Sinuhe — Statistical Machine Translation using a Globally Trained Conditional Exponential Family Translation Model

Matti Kääriäinen
Department of Computer Science
FI-00014 University of Helsinki, Finland
matti.kaariainen@cs.helsinki.fi

Abstract

We present a new phrase-based conditional exponential family translation model for statistical machine translation. The model operates on a feature representation in which sentence level translations are represented by enumerating all the known phrase level translations that occur inside them. This makes the model a good match with the commonly used phrase extraction heuristics. The model's predictions are properly normalized probabilities. In addition, the model automatically takes into account information provided by phrase overlaps, and does not suffer from reference translation reachability problems.

We have implemented an open source translation system Sinuhe based on the proposed translation model. Our experiments on Europarl and GigaFrEn corpora demonstrate that finding the unique MAP parameters for the model on large scale data is feasible with simple stochastic gradient methods. Sinuhe is fast and memory efficient, and the BLEU scores obtained by it are only slightly inferior to those of Moses.

1 Introduction

In current phrase-based statistical machine translation systems such as Moses[1] (Koehn et al., 2007), the translation model is defined in terms of phrase pairs (biphrases) extracted from a bilingual corpus as follows. The corpus is first word-aligned using a word alignment heuristic (Och and Ney, 2003). The phrase extraction heuristic then extracts all the biphrases that are compatible with the word alignment (Och et al., 1999). This way, each sentence pair may generate any number of potentially overlapping biphrases. However, when defining the phrase-based sentence level translation model, phrase overlaps are explicitly disallowed: The source sentence is segmented into disjoint phrases, which are translated independently using conditional phrase-level translation models that have been estimated from extracted biphrase counts.

The disparity between the phrase extraction heuristic and the use of the extracted biphrases can be addressed in at least three ways. One approach is to simply ignore the disparity as is done, e.g., in Moses. While empirically succesful, this approach is hard to justify theoretically, and begs the question of whether more principled methods might lead to better translation results. The other extensively studied approach is to replace the phrase extraction heuristic with a method that better matches the use of the extracted phrases (see, e.g., (Marcu and Wong, 2002; DeNero et al., 2008) and the references therein). While theoretically sound, this approach is computationally challenging both in practice (DeNero et al., 2008) and in theory (DeNero and Klein, 2008), may suffer from reference reachability problems (DeNero et al., 2006), and in the end may lead to inferior translation quality (Koehn et al., 2003).

In this paper, we study a third alternative. We propose a new translation model that is compatible with the phrase extraction heuristic. The proposed machine learning inspired translation model takes the form of a conditional exponential family probability distribution over a feature representation for word-aligned sentence pairs. The feature representation represents a word-aligned sentence pair by essentially enumerating the (multi)set of biphrases that would have been extracted from it,

[1] Throughout this paper, we refer to Moses for concreteness, but most of the discussion applies to other standard phrase-based statistical machine translation systems as well.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1027–1036,
Singapore, 6-7 August 2009.

together with the source positions at which the biphrases occur. The model's predictions are conditional probabilities for such sets of biphrases given the source sentence.

The chosen feature representation has many advantages. Since all word-aligned sentence pairs can be represented, reference reachability problems are automatically circumvented. For example, if the translation of a sentence consisted solely of words that do not occur in the phrase table, then the feature vector for the translation would be the all zero vector. As the training data receives nonzero probability, maximum likelihood or maximum a posteriori (MAP) parameters for the model can be estimated in a principled way without resorting to pseudo-references. The fact that the model is not restricted to using disjoint biphrases means that the information in biphrase overlaps is automatically taken into account. This may help in smoothing the model's predictions on long and rare phrases, and in enhancing fluency at places that otherwise would be phrase boundaries. Also, the model can be extended in a principled way by introducing additional features (e.g., translations from a dictionary, biphrases with gaps, biphrases over POS tags,...).

The proposed model has one parameter per biphrase feature, so the total number of parameters is easily millions or more. Still, the model structure is designed so that feature expectations and related quantities can be computed efficiently by dynamic programming. It is thus feasible to compute the gradient of the MAP objective, and simple gradient ascent can be used to efficiently find the globally optimal model parameters (with respect to a suitably scaled Gaussian prior used for regularization). Exact inference is also possible by dynamic programming when translations are predicted by the translation model alone. When other features like a language model are included, one needs to resort to beam search type approximate dynamic programming for decoding.

