde{f}_1^K)|pos(\tilde{e}_1^K))$, which are used in a Noisy Channel decoder, which also uses a language model over the target sentences \mathbf{e} . Overall, in (Marcu and Wong, 2002) it is shown that for a corpus of 100K sentence-pairs the JPTM performs significantly better than translating with the word-based IBM Model 4.

Impact of the JPTM The Joint Probability Translation Model still provides inspiration for phrase-based translation research as a successful attempt to estimate model parameters under a clear learning objective such as Maximum Likelihood, in contrast to the heuristics employed by the PBSMT models discussed earlier in this chapter. It builds on the prior work on estimation for word-based models to propose solutions for the computational challenges involved and highlights the feasibility of better understood learning objectives for phrase-based SMT. This thesis proceeds along similar lines to also contribute in later chapters well-understood estimators for phrase-based models.

As with PBSMT models, an MLE estimator such as that approximated in (Marcu and Wong, 2002) can be shown to heavily overfit the training data, assigning non-zero probability only to full sentence-pairs. While the authors manage to arrive at usable estimates due to the particular constraints posed during estimation, this issue remains a crucial weakness of MLE estimators for phrase-based models, including the joint-probability model. It makes little sense to aim at replacing heuristic estimation with a better understood estimator when we still have to rely on dubious constraints to arrive at estimates which generalise well. In this work, we address this overfitting behaviour of MLE estimators for phrase-based models to formulate a robust training framework with no need of artificial constraints.

Furthermore, the work on the JPTM showed how a generative, joint-probability

²The steps of the EM algorithm are discussed later in this chapter, in section 2.6.

model over sentence-pairs can provide translation performance related to that attained using conditional models. However, the JPTM as implemented above did not prove in the long run to be competitive in terms of translation performance in comparison to the discriminative PBSMT models, as the log-linear formulation of the latter makes integrating diverse translation features easier. This established the understanding that conditional-probability models are superior in terms of performance to joint-probability models, a perception strengthened by the fact that Marcu and Wong (2002) as well converted their joint-probability estimates to conditional distributions prior to decoding. In later chapters of this thesis, we provide evidence that a joint-probability model can provide strong performance as the backbone of a log-linear model employing additional features.

While the JPTM modelled contiguous phrase-pairs, the authors note the possible extension to models using non-contiguous phrases. The following section explores how *synchronous* grammars, modelling the generation of strings across two languages, can be employed to translate with non-contiguous phrase-pairs.

2.4 Hierarchical SMT

At the same time as contiguous phrase-based SMT models dominated the state-of-the-art in the first half of the past decade, three influential desiderata on SMT research were established. The first and most straightforward, although by no means trivial, was translating with non-contiguous phrase-pairs. As we mention above, this was already proposed as a desired extension of the JPTM by the time of its publication. However, the theoretical and practical challenges involved in training and decoding with such models delayed their introduction. The second issue was employing a syntactic approach for MT. Inspired by the advances in monolingual syntactic parsing, this line of research aimed at applying grammatical formalisms on the bilingual string-pairs involved in MT. The final desideratum concerned taking advantage of linguistic syntactic annotations in MT modelling. These can be used for example to constrain existing models that are not otherwise linguistically motivated or as integral parts of syntactic MT approaches. The application of Synchronous Context Free Grammars on MT has interestingly provided the foundations to pursue all three goals above.

2.4.1 Synchronous CFG Grammars

Synchronous grammars in the general sense are formal grammars whose language is a set of string-pairs. Monolingual syntactic approaches have long been extended to generate, recognise and process bilingual strings. These include the syntax-directed translation (Aho and Ullman, 1969) and syntax-directed transduction (Lewis and Stearns, 1968) approaches, as well as the more recent Multiple CFGs (Seki et al., 1991) and Multitext Grammars (Melamed, 2003), all stemming from

monolingual Context Free Grammars (CFGs). CFGs are not the only formalism to be extended to parallel strings, as we also find Synchronous Tree-Adjoining Grammars (TAGs) (Shieber and Schabes, 1990) extending monolingual TAGs for bilingual parsing, as well as Synchronous Tree-Substitution Grammars (Poutsma, 2000; Eisner, 2003) generating string-pairs by combining pairs of linked syntactic subtrees.

In this work we will confine ourselves to what is currently covered by the term Synchronous Context Free Grammars (SCFGs). While these grammars trace their foundations back to (Lewis and Stearns, 1968; Aho and Ullman, 1969), they have more recently been established as syntactic formalisms for MT after the introduction of a computationally and linguistically appealing subset of SCFGs, the Inversion Transduction Grammars (Wu, 1997) and its phrase-based extension (Chiang, 2005a).

SCFGs provide an appealing formalism to describe the translation process, which explains the generation of parallel strings recursively and allows capturing long-range reordering phenomena. Formally, an SCFG \mathbf{G} is defined as the tuple $\langle N, E, F, R, S \rangle$, where N is the finite set of non-terminals with $S \in N$ the start symbol, F and E are finite sets of words for the source and target language and R is a finite set of rewrite rules. Every rule expands a left-hand side non-terminal to a right-hand side pair of strings, a source language string over the vocabulary $F \cup N$ and a target language string over $E \cup N$. The number of non-terminals in the two strings is equal and the rule is complemented with a mapping between them.

String pairs in the language of the SCFG are those with a valid derivation, consisting of a sequence of rule applications, starting from S and recursively expanding the linked non-terminals at the right-hand side of rules. *Probabilistic* SCFGs augment every rule in R with a probability, under the constraint that probabilities of rules with the same left-hand side sum up to one. The probability of each derived string pair is then the product of the probabilities of rules used in the derivation. Unless otherwise stated, for the rest of this work when we refer to SCFGs we will be pointing to their stochastic extension.

The recursive nature of languages can be extended to the relation between them that a translation process establishes. SCFGs can crucially express both the recursive nature of translation and the reordering patterns that emerge. An example of a small grammar capturing Subject-Verb-Object (SVO) to Subject-Object-Verb (SOV) reordering and recursive construction of questions between English and Japanese can be seen in Figure 2.6.

