Learning the latent structure of translation
Mylonakis, M.

Citation for published version (APA):

General rights
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, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 3

Fragment Models Estimation with the CV-EM Algorithm

Machine Learning problems frequently involve data with an unobserved hidden structure, and these data typically cannot be described by a mere low-dimensional vector. Examples include face and character recognition that work with matrices of image pixel values, financial fraud detection operating on sequences of financial transactions, and automated medical diagnosis systems accepting as input vectors of patient medical variables. In the same category fall many of the problems in NLP like language modelling, speech recognition, parsing and machine translation.

In the first part of this chapter we occupy ourselves with modelling such complex data. We begin by a discussion of the overall challenges involved, examining in more detail the interesting case posed by natural language data. We further concentrate on generative models and treat the case of Fragment Models. These define distributions over the data modelled, by specifying how fragments of arbitrary sizes extracted from the training data can be combined together to produce novel data instances.

The highly expressive Fragment Models are nevertheless notoriously difficult to train, as they are known to have a strong tendency to overfit training data. Motivated by this, we propose a Cross-Validated MLE (CV-MLE) estimation objective and contribute the Cross-Validated Expectation-Maximization (CV-EM) algorithm. This is a general estimation algorithm, which employs the Cross-Validation criterion to induce models generalising well on yet unseen data. We show that CV-EM enjoys an array of appealing algorithmic properties, preparing the grounds for its application in the following chapters of this thesis.
3.1 Fragment Models

3.1.1 Modelling Complex Data

We employ the term ‘complex data’ to refer to instances whose representation demands a large number of numerical values, with complex patterns between these that cannot trivially be captured in a low-dimensional space. We contrast these to simpler data involving a handful of values per data point, like temperature readings from a single sensor, a collection of basic measurements of people like height and weight or a country’s key financial variables. While working with simpler data can also be highly non-trivial, modelling problems involving complex data share a number of common challenges.

Perhaps the most fundamental issue with regard to complex data modelling is that we can never hope to have access to enough training instances to model the data straightforwardly as atomic data points. For example, one can consider the height, width and weight of various animals in a 3-dimensional space and distinguish humans among them using a Gaussian model or the k-Nearest-Neighbor algorithm. In contrast, the same approaches cannot be routinely applied for the much higher-dimensional spaces of complex data such as the pixels of an image.

We may respond to this challenge by moving past the surface of the data to take advantage of their internal structure and the relations between the data’s variables. In contrast to random data where modelling efforts are in any case futile, real-life data related to ML applications often exhibit such internal organisation. Models or learning algorithms which introduce the right assumptions over these internal interdependencies are able to overcome data sparsity and adequately describe or classify the data when trained on the limited amount of available training instances.

There are multiple methods we might follow to take advantage of these internal data patterns. For example, to perform face recognition we may exploit the correlations between the pixel values to map the image data in a much lower-dimensional space using Principal Component Analysis (Turk and Pentland, 1991). In financial fraud detection we might classify transactions as fraudulent by establishing a hierarchy over the transaction’s variables using a decision tree. Medical variables are often related by encoding conditional independence assumptions between them in a Bayesian Network. Notwithstanding the differences, all of the aforementioned methods coincide in viewing the data as instances whose internal organisation can be exploited to model them from reasonably-sized training sets.

Complex Data in NLP Most of the Natural Language Processing tasks accept as input complex data, e.g. natural language sentences, and in this way are susceptible to the issues highlighted above. The space of possible values is so large that even billions of training instances are not enough to cover a substantial part
3.1. Fragment Models

of it. An interesting experiment to exemplify data sparseness when working with NLP data is searching for an exact match of medium length, well-formed sentences or even phrases against all the content of the web using a search engine, resulting in most cases in zero matches. Fortunately, despite all the talking we have only explored a small fraction of what can be stated through natural language.

However, this does render modelling NLP data challenging. For example, due to this data sparsity a language model based on the relative frequency of whole sentences in the training data can hardly be effective at all, assigning zero probability to most of the yet unseen well-formed sentences. In the same way, for parsing we cannot escape by merely learning the conditional distributions of full parse-trees for every training sentence.

One way to overcome this is to introduce independence assumptions between the variables of the data in a generative model, or employ features based on patterns between the values of these variables in a featured-based approach. For a generative parsing model, we might for example assume that expansions of parse-tree non-terminals are independent of the rest of the previous derivation steps given the non-terminal being expanded, as in Probabilistic Context-Free Grammars.

A typical solution for the purpose of language modelling is to assume a Markovian LM. For example, in a second-order LM, every new word $w_i$ generated is independent of the previous ones given the last two words $w_{i-2}, w_{i-1}$ before it, as in equation (3.2) below. With $w_1 = \langle s \rangle$ and $w_N = \langle /s \rangle$ the start and stop symbols delimiting a sentence, we can write:

$$p(w_1 w_2 w_3 \ldots w_N) = p(w_1)p(w_2|w_1) \prod_{i=3}^{N} p(w_i|w_{i-1})$$  \hspace{1cm} (3.1)

$$\approx p(w_1)p(w_2|w_1) \prod_{i=3}^{N} p(w_i|w_{i-2})$$  \hspace{1cm} (3.2)

Modelling with Data Fragments The case of Markovian LMs exemplifies how we can address the challenges of modelling complex NLP data by breaking through their surface and considering assumptions over their internal organisation. The exact chain-rule application of equation (3.1) above is associated with an opaque, rigid view of data generated as one piece. In contrast, imposing in the context of generative models independence assumptions like those of a second-order Markovian model, allows us to take advantage of the local nature of many linguistic phenomena to disentangle, as conditionally independent, the words of the sentence that are longer than 2 words apart.

These assumptions, aside from their probabilistic modelling impact on simplifying (3.1) in (3.2), crucially establish tri-grams, word sequences of length 3, as the partially overlapping building blocks of sentences. The result is that, since
gathering statistics over tri-grams is less affected by data sparsity than doing so for full sentences, a tri-gram based language model generalises much better.

Along the same lines, we can visualise the generative process of a model employing conditional independence assumptions between the data variables as building the data from partially overlapping data fragment, data, with the overlapping part matching the conditioning context between them. The size of these fragments and the way that they are combined depends on the assumptions of the model, with second order Markov LMs employing tri-grams, third order models employing four-grams etc. When training such models over data whose variables are discrete, as is the case for most NLP problems, training often consists of extracting such fragments and their associated statistics from the training data. Apart from the $n$-grams of Markov LMs, further examples include subtrees of depth one for PCFG models trained from treebanks, word-pairs for the IBM SMT models (Brown et al., 1990), contiguous phrase-pairs for the phrase-based SMT models (Och et al., 1999; Koehn et al., 2003) and synchronous subtrees of depth one for Synchronous CFG hierarchical translation models (Wu, 1997; Chiang, 2005a).

**Fixed-size and Arbitrary Fragments** As discussed above, a lot of the models employed in NLP are based on extracting and learning to recombine fragments of the training data. A common trait of the majority of these models is that they employ elementary fragments of fixed sizes, such as tri-grams, subtrees of depth one or word-pairs. They then define probability distributions over derivations combining these fragments together to generate the data being modelled.

In contrast, a particularly interesting family of models is characterised by the utilisation of fragments of arbitrary sizes. For example, the Data-Oriented Parsing models (Bod et al., 2003) are based on subtrees of arbitrary depth, while phrase-based and hierarchical SMT employs phrases, contiguous and non-contiguous respectively, of arbitrary length. In principle, the size of the fragments combined to generate the modelled data can vary up to the full size of the data points, allowing derivations generating a data point as a single fragment and in a single generative step. We refer to these models in the rest of this work using the term Fragment Models (FMs).

### 3.1.2 Data-Oriented Processing

Studying the Data-Oriented Processing paradigm (Scha, 1990; Bod, 1992), one of the earliest frameworks leading to the formulation of Fragment Models, is interesting both to highlight the potential of FMs as well as discuss the challenges involved in their training. Data-Oriented Processing was initially applied to the supervised learning of natural language parsing and later employed for unsupervised parsing (Bod, 2006) as well as translation (Poutsma, 2000; Way, 1999) among others. The application of this paradigm for NLP tasks stems from the
basic assumption that human language perception and production works with representations of concrete past language experiences rather than with abstract linguistic rules. These are maintained in the form of memorised fragments of arbitrary sizes from previous language utterances that a language user has been exposed. New linguistic input can be analysed or novel linguistic output can be produced by combining these fragments together.

The same assumptions can be employed in Artificial Intelligence terms to arrive at a Fragment Modelling approach. Here, the role of the human language user is substituted by an empirically estimated probabilistic model, with prior linguistic experiences embodied in the training corpus. From this corpus arbitrarily-sized fragments of data points are extracted, which a generative process can combine together to arrive at new data. A stochastic model instance over this process provides a distribution over these novel combinations, distinguishing between highly probable and less probable ones. In total, a framework to define such Fragment Models, along the lines it was first developed for Data-Oriented Parsing can be drawn in terms of the following components (Bod, 1995).