We have implemented a translation system called `Sinuhe` based on the proposed translation model. The system has been released under the GPLv3 open source license (Kääriäinen, 2009). Our experiments on Europarl and GigaFrEn corpora demonstrate that the proposed translation model scales well to large data, and offers translation quality that is only slightly worse than that of the baseline system `Moses`. In terms of translation speed, `Sinuhe` is already clearly better.

The rest of this paper is organized as follows. After briefly reviewing related work in Section 2, we describe the proposed translation model in Section 3. Finally, experimental results are presented in Section 4, and conclusions in Section 5.

2 Related work

The proposed translation model is strongly influenced by machine learning techniques for solving sequence prediction tasks, most notably the work on conditional random fields (Lafferty et al., 2001). The modelling task in machine translation is, however, more complicated than sequence labelling (not one-to-one, reorderings), so the standard methods cannot be directly applied here. The model we propose is also related to standard phrase-based translation models through the use of the same phrase-level translation features. However, the way we use the features is quite different.

There exists a number of discriminative approaches whose model structure, training criteria, or both, are similar to ours. However, to our knowledge, none of the other systems operates directly on biphrase features, scales up to bilingual corpora with millions of sentence pairs, and achieves translation quality comparable to fully tuned standard phrase-based systems. The approach most closely resembling ours is the independently developed global discriminative log-linear model based on synchronous context-free grammars (Blunsom and Osborne, 2008; Blunsom et al., 2008). The version presented in (Blunsom and Osborne, 2008) operates on millions of rule count features analogous to our biphrase features, and integrates a language model into training and decoding. The system can be trained on tens of thousands of short sentences yielding better translations than a baseline system `Hiero` on this data. The version presented in (Blunsom et al., 2008) scales to more than a hundred thousand short training sentences, but does not integrate a language model and thus has performance that improves upon `Hiero` without a language model only. Both versions deal with derivational ambiguity by treating derivations as a latent variables that are integrated out to get conditional probabilities for translations[2]. The downside of this

[2]In our translation model, coping with multiple derivations is not needed as there is just one derivation for each word-aligned sentence pair. However, dealing with alternative word alignments might be beneficial, though as argued

is that approximations are needed in computing the maximum probability translation in decoding, and also in computing model expectations in training when a language model is used. In addition, since the models operate directly on translations, using probabilistic training criteria for learning the model parameters is possible only if all reference translations in the training data can be generated by the model. In practice, this problem can be circumvented by discarding the training sentence pairs with unreachable reference translations, but this may mean a significant reduction in the amount of training data (24% in (Blunsom et al., 2008)).

Another closely related approach is the independently developed discriminative block bigram prediction model presented in (Tillmann and Zhang, 2007). This work proposes a global phrase-based translation model very similar to ours, but due to computational reasons, resorts to a localized approximation thereof, and is restricted to biphrases of length at most two. In (Liang et al., 2006) a standard phrase-based model is augmented with more than a million features whose weights are trained discriminatively by a variant of the perceptron algorithm. Reference reachability is again a problem, and the method has not been scaled up to use biphrase features directly.

3 The proposed translation model

3.1 Biphrase extraction

The biphrases used in `Sinuhe` are extracted from the training data with the `Moses` phrase extraction heuristics. The sentence-aligned training corpus S is first word-aligned by running Giza++ in both directions and then symmetrizing the alignments. This maps the original aligned sentence pairs (x, y) into word-aligned sentence pairs (x, a, y), where a is a many-to-many alignment between the words in x and y. Second, using a heuristic proposed in (Och et al., 1999), all the aligned phrase pairs (x', a', y') satisfying the following criteria are extracted: (1) x' and y' consist of consecutive words of x and y, and both have length at most k, (2) a' is the alignment between words of x' and y' induced by a, (3) a' contains at least one link, and (4) there are no links in a that have just one end in x' or y'. Each aligned training sentence (x, a, y) thus generates a number of potentially overlapping aligned biphrase features (x', a', y'). In our experiments, we chose $k = 7$ which is the default in `Moses`. Unlike in `Moses`, we do not map the aligned biphrases (x', a', y') back to non-aligned biphrases (x', y').