Binary SCFGs The *rank* of an SCFG is defined as the maximum number of non-terminals in a grammar rule’s right-hand side. The grammar in Figure 2.6 would be of rank 3. Contrary to monolingual Context Free Grammars, there does not always exist a conversion of an SCFG of a higher rank to one of a lower rank

$$\begin{aligned}
S &\rightarrow X_{\boxed{1}} / X_{\boxed{1}} \\
S &\rightarrow \text{Do } X_{\boxed{1}} ? / X_{\boxed{1}} \text{ ka ?} \\
X &\rightarrow NP_{\boxed{1}} VB_{\boxed{2}} NP_{\boxed{3}} / NP_{\boxed{1}} NP_{\boxed{3}} VB_{\boxed{2}} \\
NP &\rightarrow \text{I} / \text{watashi ga} \\
VB &\rightarrow \text{open} / \text{akemasu} \\
NP &\rightarrow \text{the book} / \text{hako o}
\end{aligned}$$

Figure 2.6: An SCFG rule set for SVO to SOV reordering and question construction from English to (romanised) Japanese

with the same language of string pairs. For example, even though all SCFGs of rank 3 can be converted to an equivalent one (i.e. defining the same language of string-pairs) of rank 2, the same does not apply for some SCFGs with rank 4 and above. We can convert the grammar of Figure 2.6 to one of rank 2, by replacing the third rule with the following 2 rules:

$$\begin{aligned}
X &\rightarrow NP_{\boxed{1}} Z_{\boxed{2}} / NP_{\boxed{1}} Z_{\boxed{2}} \\
Z &\rightarrow VB_{\boxed{1}} NP_{\boxed{2}} / NP_{\boxed{2}} VB_{\boxed{1}}
\end{aligned}$$

However, the following rule involving 4 non-terminals on its right-hand side cannot be binarised.

$$X \rightarrow X_{\boxed{1}} X_{\boxed{2}} X_{\boxed{3}} X_{\boxed{4}} / X_{\boxed{3}} X_{\boxed{1}} X_{\boxed{4}} X_{\boxed{2}}$$

The computational complexity and memory demands of algorithms parsing or decoding with SCFGs increases with the rank of the grammar. For this, most machine translation applications focus on SCFGs of rank two, binary SCFGs (bSCFGs) (Wu, 1997), as well as SCFGs which are *binarisable*. These are Synchronous CFGs of any rank for which a conversion to an equivalent binary SCFG exists. Fortunately, binarisable SCFGs seem to be able to cover most of the reordering patterns encountered in natural language pairs (Wu, 1997; Huang et al., 2009). This feature, coupled with the relative computational efficiency of algorithms employing bSCFGs makes the latter an appealing formalism to describe translation phenomena.

Inversion Transduction Grammars Binary SCFGs were brought to prominence by the introduction of the Inversion Transduction Grammars (ITGs) of (Wu, 1997). ITGs are a subset of SCFGs as we defined them above, where the right-hand side of rules of arbitrary length either keeps its order between the two strings or this order is inverted. Wu (1997) shows that all of these grammars can be converted to a normal form, involving either two non-terminals B, C or a word-pair $\langle e, f \rangle$. Rules leading to two non-terminals on the right-hand side can either map the two to translate monotonically across the two languages keeping the order intact, or swap the two non-terminals inverting the strings covered by them. For these grammars, we can switch to a simpler notation than that of Figure 2.6, denoting with $[]$ monotone and with $\langle \rangle$ swapping reordering. Using this notation, grammars in the ITG normal form contain only rules of the forms³:

$$A \rightarrow [B C] \qquad A \rightarrow \langle B C \rangle \qquad A \rightarrow e / f \qquad (2.28)$$

SCFG Algorithms While SCFGs are closely related to the monolingual Context Free Grammar formalism, performing tasks such as parsing with an arbitrary SCFG can be notoriously hard, with (Satta and Peserico, 2005) showing that both parsing and decoding are NP-hard. Nevertheless, while the results above apply in the general case, binary SCFGs can still be processed in polynomial time, making them an ideal candidate for practical applications. The algorithms involved then for the most usual tasks are an extension of algorithms employed for monolingual CFGs.

Parsing Parsing string-pairs using bSCFGs can be performed in polynomial time employing a modified version of the CYK algorithm (Cocke, 1969; Younger, 1967; Kasami, 1965). Running time is then $O(n^6|\mathbf{G}|)$, polynomial in both the length of each string n and the size of the grammar $|\mathbf{G}|$. However, the higher exponent makes parsing with SCFGs with the computational resources available nowadays significantly more challenging than monolingual parsing, with applications frequently having to resort to constraints or approximations.

Decoding Finding all target strings \mathbf{e} for a given source \mathbf{f} that belong in the language $\{\langle \mathbf{e}, \mathbf{f} \rangle\}$ of \mathbf{G} , or \mathbf{e} belonging to the most probable $\langle \mathbf{e}, \mathbf{f} \rangle$ according to a stochastic SCFG has interestingly a lower complexity $O(n^3)$ in respect to the sentence size. Decoding can be performed by modified versions of Earley-style parsers (Earley, 1970), such as synchronous adaptations of the CYK+ algorithm (Chappelier and Rajman, 1998). However, as we will discuss later in this chapter, state-of-the-art applications of bSCFGs for translation include the usage of a language model over the target language

³We skip productions involving an empty token in one of the two strings.

output during decoding, which complicates computations and demands non-trivial hypothesis search and pruning strategies.

Expectation-Maximization Estimating the parameters of a synchronous CFG given a corpus of parallel sentences can be performed using the EM algorithm. The Expectation step of the algorithm demands the computation of expected usage counts for all rules of the bSCFG given the current estimate. This can be performed with a modified version of the Inside-Outside algorithm (Baker, 1979; Lari and Young, 1990), running in the same complexity as SCFG parsing, namely $O(n^6|\mathbf{G}|)$.

2.4.2 The Hiero Translation System

SCFGs were initially introduced for machine translation as a stochastic *word-based* translation process in the form of the Inversion-Transduction Grammar. Simultaneously, progress in phrase-based translation showcased how translating with phrases significantly improves translation quality in comparison with word-based models. The advances in syntactic modelling of translation on the one hand and those in phrase-based translation and the related methods such as log-linear modelling for MT and MERT estimation on the other, converge with the introduction of Hierarchical Phrase-based SMT (HPBSMT) and the Hiero translation system (Chiang, 2005a; Chiang, 2007).

A key practical consideration in extending word-based ITG to the SCFG employed by Chiang is that SCFGs including phrases in the right-hand side of rules can make use of similar efficient decoding algorithms as ITGs, as long as they are binary SCFGs employing up to two non-terminals on rule expansions. Chiang takes advantage of this feature to propose an SMT system capable of employing *non-contiguous* phrase-pairs.

Synchronous Grammar Hiero is based on a synchronous grammar with a single, general-use non-terminal X covering all learnt phrase-pairs. Beginning from the start symbol S , an initial phrase-span structure is constructed monotonically using a simple ‘glue grammar’, which in practice constitutes the only rules allowed to be applied to spans larger than a predefined cut-off length⁴.

$$S \rightarrow S \boxed{1} X \boxed{2} / S \boxed{1} X \boxed{2}$$

$$S \rightarrow X \boxed{1} / X \boxed{1}$$

The true power of the system lies in expanding these initial phrase-spans with a set of recursive translation rules expanding towards non-contiguous phrase-pairs, such as those of Figure 2.7. Similarly to ITG grammars, the gaps in

⁴A usual setting of this is 10.

$X \rightarrow$	do not $X_{\boxed{1}}$ / ne $X_{\boxed{1}}$ pas
$X \rightarrow$	financial $X_{\boxed{1}}$ / $X_{\boxed{1}}$ économiques
$X \rightarrow$	this $X_{\boxed{1}}$ $X_{\boxed{2}}$ / cette $X_{\boxed{1}}$ de $X_{\boxed{2}}$
$X \rightarrow$	$X_{\boxed{1}}$'s common $X_{\boxed{2}}$ policy / politique $X_{\boxed{2}}$ commune de $X_{\boxed{1}}$

Figure 2.7: Hiero SCFG rules for English and French

the two sides of these phrase-pairs, as expressed by the X non-terminals in the synchronous productions, are mapped to each other so that the linked spans translate either monotonically or swap. However, long-range reordering is handled by the glue rules, limiting these reorderings to translating monotonically. This, together with the use of a single non-terminal X apart from the start symbol highlights that employing an SCFG grammar for the Hiero system is more of a vehicle to model and decode with non-contiguous phrase-pairs, than an attempt to learn and exploit the hierarchical structure of parallel data.