- A definition of a formal representation for data analyses.
- A definition of the fragments of the analyses that may be used as units in constructing an analysis of a new data point. The size of the fragments varies up to covering the full data point analysis as a single fragment. This last property crucially distinguishes FMs from other modelling paradigms.
- A definition of the operations that may be used in combining fragments.
- A probabilistic model over the derivations of data points through the combination of fragments.

**Data-Oriented Parsing** The first application of the Data-Oriented Processing framework was in the context of Data-Oriented Parsing (DOP) (Bod et al., 2003) which can also function as an interesting example of a Fragment Model. In DOP we are interested in modelling the constituency parsing analyses of natural language sentences.

The more traditional approach in modelling the latter is through Probabilistic Context-Free Grammars (PCFGs). Each PCFG $G$ is a 5-tuple $\langle V, T, S, R, P \rangle$: a finite set of Non-Terminal (NT) symbols $V$, a finite set of terminal symbols $T$, a designated ‘start’ symbol $S \in V$ and a finite set of rewrite rules $R$ expanding a left-hand side (a single NT) to a right-hand side string of terminals and NTs. $P$ is a set of probabilities $\{p(r|\text{LHS}(r))\}$, one for each rule in $R$, with the probabilities of all rules with the same left-hand side summing up to one.

The PCFG explains the derivation of a parse tree starting from the start symbol $S$, by recursively rewriting NTs using rules of the grammar. Each time, the rewriting operation is applied to the left-most NT which has not yet been
Chapter 3. Fragment Models Estimation with the CV-EM Algorithm

Figure 3.1: Constituency parse tree $T$ with the PCFG rules $r$ of its derivation. The probability $p(T)$ of the parse tree is the product of the derivation’s rule probabilities.

expanded, to avoid the spurious ambiguity between derivations employing the same rules in a different order. The rule applications are considered independent of the rest of the derivation given the NT that they expand. The probability of a full derivation is the product of the probabilities of the rules that were employed, with these probabilities summing up to one for all rules with the same NT as their left-hand side, as dictated by the aforementioned independence assumptions. An example of a constituency parse tree together with the PCFG rules taking part in its derivation and the computation of its probability can be seen in Figure 3.1.

We already discussed in the previous section that PCFG derivations can be seen as combining together parse fragments to derive a full parse tree, albeit of a fixed size: subtrees of depth one. In DOP, we move past this constraint to extract tree fragments of arbitrary sizes from the training corpus of sentence constituency parses and learn how to combine them together in derivations of novel sentence instances. As fragments we consider subtrees (i.e. tree fragments with a single root) of arbitrary depths, with the conditioning context remaining as in the case of the PCFGs the root of the subtree. Considering subtrees of arbitrary depth implies also including the complete parse tree in the set of subtrees. Figure 3.2 lists a subset of the subtrees that can be extracted from the tree of Figure 3.1.

More formally, a DOP probabilistic grammar is again defined as a 5-tuple $\langle V, T, S, R, P \rangle$ along the same lines as a PCFG. However, each rule $r \in R$ replaces a left-hand side single NT, to a right-hand side sub-tree of terminals and NTs having the left-hand side as root, in contrast with PCFGs where expansions lead to terminal and NT strings. For this, DOP grammars are categorised in the literature in the family of Tree-Substitution Grammars.

Each rule has an associated probability attached to it, with these probabilities again summing up to one for all rules with the same left-hand side NT. Each derivation, starting from the start symbol $S$, rewrites left-to-right non-terminals
Figure 3.2: Some of the subtrees that can be extracted from the constituency parse in Figure 3.1. The first row depicts subtrees of depth one that are also the units of PCFG derivations. However, DOP extracts and reuses in derivations also subtrees of arbitrary depth, up to the complete parse tree.
to their subtree expansion, by applying a rule with the current leftmost NT as its left-hand side. The probability of a derivation $D$ is, along the same lines as PCFGs, equal to the product of the probabilities of the rules $r$ taking part in it.

$$p(D) = \prod_{r \in D} p(r|\text{LHS}(r))$$ (3.3)

Three derivations of a novel sentence based on a subset of the fragments of Figure 3.2 are listed in Figure 3.3. The first derivation reuses the same fragments as a PCFG derivation of the parse would employ, highlighting that PCFG derivations with elementary fragments is also part of the space of derivations that DOP considers. The second and third derivations however employ a subtree reaching down to depth two, which in the second derivation encodes the dominant Subject-Verb-Object sentence structure of English, while the third one memorises the argument structure of the verb ‘likes’.

**Latent Segmentation** The introduction of Data-Oriented Parsing, apart from showcasing the descriptive power of Fragment Models, also gradually revealed the challenges involved into estimating such models from training data. Context-Free Grammars provide only a single derivation for each parse tree, if we agree to only substitute NTs in a left-to-right fashion as mentioned above. This is a crucial point, as it allows us to relate a training parse tree to a single derivation behind it and in this way translate the observation of the tree to the observation of the unique sequence of CFG rules taking part in its derivation. For the purpose of PCFG training, this enables us to treat a corpus of training parse trees (a tree-bank) as complete data\(^1\), which simplifies training under a Maximum-Likelihood objective to an instance of Relative Frequency Estimation.

However, while the same left-to-right constraint allows us to avoid spurious ambiguity between DOP derivations employing the same subtrees, as Figure 3.3 reveals under DOP we can still arrive at the same parse tree under multiple derivations, each employing a different set of fragments for this purpose. The probability of a full parse $T$ is then the sum of all derivations $D \Rightarrow T$ leading to $T$.

$$p(T) = \sum_{D \Rightarrow T} p(D)$$ (3.4)

Most importantly, DOP and Fragment Models in general introduce in this manner a latent segmentation variable. This dictates how a data point is segmented into fragments, as there is more than a single way to do this. While in DOP latent segmentation is not explicitly encoded in a separate model variable, it still is embedded in the model in the form of subtree expansion probabilities, which indicate preferences for substitution of NTs by larger or smaller fragments.

\(^1\)For a discussion of complete vs. incomplete data please refer to section 2.6.
3.1. Fragment Models

Figure 3.3: Three DOP derivations of the same parse tree.
Because of the latent segmentation, the treebank training set must be considered as incomplete data, as it does not contain information on the segmentation of parse trees into DOP subtrees. For parameter estimation, we need thus to disambiguate between the segmentations of data points into fragments.

### 3.1.3 DOP Estimation

At first sight, one could apply Maximum Likelihood Estimation for DOP, as it is successfully applied for PCFGs in the form of Relative Frequency Estimation. The fact that treebanks are incomplete data for the purpose of estimating the parameters of a DOP model, disallows the application of the relatively easy RFE. Nonetheless, we can still formulate an MLE estimation objective, as we already discussed in section 2.6, and maximise the likelihood $L(T)$ of the training data $T$ by summing through the alternative derivations of each training instance $T$.

$$L(T) = \prod_{T \in T} \sum_{D \Rightarrow T} p(D) \quad (3.5)$$

However, this vanilla MLE objective is of little use to estimate the parameters of DOP models, as we discuss in more detail in the wider context of Fragment Models in the next section. In rough terms, the problem is that the MLE objective of fitting the training data leads to allocating all probability mass to full training parse trees, as these are also part of the subtrees extracted from the training set (Prescher et al., 2004). Doing so allows the model to exactly predict the training treebank. Crucially, while it seems superficially desirable to arrive at parameters which fit well the training data, in this case it completely defeats the purpose of learning such a model, as it assigns no probability mass to any analyses of sentences past those included in the training parses (Zollmann and Sima’an, 2006). The MLE estimate of DOP probabilistic grammars has an extremely limited ability to generalise to yet unseen data instances.

In the face of this, right from the introduction of DOP, there has been a barrage of work on estimating DOP models, resulting in a fair amount of progress and increased understanding of the issues involved, but still failing to decisively resolve the problem of estimation for DOP models. We briefly examine the key points of such estimation approaches, as an interesting overview of possible solutions to Fragment Model estimation that have already been tried.

#### DOP1 Estimator

The introduction of the DOP model for parsing was coupled with the DOP1 estimator (Bod, 1995). Under DOP1, the probability of a DOP subtree is set to the relative frequency of its ‘appearance’ in the training treebank. More accurately, this estimator belongs to the family of estimators employing extraction heuristics, that we discuss in the next section. Namely, we compute the DOP1 estimate of a treebank by first extracting all DOP subtrees of the
3.1. Fragment Models

training parses, counting how many times each fragment can be extracted. The DOP1 estimate is then set to the relative frequency of the subtree fragments in this multiset of subtrees constructed in the previous step.