To reduce the number of extracted biphrases, for each source phrase x', only biphrases (x', a', y') whose occurrence count is among the top $K = 20$ in the training data are retained (rank ties broken by including all biphrases with rank equal to the limit K). For technical reasons related to our dynamic programming algorithms, we also drop biphrases whose source phrase begins or ends with unlinked words. Finally, we drop all biphrases that occur only once in the training data. This can be motivated by a leave-one-out argument (cf the derivation of Good-Turing estimates): Dropping the biphrases that occur only once in the training data means that the feature representation for a training sentence pair (see Section 3.2) contains only biphrases that occur also in other training examples. Without the leave-one-out pruning, the feature vectors for training sentence pairs would be maximally dense, whereas such feature density cannot be expected on test data. Our system can also be used without the leave-one-out pruning, but according to our preliminary experiments this has little effect on translation quality. An exception seems to be morphologically rich languages with scarce training data on which pruning seems to reduce translation quality.

All the pruning steps combined reduce the phrase table size considerably, but in our experiments, millions of biphrases per language pair still remain (2-4 million for Europarl data and over 95 million for GigaFrEn data).

3.2 Features

Our primary feature representation is a binary feature vector that indicates which aligned biphrases in the phrase table occur in an aligned sentence pair and where. More specifically, a source sentence x aligned to a target sentence y by an alignment a is represented by a binary feature vector $\phi(x, a, y)$ whose component $\phi(x, a, y)_{(x', a', y'), i}$ is 1 iff the aligned biphrase (x', a', y') occurs at source position i in (x, a, y), and 0 otherwise. Here, (x', a', y') occurs in (x, a, y) at source position i iff the phrase extraction process described in Section 3.1 would have extracted it from (x, a, y) at source position i.

in (DeNero et al., 2006), the ambiguity in word alignment is less prevalent than in phrase segmentation.

The weights for the aligned biphrases are tied together by mapping the binary feature vector ϕ (indexed by pairs of an aligned biphrase and a source position) to an integral feature vector $\tilde{\phi}$ (indexed by aligned biphrases only) using the formula $\tilde{\phi}_{(x',a',y')} = \sum_i \phi_{(x',a',y'),i}$. The "real" features that drive the translation process are thus the lowest level binary features ϕ, whereas the higher level representation $\tilde{\phi}$ is convenient in defining the conditional probabilities given by the translation model.

3.3 The model

Instead of modelling the conditional distribution $P(y|x)$ directly, we model the conditional distribution $P(\phi(x,a,y)|x)$ by the following conditional exponential model:

$$P(\phi(x,a,y)|x) = \frac{\exp(w \cdot \tilde{\phi}(x,a,y))}{\sum_{\phi \in \Phi_x} \exp(w \cdot \tilde{\phi})}.$$

Here, w is a parameter vector with one component for each aligned biphrase feature in the phrase table. The set Φ_x defines the set of possible predictions given x, and includes all feature vectors ϕ satisfying the following criteria:

1. There exists a translation y' and an alignment a' such that all active features in ϕ occur in (x, a', y')
2. Features corresponding to aligned biphrases that occur inside aligned biphrases whose features are active in ϕ are also active in ϕ.

Thus, the set Φ_x has a feature representation for all possible aligned sentence pairs (x, a', y') that have x as the source side, so all reference translations y' word-aligned to x in any way a' are representable by features in Φ_x. By condition 1, the predictions given by the model never contain conflicting biphrases, so given any prediction of the model, there always exists a translation y' where all the predicted biphrases do occur. However, since our dynamic programming algorithms can only force active super-phrases implying active sub-phrases (condition 2) but not active sub-phrases implying active super-phrases, the set Φ_x also contains some feature vectors in which the latter type of implications are not enforced. Having such redundant representations for some translations is a waste of probability mass, but we hope it has little effect in practice.

The choice of modelling $P(\phi(x,a,y)|x)$ instead of modelling $P(y|x)$ directly is crucial, both from a modelling and from a computational perspective. From the modelling perspective, the crucial point is that in our approach, any aligned sentence pair (x,a,y) has an associated feature vector $\phi(x,a,y) \in \Phi_x$ that is reachable (i.e., receives non-zero probability) by the model. This means it is straightforward to use probabilistic criteria in learning the model parameters. In contrast, systems modelling $P(y|x)$ directly are often plagued by the reference reachability problem. To use probabilistic training criteria for such systems one needs to circumvent the reference reachability problem, e.g., by using pseudo-references or by dropping out the non-reachable portion of training data.

Working with the feature vectors $\phi(x,a,y)$ instead of working with a and y directly means that we model the ordering and choice of words in y only partially. This way, when computing the normalizing constants and feature expectations, we can partition the unbounded set of potential translations y and alignments a into a smaller set of equivalence classes given by $\phi(x,a,y)$. Though the number of feature vectors $\phi \in \Phi_x$ may be large (exponential in length of x), all the necessary computations can be done exactly and efficiently by dynamic programming. For more details, see Section 3.4.3.

