Forest Reranking for Machine Translation with the Perceptron Algorithm

Zhifei Li and **Sanjeev Khudanpur**
Center for Language and Speech Processing and Department of Computer Science
Johns Hopkins University, Baltimore, MD 21218, USA
zhifei.work@gmail.com and khudanpur@jhu.edu

Abstract

We present a *scalable* discriminative training framework for parsing-based statistical machine translation. Our framework exploits *hypergraphs* (or packed forests) to compactly encode exponentially many competing translations, and uses the perceptron algorithm to learn to discriminatively prefer the *oracle*-best tree in the hypergraph. To facilitate training, we present: (i) an *oracle extraction* algorithm to efficiently extract the oracle trees from a hypergraph that best match the reference translations; (ii) a *hypergraph pruning* algorithm that substantially reduces the disk space required for storing the hypergraphs without degrading the translation quality; and (iii) simple yet effective *data* and *feature selection* algorithms by which an equally good or better model is obtained using a fraction of the training data and features. We experimentally show that our approach is scalable and is able to improve over a full-scale state-of-the-art hierarchical machine translation system.

1 Introduction

Discriminative training has been attempted for the task of statistical machine translation (SMT) both in a *small-scale* setting (i.e, finding optimal weights among a small set of generative models) and in a *large-scale* setting (i.e., training optimal weights for thousands/millions of features). In this paper, we focus on the *large-scale* discriminative training.

In the *small-scale* setting, minimum error rate training (Och et al., 2003) has become the de facto standard in SMT systems. Smith and Eisner (2006) propose an annealed minimum risk approach, while Zens et al. (2007) give a systematic experimental comparison for different training criteria. Shen et al. (2004) use perceptron-inspired algorithms to tune weights for tens of features. Chiang et al. (2008) use an online max-margin method to tune tens of syntax features for a hierarchial system.

Large-scale discriminative training for SMT is nontrivial due to several reasons. Learning a discriminative model normally involves in running an *iterative* training algorithm, which may require decoding the training data at each iteration. The decoding process is computationally expensive. In particular, the decoding of a single sentence often takes several CPU-seconds, and the parallel corpora available for discriminative training typically contain millions of sentence-pairs. Therefore, a single decoding pass over the training data may take tens of CPU-days. Another reason that makes discriminative training nontrivial for a SMT task is that the number of features needed to improve MT performance is enormous. For example, the number of phrase pairs extractable from the training bitext alone runs into the tens of millions.

To address these problems, previous approaches have resorted either to an n-best approximation (e.g., Watanabe et al. (2007)), to a computationally cheap baseline system (e.g., a monotone translation system as assumed in Liang et al. (2006)), or to a small-scale setup (e.g., only sentences less than fifteen words are used as in Blunsom et al. (2008)).

In this paper, we present a *scalable* discriminative reranking framework which discriminatively *reranks* hypotheses on a *hypergraph*, instead on an n-best list.Specifically, for each sentence, we generate a hypergraph using a baseline SMT system and save it to disk. In each iteration of the discriminative training, the learning algorithm updates towards an *oracle tree* in the *fixed* hypergraph. The **reranking** approach has the advantage of being simple and scalable (since it does not require re-decoding the training data at each iteration of training).

Our hypergraph-based discriminative reranking approach is particularly appealing since a hypergraph compactly encodes exponentially many hypotheses, representing a much larger hypothesis space than an n-best. In this respect, the hypergraph-based reranking occupies an intermediate position between reranking on a fixed n-best and redecoding the training data at each iteration. To support the hypergraph-based discriminative training for SMT, we use an *oracle extraction* algorithm[1] that is able to efficiently extract oracle trees from hypergraphs, and a *hypergraph pruning* algorithm that substantially reduces the required disk space for saving hypergraphs without degrading the hypergraph quality. The scalability of our approach is also due to our simple yet effective *data selection*[2] and *feature selection* algorithms, by which an equally good or better model is obtained using a fraction of the training data and features. Our hypergraph-based discriminative reranking approach is analogous to a lattice-based discriminative training approach for speech recognition (Woodland and Povey, 2002) and a forest reranking approach to monolingual parsing (Huang, 2008), both of which have been shown to be quite effective for their respective tasks.