Nevertheless, non-contiguous phrase-pairs with binary reordering greatly increase the descriptive power of the phrase-table. In the first place, they allow memorising phrase patterns whose words might lie far apart in the training corpus, without the need to couple them to the particular in-between context that they appear with. Secondly, they provide the means to learn context-driven reordering patterns, some examples of which can be seen in Figure 2.7. This reduces the need for a separate reordering model such as those employed for PBSMT, with the original Hiero system relying solely on the non-contiguous phrase-pairs to reorder during decoding.

Even more crucially, non-contiguous phrase-pairs offer a generalisation of the phrase-table that reduces the effect of data sparsity. Taking the famous ‘ne . . . pas’ negation construction in French as an example, PBSMT models can memorise and reuse during decoding only translations of instances of it appearing in the training data with particular verbs, such as ‘ne veux pas’ or ‘ne peux pas’. Phrase-tables that employ non-contiguous phrase-pairs such as Hiero’s are able to successfully generalise these instances to ‘ne X pas’, greatly expanding the generalisation power of the phrase-table.

This power does not come without its challenges. As the space of possible contiguous phrase-pairs that the rules of the synchronous grammar can lead to increases, so does the need to avoid generalising towards erroneous phrase translations. For example we would like to somehow consider as more probable

expanding the non-terminal X of ‘ne X pas’ towards a verb than a noun. The grammar of the HPBSMT does not consider this and instead relies on additional features and most prominently on the language model feature to disambiguate between stronger and weaker expansions.

Finally, restricting the right hand side of the SCFG rules to two non-terminals also limits the descriptive power of bSCFG grammars as they can only cover binary reordering patterns. However, given the evidence that most of the actual reordering taking place in natural language pairs does follow these constraints as discussed in (Huang et al., 2009), it might well be that this shortcoming is actually a strength of bSCFGs. Namely, that they greatly decrease the size of the search space in a manner that not only improves computational efficiency, but also correlates with the transformations found in natural language translation, avoiding search errors and preventing distributing probability to translations that have little to do with natural language correspondences.

Translation Model and Training Establishing the form of the translation model and training follows the same pattern as PBSMT models. Firstly, the grammar is built by complementing the fixed glue-grammar rules with non-contiguous phrase-pair rules like those of Figure 2.7, which are extracted from the training corpus. Non-contiguous phrase-pair extraction from a word-aligned parallel corpus follows the same heuristics in regard to what constitutes a phrase-pair as contiguous phrase-pair extraction in alignment template and PBSMT systems. In addition however to pairs of contiguous phrases, Hiero extracts also phrase-pairs with ‘gaps’, from those which include internal spans that are themselves considered phrase-pairs. This process continues recursively, extracting rules that include up to two ‘gaps’ on the right hand side. The phrasal alignment pattern between the internal phrase-pairs is preserved by the mapping of the X non-terminals in the rule’s right-hand side. All grammar rules created in this process have again the single non-terminal X as their left-hand side.

The grammar extracted is further augmented to become a *weighted* bSCFG by assigning weights to the extracted rules using similar heuristics based on extraction counts as these used in PBSMT, for example by normalising extraction counts per source or target right-hand side. The weighted bSCFG provides a score for a derivation as the product of the weights of the rules taking part in it. This score is used as a feature to form part of a log-linear, feature-based translation model. For every derivation D with \mathbf{f} as the source string, $p(D)$ is proportional to a log-linear function of a language model feature and a number of additional features examining each rule application r independently from the rest of the derivation.

$$p(D) \propto \phi_{\text{LM}}^{\lambda_{\text{LM}}} \times \prod_{r \in D} \prod_{i \neq \text{LM}} \phi_i^{\lambda_i}(r) \quad (2.29)$$

The feature weights λ are trained by Minimum Error Rate Training, in the same way as the similar log-linear models employed for phrase-based translation.

Equation (2.29) can be misleading, given that the bSCFG could be considered as merely providing the scores for a handful of features among a multitude of these. On the contrary, the bSCFG functions as the *backbone* of the log-linear model, as the space of translations \mathbf{e} considered for the input sentence \mathbf{f} are exactly those which take part in string pairs $\langle \mathbf{e}, \mathbf{f} \rangle$ in the language of the bSCFG. In addition, the importance of the weighted bSCFG scores is central, given that the rest of the features are either monolingual (as the LM feature) or are smoothing features similar to those used in PBSMT models.

Impact of the Hiero system At first sight, Hiero introduces a hierarchical phrase based translation system capable of employing non-contiguous phrase-pairs. This allows to generalise the PBSMT phrase-table and address training data sparsity by enabling the memorisation and reuse of non-contiguous phrase patterns, disentangled from their particular application in the parallel corpus.

More importantly though, Hiero showcased a *syntactic* approach for SMT that was both computationally scalable and offered state-of-the-art performance. The fact that Hiero offered a simple instantiation of such an approach stimulated further research in a number of open questions.

- As is the case with PBSMT’s phrase translation probabilities, the weights assigned to bSCFG rules play a central role in discerning strong from weak translations. How can we estimate these crucial model parameters with a more meaningful learning objective than heuristic estimation ?
- The hierarchical and compositional nature of syntactic SMT can lead to weak generalisations if the structural part of the synchronous grammars does not focus on productions which make linguistic sense. How can we learn a richer syntactic MT structure, possibly taking advantage of linguistic syntax, which better models the hierarchical nature of natural language translation ?

These questions will be the central themes of chapters 5 and 6 of this thesis.

2.4.3 Linguistically Augmented Hierarchical Translation

While the Hiero system exhibited a syntactic approach to MT it stopped short of employing *linguistic* syntax. The latter provides linguistic annotations to sentences which are generally understood to correlate with translation transformations up to a certain extent. Even more, these annotations are recursive, a feature which promotes them as prominent candidates to be integrated in a hierarchical translation system.

We may categorise MT systems employing linguistic syntax in two categories. The first brings together systems where linguistic syntax is the primary vehicle to describe the translation process. An example could be models which explain translation through transformations on the source parse tree or which build the target sentence through combining target parse fragments, such as (Yamada and Knight, 2001; Galley et al., 2006). The second category includes systems which, recognising their informative value for translation, take advantage of linguistic annotations while not strictly adhering to them to explain the translation process. For example, in (Venugopal et al., 2009; Chiang et al., 2009) the linguistic plausibility of translations is assessed through additional features in an otherwise non-contiguous phrase-based system. Below we focus on SAMT, a linguistically enriched extension of the Hiero system.