3.4 Learning the model parameters

3.4.1 The objective

We use maximum a posteriori (MAP) estimation to estimate the model parameters w. To control overfitting, we regularize the parameters by a suitably scaled Gaussian prior. This can be also viewed as $L2$ regularization. The prior guarantees that the MAP parameters are unique, and models our belief that the observed feature occurrence counts randomly deviate from their "true" values roughly proportionally to the standard deviations of the occurrence count distributions. The prior variance $\sigma^2_{(x',a',y')}$ for feature (x',a',y') is given by the formula $\sigma^2_{(x',a',y')} = \alpha/\rho_{(x',a',y')}$, where $\alpha > 0$ is a free regularization parameter, and $\rho^2_{(x',a',y')}$ is an empirical estimate of the variance of the occurrence count of (x',a',y') in the training data. This is similar to (Chen and Rosenfeld, 2000), except that we use standard deviations in place of variances. As the estimate for the vari-

ance of a feature we use the occurrence count of the corresponding biphrase in the training data. This could be justified by assuming that the occurrence counts follow a Poisson distribution. We have also run preliminary experiments with other forms of regularization (different ways of computing $\sigma_{(x',a',y')}$, exponential priors corresponding to $L1$ regularization, no regularization), and it looks like the system is not very sensitive to the chosen prior.

Combining the prior with the model, we see that the negative log-posterior $\mathcal{L}(w)$ is given by the formula

$$\sum_{(x',a',y')} \frac{w^2_{(x',a',y')}}{2\sigma^2_{(x',a',y')}} - \sum_{(x,a,y)\in S} \log P(\phi(x,a,y)|x) + C,$$

where the sum over (x', a', y') is understood to go over all aligned biphrase features in the model. This is our criterion for learning w.

3.4.2 Optimization

We solve the optimization problem related to learning w by first order gradient ascent methods. The gradient $\nabla\mathcal{L}(w)$ of $\mathcal{L}(w)$ with respect to w can be written as

$$\sum_{(x',a',y')} \frac{w_{(x',a',y')}}{\sigma^2_{(x',a',y')}} - \sum_{(x,a,y)\in S} \left[\tilde{\phi}(x,a,y) - \mathbb{E}_w[\tilde{\phi}|x]\right],$$

where $\mathbb{E}_w[\tilde{\phi}|x]$ denotes the conditional expectation of the aligned biphrase occurrence count features given x with respect to model parameters w. Feature expectations can be computed by combining the results of a left-to-right and right-to-left dynamic programming sweep over the source sentence. For more details, see Section 3.4.3.

Inspired by the empirical results in (Vishwanathan et al., 2006), we use classic stochastic gradient ascent to solve the optimization problem. At each step t, we sample with replacement a batch S_t of b examples from S. We start from $w_0 = 0$, and use the update rule

$$w_{t+1} = w_t - \eta_t \nabla\mathcal{L}_t(w_t), \qquad (1)$$

where $\eta_t > 0$ is the learning rate, and $\nabla\mathcal{L}_t(w)$ is the stochastic gradient of the negative log-posterior

$$\mathcal{L}_t(w) = \frac{|S_t|}{|S|} \sum_i \frac{w_i^2}{2\sigma_i^2} - \sum_{(x,a,y)\in S_t} \log P(\phi(x,a,y)|x)$$

restricted to batch S_t. The second term of the stochastic gradient involves only biphrases whose source sides match the source sentences in the batch. Though the gradient of the regularizer is non-zero for all non-zero biphrase features, the updates of features that are not active in the second term of the gradient can be postponed until they become active again. Due to feature sparsity, the number of features that are active in a small batch is small, and thus also the updates are sparse. Hence, it is possible to handle even feature vectors that do not fit into memory.

Another advantage of the stochastic gradient method is that many processes can apply updates (1) to a weight vector asynchronously in parallel. We have implemented two strategies for dealing with this. The simpler one is to store the weight vector in a database that takes care of the necessary concurrency control. This way, no process needs to store the entire weight vector in memory. The downside is that all training processes must be able to `mmap()` to the common file-system due to limitations in the underlying Berkeley DB database system. We have also implemented a client-server architecture in which a server process stores w in memory and manages read and update requests to its components that come from training clients. In this approach, the degree of parallelism is limited only by the number of available machines and server capacity. The server could be further distributed for managing models that do not fit into the memory of a single server.