We report experimental results for both the n-best and hypergraph-based discriminative reranking on Hiero (Chiang, 2007), showing that our approach is able to improve over a full-scale state-of-the-art hierarchical machine translation system.

2 Hiero and Hypergraphs

In Hiero (Chiang, 2007), a synchronous context-free grammar (SCFG) is extracted from automatically word-aligned corpora. An illustrative grammar rule for Chinese-to-English translation is

$$X \;\rightarrow\; \langle\, X_0 \text{ 的 } X_1\,,\; X_1 \text{ of } X_0\,\rangle\,,$$

where the Chinese word 的 means *of*, and the alignment, encoded via subscripts on the nonterminals, causes the two phrases around 的 to be reordered around *of* in the translation. Given a source sentence, Hiero uses a CKY parser to generate a hypergraph, encoding many hypotheses (derivation trees along with the translation strings).

[1]The details of the oracle extraction algorithm have been presented in Li and Khudanpur (2009).

[2]The details of the data selection method have been discussed in Li and Khudanpur (2008).

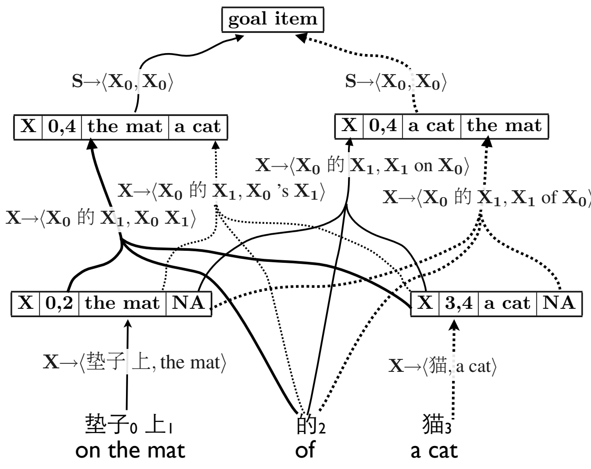

Figure 1: A toy hypergraph in Hiero. When generating the hypergraph, a trigram language model is integrated. Rectangles represent items, where each item is identified by the non-terminal symbol, source span, and left- and right-side language model states. An item has one or more incoming hyperedges. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

2.1 Hypergraphs

Formally, a hypergraph is a pair $\langle V, E\rangle$, where V is a set of *nodes* (vertices) and E is a set of *hyperedges*, with each hyperedge connecting a *set* of *antecedent nodes* to a single *consequent node*.In parsing parlance, a node corresponds to an *item* in the chart (which specifies aligned spans of input and output together with a nonterminal label). The root node corresponds to the *goal item*. A hyperedge represents an SCFG rule that has been "instantiated" at a particular position, so that the nonterminals on the right and left sides have been replaced by particular antecedent and consequent items; this corresponds to storage of backpointers in the chart.

We write $T(e)$ to denote the set of antecedent nodes of a hyperedge e. We write $I(v)$ for the set of *incoming hyperedges* of node v (i.e., hyperedges of which v is the consequent), which represent different ways of deriving v. Figure 1 shows a simple Hiero-style hypergraph. The hypergraph encodes four different derivation trees that share some of the same items. By exploiting this sharing, a hypergraph can compactly represent exponentially many trees.

3 Discriminative Modeling

We first describe a general discriminative reranking framework, inspired by Roark et al. (2007) for n-

gram language modeling in speech recognition.

3.1 Global Linear Models

A linear discriminant aims to learn a mapping from an input $x \in X$ to an output $y \in Y$, given

1. training examples (x^i, y^i), $i = 1 \cdots N$,
2. a representation $\Phi : X \times Y \rightarrow R^d$ mapping each possible (x, y) to a feature vector,
3. a function $\mathrm{GEN}(x) \subseteq Y$ that enumerates putative labels for each $x \in X$, and
4. a vector $\alpha \in R^d$ of free parameters.