Syntax Augmented MT The Syntax Augmented MT (SAMT) system (Zollmann and Venugopal, 2006) can be classified in the latter category. It extends HPBSMT by extracting linguistically augmented hierarchical translation rules. For this it utilises constituency parse trees of the *target* sentences. Whenever a rule like those in Figure 2.7 is extracted, if the spans substituted by the generic non-terminal X are also covered by a constituent non-terminal NT in the target sentence parse, additional rules substituting X with NT are extracted. This leads to SCFG rules enriched with the use of a linguistically augmented non-terminal set, such as those in Figure 2.8. The rest of the SAMT model details mostly follow those of the Hiero system. A log-linear model employs features based on heuristic extraction counts and feature weights are optimised by MERT.

The rules of Figure 2.8 delineate how linguistic annotations are used towards an SCFG grammar which is, depending on the assigned rule scores and feature weights, possibly more selective in the recursive expansion of non-contiguous phrase-pairs. This is performed without completely committing to the linguistic structure itself. This is further exemplified by an additional set of rules employed by SAMT, which are augmented by SCFG non-terminals crossing linguistic constituent brackets. These may substitute X in the extracted rules when the underlying phrase-pair is a concatenation $NT1 + NT2$ of two or more constituents. They can also constitute a partial syntactic category $NT1$ missing a non-terminal $NT2$ on its right $NT1/NT2$ or its left $NT1\backslash NT2$, as in Categorical Grammar (Bar-Hillel, 1953). Examples of rules utilising these non-terminal categories appear in Figure 2.9

Overall, SAMT highlights a flexible approach on how to utilise linguistic syntax of the target language for MT. Decoding with a SAMT model results in the construction of target syntactic analyses which are able to take advantage of SCFG rules that are more linguistically aware than Hiero, hopefully producing translations which are more grammatically sound.

$X \rightarrow$	do not $X_{\boxed{1}}$ / ne $X_{\boxed{1}}$ pas
$X \rightarrow$	do not $VB_{\boxed{1}}$ / ne $VB_{\boxed{1}}$ pas
$VP \rightarrow$	do not $VB_{\boxed{1}}$ / ne $VB_{\boxed{1}}$ pas
$X \rightarrow$	financial $X_{\boxed{1}}$ / $X_{\boxed{1}}$ économiques
$NP \rightarrow$	financial $NN_{\boxed{1}}$ / $NN_{\boxed{1}}$ économiques
$X \rightarrow$	$X_{\boxed{1}}$'s common $X_{\boxed{2}}$ policy / politique $X_{\boxed{2}}$ commune de $X_{\boxed{1}}$
$X \rightarrow$	$X_{\boxed{1}}$'s common $JJ_{\boxed{2}}$ policy / politique $JJ_{\boxed{2}}$ commune de $X_{\boxed{1}}$
$NP \rightarrow$	$X_{\boxed{1}}$'s common $JJ_{\boxed{2}}$ policy / politique $JJ_{\boxed{2}}$ commune de $X_{\boxed{1}}$

Figure 2.8: SAMT SCFG rules for English and French, extending Hiero's X -rules

$DT + NN \rightarrow$	$DT_{\boxed{1}}$ $NN_{\boxed{1}}$ / $DT_{\boxed{1}}$ $NN_{\boxed{1}}$
$NP \setminus DT \rightarrow$	financial $NN_{\boxed{1}}$ / $NN_{\boxed{1}}$ économiques
$S \setminus VP \rightarrow$	financial $NN_{\boxed{1}}$ / $NN_{\boxed{1}}$ économiques
$VP \setminus VB \rightarrow$	$DT + NN_{\boxed{1}}$'s common $X_{\boxed{2}}$ policy / politique $X_{\boxed{2}}$ commune de $DT + NN_{\boxed{1}}$

Figure 2.9: SAMT rules with compound non-terminals

2.5 Limits of Estimation Heuristics

SAMT also serves to exemplify the limits of heuristic rule scores based on rule extraction heuristics. Their usage was already debatable when counting contiguous or non-contiguous phrase-pairs in PBSMT and its hierarchical extension, given that this ignores the latent segmentation variable splitting sentence-pairs down to phrase-pairs. We might argue that this issue is somehow mitigated by the fact that at least we are counting events (phrase-pairs) in the observed part of the data (sentence-pairs), even though we skip explaining how we arrived there (segmentation).

In approaches like SAMT, we, somehow silently but still crucially, move forward to heuristically extract counts from the latent structure itself: the Synchronous CFG parse of the sentence-pair. This parse, for the linguistically augmented rules of SAMT, goes a long way past a simple segmentation in non-contiguous phrase pairs. While extracting event surface counts without completely disambiguating the latent structure underlying them might already feel uncomfortable, relying on artificial counts over the unobserved variables can completely undermine our confidence in the scores derived from them.

This issue is shared by all MT approaches employing a richer syntactic structure to explain the translation process (generative models) or discriminate between strong and weak translations (discriminative models). As the latent structures involved in these models become more complex, the risks from heuristically estimating the parameters of these structures increase. Crucially, this also increases the possible gains from learning these latent structures from parallel data.

2.6 Expectation-Maximization Algorithm

A popular and widely successful method to estimate the parameters of generative models using training data $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ is Maximum Likelihood Estimation. Under MLE, we seek to find the parameter set $\hat{\theta}$ for a stochastic model $p(X = \mathbf{x}; \theta)$ which maximises the likelihood of observing the training set \mathcal{X} . Assuming all samples \mathbf{x} are independent and identically distributed (i.i.d), we have:

$$\hat{\theta} = \arg \max_{\theta} \mathcal{L}(\mathcal{X}; \theta) = \arg \max_{\theta} \prod_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x}; \theta) \quad (2.30)$$

For a number of modelling problems, we might be fortuitous enough to possess training data which include the outcomes of all the random variables the model assumes for every data point, i.e. we can infer the model from *complete* data. For example, for a straightforward stochastic model over a loaded dice, complete data can refer to a number of rolling outcomes for it. If we interpret the model as a generative process, we might say that we then have, for every training data

point, access to all the generative steps involved in their emission. Training a PCFG from a parsed sentence corpus (a treebank) falls under this case.

In these cases, MLE boils down to Relative Frequency Estimation (RFE), i.e. assigning to the model parameters the relative frequencies of the model’s random variable outcomes in the observed data. While a straightforward RFE estimate is generally speaking readily computable, arriving at estimates which generalise well, i.e. predict unseen data accurately, is highly non-trivial. The additional estimation efforts are then mostly directed to *smoothing*: accounting for data sparsity, the fact that our training data provides but a limited glimpse in the distribution of outcomes of random variable X .

In other cases however, we might assume a model with *latent variables*, i.e. involving random variables whose outcome is not observed in the training data. One motivation towards formulating such a model could be aiming to uncover hidden patterns in the data, such as the word-alignments between sentence pairs of Figure 2.3. An additional objective might plainly be to better model the unknown data distribution, hoping that the assumptions behind our model can aid to better describe the data. An example might be using a mixture model to fit data, when a single standard distribution such as the Gaussian is not considered to be enough to accurately describe the underlying distribution. In these cases we have to estimate model parameters using what we then consider as *incomplete* data with *missing* values. The missing information in two aforementioned examples would be respectively the word-alignments for the parallel sentences and the indication of each data point’s origin among the mixture’s distributions.