3.4.3 Computing gradients etc

The computationally most challenging part in learning the model parameters is computing $\nabla \log P(\phi(x,a,y)|x)$, i.e., the vector of differences between the observed occurrence counts of biphrase features in (x, a, y) and their conditional expectations under the current model parameters.

The conditional feature expectations can be computed by a dynamic programming procedure similar to the one used in training conditional random fields. We combine the results of a left-to-right and right-to-left dynamic programming

sweep over x. In the left-to-right sweep, we have for each biphrase feature $(x', a', y'), i$ a state $s_{(x',a',y'),i}$ for translations starting from the beginning of x and ending in an occurrence of the biphrase (x', a', y') at source position i. This state records the contribution of all partial translations whose right-most active biphrase feature on the source side is $(x', a', y'), i$ to the conditional expectation of feature $(x', a', y'), i$ (in log scale). The score for $s_{\text{empty},0} = 0$, and $s_{(x',a',y'),i}$ is obtained from the recurrence

$$\text{score}\left(s_{(x',a',y'),i}\right) = \sum_{(x'',a'',y''),i'' \in A((x',a',y'),i)} \Big[\text{score}\left(s_{(x'',a'',y''),i''}\right) + \sum_{(x''',a''',y'''),i''' \in B} w_{(x''',a''',y'''),i'''}\Big]$$

Here, $A((x', a', y'), i)$ is the set of predecessor states of $s_{(x',a',y'),i}$ and includes all states $s_{(x'',a'',y''),i''}$ such that a proper suffix of $(x'', a'', y''), i''$ (i.e., a biphrase whose source and target are proper suffices of x'' and y'' respectively) is equal to a prefix of $(x', a', y'), i$. As a special case, A includes all states $s_{(x'',a'',y''),i''}$ for which $(x'', a'', y''), i''$ ends before or at position i. This takes care of translation paths that leave some words in x untranslated. Since the starting position of a proper suffix of a biphrase is always after the biphrase's original starting position, going through the states in order of increasing i guarantees that the scores for biphrases in $A((x', a', y'), i)$ are available when computing score $\left(s_{(x',a',y'),i}\right)$.

The set B that depends on $((x', a', y'), i)$ and $((x'', a'', y''), i'')$ is defined by the formula

$$B = \text{sub}\left((x', a', y'), i\right) \setminus \text{sub}\left((x'', a'', y''), i''\right),$$

where sub $((x', a', y'), i)$ denotes the set of sub-biphrases of $(x', a', y'), i$ (including the biphrase $(x', a', y'), i$ itself). Thus, summing over the weights of biphrases in B adds the contribution of features introduced by extending translation paths ending in $(x'', a'', y''), i''$ by $(x', a', y'), i'$.

From the right-to-left dynamic programming, we get analogously the contribution of right-to-left partial translations whose left-most active biphrase is $(x', a', y'), i$. The partition function used for normalizing the expectations can be obtained as a side product of either of the sweeps.

In conditional random fields, the (unnormalized) expectations for the feature can be obtained by multiplying the scores of the states corresponding to the same feature in the left-to-right and right-to-left dynamic programming memories. In our case, combining the two values stored in the states for a feature $(x', a', y'), i$ only gives the contribution of the translation paths where $(x', a', y'), i$ is active but not covered by any longer biphrase that extends $(x', a', y'), i$ both left and right. To include the contribution of the remaining translation paths, we need to go through states corresponding to super-biphrases of $(x', a', y'), i$. Special care has to be taken in order to include the contribution of all feature vectors in which such super-biphrases are active exactly once. An efficient way to do this is to process the states for super-biphrases in topological order with respect to biphrase inclusion, and to include only the contributions of states for super-biphrases that extend the previously included states both left and right.

Another complication in the dynamic programming is that a biphrase can extend the source side of another overlapping biphrase to the right, but the target side to the left, or visa versa. Such overlaps are not directly covered by our dynamic programming. To deal with them, we construct new virtual biphrases that correspond to the results of such overlaps in a pre-processing step. The number of such virtual combinations can in theory grow exponentially, but in practice only a small number of virtual biphrases seems to suffice.