A somewhat subtle but crucial point is that the DOP1 estimate has no connection to the relative frequency of any events in the training data, as the training treebank does not contain any information relating to the segmentation of parses into fragments as we already mentioned. As the extraction step lacks any clear link to the training data by means of an optimisation objective, the DOP1 estimate is a heuristic estimator, with an appealingness relating only to the strength of the results from its empirical application. Its arbitrary nature from a theoretical standpoint is further highlighted by (Johnson, 2002) who shows that DOP1 is a biased and inconsistent estimator. Additionally, (Bonnema et al., 1999) notice that the DOP1 extraction heuristic in practice favours larger extracted subtrees over smaller ones.

**Bonnema et al. Estimator** Bonnema et al. (1999) propose instead an alternative estimator which assigns to every appearance of a subtree in the training data, a count equal to the fraction of the number of possible DOP derivations using the subtree fragment, against the total number of DOP derivations of the parse where it appears. Essentially, this boils down to Maximum Likelihood estimation on a treebank whose segmentation of every parse tree in DOP subtree fragments has already been disambiguated by the strong assumption that all segmentations are equally likely. Sima’an and Buratto (2003) show that this estimator is inconsistent and discuss that it is biased towards smaller subtrees and does not perform well in practice.

**Back-Off Estimator** Starting from the DOP1 estimate, (Sima’an and Buratto, 2003) propose a back-off estimator based on Katz smoothing technique (Katz, 1987) to discount probability mass from the larger subtrees towards smaller ones. As (Zollmann and Sima’an, 2006) discuss, this estimator both inherits the weaknesses of the DOP1 estimate, as well as loses some of the appealing properties of Katz smoothing through problems related to the estimator’s practical implementation.

**Parsimonious-DOP Estimator** Starting from the PCFG relative-frequency estimate of the training treebank, which employs minimal subtrees of depth one, (Zuidema, 2007) propose an estimator which distributes probability mass to larger subtrees by evaluating their extraction frequency against its expectation. It also includes a counter-balancing bias against large trees to avoid completely overfitting the treebank. While the Parsimonious-DOP estimator works in the opposite manner than the Back-Off estimator, moving overall probability mass from smaller to larger subtree fragments instead, it shares with it the weakness of being
based on the DOP1 heuristic, as it compares model subtree extraction expectations against the DOP1 extraction counts.

**Shortest Derivation Parsing** While not involving the estimation of a model, an alternative approach to the problems related to estimating DOP model parameters is brought forward by (Bod, 2000): shortest derivation parsing. It abandons probabilistic modelling altogether, in favour of parsing by recovering the shortest DOP derivation of a parse tree covering a test sentence, by employing subtree fragments extracted from the training treebank. This parsing heuristic objective favours large subtrees as does the DOP1 estimator and, perhaps for exactly this reason, it is shown to perform competitively against the latter.

**DOP* Estimator** The DOP* estimator proposed in Zollmann and Sima’an (2006) is of particular interest to this work. The authors briefly consider the possible use of Cross-Validation (CV) to avoid overfitting towards a Maximum Likelihood estimate of a DOP model’s parameters which fails to generalise. However, they do not pursue this approach. They argue that while such a learning objective might be theoretically appealing, in practice it involves employing hill-climbing algorithms such as the Expectation-Maximization algorithm, which do not guarantee arriving at the overall ML estimate but only at a local likelihood optimum.

With a primary objective of arriving at a consistent estimator, the DOP* estimator instead couples the shortest derivation principle with CV to disambiguate the segmentation of the treebank parses in DOP derivations: after partitioning the treebank in 10 parts, for every part they consider the shortest DOP derivation(s) of each parse utilising exclusively subtree fragments from the remaining parts of the treebank. From this subtree-segmented treebank they arrive at a DOP estimate by ML estimation which, operating on DOP derivations in place of unsegmented parse trees, boils down to relative frequency estimation. This estimation approach is shown to be consistent, i.e. to arrive with a probability approaching one at an accurate estimate of the true distribution of parse trees when the training treebank size grows towards infinity.

Despite the asymptotic consistency properties of the DOP* estimator, its application on treebanks of real-life sizes still demands a ‘leap-of-faith’ concerning the use of the shortest derivation principle. In contrast, in this thesis we draw inspiration from certain aspects of the work on the DOP* estimator to pursue and study the implications of a Cross-Validated Maximum-Likelihood learning objective, implemented in the form of our Cross-Validating Expectation-Maximization algorithm.
3.1. Fragment Models

3.1.4 Modelling with Fragment Models

In the previous section we reviewed in some detail the developments in employing a Fragment Model for natural language parsing in the form of the DOP framework. However, models which can be categorised as FMs have been also introduced for other tasks, with contiguous and non-contiguous phrase-based Statistical Machine Translation being the most relevant in respect to this work. Irrespective of the particularities linked to every kind of data or process that needs to be modelled, FMs enjoy certain common properties which we will now discuss.

A Fragment Model can be considered a hybrid between more traditional generative models on the one hand and example-based models on the other. Typically, generative models describe the derivation of data points through generative steps involving minimal, usually fixed-size, units. Defining what constitutes such a unit rests with the modeller: for example, a language model can operate at the word, morpheme or even letter level. Still, the crucial property of these units is that they are considered atomic, as they cannot be in turn constructed from other such modelling units. Furthermore, learning and applying a generative model is usually a two-step process. First, generative models are induced from the training data according to a learning objective or using a training algorithm, and the learnt model is subsequently applied to process the test data.

In contrast, frameworks categorised as memory-based or example-based process novel data by reusing memorised training examples. For instance, Example-Based Translation (EMBT) (Nagao, 1984) strives to employ these to translate novel sentences by recombining translation fragments from the memorised examples. These fragments can be of arbitrary sizes, with the larger fragments preferred (Sato and Nagao, 1990). Part of an EBMT system is the process of choosing which translation fragments to use and how to recombine them. Typically, this process does not depend on a prior model induction step, accepting the entire training data as input and outputting a model describing translation by means of such fragments, as is usually the case with generative models. On the contrary, the related decisions are considered and scored for every test data point.

Fragment Models stand in the middle, aiming to pick the ‘best of both worlds’. On the one side, in comparison to traditional generative models employing minimal atomic units, they crucially consider generating data from units whose involvement in the derivation of a data point can be replaced by an alternative analysis employing other, smaller units. As examples, a parse analysis involving a larger DOP subtree can be substituted by an analysis which arrives at the same constituency structure by a combination of smaller subtree fragments; a translation phrase-pair can be constructed as a combination of smaller phrase-pairs or even word-pairs. On the other side, in comparison to memory-based frameworks, their formulation by means of a stochastic generative process, allows us to encode the construction of novel data instances from fragments extracted from the
training corpus, by means of the perhaps better-founded and easier to understand probabilistic models.

By moving past atomic units to larger fragments, while crucially remaining within the context of a generative framework, Fragment Models have a chance to overcome the inherent weaknesses due to the blanket independence assumptions behind traditional generative models. An interesting example of when this is desirable is the modelling of the translation of idiomatic expressions in natural language such as ‘kick the bucket’. Such expressions typically exhibit correlations between their words and their translations that severely violate the independence assumptions behind word-based models; word-to-word translation statistics are not enough to direct us toward an adequate translation for the aforementioned example phrase. An FM enjoys the ability to reserve some probability mass from more fine-grained analyses that assume some degree of conditional independence between their generative steps. It can then assign this mass to directly model the translation of such idiomatic expressions as single units, in this case translate ‘kick the bucket’ as a single contiguous phrase.

### 3.1.5 MLE and Fragment Models

As already highlighted above through following the development of the literature on DOP estimation, using a Fragment Model does not come without disadvantages. The central issue is that due to the introduction of the latent segmentation variable, estimating the parameters of FMs is far from straightforward. In addition, FMs ability to bypass the independence assumptions of models employing smaller fragments allows it to fit the training data arbitrarily well. This is a blessing that, unless properly treated, can easily develop into a curse in the form of overfitting.

For example it is easy to see that the often successful Maximum Likelihood Estimation objective completely overfits the training data when applied to estimate FMs, as discussed in (Prescher et al., 2004). The MLE objective of equation (2.30) can be interpreted as minimising the Kullback-Leibler divergence between the empirical relative-frequency distribution $\tilde{p}(x)$ of values of the data variable $X$ in the training data $X$, and the model estimate $p(x; \theta)$ parameterised by $\theta$.

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

$$= \arg\min_{\theta} KL(\tilde{p}(x) \parallel p(x; \theta)) = \arg\min_{\theta} \sum_{x \in \mathcal{X}} \tilde{p}(x) \log \frac{\tilde{p}(x)}{p(x; \theta)}$$

We already mentioned that we considered a key feature of FMs to be able to model the training data with fragments of arbitrary sizes, up to considering a data point as a single fragment emitted in a single generative step. This entails
that there exists a parameter setting $\tilde{\theta}$ for which all values $x$ of random variable $X$ are emitted in a single generative step, according to their relative frequency in training data $\mathcal{X}$. As these relative frequencies sum up to one, this leaves no probability mass for analyses employing smaller fragments than whole data points.