In general, for a SMT task, x is a sentence in the source language, $\mathrm{GEN}(x)$ enumerates possible translations of x into the target language, and can be the set of strings in an n-best list or hypergraph. y is the *desired* translation: either a *reference* translation produced by a bilingual human or a so-called *oracle* translation in $\mathrm{GEN}(x)$ that is most similar to such a human-reference. In our training, y is an *oracle tree* (which contains a translation *string* as well) since we use the perceptron algorithm, which, unlike the conditional random field used by Blunsom et al. (2008), cannot easily treat the derivation trees as latent variables.

Given an input x, the model assigns every hypothesis $y \in \mathrm{GEN}(x)$ a score,

$$s(x, y) = \Phi(x, y) \cdot \alpha = \sum_j \Phi_j(x, y)\alpha_j\,, \qquad (1)$$

where j indexes the feature dimensions.

The **learning** task is to obtain the "optimal" parameter vector α from training examples, while the **decoding** task is to search over $\mathrm{GEN}(x)$ to obtain the y that has the maximum score $s(x, y)$. These tasks are discussed next in Section 3.2 and Section 3.3, respectively.

3.2 Parameter Estimation

Given a set of training examples, a parameter estimation algorithm is used to find an optimal α by maximizing a certain objective function of the training data. Different algorithms use different objective functions, e.g., maximum conditional likelihood (Blunsom et al., 2008), minimum risk (Li and Eisner, 2009), or max-margin (Chiang et al., 2008). We use the *averaged perceptron* algorithm (Collins, 2002) due to its simplicity and suitability to large data settings. Given a set of training examples, the algorithm *sequentially* iterates over the examples, and adjust the parameter vector α, as illustrated in Figure 2. After iterating over the training data a few times, an *averaged* model, $\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{i=1}^{N}\alpha_t^i$, is computed and is used for testing, where α_t^i represents the parameter vector after seeing the i-th example in the t-th iteration, N represents the size of the training set, and T is the number of iterations of the perceptron algorithm.

```
**Perceptron**$(x, \mathrm{GEN}(x), y)$
1  $\alpha \leftarrow \vec{0}$        $\triangleright$ initialize as zero vector
2  for $t \leftarrow 1$ to $T$
3      for $i \leftarrow 1$ to $N$
4          $z^i \leftarrow \arg\max_{z \in \mathrm{GEN}(x^i)} \Phi(x^i, z) \cdot \alpha$
5          if $(z^i \neq y^i)$
6              $\alpha \leftarrow \alpha + \Phi(x^i, y^i) - \Phi(x^i, z^i)$
7  return $\alpha$
```

Figure 2: The Basic Perceptron Algorithm

3.3 Decoding/Inference

During the perceptron training-time (see line-4 in Figure 2) as well as the test-time, the following **decision rule** is normally used to select the optimal output y^*,

$$y^* = \arg\max_{y \,\in\, \mathrm{GEN}(x)} s(x, y), \qquad (2)$$

This is called Viterbi decoding. Other decision rules like minimum risk (Tromble et al., 2008) or variational decoding (Li et al., 2009b) can also be used. The decoding complexity depends on the *size* and *structure* of $\mathrm{GEN}(x)$ (e.g., n-best or hypergraph).

4 Discriminative Forest Reranking

4.1 Discriminative Reranking

The above framework (models, parameter estimation, and decoding) is quite general and can be applied in many structured prediction tasks. We use it as a **reranking** framework: train a discriminative model to rerank the hypotheses (encoded either in an n-best list or a hypergraph) produced by a baseline SMT system. The reranking approach has the advantage of being simple and scalable (as we do not need to re-decode the training data following each iteration of training).