3.5 Prediction with translation model alone

Prediction is done in two phases. First, we find (by a dynamic programming procedure similar to the one outlined in Section 3.4.3) the highest probability feature vector $\hat{\phi}(x)$ defined by

$$\hat{\phi}(x) = \underset{\phi \in \Phi_x \,:\, x \text{ covered by biphrases in } \phi}{\arg\max} P(\phi|x).$$

Note that we restrict the search to feature vectors that cover the whole of x, i.e., to feature vectors $\phi(x, a, y)$ in which each word in x is covered by at least one active aligned biphrase (x', a', y'). This forces the system to translate all words in the source sentence even if the translation model predicts that none of the translations are very likely.

To translate words that are not covered by any aligned biphrase feature in the model, we use the following strategy: If the word is found from an

optional out-of-vocabulary dictionary, we use the translation from the dictionary, and otherwise resort to an implicit zero weight aligned biphrase that copies the input word to the output as is. In our experiments, the out-of-vocabulary dictionary is constructed from the word translations that occur once in the training data, so the out-of-vocabulary dictionary only compensates for the word translations lost in phrase table pruning. If available, a real dictionary could be used as well.

The second step in predicting a translation is solving the pre-image problem, i.e., constructing a translation y from the predicted feature vector $\hat{\phi}(x)$. Since $\hat{\phi}(x) \in \Phi_x$, there always exists an alignment a and a translation y such that all the aligned biphrases in $\hat{\phi}(x)$ occur in $\phi(x, a, y)$, but the a and y may not be unique. We choose the a and y given by concatenating the target sides of the biphrases active in $\hat{\phi}(x)$ in the order induced by their positions in the source sentence. Thus, there is no phrase-level reordering, and the fluency of the target language output is induced by the phrase overlaps only.

3.6 Predicting with an integrated LM

The prediction strategy outlined in the previous section is simple and conceptually clean. However, biphrase overlaps alone may not be enough to enforce fluent output, especially given that bilingual data is typically more scarce than monolingual data. Also, the lack of a reverse translation model means the system is unable to identify phrase extraction errors in which rarely seen source phrases are translated to common target phrases by chance.

To address these shortcomings, we augment the translation model with the following additional features that have been observed to enhance translation quality in other SMT systems.

1. **Language model:** $\log P(y)$, where $P(y)$ is given by a smoothed n-gram language model
2. **Lexical translation model (reverse direction):** $\log P(x|y, a)$ given by a word-level reverse translation model
3. **Translation length:** number of words in y
4. **Distortion:** number of source words in phrases with swapped translations

The final score driving the translation process is given by a linear combination of the translation model score $\log P(\phi(x, a, y)|x)$ and these features. Besides the translation model, the language model feature is clearly the most influential, while the lexical translation feature has only a minor positive effect on translation quality.

We use an approximate dynamic programming variant of the commonly used beam search procedure to find the highest scoring candidate translation. We compute the translation model log-probability $\log P(\phi(x, a, y)|x)$ incrementally while building up the corresponding candidate translation y and word alignment a from left to right. We allow phrase-level distortions given by swapping the order of translations of consecutive non-overlapping source phrases. Unlike in `Moses`, our beam search is structured around state transitions, not around states. This means that we apply each biphrase (state transition) simultaneously to all applicable partial translations (states). This strategy is in our experience more efficient, does not rely on future score estimates, and is implementationally very similar to the dynamic programming procedures that we use in training the model parameters and in prediction with a language model alone.

The weights of the features are tuned by optimizing the BLEU score of development set translations with amoeba search. This simplistic strategy is feasible given our system's fast translation speed, and extends easily to cover non-linear feature combinations. The reason for using amoeba is that it is simpler to implement — we do not believe amoeba yields any better values for the parameters in the end.

4 Experiments

4.1 Experimental setup

Our experiments are on the Europarl translation tasks following the setup used in the shared translation task of the ACL 2008 Third Workshop on Statistical Machine Translation (Callison-Burch et al., 2008), and on the French-to-English translation task of the EACL 2009 Fourth Workshop on Statistical Machine Translation (Callison-Burch et al., 2009). The size of the Europarl training corpora is about 1M sentence pairs per language pair, while the larger GigaFrEn corpus contains about 22M sentence pairs. The corpora were used for biphrase extraction and translation model training. Decoder feature weights were tuned on the provided development sets. In case of Europarl, language models were trained on the target sides of

	es-en	en-es	fr-en	en-fr	de-en	en-de	time
Sinuhe	31.38	30.94	31.50	28.91	25.03	19.26	338.0
Moses	32.18	31.88	32.63	29.92	27.30	20.57	3729.5
$\text{Sinuhe}_{\text{trans}}$	29.14	27.12	28.74	26.06	22.38	17.14	44.2
$\text{Moses}_{\text{trans}}$	24.32	22.75	23.84	21.22	19.62	13.59	1321.5

Table 1: Left: The translation quality of the SMT systems as measured by the BLEU score. Translations were detokenized but not recased before evaluating their quality against lowercased reference translations by the `mteval-v11b.pl` script. Right: Average total translation time in seconds.

the bilingual corpora. In the GigaFrEn experiments we used the provided monolingual news domain data. All data was tokenized and lowercased using the tools in the Moses distribution.