But for $\theta = \tilde{\theta}$ we have $KL(\tilde{p}(x) \mid \mid p(x; \tilde{\theta})) = 0$ and since the KL divergence between any two distributions is always larger or equal to zero, $\tilde{\theta}$ is the ML estimate of $\theta$ as it minimises (3.6). Overall, the MLE estimate of Fragment Models completely overfits the training data by predicting nothing more than the training points according to their empirical relative frequency.

### 3.1.6 Expected Error of MLE for Fragment Models

Taking the result of the previous section into account allows to shed some light on the source of the generalisation error incurred by the MLE estimates of FMs, as analysed in terms of estimator bias and variance. In section 2.7.1 we discussed how the expected generalisation error for an estimator of a distribution $\hat{p}(\mathcal{X})$ trained on data $\mathcal{X}$, can be analysed in estimator bias and variance terms. Let us measure the estimator’s expected error $Err(\hat{p})$ in terms of the expected Kullback-Leibler (KL) $E_{\mathcal{X}}KL(q, \hat{p})$ divergence between the estimator’s output and the target distribution $q$. Then, the error $Err(\hat{p})$ can be analysed in a bias and a variance term according to equation (2.38), which we repeat here for the reader’s convenience.

$$Err(\hat{p}) := E_{\mathcal{X}}KL(q, \hat{p}) = KL(q, \tilde{p}) + E_{\mathcal{X}}KL(\tilde{p}, \hat{p})$$

The bias term is the KL-divergence between $q$ and the mean estimate over all training data $\tilde{p} = E_{\mathcal{X}}\hat{p}(\mathcal{X})$. Variance is the expected divergence between the average estimate and the estimator’s actual choice for each training input $\mathcal{X}$.

The fact that the MLE estimate $\hat{p}(\mathcal{X})$ in the case of FMs predicts exactly the training data according to their empirical relative frequency, has consequences for both terms of the estimator’s error. On one hand, MLE is a zero-bias estimator for FMs: The average estimate $\tilde{p}$ will coincide with the target distribution $q$ when we average over all training sets $\mathcal{X}$, which themselves are sampled from $q$. On the other hand, since every estimate assigns zero probability to any value not appearing in the training data $\mathcal{X}$, in all but trivial cases the variance term will be unboundedly large.

The end result is that the expected error of the MLE estimate is extremely large even though the estimator is zero-biased. This is a typical instance of the estimator bias-variance trade-off, when merely aiming to minimise one of the two error terms severely increases the other. To arrive at an overall low expected error, we need to trade bias error by relaxing how closely our estimator fits the training data, in order to reduce the error attributed to the variance between
the estimates. For Fragment Models this translates to shifting probability mass from excessively large fragments to smaller, more reusable ones, according to our expectation for them to appear in further samples from the target distribution than the training data we currently have at hand.

3.1.7 Fragment Models and Generalisation

The overfitting behaviour of MLE estimators for FMs should not discourage us from employing them. Part of the model space of FMs are also estimates which solely employ atomic fragments in their analyses. This means that an FM always includes an estimate which fits the data as well as models employing atomic, fixed-size fragments. Combining this with the result above, what we essentially learn is that FMs have the ability to provide model estimates which cover the continuum between the best data fit provided by their traditional generative counterparts and completely overfitting the training data, by shifting probability mass from more generic explanations of the data to more specific ones. FMs should be thus seen as very versatile models which can accurately describe any training data set. Our focus should then be directed towards employing this ability to also accurately model yet unseen data.

Choosing how closely the training data should be fit is, as is often the case in Machine Learning, a data-centric issue. The more training data we have at hand, the larger the probability mass we can reserve for larger fragments, with the overfitting unconstrained MLE estimate we discuss above being the optimal choice as data grows towards infinity and \( \hat{p}(x) \) converges towards the true data distribution. Smaller training data sizes demand focusing probability mass on derivations which employ smaller fragments which we hope will generalise better. But not all fragments of the same size are equally good at generalising: some of them might come forward in data as noise, other as particular instantiations involving combinations of smaller fragments which roughly follow the independence assumptions assumed, while others might signify a departure from the same assumptions that needs to be captured. We believe that navigating this treacherous field demands a data-driven approach, as the one we propose in the form of the CV-EM algorithm later in this chapter and apply empirically in the rest of this work.

3.1.8 Inducing Fragment Models

A direct application of Maximum Likelihood Estimation, apart from the unconstrained case, has also been empirically shown to perform poorly even when the size of the fragments is constrained so that complete overfitting is avoided (DeNero et al., 2006). Given the difficulties of establishing an ML objective that leads to estimates which generalise well, there has been an array of research directions towards alternative FM estimation approaches. We discuss below two of
3.1. Fragment Models

these: the extraction heuristic, which despite its heuristic nature is still applied in most state-of-the-art implementations of FMs, and Bayesian induction of FMs which employs probabilistic priors to arrive at reasonable FM estimates in a more principled manner.

The Extraction Heuristic The Extraction Heuristic estimates the parameters of a Fragment Model in two steps. Firstly, a corpus of extracted fragments is constructed from the training data, extracting all fragments assumed from the FM from every training data point and assigning them a frequency equal to the number of times they were extracted. Then, we arrive at an FM estimate by applying Relative Frequency Estimation on this fragments corpus. This leaves the estimate only related to the original training corpus of complete data points by means of the heuristic extraction process. As we have already commented, the Extraction Heuristic has nothing to do with RFE on the training corpus itself, as the latter does not provide information on any events that are related to its segmentation in fragments. We cannot then just ‘count’ fragment appearances on the corpus, as the segmentation variable is hidden.

Nevertheless, while it remains difficult to understand what the Extraction Heuristic optimises, its straightforward implementation and the relatively strong results obtained through its employment resulted in it being the estimator proposed during the introduction of both DOP (Bod, 1995), as well as phrase-based translation by means of both contiguous (Och et al., 1999; Koehn et al., 2003) and non-contiguous phrase-pairs (Chiang, 2005a). Going further, the Extraction Heuristic remains a competitive estimator for state-of-the-art systems up to this day.

However, due to their heuristic nature, these estimates have limited theoretical appeal and leave open the question how much better an estimate that maximises some meaningful objective function can do. Also, as they operate on the surface of the training data ignoring the latent variables of FM models, the risks involved in their application grow as the latent variable of the proposed Fragment Models becomes more involved, as we have discussed when examining the Syntax Augmented MT models in section 2.4.3. For this, there has been a significant amount of work on alternative, better founded and understood approaches to induce FMs, such as a large part of this work, as well as the work on the Bayesian induction of FMs which we discuss next.

Bayesian Induction One way to address the inherent tendency of Fragment Models to overfit the training data is by means of a Bayesian prior over the model space. This allows us to encode in the prior a certain preference over parts of the model space which we believe might better generalise. The actual parameterisation of the model is then typically marginalised out. Two practical Bayesian inference approaches that have been applied to induce FMs are Variational Bayes
Chapter 3. Fragment Models Estimation with the CV-EM Algorithm

and Gibbs sampling. Variational Bayesian EM (also named Variational Bayes)\(^2\) provides an iterative algorithm to arrive at a local maximum of the marginal likelihood:

\[
p(\mathcal{X}) = \int_{\theta} p(\theta) \ p(\mathcal{X}; \theta)
\]

A Gibbs sampler (Geman and Geman, 1984) is a Markov Chain Monte Carlo method to sample from the model distribution where the model parameters have also been marginalised out.

There is no doubt that prior knowledge reaching past the training data can often prove highly successful. However, in this case there is no ‘expert’ to consult on which fragments or which parts of the model space to prefer and no clear reason to favour one model estimate over the other prior to observing the training data, apart from our experience that models which favour extremely large fragments tend to overfit and do not generalise well. There have been two main directions on choosing such priors, both of which aim to avoid models reserving too much probability mass for overly large fragments: preferring sparse fragment distributions and preferring smaller fragments.

The first direction is employing priors preferring sparse fragment distributions which assign most probability to a small subset of the data fragments, favouring more parsimonious model formulations. For example, this can be achieved by means of a sparse Dirichlet prior, with (Zhang et al., 2008a) employing such sparse prior in a Variational Bayesian approach to disambiguate the segmentation of sentence pairs in contiguous phrase-pairs. The second direction is priors preferring smaller fragments. For example, (Blunsom et al., 2009) employ a Dirichlet Process prior with a base distribution including a Poisson distribution over the phrase-pair length with unit mean.

