Machine Translation Using Overlapping Alignments and SampleRank

Benjamin Roth*
Spoken Language Systems
Saarland University
66123 Saarbrücken, Germany
beroth@coli.uni-saarland.de

Andrew McCallum
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
mccallum@cs.umass.edu

Marc Dymetman and **Nicola Cancedda**
Xerox Research Centre Europe
38240 Meylan, France
{marc.dymetman,nicola.cancedda}@xrce.xerox.com

Abstract

We present a conditional-random-field approach to discriminatively-trained phrase-based machine translation in which training and decoding are both cast in a sampling framework and are implemented uniformly in a new probabilistic programming language for factor graphs. In traditional phrase-based translation, decoding infers both a "Viterbi" alignment and the target sentence. In contrast, in our approach, a rich overlapping-phrase alignment is produced by a fast deterministic method, while probabilistic decoding infers only the target sentence, which is then able to leverage arbitrary features of the entire source sentence, target sentence and alignment. By using SampleRank for learning we could in principle efficiently estimate hundreds of thousands of parameters. Test-time decoding is done by MCMC sampling with annealing. To demonstrate the potential of our approach we show preliminary experiments leveraging alignments that may contain overlapping bi-phrases.

1 Introduction

Traditional approaches to phrase-based machine translation use dynamic programming to search for the derivation (or phrase alignment) that gets the maximum probability (or score) given the source sentence. The following setting is standardly assumed: the source sentence is partitioned into spans of words, each span covered by a translation phrase (biphrase), which are then reordered. While this method performs well in practice, three potential downsides are associated with it: First, finding the optimal partitioning of the source sentence and selection of translation phrases can be done efficiently only if the global score is composed of several local scores. Corresponding short-range Markov assumptions may be too limited to capture all dependencies of interest. Second, unrestricted reordering makes decoding NP-hard, and search has to be approximated by beam-search techniques, which work by generating the target sentence left-to-right, and have difficulty recovering from wrong decisions taken on early prefixes. Third, maximizing the joint probability of translation and an auxiliary (hidden) phrase alignment ("Viterbi decoding") is clearly not an intuitive or easily motivated objective when rather just the translation itself is of interest.

While in current models these restrictions may not harm performance seriously if associated with strong heuristic guidance for the beam-search, it is not clear that they are still appropriate for more expressive statistical models. Therefore alternative frameworks that allow a richer representation of translation pairs and a more flexible search are a worthwhile object of study.

In this paper we show how a translation model that does not rely on a rigid partition in non-overlapping phrases can be trained and used for inference in a sampling setup, leveraging the generic facilities offered by the recent probabilistic programming language FACTORIE (McCallum et al., 2009). The model is trained with SampleRank (Wick et al., 2009) and can in principle incorporate arbitrary features of the translation pair. De-

*This work was conducted during an internship at XRCE.

coding is done by a random walk with a sharpened (annealed) version of the learned model distribution. Our work differs from previous approaches in that SampleRank is used as the training method and in that the features and the proposal steps depend on an overlapping alignment which is deterministically associated with every source-target sentence pair.

2 Related Work

Arun et. al (2009) use operations in a non-overlapping phrase-based setting and traditional features to obtain random samples of alignments, where translation samples are obtained by a form of marginalization that corresponds to "forgetting" the alignment and only outputting the target string. In contrast to our work, their approach is strongly connected to the traditional phrase-based setting, and is concerned with the question of whether sampling and marginalization can reach equally good translations as dynamic programming for the same type of model.

Kääriäinen (2009) has developed a system for phrase-based translations using overlapping biphrases, which allows decoding to use a representation which is consistent with that used when heuristically extracting the biphrases from bilingual data, contrary to what is the case with standard phrase-based systems. Each biphrase is a feature, and biphrase parameters are estimated on the basis of maximizing the likelihood of a large bilingual corpus relative to a simplified translation model that does not incorporate a language model or distortion. Decoding is done through dynamic programming, either in the simplified mode not including a language model, or in a full mode including one, in which case a heuristic beam-search is used. While we also employ overlapping biphrases here, we perform decoding and training using a sampling approach on a set of features including a language model and a distortion model.