We experimented with four translation systems: $\text{Sinuhe}_{\text{trans}}$, Sinuhe, $\text{Moses}_{\text{trans}}$, and Moses. $\text{Sinuhe}_{\text{trans}}$ uses only the translation model in producing translations (see Section 3.5), while the full system Sinuhe uses also a language model and some additional features (see Section 3.6). As a baseline, we used the Moses translation system, which is known to be very competitive on the Europarl translation tasks as evidenced by the University of Edinburgh entries in the translation challenge (Callison-Burch et al., 2008). The other comparison point $\text{Moses}_{\text{trans}}$ was obtained from Moses by disabling distortions and setting setting the weights of all features except the forward translation model to 0. By comparing $\text{Sinuhe}_{\text{trans}}$ and $\text{Moses}_{\text{trans}}$, we hope to indirectly compare the performance of the underlying translation models. A more direct comparison was not possible as it is not feasible to normalize the "probabilities" predicted by the Moses translation model.

4.1.1 Training the models

We trained Moses exactly as suggested in (Callison-Burch et al., 2008), except that we used the `-unk` option for SRILM in training the language models (both for Sinuhe and Moses). The translation model for Sinuhe (and $\text{Sinuhe}_{\text{trans}}$) was built from the phrases extracted by Moses as described in Section 3.1. We chose $\alpha = 1.0$ and set batch size to 1. The learning rate was initially set to 0.1, and decayed proportional to $1/t$ after 2M or 100M iterations of training for Europarl and GigaFrEn tasks, respectively. These choices may not be optimal as we did not experiment with other choices yet. In case of Europarl, training was run for 70-100M iterations using the Berkeley DB based distribution strategy (4 CPU cores per language pair). This took 10 days. For GigaFrEn, we used the client-server architecture, and trained the model for 620M stochastic gradient iterations on about 200 CPUs. This took 2 days, which is a lot less than the time needed to run (parallel) Giza on this data. The number of biphrase features in Sinuhe's model was 2-4 million on the Europarl tasks, and about 95 million on the GigaFrEn task.

The decoder parameters for Sinuhe were tuned on the development sets by amoeba, and for Moses by MERT. As both amoeba and MERT try to solve the same optimization problem, we believe the difference in optimization methods has little influence on the results.

4.1.2 Translation results

Europarl tasks The systems were tested on the 2000 sentence Europarl domain development test sets provided for the shared translation task (Callison-Burch et al., 2008). The resulting BLEU scores and total translation times averaged over the datasets are reported in Table 1. While Moses has the highest BLEU score for all the language pairs, the BLEU score for Sinuhe is worse by only at most 1.31 BLEU points except on the de-en task, where the difference is 2.27. $\text{Sinuhe}_{\text{trans}}$ is clearly inferior to Sinuhe but equally clearly superior to $\text{Moses}_{\text{trans}}$.

It takes less than a minute to translate the development test set by the fastest system $\text{Sinuhe}_{\text{trans}}$. The slowest system Moses needs around an hour for the same task. Memory usage follows a similar pattern. For example, Sinuhe requires roughly one tenth of the memory used by Moses. Thus, in terms of resource usage, $\text{Sinuhe}_{\text{trans}}$ and Sinuhe seem clearly superior to Moses. The quantitative results would change if the systems' parameters were optimized for speed rather than quality, but the differences

are so clear that the general pattern would probably remain the same. For example, Moses with no distortion is still clearly slower than Sinuhe.