Machine Translation is a prominent ML field where estimating model parameters from incomplete data is a central issue. In most experimental settings training data are merely sentence-aligned, with no further information on how, starting from the source sentence, we arrived at the target language output. The only model for which these are considered complete data is one which tracks the translation of whole sentences as a single unit, which has very limited applicability given the sparsity of the training data. Any meaningful generative SMT model which aspires to generalise well is thus bound to employ latent variables. We have already considered such cases in this chapter, such as the word-alignments for the IBM models, phrase segmentation for PBSMT models and the hierarchical translation structure for SCFG-based models.

In this section we will examine the challenges of estimating model parameters with MLE using incomplete data and how the Expectation-Maximization algorithm can address these, focusing on discrete distributions.

MLE with Incomplete Data Let us consider a model, parametrised by vector θ , over random variable $Z = \langle X, Y \rangle$, whose values \mathbf{z} are tuples consisting of values \mathbf{x} of the *observed* data variable X and values \mathbf{y} of *missing* data variable Y . We use these names for X and Y , because the incomplete training data $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$

that we possess present only the values for X , while information over the value of Y for each data point is missing.

Given this setting, we have to rewrite the MLE criterion of (2.30) so that the unobserved variable is marginalised. For this we will need a function $Z(\mathbf{x})$ which maps every observed data point \mathbf{x} to the set of all possible complete data from which it could descend from.

$$Z(\mathbf{x}) = \{\mathbf{z}_1 = \langle \mathbf{x}, \mathbf{y}_1 \rangle, \mathbf{z}_2 = \langle \mathbf{x}, \mathbf{y}_2 \rangle, \dots\} \quad (2.31)$$

The Maximum-Likelihood Estimate of a model over complete data can then be computed from incomplete data as follows.

$$\begin{aligned} \hat{\theta} &= \arg \max_{\theta} \prod_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x}; \theta) \\ &= \arg \max_{\theta} \prod_{\mathbf{x} \in \mathcal{X}} \sum_{\mathbf{z} = \langle \mathbf{x}, \mathbf{y} \rangle \in Z(\mathbf{x})} p(\mathbf{x}, \mathbf{y}; \theta) \end{aligned} \quad (2.32)$$

The problem is that, in most non-trivial cases, the optimisation above involving a product of sums cannot be solved analytically. We may however still iteratively arrive at increasingly better fitting parameter settings, by means of the EM algorithm.

The Expectation-Maximization (EM) algorithm (Dempster et al., 1977)⁵ provides a method to iteratively arrive at a *local* optimum of the likelihood function of (2.32), many times in a computationally appealing way. It stands out from superficially similar looking iterative algorithms, as it provides theoretical guarantees over its operation and its output.

Initialisation We begin by initialising the model's parameter set by an initial setting $\hat{\theta}_0$. Parameter initialisation can be crucial as EM can only climb towards a local optimum of the data likelihood function. For complex likelihood functions with more than one local maximum, the initialisation point determines towards which of these we will converge. As a result, a weak initialisation can result in a globally suboptimal estimate.

However, as in most cases we are not aware of the likelihood function's exact form and there is usually no clear way to judge the quality of an initialisation point, popular initialisation choices are uniform distributions or a randomly set parameter set. If practically possible, it makes also sense to run the EM algorithm multiple times, each one starting from a different initialisation point (random restarts). Hopefully, each will climb towards a different local optimum and we can select the estimate which corresponds to the best local optimum reached.

⁵(Prescher, 2004) and (Bilmes, 1997) provide interesting tutorials of the EM algorithm.

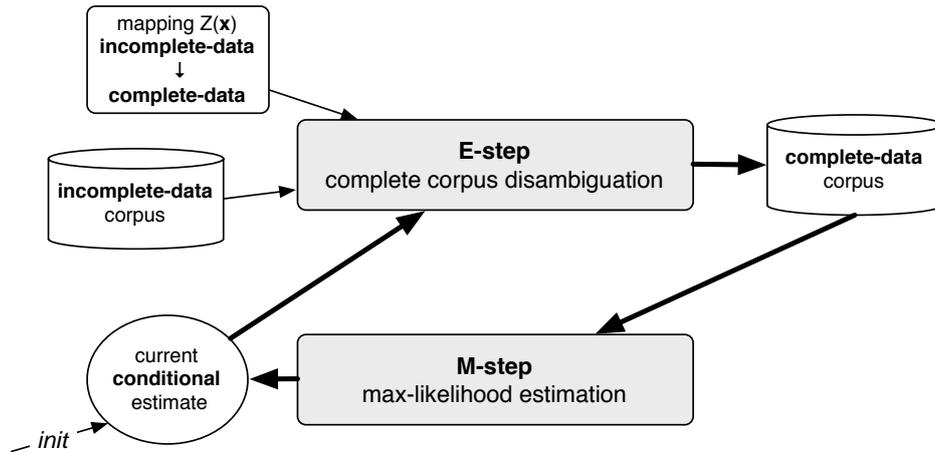


Figure 2.10: The two steps of the EM algorithm, adapted from (Prescher, 2004).

Iterative Procedure The initialisation point $\hat{\theta}_0$ together with the incomplete data training corpus \mathcal{X} and function $Z(\mathbf{x})$ form the input of the EM algorithm. The basic operating principle behind EM can be described along the following lines.

If we apply function $Z(\mathbf{x})$ to every $\mathbf{x} \in \mathcal{X}$, we can build from the incomplete-data corpus \mathcal{X} a *complete*-data corpus $\mathcal{Z}(\mathcal{X})$, extending each \mathbf{x} to all its possible complete data expansions. For example, if the incomplete corpus consists of aligned sentence-pairs, we can expand each unaligned sentence-pair to multiple aligned ones, considering all possible alignments between the words of the two strings. The problem is that we do not know how to distribute the unit count of each observed data point among all its complete-data expansions; e.g. we do not know what the count of a particular alignment pattern would be given that it's sentence pair has been observed once. If we had that information and could thus disambiguate corpus $\mathcal{Z}(\mathcal{X})$, MLE would boil down to relative frequency estimation.

The EM algorithm breaks out of this impasse following a two steps procedure. First, it uses in each iteration r the previous parameter estimate $\hat{\theta}_{r-1}$ (starting with $\hat{\theta}_0$) to disambiguate the complete-data corpus. This is performed by computing the expected counts of each complete-data expansion $\mathbf{z} \in Z(\mathbf{x})$ with respect to $\hat{\theta}_{r-1}$ (*E-step*). In the word-alignment example, we would compute for every sentence the expected counts of each possible word-alignment, as if it was produced from a model parametrised by $\hat{\theta}_{r-1}$. Subsequently, putting then $\hat{\theta}_{r-1}$ aside, EM moves on to compute a new estimate $\hat{\theta}_i$ by MLE on this disambiguated complete corpus (*M-step*). As this concerns maximising the likelihood of a corpus where for every data point all values of the model's variables are observed, this optimisation is in most cases feasible, and many times is equivalent to RFE.

Remarkably, this new estimate is guaranteed to better fit the training data.