Sampling has also been used in the synchronous grammar paradigm (hierarchical models), for the training of synchronous grammars when Bayesian priors are given (Blunsom et al., 2009) and for estimating the partition function for a model using a synchronous grammar intersected with a language model (Blunsom and Osborne, 2008).

initialize each source sentence with gloss and alignment
if *training* **then** initialize model
if *decoding* **then** load model
for *num iterations* **do**
  foreach *sentence pair* **do**
    generate translation neighbors
    compute alignments for neighbors
    score neighbors
    if *training* **then**
      update parameters w.r.t. objective function (BLEU) using *SampleRank*
    end
    sample next translation from neighbors (proportional to score)
  end
end

Algorithm 1: Training and decoding

3 Sampling For Training and Decoding

3.1 Overview

Algorithm 1 gives an overview of the procedure used for training and decoding. The flow for training and decoding differs only in two places, which concern loading and updating the model parameters; otherwise exactly the same code can be used.

The algorithm starts by initializing each source sentence with a gloss (choosing the most likely translation per source word). If the task is decoding, a previously trained model is loaded, while for training the model parameters are initialized. Starting from the glosses, an iterative Markov-Chain Monte Carlo (MCMC) sampling procedure is performed according to the following scheme: For each source sentence x a set of translations y (*neighbors*) is generated by changing the current translation according to *neighborhood operators*. For each of the pairs (x, y), we first compute deterministically its alignment, then compute features of (x, y) using the value of this alignment and then score the result according to the current model.

Since we are using the FACTORIE toolkit (McCallum et al., 2009) as an implementation platform and its embedded SampleRank training algorithm, we actually model this as a very simple factor graph, in which each factor directly connects the nodes x

and y; however, this could easily be extended to a more sophisticated graphical model. In training, the model parameters (feature weights) are updated if the model's ranking of two neighbors disagrees with the ranking given by the objective function. Our objective function is the BLEU score of the current model translations relative to the training corpus[1]. Finally, the next translation is chosen by randomly sampling one of the neighbors with probability proportional to the score it was assigned by the model.

3.2 SampleRank

A translated text is scored according to a "log-linear" model of the form $\exp\left(\sum_k \Theta_k \phi_k(x, y)\right)$, where the $\phi_k(\cdot, \cdot)$ are feature functions and Θ is a vector of feature weights. We write x for the ("observed") source sentence and y for the ("unobserved") translation. The normalized score gives an estimate $P(y|x, \Theta)$ of the probability of the unobserved variable given the observed variable.

In SampleRank (Wick et al., 2009; Rohanimanesh et al., 2009), the goal is to learn parameters Θ so that the model agrees with the objective function in ranking configurations. Candidates are taken from the neighborhood, top-scoring candidates are compared. In case of disagreement between objective function and model in ranking two candidates, a perceptron-style update of model parameters at iteration t is performed:

$$\Theta^{t+1} \leftarrow \Theta^t + \eta(\phi(x, \hat{y}') - \phi(x, y')),$$

where $\hat{y}'$ is the candidate preferred over y' by the objective function, and where η is a learning rate that is set differently in different variants of the method.

The translation for the next iteration is sampled according to a transition distribution $Q(y'|y)$. In our case, $Q(y'|y)$ is zero for translations not in the neighborhood of y, and proportional to the current learnt model probability $P(y'|x, \Theta)$ otherwise (normalized by the sum of all neighbor probabilities).

In preliminary experiments we tried several of the possibilities provided by FACTORIE for setting the learning rate (averaged perceptron (Collins, 2002), MIRA (Crammer and Singer, 2003) and confidence weighted updates (Dredze et al., 2008)) and found that the averaged perceptron, which amounts to setting the learning rate to 1.0 and averaging the parameters over all iterations, works best in practice. This might be related to the fact that in our current experiments there are only around a dozen features: if there were many sparse features (e.g. one binary feature per biphrase) we would expect a confidence weighting scheme to perform better.

[1]Note that, while the neighborhood is local (per-sentence), the objective score is global (corpus-wide). This corresponds to a factor model of the corpus with shared parameters for sentence factors.