GigaFrEn task The fr-en model was tested on the 2525 sentence news domain test data used in the preliminary evaluation of the translation challenge results. The BLEU scores for Sinuhe and Moses were 26.32 and 26.98, respectively. The total translation time was 13m 50s with Sinuhe, whereas Moses needed 82m 28s. Thus, the pattern that was observed on Europarl tasks is repeated here: The translation quality of Moses is slightly better, but Sinuhe is significantly faster. Surprisingly, both Sinuhe and Moses fare well in comparison to the participants of the actual challenge: According to the preliminary results on the same test data we used (Koehn, 2009), the Moses baseline would have been beaten only by Google, and Sinuhe would have been sixth among the 23 participating systems with a difference of only 0.57 BLEU points to the second best entry. A partial explanation for the good relative performance could be that the challenge participants had only a week to train their models on the full version of GigaFrEn data, so they may not have had time to take full advantage of it. On the other hand, many of the top ranked systems relied on external resources that were not available for us.

Based on an informal human evaluation of the outputs of Sinuhe and Moses, it looks like the translations of Sinuhe are slightly more accurate in conveying the meaning of the original sentences, but especially the translations of long rare expressions (e.g., multi-word names of institutions) are less fluent. This hints that the parameters for (rare) biphrases may have been regularized too heavily — it looks like Sinuhe is underfitting rather than overfitting. We will conduct more experiments to see how much the translation quality can be improved by a better choice of α or by using a different prior for regularization. Of course, there is room for tuning elsewhere, too. For example, it would be a surprise if the phrase extraction pipeline that has been optimized for Moses would be optimal for Sinuhe.

5 Conclusions

In this paper, we have shown that phrase-based SMT can be viewed as an instance of structural prediction. The word alignment and phrase extraction heuristics serve as a strategy for feature extraction, and the translation task can be modelled as a structural prediction problems over these features. Our methods scale to large corpora and are fast at predicting translations. While speed is not the primary goal, the faster translation times may be a key to success in applications where the amount of text that needs to be translated is large. In terms of BLEU scores, the results do not improve the state-of-the-art so far. However, fine-tuning the standard phrase-based approach over the years has increased its performance significantly, and we see no reason why the same would not happen with the proposed approach, especially if the model is augmented with additional features like gapped biphrases and biphrases over POS tags.

The fact that our translation model is a properly normalized conditional probability distribution opens up many new possibilities. For instance, instead of predicting translations, it is possible to efficiently compute the expected number of times each word would appear in them. Such output might be useful, e.g., if the translations are to be post-processed by models relying on bag-of-words representation. Another research direction we are currently looking into is training the proposed translation model in the reverse direction, and then predicting translations using the noisy channel approach, i.e., by maximizing $P(x|y)P(y)$. The key difference to previous work here is that since $P(x|y)$ is properly normalized, the noisy channel approach would not in our case suffer from the potentially negative effects caused by ignoring the normalizer that depends on y. Besides being a viable (though computationally demanding) alternative criterion for predicting translations, the noisy channel approach could easily be used for, e.g., reranking n-best lists and for system combination.

Acknowledgments

This work has been partially funded by the SMART EU project. We wish to thank the anonymous reviewers for their constructive comments and especially for pointing out the related paper (Blunsom and Osborne, 2008) that we had missed. Special thanks to Vladimir Poroshin for all the bugs he found in beta-testing and to Esther Galbrun for her comments on a draft version of this paper.

References

Phil Blunsom and Miles Osborne. 2008. Probabilistic inference for machine translation. In *EMNLP*.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *ACL*.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Josh Schroeder, and Cameron Shaw Fordyce. 2008. ACL 2008 third workshop on statistical machine translation. `http://www.statmt.org/wmt08`.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. EACL 2009 fourth workshop on statistical machine translation. `http://www.statmt.org/wmt09`.

Stanley Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for ME models. *IEEE Transactions Speech and Audio Processing*, 8(1):37–50.

John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *ACL*.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *Workshop on SMT at NAACL*.

John DeNero, Alex Bouchard, and Dan Klein. 2008. Sampling alignment structure under a bayesian translation model. In *EMNLP*.

Matti Kääriäinen. 2009. Sinuhe source code distribution (v1.2). Website. `http://www.cs.helsinki.fi/u/mtkaaria/sinuhe`.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, and Chris Callison-Burch et al. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Philipp Koehn. 2009. BLEU/NIST scores for submissions. `http://groups.google.com/group/WMT09/browse_thread/thread/bfbce7b219648a4c`.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Percy Liang, Alexandre Bouchard-Cote, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *ACL*.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *EMNLP*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:2003.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *EMNLP*, pages 20–28.

Cristoph Tillmann and Tong Zhang. 2007. A block bigram prediction model for statistical machine translation. *ACM Transacions on Speech and Language Processing*, 4(3).

S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. 2006. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML*.