However, the strong overfitting behaviour of FMs employing large fragments entails that they can assign extremely large likelihood values to the training data. Examining equation (3.7) above where we marginalise over the product of data likelihood and prior probability, reveals that for a Bayesian prior to counter this it needs to penalise models employing large fragments equally strongly. Zhang et al. (2008a) find that a good choice for the Dirichlet hyperparameter \(\alpha\) (which must satisfy \(\alpha > 0\)) is the extremely low value \(\alpha = 10^{-100}\). Blunsom et al. (2009) use a base distribution including a Poisson distribution over the phrase-pair length with unit mean.

Overall, for both prior designs, small fragments are strongly preferred, with larger fragments having a higher chance to be sampled only when they appear very frequently in the data. While this approach does allow to expand to fragments past the minimal set without overfitting, imposing a blanket preference for small fragments is a bias which might prevent discovering larger fragments that could

\(^2\)An overview of Variational Bayesian approaches and applications can be found in (Beal, 2003).
be nevertheless useful to better model yet unseen data.

In the next section, we introduce the CV-EM algorithm, a data-driven approach towards estimation which explicitly aims at model estimates which generalise well. In later chapters, we see how the CV-EM can be applied to arrive at Fragment Model estimates for Machine Translation which perform well on test data, with both a clear optimisation criterion (in contrast to the Extraction Heuristic) and a data-driven approach to avoid overfitting (in contrast to enforcing an external prior).

### 3.2 Cross-Validated Expectation-Maximization

The central problem in Machine Learning is bridging the gap between the limited sample that makes up our training data and the yet unseen test data. This necessarily involves abstracting away from the actual training sample to capture the general properties of the data being modelled, so that what is learnt from the training set can hopefully extend to novel data instances. If no abstraction from the training data is necessary to solve a problem and merely looking them up is sufficient, then this falls more into the realm of databases and could hardly be considered an ML problem. The art of Machine Learning then lies in successfully choosing the level of abstraction from the training points and sorting out the characteristics of the underlying data distribution from the peculiarities of the training instances. Cross-Validation (CV) provides a simple yet powerful method to evaluate how well a learner does in this respect.

As we presented in more detail in section 2.7, given a model for the random variable behind the training data and an estimator for its parameters, \(K\)-fold Cross-Validation (CV) provides a method to estimate the Generalisation Error (GE), the error over yet unseen data of the model instances selected by the estimator, by employing the training data itself. It is able to do so, by first partitioning the training data in \(K\) parts. Then, in \(K\) rounds, each time a different part from the training data is held out, to assess on it the prediction error of the model instance selected by training on the rest of the data. The outcomes of these \(K\) rounds are then combined together to arrive at a single estimate of the GE. Notably, as we discussed in the previous chapter, CV is a low bias, low variance estimator of the GE, allowing a fairly accurate prediction of how useful a learner applied on the training data at hand is expected to be for yet unseen data points.

These features have established CV as a widely used approach for model selection, i.e. choosing which model, out of a limited set of possible options, is best suited for a particular learning problem, by picking the model which offers the lowest GE as estimated by CV. Here in this work, we move further to describe how Cross-Validation can be employed for parameter estimation of models employing latent variables.

We begin by discussing how the training data themselves are used in practice
during the modelling process to define which hypotheses over the values of the model latent variables we will consider, with the risk of overfitting the training corpus. To avoid this, we formulate a Cross-Validated MLE (CV-MLE) learning objective, aiming at model estimates which generalise better. CV for GE estimation considers the error on each part of the training data of a model trained by excluding that part. In the same way, CV-MLE seeks the estimate which maximises the likelihood of each training data part, by excluding hypotheses over the values that the latent variables take for it, and which we would not consider if we excluded this part from the training data.

As for plain MLE, it is frequently not possible to compute the CV-MLE estimate analytically for models with latent variables. With this in mind, we propose CV-EM, a Cross-Validated instance of the EM algorithm to allow CV-MLE parameter optimisation from incomplete data. In the rest of this chapter, we present both the CV-MLE estimation criterion and the CV-EM algorithm that allows us to optimise parameters according to it. We compare our framework with related approaches on estimation towards increased generalisation and discuss how CV-EM can be applied to estimate the parameters of Fragment Models.

### 3.2.1 Pitfalls of Model Extraction

Maximum Likelihood Estimation, estimating the parameters of statistical models so as to maximise the likelihood of the training data, is one of the most widely applied estimators in the Machine Learning literature. When a model with no latent variables is trained from complete data, MLE boils down to the familiar Relative Frequency Estimation. However the MLE estimation objective can also be applied to train models with latent variables from incomplete data. In these cases, an MLE estimator is frequently implemented as an instance of the Expectation-Maximization algorithm, which allows us to climb the likelihood with respect to the model parameters until a local maximum is reached. Pairing the MLE optimisation objective with the EM-algorithm in this way allows us to discover latent data patterns, such as the word-alignments between sentence pairs in Statistical Machine Translation (see section 2.2).

Crucially, for many models explaining complex data, the parameter estimation process where MLE is applicable is preceded by an implicit step, where the training data are used to establish the model parameter space. A particular modelling framework, for example Phrase-Based Statistical Machine Translation, can be seen as a function which when applied on the training data returns a parametric model, using the training data in this way to set up the model’s parameter space. For our example, the output of this function would be the space of conditional distributions between source phrases and their possible target phrase translations, as they appear on a word-aligned training parallel corpus. We will refer to this as the *model extraction* step.

A model extracted in this way establishes a set of hypotheses over the target
distribution \( q(x) \) we are trying to model, with every parameter setting mapping to one such hypothesis. Interestingly, all of these hypotheses can be considered to be suggested by the training data themselves, as these are employed to set the model parameter space during the model extraction step. This renders choosing between these hypotheses during estimation by fitting the training data (as we do in MLE) dangerous, as we might be testing hypotheses suggested by the data. In this way, we risk arriving at estimates which succeed in little more than predicting the particular instances contained in the training data, missing the chance to discover the underlying patterns.

In the case of models with latent variables, each such hypothesis over the model parameter space also leads to an expectation over the values of these hidden variables for the training data points. Since the values for the hidden variables that we will consider in practice frequently arise from examining the training data, we need to make sure that our estimators are not misled into preferring hypotheses over the values of the latent variables which overly fit the training material. For example, if we assume a generative model where the derivation of each training data point is (partially) hidden, every parameter setting disambiguates between all the different derivations of a data point by establishing a distribution over them. For such a model extracted using the training data, the danger then lies at erroneously preferring derivations which overfit the training data points, such as derivations which generate the data by combining large data fragments instead of smaller, more reusable ones.

These are issues that apply in various extents to the training of most models of complex data. For example, the large majority of models in NLP, such as probabilistic CFGs, language models etc, are extracted from training data which are also used to estimate their parameters. The actual extent that the pitfalls discussed above affects empirical work relates to the level that the model extracted from the training data abstracts from them. As we move from coarse-grained models to more fine-grained models (e.g. as we increase the maximum history size of an interpolated Markovian LM), the risk of overfitting the training data increases, with Fragment Models standing at the far end of this continuum.

FMIs are based on analyses of the data of arbitrary granularity, and for these models estimation by fitting the training data leads to degenerate estimates which fail to generalise. This is not surprising, given that an extracted FM includes hypotheses over the data distribution which closely predict the training corpus, as we discussed in sections 3.1.5 and 3.1.6. When these degenerate hypotheses are then tested against the same data they were extracted from, they easily emerge as the strongest (best fitting) ones. For many modelling frameworks, such as the Fragment Models family or interpolations of models of different granularities, straightforwardly maximising the likelihood of the training data as an estimation criterion fails to arrive at model parameter values which generalise well.
3.2.2 Cross-Validated MLE

The issues highlighted above render the application of the training data Maximum Likelihood optimisation objective problematic for models (or the relevant part of the model parameters) whose estimation needs to also establish how closely they should predict the training data. Nevertheless, we need not abandon ML estimation altogether, as it is a well-understood, widely applied estimator enjoying desirable statistical properties. In contrary, we will formulate here an alternative, Cross-Validated Maximum-Likelihood estimation objective, which avoids the problems arising from establishing model hypotheses from the training data.

**Deleted Estimation** Some of the issues and core principles behind the solution we propose here are long established in NLP research, as exemplified by the literature on LM parameter estimation. Jelinek and Mercer (1985) observe that the n-gram relative frequencies of the training data can diverge from those in test data, especially as the length of the n-grams increases. Furthermore, the MLE estimate of the interpolation weights for a linear interpolation of Markovian language models of different orders does not generalise well (Jelinek and Mercer, 1980).