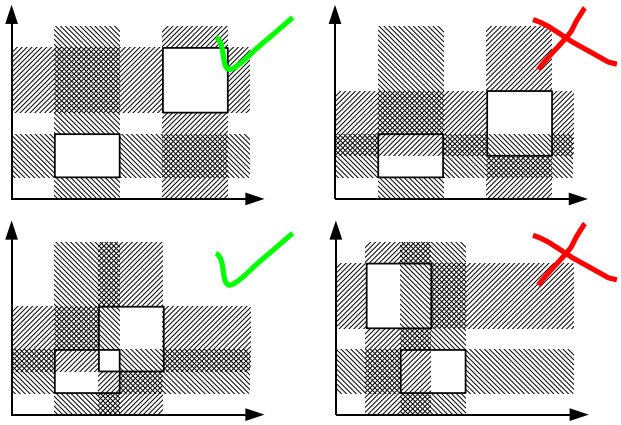

Figure 1: Consistent (left) and inconsistent (right) pairs of biphrases. The x-axis (y-axis) denotes positions in the source (target) sentence. The squares indicate the spans of words covered on either side by the biphrases.

3.3 Phrase Alignment

We call a phrase alignment a set of biphrases that express correspondences between spans of words in source and target sentences. In our case a phrase alignment is used for two purposes: first, to compute some features of a translation and secondly, for constructing part of the proposal neighborhood (proposed changes to current translation). We employ a greedy phrase alignment algorithm of source and target sentences similar to *Competitive Linking* (Melamed, 2000): first, biphrases[2] that match on both source and target side are ordered by a heuristic weight (which does not depend on the model Θ). Then, biphrases are added to the alignment set, in descending order of their weight, if they are consistent with the current alignment set.

Our notion of consistency allows for overlapping

[2]We use a biphrase table extracted by the heuristics in a standard run of the Moses pipeline.

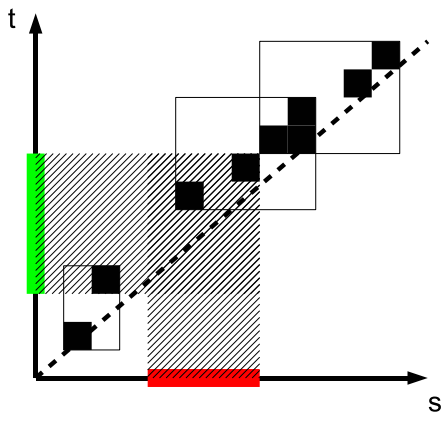

Figure 2: Finding a target span (on the y-axis, marked) for a source span (x-axis, marked). Here, the rightmost point of the source span is mapped using the internal word-alignment, the leftmost by choosing the point at the corresponding relative position on target side.

biphrases: Two (alignment) biphrases are *consistent* if the *matching criterion* is the same for source and target side, otherwise they are *inconsistent*. Existence of (partial) *overlap* is used as the matching criterion. A biphrase is consistent with an alignment set, if it is consistent with all biphrases in this set, see Figure 1 for an example.

For weighting the biphrases we resort to a heuristic similar to the conditional link probability scoring in (Moore, 2005). However, we use weights that can be read off directly from the phrase-table as used by the popular Moses (Koehn et al., 2007) pipeline: the phrasal and lexicalized biphrase probabilities $P(f|e)$ and $P(e|f)$ are multiplied instead of using $P(e, f)$. Additionally we normalize the biphrase weights for length (l_f and l_e), using the geometric mean, in order not to penalize longer biphrases. This has also the effect of increased contiguity of alignments. The resulting biphrase weight is:

$$\sqrt[l_f]{P_{lex}(e|f)P_{phr}(e|f)}\ \sqrt[l_e]{P_{lex}(f|e)P_{phr}(f|e)}$$

3.4 Neighborhood Operators

At present, four operators are used to generate a neighborhood of the current translation:

1. **Remove:** For each target position the word at this position is removed.

2. **Insert:** For each target position, if the trigram at this position is not present in the language model, a random word is inserted according to the trigram language model.

3. **Replace:** A span of words on the target side of the sentence pair is replaced by the following mechanism: First, a biphrase is sampled randomly out of all biphrases that can match somewhere on the source side of the sentence pair (in general, such a biphrase does not match on the target side). Then the first and last position of the match on the source side are mapped onto positions of the target side, following the internal word alignment of the highest weighted biphrases from the previous alignment. If no word alignment is available at these positions, a mapping is done along the sentence diagonal (see Figure 2). The span of the target sentence identified by this mapping is replaced by the target side of the sampled biphrase.