The process then continues iteratively, each time using the previous estimate to arrive at a better one until convergence, as illustrated in Figure 2.10.

Let us now more thoroughly describe the two steps that each EM iteration consists of. We switch to the equivalent optimisation criterion of *log*-likelihood maximisation which is sometimes easier to formulate and solve. Also, to simplify the exposition, we describe the two steps for a single data point \mathbf{x} . The relevant equations easily extend to the full training corpus by summing⁶ through all the i.i.d sampled data points of the corpus \mathcal{X} .

Expectation Step In the Expectation step (E-step), we formulate the *expected* log-likelihood $Q(\theta|\hat{\theta}_{r-1})$ of the complete-data $\mathbf{z} = \langle \mathbf{x}, \mathbf{y} \rangle$, given the observed incomplete-data point \mathbf{x} and the parameter estimate from the previous iteration $\hat{\theta}_{r-1}$.

$$Q(\theta|\hat{\theta}_{r-1}) = E \left[\log p(\mathbf{z}|\theta) | \mathbf{x}, \hat{\theta}_{r-1} \right] = \sum_{\langle \mathbf{x}, \mathbf{y} \rangle \in Z(\mathbf{x})} \log \{p(\mathbf{x}, \mathbf{y}|\theta)\} p(\mathbf{y}|\mathbf{x}, \hat{\theta}_{r-1}) \quad (2.33)$$

In the somewhat abstract equation above, it is crucial to notice that $p(\mathbf{y}|\mathbf{x}, \hat{\theta}_{r-1})$, the expected counts of the complete-data expansions of \mathbf{x} given $\hat{\theta}_{r-1}$, can be readily computed and will function as a constant in the M-step that follows. Substituting it with $q(\mathbf{x}, \mathbf{y}|\hat{\theta}_{r-1})$ to denote this, we have:

$$q(\mathbf{x}, \mathbf{y}|\hat{\theta}_{r-1}) = \frac{p(\mathbf{y}|\mathbf{x}, \hat{\theta}_{r-1})}{\sum_{\langle \mathbf{x}, \mathbf{y}' \rangle \in Z(\mathbf{x})} p(\mathbf{y}'|\mathbf{x}, \hat{\theta}_{r-1})} \quad (2.34)$$

In practice, E-step implementations involve computing these expected counts. While it involves going through all possible complete-data expansions of \mathbf{x} which can be exponential in number, for many practical applications dynamic programming algorithms allow them to be efficiently computed.

Maximization Step With the $q(\mathbf{x}, \mathbf{y}|\hat{\theta}_{r-1})$ counts already computed in the E-step, the Maximization step (M-step) of the EM algorithm involves maximising $Q(\theta|\hat{\theta}_{r-1})$ with respect to θ to retrieve the next parameter estimate $\hat{\theta}_r$.

$$\hat{\theta}_r = \arg \max_{\theta} Q(\theta|\hat{\theta}_{r-1}) = \arg \max_{\theta} \sum_{\langle \mathbf{x}, \mathbf{y} \rangle \in Z(\mathbf{x})} \log \{p(\mathbf{x}, \mathbf{y}|\theta)\} q(\mathbf{x}, \mathbf{y}|\hat{\theta}_{r-1}) \quad (2.35)$$

Equation (2.35) involves optimising the model parameters from what is now, with the help of the counts computed during the E-step, a complete-data corpus. This is usually much easier than the original incomplete-data optimisation problem of (2.32) and many times translates to the usually easy to implement and compute Relative-Frequency Estimation.

⁶We are employing log-likelihood optimisation.

Theoretical Guarantees The EM algorithm’s appeal is strengthened, by the fact that it is coupled with theoretical guarantees concerning its operation and output. Dempster et al. (1977) show⁷ that the iterations of the EM algorithm provide:

Guarantee to Non-Decrease Likelihood After every iteration, the new estimate raises or leaves equal the likelihood of the incomplete-data training corpus in comparison with the estimate of the previous iteration, i.e. $\mathcal{L}(\mathcal{X}; \hat{\theta}_r) \geq \mathcal{L}(\mathcal{X}; \hat{\theta}_{r-1})$.

Guarantee to Converge The iterative process will converge to a local maximum of the likelihood function.

These guarantees distinguish EM as a well-understood and powerful optimisation algorithm for MLE using incomplete data. While other heuristic optimisation procedures might somehow work for specific tasks, we are still left unsure of the exact pre-conditions that favour their use, as well as over the quality of their output for various input data. In contrast, the two guarantees discussed above clearly delineate what EM can do for the modeller. Namely, given a starting point it will climb and converge to the ‘nearest’ local maximum of the incomplete data’s likelihood under the model.

Discussion In addition to being well-understood, the EM algorithm has proven its effectiveness as an essential estimation tool for various machine learning tasks for more than three decades. Its popularity however has sometimes led to the formulation of iterative estimation procedures, which are then casually presented as ‘EM’-algorithms. This confusion is further increased by the fact that EM is more of an algorithmic framework, which still needs to be applied for every estimation problem, than a concrete set of instructions. However, only true instances of the Expectation-Maximization algorithm, following the principles described in this section, inherit the algorithmic guarantees associated with EM. It is important for the Machine Learning practitioner to distinguish between EM and ‘EM-like’ algorithm instantiations.

EM’s magnificent ability to fit latent variables over observed data has in the past sometimes led to its promotion as an omnipotent out-of-the-box tool to perform unsupervised induction of hidden patterns in data, like parses, part-of-speech tag sequences and others. EM is certainly effective in this task, as showcased in its application for the induction of word-alignments under the IBM Models discussed in this chapter. However, EM merely provides us with a tool to fit models involving latent variables to incomplete data. The extent to which this will be helpful in inducing interesting latent patterns is linked to the following factors:

⁷Somewhat more accessible proofs of EM’s algorithmic properties can be found in (Beal, 2003; Chen and Gupta, 2010).

Model We need to consider the model’s ability to effectively describe the data through latent variables. The increasingly more refined IBM Models again showcase how better models induce better latent patterns.

Maximum Likelihood Objective As EM performs MLE, we must assess the appropriateness of the ML objective for our problem. For models using more parameters than necessary to capture the regularities underlying the data, MLE might overfit the data producing degenerate estimates, as we further discuss in Chapter 3. Also MLE coupled with the model at hand might reveal latent patterns that have little to do with what we were after, e.g. sentence bracketings similar to human-derived references.

Initialisation Point EM converges towards a local maximum beginning from the initialisation parameters setting, which, depending on the form of the likelihood function, can greatly affect the algorithm’s output (e.g. see (Goldberg et al., 2008)).

Number of Iterations Depending on the application, sometimes it is better to stop the algorithm well before convergence to avoid overfitting, as is usually performed during word-alignment with the IBM Models. Other times, EM needs surprisingly many iterations to arrive at a good estimate, as discussed in (Johnson, 2007).

Overall, Expectation-Maximization stands out as a highly potent estimation algorithm, whose careful empirical application can lead to discovering latent patterns that go well beyond the surface of the observed training data.