In both cases, validation or Cross-Validation, with the application of the latter often referred to as ‘Deleted Estimation’ in LM literature, are successfully employed to address the problems related to the skewed n-gram distributions suggested by the training data. Jelinek and Mercer (1985) cross-validate the n-gram LM, extracting in each CV-round the model from one part of the data while estimating its parameters from the held-out part. Additionally, the interpolation weights of interpolated LMs are trained by maximising the likelihood of a held-out corpus or by means of cross-validation, so as to avoid training both the n-gram models and the interpolation weights from the same part of the corpus. Below, we base ourselves on these approaches to formulate a comprehensive CV-MLE optimisation criterion.

**Re-examining MLE on Incomplete Data** Let us now examine how we can formulate a Cross-Validated Maximum Likelihood Estimation objective for generative models estimated from an incomplete data corpus \( \mathcal{X} \) made of observed data points \( \mathbf{x} \). As a first step however, it is interesting to begin by revisiting MLE in this setting (see also section 2.6).

This time however, we make explicit the use of training data to establishing the hypotheses \( Z(\mathbf{x}) \) over how each observed data point \( \mathbf{x} \) can be completed with its unobserved part \( \mathbf{y} \) to arrive at a complete data point \( \mathbf{z} = \langle \mathbf{x}, \mathbf{y} \rangle \). Towards this aim, as part of the input of the function \( Z \) which maps incomplete to complete data, we will include the training data \( \mathcal{X} \) which are implicitly employed by this function, using the notation \( Z(\mathbf{x}; \mathcal{X}) \). We can then rewrite equation 2.32, this
time highlighting the role of the training corpus itself as part of the process to arrive at the hypotheses over the missing information $y$.

$$\hat{\theta} = \arg \max_{\theta} \prod_{x \in \mathcal{X}} \sum_{z = \langle x, y \rangle \in Z(x; \mathcal{X})} p(x, y; \theta)$$  \hspace{1cm} (3.8)

For example, in the case of Phrase-Based SMT, the incomplete data corpus $\mathcal{X}$ refers to the parallel training data, a corpus of observed word-aligned sentence-pairs $x$, where each $x$ misses the hidden phrase-pair segmentation $y$. To train the PBSMT model, which assumes a segmentation in phrase-pairs, we must establish hypotheses $Z(x; \mathcal{X})$ over the phrase-segmentation, by considering which phrases can be translations of each other. Crucially, these hypotheses are constructed by examining during the model extraction step the training corpus $\mathcal{X}$ in its entirety and extracting all phrase-pairs according to the phrase-extraction heuristic, as presented in section 2.3. As discussed in section 3.2.1 above, for some model frameworks and in particular for fragment models such as those employed in PBSMT, this entails the danger of favouring hypotheses suggested by the training data itself, leading to estimates which overfit the training corpus.

**Cross-Validated Likelihood** To avoid this pitfall, we will employ $K$-fold Cross-Validation during the process of establishing the hypotheses $Z(x; \mathcal{X})$ on the complete data $z = \langle x, y \rangle$ from which the incomplete observations $x$ might stem. In more detail, we begin by splitting the training corpus $\mathcal{X}$ in $K$ roughly equal-sized parts $\mathcal{X}^1 \ldots \mathcal{X}^K$. For every $1 \leq k \leq K$, we consider for the data points $x$ belonging to part $\mathcal{X}^k$ only hypotheses $Z(x; \mathcal{X}^\neg k)$ over the completion of the observed data which stem from the rest of the data $\mathcal{X}^\neg k = \{\mathcal{X}^1 \ldots \mathcal{X}^{k-1}, \mathcal{X}^{k+1} \ldots \mathcal{X}^K\}$. For the example of Phrase-Based SMT, this would translate into considering for every $x \in \mathcal{X}^k$ only phrase-pair segmentations $Z(x; \mathcal{X}^\neg k)$ which employ phrase-pairs extracted from the rest of the training corpus $\mathcal{X}^\neg k$, excluding the part where the data point which we currently examine belongs. We will refer to the likelihood of the incomplete corpus according to the model when only the cross-validated hypotheses over the unobserved data are considered, as the cross-validated likelihood $\mathcal{L}^{CV}$.

$$\mathcal{L}^{CV}(\mathcal{X}; K, \theta) = \prod_{k=1}^{K} \prod_{x \in \mathcal{X}^k} \sum_{z = \langle x, y \rangle \in Z(x; \mathcal{X}^\neg k)} p(x, y; \theta)$$  \hspace{1cm} (3.9)

**Cross-Validated MLE** Cross-Validated MLE aims at arriving at the parameter set which maximises the likelihood of the incomplete training data just as plain MLE does. However, by maximising the cross-validated incomplete data likelihood during CV-MLE we are more selective when choosing which hypotheses over the hidden part of the data to consider, by cross-validating the set of
Chapter 3. Fragment Models Estimation with the CV-EM Algorithm

these hypotheses as just described. The CV-MLE estimate $\hat{\theta}^{CV}$ is then computed according to the following equation.

$$
\hat{\theta}^{CV} = \arg \max_{\theta} L^{CV}(X; K, \theta)
$$

$$
\hat{\theta}^{CV} = \arg \max_{\theta} \prod_{k=1}^{K} \prod_{x \in X^k} \sum_{z=(x,y) \in Z(x; X^{-k})} p(x, y; \theta) \quad (3.10)
$$

Properties of CV-MLE On the one hand, CV-MLE deviates from standard applications of MLE little enough so as to retain the well-understood and desirable properties of MLE as an estimator. Firstly and most importantly, the estimation objective is still likelihood maximisation, albeit operating on a more constrained space of hypotheses $Z(x; X^{-k})$ on how the complete training corpus might look like. When working with incomplete data, choosing which such hypotheses to consider is a modelling choice and after this choice is made, CV-MLE proceeds as plain MLE would.

Moreover, let us assume that the incomplete to complete data mapping function $Z(x; X)$ utilises only the set of training points $\{x_1, x_2, \ldots, x_N\} \in X$ and not their frequencies in $X$. This assumption holds for all models discussed in later chapters of this thesis and makes sense as the mapping function $Z$ must output only which hypotheses must be considered and not how probable these are, the latter task assigned to the estimator.

For these mapping functions $Z$, CV-MLE likewise with plain MLE can be shown to be asymptotically consistent, following similar steps as the related proof in (Zollmann and Sima’an, 2006) for the asymptotic consistency of the DOP* estimator which also employs Cross-Validation. Intuitively, this property holds because, as the size of the training corpus grows, the probability that every training point in $X^k$ is also in $X^{-k}$, rendering equal the outputs of $Z(x; X)$ and $Z(x; X^{-k})$, increases towards one. In that case the estimate of CV-MLE in equation (3.10) converges towards the consistent estimate of plain MLE of equation (3.8).

On the other hand, the crucial step of cross-validating the hypotheses over the unobserved part of the data avoids overfitting towards hypotheses which do not generalise well. On the contrary, CV-MLE favours estimates that prefer such hypotheses which, according to the cross-validation criterion, are similar to those that can be employed to model yet unseen data. These properties promote CV-MLE as a well-understood estimator with good statistical properties that directly aims towards estimates which generalise well.

3.2.3 Cross-Validated EM

Similarly to the estimate of equation (3.8) for plain MLE using incomplete data, we are in most cases not able to analytically compute the CV-MLE estimate
3.2. Cross-Validated Expectation-Maximization

\( \hat{\theta}^{CV} \) of equation (3.10). For this, in this work we formulate the Cross-Validated Expectation Maximization (CV-EM) algorithm, which iteratively maximises the cross-validated likelihood of the incomplete data until convergence towards a local optimum. CV-EM is a true instance of the EM algorithm, fully enjoying the same algorithmic and statistical estimation properties as those presented in section 2.6. In a nutshell, we could say that CV-EM is for CV-MLE the equivalent of EM for MLE: an iterative algorithmic optimisation framework with a well-understood operation and favourable properties.

The CV-EM algorithm as an instance of the EM algorithm follows the same algorithmic workflow of initialisation, followed by iterations between an E-step and an M-step until convergence. The crucial difference with a standard application of EM is that, as we are now climbing the cross-validated likelihood of the incomplete training data as defined above; we will only consider the cross-validated set of complete data hypotheses for every training point.

In the description of the algorithm below, we will follow the same notation as we used already for the CV-MLE in section 3.2.2. Namely, to employ K-fold cross validation during CV-EM, we begin by splitting again the training corpus in K equal sized parts \( \mathcal{X}^1, \mathcal{X}^2, \ldots, \mathcal{X}^K \). An essential part of an application of the EM algorithm is establishing the ambiguous complete data hypotheses by employing the incomplete to complete data mapping function \( Z \). Since CV-EM optimises the parameters according to the CV-MLE estimation criterion, we will employ for every data point \( x \in \mathcal{X}^k \), the cross-validating mapping function \( Z(x; \mathcal{X}^{-k}) \), returning complete data hypotheses from the rest of the training data after \( \mathcal{X}^k \) has been excluded.

Along the same lines as for standard EM in section 2.6, the iterative procedure of CV-EM is as follows.