   The number of neighbors added by the replace operator is set to the number of words in the source sentence.

4. **Shift:** The target side of a random biphrase in the alignment is shifted to all possible positions.

In total, this amounts to about $4 \times$ source-length operations per iteration. All operators are only concerned with changing the target side and do not directly change the phrase alignment but rather trigger changes to it, since the alignment follows deterministically from a sentence pair.

3.5 Features

All current features are most naturally directly expressed as features $\phi(x, y)$ about a sentence pair, since all other sources of information (e.g. alignments) are deterministically computed from this configuration. The features group into length features, a trigram language model feature, a distortion feature, biphrase and alignment features.

There are two length features: first the bare target length in order to counteract a potential length bias of other features. Second, the relative length difference between target and source in order to control

source:	furthermore , there are six basic aspects which worry us because of the method employed .
gloss:	en outre , il y sont six de base aspects qui vous inquiétez nous parce que de la méthode employées .
sampling:	en outre , il y a six les aspects ∅ qui inquiètent nous à cause de la méthode ∅.
moses:	en outre , il y a six aspects fondamentaux qui nous inquiète parce que la méthode utilisée.
reference:	en outre , six autres questions de fond nous préoccupent à cause de la méthode suivie .

Figure 3: Example translations by initializing with maximal per-word phrase translation, the sampling model and Moses. Obvious errors are underlined, omissions indicated by ∅.

length variation between the two languages. Formally:

$$\left(1 - \frac{l_f}{l_e}\right)^2,$$

where l_f and l_e are the lengths of source and target sentence respectively.

As the language model feature we include the mean of the trigram log-probabilities of the translation (that is, sum of these log-probs normalized by the length of the target). The distortion feature is computed as follows: For each alignment biphrase the actual positions of start and end point on the target side is compared to the projection of these points from the source side along the sentence diagonal. The distortion feature is the average distance between this projection and the actual position.

Several features are computed from the inferred alignment. They are

- the number of biphrases in the alignment,
- the sum of alignment biphrase probabilities (one feature for each direction, lexicalized and unlexicalized),
- the sum of alignment biphrase weights and
- the number of unaligned source (target) words.

3.6 Implementation in FACTORIE

Our model uses the FACTORIE (McCallum et al., 2009) toolkit, a library implemented in the Scala programming language[3] for performing learning and inference with arbitrary factor graphs. With FACTORIE, features need not be stored or represented in a particular form (e.g. expanded as vectors), but can be extracted at run time as necessary. Changes to a variable-value configuration (a translation corresponds to an assignment to unobserved variables) are represented efficiently by differences (`DiffList`'s) to the preceding configuration. This has the advantage that for sampling several alternatives can be evaluated and scored with little overhead. In learning, only weights of factors that are affected by a particular change need to be updated.

In FACTORIE, a `Variable` stores everything that changes and is used for features. In the translation model there are Variables for the target sentence and the source-target sentence pair. Any change to a Variable must be reflected by a `Diff`. A Diff is an object that implements `do` and `undo` methods for efficiently controlling the change. In addition, it stores which Variable is affected. Diff's add themselves to a DiffList which keeps track of consecutive changes. For example, removing a word from a target sentence adds `Diffs` that capture

1. **removal** of the target token (`Diff` for `TokenVariable`),
2. **addition** (**removal**) of biphrases that become (cease to be) part of the *cover*[4],
3. updating of alignment to account for modified target sentence, **addition** and **removal** of biphrases w.r.t. new and old alignment.

`Template`'s are defined between types of `Variable`'s. When a variable of the respective type is changed, they gather statistics (feature values) and return the involved factors of the factor graph. Finally, a `Model` is a set of `Template`'s. Once all the components are defined, training the machine translation model can be performed in just a few lines, e.g.

[3] `www.scala-lang.org`

[4] The cover is the set of all biphrases from the phrase table that match on source and target side.

Table 1: Evaluation results on the WMT08 subset.

Method	BLEU	NIST
gloss	0.1039	4.5052
sampling	0.2213	5.6124
Moses	0.3023	6.3983

- Initialize the sampler:
  `new TranslationRandomSampler(model, objective) with SampleRank`
- Iteratively process sentence pairs:
  `sampler.process( pair )`

4 Experiments

The WMT08-Europarl training set is used for the heuristic extraction of biphrases, the corresponding development set (2000 sentences) for estimating the feature weights and 250 sentences of the test set for evaluation. The model is trained using SampleRank with averaged perceptron, the training sampling is run for 1000 iterations (corresponding to about 100K proposals per sentence).