2.7 Generalisation Error and Cross-Validation

Most statistical estimators for parametric models, including Maximum-Likelihood Estimation with which we are primarily occupied in this thesis, select model parameters by fitting the model to the training data. That entails optimising the parameters so that training data error, as computed according to an error function⁸, is minimised. In general however, our interest in the model estimate goes well beyond the training data, as our primary concern relates to its *prediction* capability on independent test data. The problem is that frequently, minimising loss on the training data does not necessarily translate to reducing error on test data. Below we try to localise the reasons for this, by distinguishing between two sources of estimator errors, bias and variance.

In addition, we discuss Cross-Validation (CV), a method to arrive at an estimate of the *Generalisation Error* (GE) of a model estimate, i.e. the expected error over all the independently drawn test sets. As for most applications GE

⁸Also sometimes called a loss function.

is exactly the error we would like to minimise, CV can aid in model assessment, evaluating the expected performance of a particular model estimate derived from the training data on yet unseen data. In the next chapter we will also present how CV can be used directly for model estimation, finding the model estimate which is most expected to perform well with future data.

2.7.1 Estimator Bias and Variance

An estimator can be defined as a function of the data which, for a particular data set, returns an estimate of a given quantity. This quantity can be for example a number, like an estimate of the true average of a random variable, but it can also be our estimate of the true distribution from which we are sampling. Two characteristics of estimators that relate to their generalisation error are estimator bias and variance.

Estimator *bias* is the accuracy of our average estimate (as measured by our error function), when we average over all training sets \mathcal{X} that we can sample. *Asymptotically unbiased* estimators converge to the true value of the quantity estimated as the training set size approaches infinity. This is an appealing property as it guarantees that an estimator will ultimately arrive at an accurate estimate given enough data.

However, we never possess training data of sizes close to infinite. Somewhat surprisingly, an unbiased estimator for smaller training sample sizes frequently produces a large generalisation error. Bias relates to the strength of prior assumptions employed by the estimator, with an unbiased estimator enforcing no such assumptions over the quantity estimated, opting instead to completely rely on the training input. The estimates may become then too sensitive to the training input.

This can lead to increased *variance* between them for different training sets, which entails that many of them will deviate significantly from the true value, leading to generalisation errors. Lowering GE due to estimate variance usually entails increasing our assumptions over the quantity estimated and in this way abstracting away from the training data, i.e. increasing the estimator's bias. Still, at the other extreme, an estimator with zero GE due to estimate variance always outputs the same estimate irrespective of the training data. Unless our strict assumptions behind this estimate are somehow correctly guessed, this is bound to lead to a large GE.

This establishes a trade-off between errors due to estimator bias and those due to variance, where decreasing errors due to bias increases errors due to variance and vice versa. The curve-fitting example of Figure 2.11⁹ showcases this trade-off. Both low bias as well as low variance estimators return estimates which widely deviate from the true function $f(x) = x^2$ behind the three noisy samples.

⁹Adapted version of a similar example in (Duda et al., 2001).

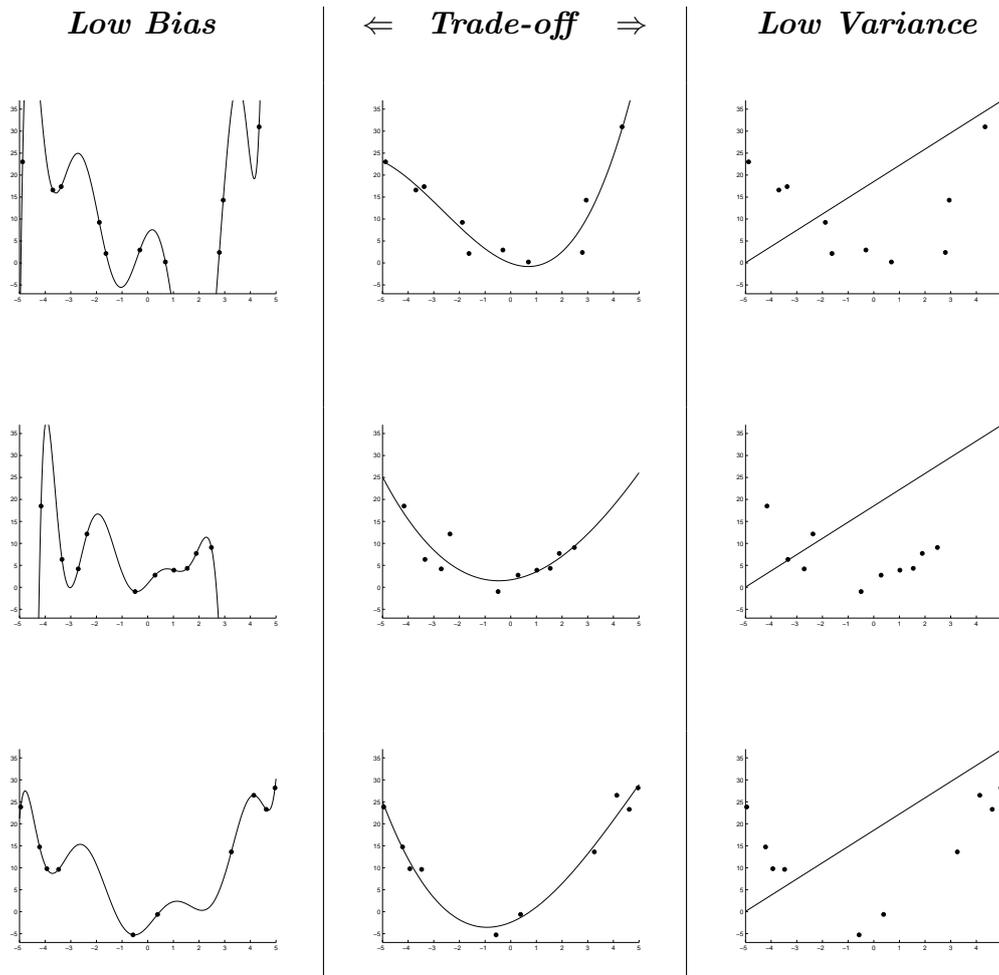


Figure 2.11: A *low bias* estimator (9-th degree polynomial) precisely fits each sample, but is penalised in terms of estimate variance. A *low variance* estimator (fixed linear) has zero estimate variance, but its high inherent bias results in a bad estimate of the underlying function. A *trade-off* between bias and variance (cubic) is needed to lower the Generalisation Error.

The challenge then lies in finding the correct balance between estimator bias and variance which minimises generalisation errors overall, as is the case in the second column in the figure.

Bias-Variance Decomposition We may gain additional understanding in the source of an estimator’s errors using the GE’s *bias-variance decomposition*, breaking down the GE into terms attributed to estimator bias and variance respectively. This decomposition relies on the kind of the estimation and the error function used. In the context of this thesis, it is interesting to consider estimators where the target of our estimation efforts is the distribution generating the data we model.