**Initialisation** We begin by initialising the model’s parameter set by an initial setting \( \hat{\theta}_0^{CV} \). As for all instances of the EM algorithm, initialisation can sometimes crucially determine the outcome of CV-EM’s output, given that the latter climbs towards a local optimum of the cross-validated likelihood starting from the initialisation point. In cases where the shape of the CV-likelihood function in respect to the model parameters is complex, random restarts might provide a solution to the sensitivity of CV-EM to the initial parameter set.

After initialisation, the algorithm proceeds to iteratively compute estimates which raise the CV-likelihood of (3.9) (or equivalently its logarithm) until convergence. Every iteration \( r \) entails two steps, the E-step and the M-step.

**E-step** In the Expectation step (E-step), we formulate the expected cross-validated log-likelihood \( Q^{CV}(\theta | \hat{\theta}_{r-1}^{CV}) \) of the incomplete corpus \( \mathcal{X} \) given the parameter estimate from the previous iteration \( \hat{\theta}_{r-1}^{CV} \), by marginalising out the cross-validated set of complete data hypotheses \( z = (x, y) \) provided by \( Z(x; \mathcal{X}^{-k}) \).
\[ Q^{CV}(\theta | \hat{\theta}_{r-1}^{CV}) = E \left[ \log L^{CV}(X; K, \theta) | X, K, \hat{\theta}_{r-1}^{CV} \right] \]
\[ = \sum_{k=1}^{K} \sum_{x \in X^k} \sum_{(x, y) \in Z(x; X^k)} \log \{ p(x, y | \theta) \} \ p(y | x, \hat{\theta}_{r-1}^{CV}) \]  

(3.11)

If from a mathematical point of view the E-step involves formulating the expectation over the CV log-likelihood in (3.11), from an implementation one, as for standard EM, it relates to computing the expected counts \( q^{CV} \) of the cross-validated complete data hypotheses given \( \hat{\theta}_{r-1}^{CV} \).

\[ q^{CV}(x, y | \hat{\theta}_{r-1}^{CV}) = \frac{p(y | x, \hat{\theta}_{r-1}^{CV})}{\sum_{(x, y') \in Z(x; X^k)} p(y' | x, \hat{\theta}_{r-1}^{CV})} \]  

(3.12)

The expected counts \( q^{CV} \) disambiguate between the complete data expansions of each incomplete training point employing the current parameter estimate \( \hat{\theta}_{r-1}^{CV} \) and their computation prepares the ground for the M-step that follows.

**Maximization Step**  In the Maximization step (M-step) of the CV-EM algorithm, we maximise the objective function \( Q^{CV}(\theta | \hat{\theta}_{r-1}^{CV}) \) of (3.11) with respect to \( \theta \) to retrieve the next parameter estimate \( \hat{\theta}_{r}^{CV} \).

\[ \hat{\theta}_{r}^{CV} = \arg \max_{\theta} Q^{CV}(\theta | \hat{\theta}_{r-1}^{CV}) \]
\[ = \arg \max_{\theta} \sum_{k=1}^{K} \sum_{x \in X^k} \sum_{(x, y) \in Z(x; X^k)} \log \{ p(x, y | \theta) \} \ q^{CV}(x, y | \hat{\theta}_{r-1}^{CV}) \]  

(3.13)

The optimisation step of equation (3.13) above is much easier than that of (3.10), given the expected counts \( q^{CV} \) computed during the E-step and which are kept constant during the arg max operation in (3.13). In many applications, as is the case in all of our own work using CV-EM in later chapters of this thesis, the M-step of the CV-EM framework will translate in Relative-Frequency Estimation over the disambiguated corpus of complete data hypotheses.

In any case, the new current estimate \( \hat{\theta}_{r}^{CV} \) computed in (3.13) is then fed as input to the E-step of the following iteration, similarly to the the workflow of Figure 2.10.

**Convergence**  Given that CV-EM is an instance of the EM algorithm as we discuss in more detail below, its application is paired with the guarantee that the iterative process will converge when a local optimum of the CV log-likelihood is reached. A stop condition can then terminate the algorithm when the parameters have sufficiently converged or when the increment of the CV log-likelihood is smaller than a predefined value.
CV-EM as an instance of EM  A comparison of the algorithmic workflow and the equations related to the E-steps and M-steps of the CV-EM algorithm presented in this section and the EM algorithm discussed in section 2.6 easily reveals that the two are closely related. However, a crucial point is that CV-EM is not an EM-like algorithm, somehow reminiscent of EM because of its iterative nature. On the contrary, CV-EM is an instance of the EM algorithm and as such inherits all the algorithmic and statistical estimation properties of the latter.

The single but essential difference between standard MLE and CV-MLE is that the latter employs a different, cross-validated set of hypotheses over the complete data $z = \langle x, y \rangle$ that every observed data point $x$ stems from. After splitting the training data in $K$ parts, we arrive at this new set of hypotheses by making sure that the mapping function $Z(x; \mathcal{X}^{-k})$ between incomplete and complete data excludes the training data part $\mathcal{X}^k$ for every $x \in \mathcal{X}^k$ to compute its output, but rather relies on the rest of the training data $\mathcal{X}^{-k}$.

Similarly, the single but again essential difference between CV-EM and EM is that the latter employs a mapping function $Z(x; \mathcal{X})$ examining all the training data to output complete data hypotheses, while CV-EM replaces this with the cross-validating function $Z(x; \mathcal{X}^{-k})$. Employing $Z(x; \mathcal{X}^{-k})$ as a mapping function in place of $Z(x; \mathcal{X})$, leads from the formulation of the EM algorithm in section 2.6 to that of CV-EM here.

Crucially, the mapping function $Z$ is not part of the EM algorithm’s internals but part of its input. For this reason, CV-EM is a true EM-instance, where we alter the EM algorithm’s input so as to maximise the CV-likelihood of (3.9). This is far from an observation of a purely theoretical nature. On the contrary, it guarantees the Machine Learning practitioner that CV-EM inherits the highly desirable properties of the EM algorithm as set out in section 2.6. In our case, this translates to:

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

Guarantee to Converge  The iterative process will converge to a local maximum of the Cross-Validated likelihood function $L^{\text{CV}}(\mathcal{X}; K, \theta)$.

Overall, these features promote CV-EM as an algorithm with both a clear objective and a well-understood operation. The objective of CV-EM is to discover parameter estimates for generative models with latent variables, which maximise the likelihood of an incomplete data corpus when the set of complete data hypotheses for it is cross-validated. During the iterative operation of CV-EM a series of such estimates is output, each increasing this cross-validated likelihood until a guaranteed convergence towards a local optimum is reached. The combination of Maximum Likelihood Estimation and a Cross-Validated space of complete
Chapter 3. Fragment Models Estimation with the CV-EM Algorithm

data hypotheses, as practically implemented in terms of the CV-EM algorithm, aims towards strong parameter estimates which generalise well, something that we empirically validate successfully in the following chapters of this thesis.

3.2.4 Related Approaches

Cross-Validation Based As we have already discussed in section 3.2.2, we trace the origins of our work in the applications of Deleted Estimation for estimating Language Model parameters. Jelinek and Mercer (1985) employ CV in the process of estimating the parameters of an LM under Maximum Likelihood Estimation. However their model has no hidden variables and the use of CV is confined in identifying n-grams and estimating their conditional probabilities from different parts of the data. After the n-grams participating in the language model have been identified in one part of the training data, the LM estimate is computed analytically from the rest of the data under Relative Frequency Estimation, i.e. complete-data estimation.

Jelinek and Mercer (1980) use CV and an instance of the EM algorithm, the Baum-Welch algorithm (Baum et al., 1970), to estimate model interpolation weights while other model parameters remain constant. Here we describe in general terms both a CV-MLE estimation objective and a Cross-Validated instance of the EM algorithm aiming to estimate all parameters of a model with latent variables. This is of particular importance for cases like the Fragment Models, which do not employ a distinct set of parameters which regulate the balance between fitting the training data and generalising over yet unseen data.

The possibility to employ cross-validation in an iterative estimation procedure has also been explored in (Shinozaki and Ostendorf, 2008). The authors propose an EM-like iterative procedure which keeps a separate model estimate for each of the $K$ parts of the training data $X^k$, which is specifically estimated from and applied on different parts of the training data. The end result is a heuristic estimation algorithm which, while somehow inspired by the workflow of the Expectation Maximization algorithm, is not an instance of the latter and does not inherit its properties. The authors acknowledge this and observe that the training data likelihood can decrease after some iterations and that there is no guarantee for convergence. In contrast, the Cross-Validated EM presented in this chapter is based on a clear learning objective and enjoys desirable properties inherited from EM by being an instance of the latter.