Decoding with trained model is done in an annealing setup, i.e. the model probability $P(y|x)$ used for sampling the next successive translation is taken to an exponent resulting in $P(y|x)^{\frac{1}{\alpha}}$, with the temperature α decreasing over time according to a cooling factor. We set the initial temperature to 100 and the cooling factor to .9. We observe that after about 50 iterations (5K proposals), the objective score oscillates around the same value and the algorithm can be stopped. Evaluation is done on the lowercased and tokenized outputs of three methods: the gloss the translation was initialized with, the result of the sampling method and output of a standardly trained Moses (Koehn et al., 2007) model.

Table 1 shows that the sampling method can achieve a good improvement over the baseline with a gloss initialization but does not yet reach the performance of a mature machine translation system such as Moses. Figure 3 gives an example of translations produced by each method. In contrast to Moses, here, the sampling model produces omissions of words that are actually contained in the phrase-table and also usually translated. This is related to the fact that we do not hard-constrain our overlapping alignments to cover all source words (as is implicitly enforced with the alignments used by Moses), which is sometimes a good thing — as when some function words are better omitted in translation — but can be detrimental with content words. Currently we do not have different factors accounting for the two types of omissions, but this could be easily added to the model.

5 Future Work

We plan to explore several directions in order to improve model performance: First, the neighborhood operators could be improved by focusing on such operators that empirically yield more promising neighbors. Especially an insertion mechanism that is more connected to the source side might be an interesting option to try. Second, we do not yet make use of the ability of the SampleRank algorithm to easily learn many thousands of feature weights. We plan to include more fine-grained features possibly even to the degree that every biphrase is regarded as a feature itself. Such an approach could also demonstrate FACTORIE's ease in handling a more sophisticated graphical model structure.

6 Conclusion

We outlined how a machine translation model can be implemented from scratch in a just a few hundred lines of code that deviates in many ways from the standard phrase-based approach popularized by Moses. Although we did not reach state-of-the-art performance, we showed how to implement a translation system that allows great freedom with respect to feature design and search operations. Both training as well as decoding are included in our approach. In the current version, the use of overlapping alignment emphasizes this gained freedom in modeling. However, more exotic features, long range dependencies and settings with many more weights to be estimated can be incorporated into such a model without changing its basic principles. Models of this kind may therefore become a valuable tool for experimenting with novel setups that do not fit in any of the traditional settings.

References

A. Arun, C. Dyer, B. Haddow, P. Blunsom, A. Lopez, and P. Koehn. 2009. Monte Carlo inference and maximization for phrase-based translation. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 102–110. Association for Computational Linguistics.

P. Blunsom and M. Osborne. 2008. Probabilistic inference for machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 215–223. Association for Computational Linguistics.

P. Blunsom, T. Cohn, C. Dyer, and M. Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 782–790. Association for Computational Linguistics.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, page 8. Association for Computational Linguistics.

K. Crammer and Y. Singer. 2003. Ultraconservative Online Algorithms for Multiclass Problems. *Journal of Machine Learning Research*, 3:951–991.

M. Dredze, K. Crammer, and F. Pereira. 2008. Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning*, pages 264–271. ACM.

Matti Kääriäinen. 2009. Sinuhe – statistical machine translation using a globally trained conditional exponential family translation model. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1027–1036, Singapore, August. Association for Computational Linguistics.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual meeting-Association for Computational Linguistics*.

A. McCallum, K. Schultz, and S. Singh. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems Conference (NIPS)*.

I.D. Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

R.C. Moore. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, page 88. Association for Computational Linguistics.

K. Rohanimanesh, M. Wick, and A. McCallum. 2009. Inference and learning in large factor graphs with adaptive proposal distributions. Technical report, Technical Report #UM-CS-2009-028, University of Massachusetts, Amherst.

M. Wick, K. Rohanimanesh, A. Culotta, and A. McCallum. 2009. SampleRank: Learning Preferences from Atomic Gradients. *NIPS WS on Advances in Ranking*.