Assume that we wish to recover the target distribution q by means of an estimator \hat{p} returning the probability estimate $\hat{p}(\mathcal{X})$ when trained on training set \mathcal{X} . A sensible error function in this setting can be the Kullback-Leibler (KL) divergence between the target q and estimate \hat{p} . For the distinct random variable case this is:

$$KL(q, \hat{p}(\mathcal{X})) = \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{\hat{p}(\mathbf{x}; \mathcal{X})} \quad (2.36)$$

Denoting with $E_{\mathcal{X}}$ the expectation over all training samples, generalisation error is then the expected KL-divergence between q and \hat{p} .

$$Err = E_{\mathcal{X}} KL(q, \hat{p}) = E [KL(q, \hat{p}(\mathcal{X})) | \mathcal{X}] \quad (2.37)$$

Heskes (1998) shows that the GE Err can then be decomposed in bias and variance terms. The bias term is the KL-divergence between q and the mean estimate over all training data $\bar{p} = E_{\mathcal{X}} \hat{p}(\mathcal{X})$. Variance is the expected divergence between the average estimate and the estimator’s actual choice for each training input \mathcal{X} .

$$Err = \overbrace{KL(q, \bar{p})}^{bias} + \overbrace{E_{\mathcal{X}} KL(\bar{p}, \hat{p})}^{variance} \quad (2.38)$$

An example of an unbiased estimator in this setting would be one which only predicts the training data according to their frequency in the training set. It is easy to show that the average over all sampled training sets \bar{p} would then coincide with the target distribution q leading to a zero bias term. However, excluding random variables taking only a handful of values or having access to extremely large training sets, the variance term of an unbiased estimator becomes unboundedly large, leading to a large GE. In Chapter 3, we shall revisit the bias-variance decomposition in the context of Fragment Models.

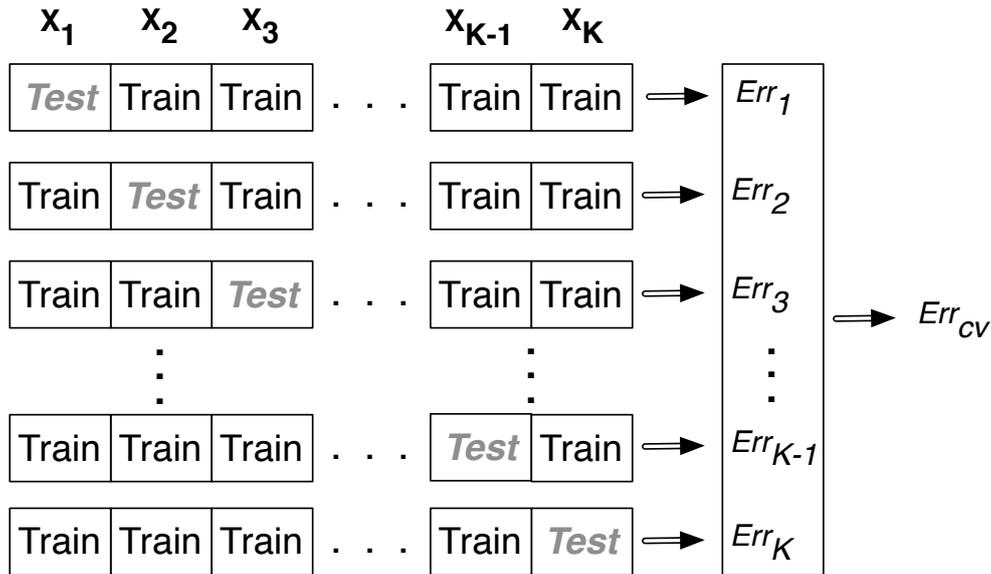


Figure 2.12: K-fold Cross-Validation.

2.7.2 Cross-Validation

The Bias-Variance analysis of estimator Generalisation Error highlights that low training data error alone does not always translate to low GE, due to training data *overfitting*. While providing low training set error remains an intuitive model selection criterion, we need to discriminate between model-estimator combinations which are able to capture the underlying random variable's statistical properties and those which fixate on the training sample's peculiarities.

Given enough data, we could set aside a *validation* set, the error on which can aid in assessing GE. Nevertheless, reserving data for the validation set reduces the size of the training set. This is further aggravated from the fact that often, in order to attain a reasonable estimate of the GE, the size of the validation set must be substantial. Validating thus on a reserved data set is inefficient in employing training data and its use is often prohibitive when assembling training sets is particularly costly.

A simple but highly effective method to arrive at an estimate of the GE without sacrificing possibly scarce training data is *K*-fold Cross-Validation (CV) (Hastie et al., 2001; Duda et al., 2001). The basic concept is using part of the training data to fit our models and a different *holdout* part to test them, while rotating which part functions as the holdout set. This allows a more efficient usage of our data, as in the end we allow all data points to take part in both fitting the model as well as validating its generalisation capacity over new data.

More precisely, we begin by splitting the data in *K* roughly equal-sized parts $X^1 \dots X^K$. For every $1 \leq k \leq K$, we test against part X^k a model trained on the

rest of the data X^{-k} . In this manner, K estimates of the generalisation error are computed, which we can further combine together to output an overall estimate of the GE, for example by averaging them together. This process is depicted in Figure 2.12.

Usual settings for K which have been shown to work well for a range of modelling problems (Kohavi, 1995) are in the range of 5 to 20, with 10 a popular choice. The case when K is equal to the size of the training set X is referred to as *leave-one-out* CV, as in each CV-round we hold out a single training data point.

CV as an Estimator of GE Cross-Validation is itself an estimator of the Generalisation Error of model-estimator combinations. Due to the No-Free-Lunch Theorem (Wolpert, 1996), which is applicable to all estimators, we cannot of course prove that CV is an overall superior estimator of the GE under all circumstances.

Nevertheless, it has been shown (Kohavi, 1995) that, under some assumptions, CV is both an unbiased as well as low-variance estimator of the Generalisation Error, promoting CV as a highly accurate estimator of GE. The key assumption for this to hold is that the estimators tested under CV are *stable* under the perturbations of the training data set during CV. In other words, that their predictions do not change when trained on the training set with a CV holdout part removed. While this assumption does not strictly hold for most estimators and experimental settings, this result exhibits that CV is expected to be a good estimator of prediction error when for the estimator, training data and number of CV-folds chosen, the predictions of the estimator do not greatly change when presented with the CV holdout parts removed.

Practical Applications Apart from these theoretical properties, CV has been also shown to provide a low-bias, low-variance estimator of GE for a host of ‘real-life’ problems (see e.g. (Kohavi, 1995; Schaffer, 1993)). In addition, CV has also found numerous applications for NLP problems. Examples include estimating back-off parameters of Language Models (Jelinek and Mercer, 1980; Kneser and Ney, 1995), as well as estimating the parameters of Data-Oriented Parsing models (Zollmann and Sima’an, 2006) and selecting the feature set of a discriminative parsing model (Collins, 2000).

Cross-Validation is mostly applied in the context of *model selection*, picking out the model which the best prediction properties. In the next chapter, we take advantage of the theoretical and practical appealingness of CV as an estimator of Generalisation Error to formulate a model *parameter estimation* objective, which aims at increased generalisation over yet unseen data. We find that this learning objective is a preferred alternative to plain Maximum-Likelihood Estimation for models with a strong tendency to overfit training data. We then discuss this in

detail for Fragment Models: a family of models which has already been employed with success for syntactic parsing and machine translation.