Information Theoretical Apart from solutions employing Cross-Validation, the problem of avoiding overfitting the training data has been widely addressed in the context of model selection: selecting among a host of models the one which, taking the training data that we have available in consideration, is most likely to generalise well. Of particular interest in the context of this thesis are model selection approaches which examine models belonging in the same model family.
which differ in their complexity (e.g., as measured by the number of parameters),
which usually intuitively translates to how fine or coarse grained is the view that
they take on data.

Examples of such approaches are the Akaike Information Criterion (AIC) (Akaike, 1974)
and the Bayesian Information Criterion (BIC) (Schwarz, 1978). Both model selection
criteria are derived starting from information theoretical arguments and penalise models
with a large number of parameters. Typically, increasing the number of free parameters in a
model leads to estimates which further increase the likelihood of the training data. When
selecting a model, both AIC and BIC add a penalising term for the number of parameters in
the model in addition to the likelihood that their estimates assign to the training data, aiming
to mitigate the danger of overfitting them.

A further model selection criterion that also aims to balance how closely a
model fits the training data with the model’s complexity is the Minimum
Description Length (MDL) (Rissanen, 1978; Rissanen, 1983). MDL evaluates
the description length of a model as an indication of its complexity together with
the length of the description it assigns to the training data, which relates to how
closely it predicts them, and it selects the model which minimises the sum of
the two. Model selection criteria such as these have found applications in NLP
such as (Grünewald, 1996; Goldsmith, 2001; Adriaans and Jacobs, 2006; Poon et
al., 2009), either by being directly applied or by inspiring approaches which try
to counterbalance the increasingly better fit of the training data offered by large
models, aiming towards larger generalisation capacity. Nevertheless, the applica-
tion of such criteria is often challenging (Adriaans and Vitanyi, 2007), and the
difficulties mount up when applying MDL learning for the frequently complex
models employed in NLP.

Regularisation  A wider framework that, from a technical and a formalisation
perspective, encompasses the AIC and BIC criteria as well some applications of
the MLD principle (e.g., ‘crude’, two-part MDL (Grünewald, 2007)), is regular-
isation. The regularisation of the log-likelihood optimisation objective involves
adding a penalty term $R(\theta)$, whose role is to penalise models of increased
complexity or model instances which overfit the training data. The impact of the
penalty term in the optimisation criterion is weighted by a parameter $\alpha$, which
allows us to adjust the tradeoff between data fit and model complexity.

$$\hat{\theta} = \arg \max_{\theta} \log L(\mathcal{X}; \theta) - \alpha R(\theta) \quad (3.14)$$

The frequently applied $L_p$-norm regularisation involves using the $L_p$-norms
$||\theta||_p$ of the parameter vector $\theta$ in the regularisation term $R(\theta)$. 
Chapter 3. Fragment Models Estimation with the CV-EM Algorithm

\[ ||\theta||_p = \left( \sum_i |\theta_i|^p \right)^{\frac{1}{p}} \]  

(3.15)

A member of this family that has been widely applied in Machine Learning literature is \( L_1 \)-norm regularisation, which uses a penalty term equal to the sum of the parameter values. This has been shown to prefer in practice model instances with many parameters equal to zero (Tibshirani, 1996); when this happens, an \( L_1 \)-norm optimisation objective essentially performs model selection and model smoothing at the same time. The AIC and BIC criteria can be interpreted as instances of \( L_0 \)-“norm” regularisation; \( ||\theta||_0 \), as \( p \) approaches zero and under certain simplifying assumptions, merely counts the number of non-zero parameters in vector \( \theta \).

**Bayesian** Instead of penalising whole model spaces, as the model selection criteria do based on their complexity as measured by their number of parameters or description length, the use of Bayesian inference allows us to employ external preferences over the structure and the parameter space of a model. These are stated in terms of probabilistic priors which allow preferring certain model structures, or parts of the parameter space for which the modeller hopes that estimates situated there generalise better. Frequently used choices for such priors for models employing multinomial distributions, such as is commonly the case in NLP, are the Dirichlet distribution (Johnson et al., 2007; Zhang et al., 2008a) and the Dirichlet Process (Liang et al., 2007; DeNero et al., 2008; Blunsom et al., 2009), which can both be tuned with the help of hyperparameters to prefer more compact model estimates, as discussed for the case of Fragment Models in section 3.1.8. Inference with Bayesian priors such as these typically involves marginalising out the model parameters given the training data.

While Bayesian inference provides an interesting theoretical framework to employ external biases to arrive at models and model estimates which generalising better, its practical application does not come without shortcomings. Bayesian methods are centred around the modelling step of introducing external knowledge, which while it can prove beneficial in many cases, still entails the dangers of arriving at suboptimal solutions and overwhelming the empirical evidence under the strength of the external prior imposed. Furthermore, in practice Bayesian approaches are frequently sensitive to the choices involved in approximating model parameter marginalisation using sampling or variational methods, as well as in the selection of the prior’s hyperparameters.

Overall, CV-EM takes a data-driven approach on the problem of finding estimates which generalise well, focusing on avoiding formulating and validating hypotheses using the same data set. In contrast to CV-EM’s concentration on the empirical evidence, most of the approaches mentioned above either emphasise employing external knowledge for the task, or take an information theoretic
view on the problem. It is interesting to note that CV-EM is not mutually exclusive with these alternative approaches. While we believe that on one hand the data-driven nature of CV-EM safeguards against arbitrary modelling choices, we still find interesting investigating the crossroads between ours and alternative approaches on the problem of model estimation for increased generalisation.

3.2.5 CV-EM for Fragment Models

In the next chapters of this thesis, we will experiment with applying the CV-MLE optimisation criterion as implemented through the CV-EM algorithm for an array of Statistical Machine Translation models, all of which belong to the family of Fragment Models. Before we delve into the details of applying CV-EM for each particular problem, we close this section by discussing what we might expect from employing our methods for the estimation of Fragment Models in more abstract terms.

Balancing MLE Bias and Variance

In section 3.1.6, we showed that the large expected error of the MLE estimator for Fragment Models can be attributed to an unbalanced correspondence between errors attributed to estimator bias and variance. The generalisation error due to estimator bias is zero, or very low in case we constrain the size of the fragments, which however gives rise to a large error due to the variance of the estimates in respect to the training data. CV-EM reduces the overall expected test error by increasing the estimator’s bias in a targeted manner.

For Fragment Models, hypotheses over the value of the unobserved data variables relate to the segmentation of training instances in data fragments and the generative steps that are followed to arrive at the observed data points. CV-EM cross-validates this hypothesis space, so that all the hypotheses that will be considered employ both reusable fragments and reusable derivational steps according to the cross-validation criterion. This brings about an increase in estimator bias error due to moving away probability mass from the largest fragments, most of which will fail to survive the application of CV.

However, in contrast to arbitrary constraints such as fragment length cut-off points, increasing the generalisation error’s bias term in this way directly aims to greatly reduce the error due to estimate variance. The CV-MLE estimates, focusing on reusable fragments and derivation steps, will differ with each other much less than the MLE estimates, each of which only predicts each different sampled training data set. The end result is a better trade-off between the expected test error’s bias and variance terms\(^3\), which lowers the overall generalisation error of CV-EM Fragment Model estimates.

\(^3\)See section 2.7.1 for more details on the trade-off between bias and variance.
Model Selection and Estimation A further interesting facet of CV-EM estimation that is relevant to FM estimation, is that it combines features of both model selection and model estimation. This property of CV-EM is highly applicable to Fragment Model estimation, where parameters related to the model’s level of abstraction from the training data are not clearly separated from the rest of the model parameters, as is for example the case for mixture models. Easy solutions such as separating FMs into an array of models which each employs fragments of a certain size are not working either, as some larger fragments capture data regularities while others simply overfit training data particularities. This makes it difficult for FMs to precede estimation with a clearly separated model selection step. CV-EM addresses it by applying features of model selection during estimation of the model’s parameters.

On the one hand, the cross-validated complete data hypothesis space which CV-EM considers is a subset of that employed in standard applications of EM. Assuming this hypothesis space when maximising training data likelihood as CV-EM does, effectively shapes the model set that will be considered, by not considering models leading to hypotheses which do not survive cross-validation. While no single model is selected, models employing fragments which do not appear to generalise well according to CV are either eliminated or penalised, depending on the smoothing choices made in each application of CV-EM. On the other hand, when the extent of the model space and the preferences over it have been set, estimating model parameters by maximising training data likelihood allows us to discover which estimate seems to better capture the latent patterns of the training data, cross-validating our hypotheses over them to safeguard against overfitting.

In total, these features of CV-EM together with the algorithmic and statistical estimation properties inherited by the Expectation-Maximisation algorithm promote it as a well-founded and highly suitable estimation framework for Fragment Models. In the next chapters of this thesis, we empirically evaluate CV-EM for Fragment Model estimation, for three Statistical Machine Translation models of increasing sophistication belonging to this family.