Each component $\Phi_j(x, y)$ of the feature vector can be any function of the input x and the output y. To facilitate reranking, we first define a *baseline feature* $\Phi_0(x, y)$, which is the score assigned to y by the baseline SMT system.[3] We then need to define many additional *reranking features*. For example, an n-gram feature might be:

$$\Phi_1(x, y) = \text{Count of the bigram "}\texttt{the of}\text{" in } y.$$

Given the feature- and the parameter-vectors, the total score assigned to an output $y \in \mathrm{GEN}(x)$ for a given input x is

$$s(x, y) = \beta\Phi_0(x, y) + \sum_{j\in[1,F]} \alpha_j \Phi_j(x, y)\,, \qquad (3)$$

where β is the weight for the baseline feature and F is the number of discriminative reranking features. To find the optimal weight β for the baseline feature, one could simply treat Φ_0 as a feature in the discriminative reranking model and set the value of β via the perceptron algorithm. This, however, may lead to *under-training* (Sutton et al., 2006) of the discriminative reranking features in the reranking model: the baseline feature is strongly indicative of the *overall* goodness of y for x, relative to any single discriminative reranking feature which indicates the *local* goodness of y. Therefore, we use a fixed value for β during the discriminative training, as suggested by Roark et al. (2007).

In the general framework described so far, two algorithms are specific to the size and structure of $\mathrm{GEN}(x)$: *decoding* and *oracle extraction*. When $\mathrm{GEN}(x)$ is an n-best list as in Li and Khudanpur (2008), these two algorithms are straightforward as we can simply perform a brute-force linear search over the n-best list to obtain the Viterbi or oracle translation. It will be more complex when $\mathrm{GEN}(x)$ is a hypergraph, which encodes exponentially many hypothesis. We will present the decoding algorithm in Section 4.2, and review the oracle extraction algorithm (Li and Khudanpur, 2009) in Section 4.3.

[3]This score itself is often a linear combination of several models, with the relative weights among these models obtained via a minimum error rate training procedure (Och et al., 2003).

```
Decoding-on-Hypergraph(GEN(x))
1 for $v$ in topological order    ▷ each node
2     $\hat{s}(v) \leftarrow$ negative infinity
3     for $e \in I(v)$    ▷ each incoming hyperedge
4         $s(e) \leftarrow \beta\Phi_0(e) + \sum_{j\in[1,F]} \alpha_j\Phi_j(e)$
5         $s(e) \leftarrow s(e) + \sum_{u\in T(e)} \hat{s}(u)$
6         if $s(e) > \hat{s}(v)$    ▷ better derivation?
7             $\hat{d}(v) \leftarrow \langle e, 1\rangle$
8             $\hat{s}(v) \leftarrow s(e)$
9 return $\hat{d}$(goal)
```

Figure 3: Discriminative reranking on a hypergraph $\mathrm{GEN}(x)$ of a source sentence x. $\hat{d}(v)$ and $\hat{s}(v)$ are the Viterbi derivation and Viterbi score for a node v, respectively.

4.2 Decoding on Hypergraphs

The decoding algorithm that finds y^* on a hypergraph, as defined in (2), is outlined in Figure 3. Recall that the total score assigned to an output $y \in \mathrm{GEN}(x)$ for a given input x is defined in (3). For each node v, the algorithm adjusts its Viterbi derivation $\hat{d}(v)$ and Viterbi score $\hat{s}(v)$ by applying the discriminative model α. To update the Viterbi derivation $\hat{d}(v)$, the algorithm processes each incoming hyperedge $e \in I(v)$. A hyperedge e's score $s(e)$ is the model score on that hyperedge (see line-4) plus the Viterbi scores of all its antecedent nodes $u \in T(e)$ (see line-5). If e leads to a better derivation, $\hat{d}(v)$ is updated with $\langle e, 1\rangle$, the best derivation along the hyperedge e. The algorithmic complexity is linear with the hypergraph size, that is, O($|E|$).

4.3 Oracle Extraction on Hypergraphs

While a hypergraph represents a very large set of translations, it is quite possible that the reference translations are not contained in the hypergraph, due to pruning or inherent deficiency of the translation model. In this case, we want to find the translation in the hypergraph that is most similar to the reference translations, with similarity computed by BLEU (Papineni et al., 2002). Such maximally similar translation will be called *oracle translation*, and the process of extracting them *oracle extraction*. As mentioned, we use oracle *tree* (instead of oracle *string*) in our training. In Hiero, many distinct derivation trees may yield the *oracle string*. This is called *spurious ambiguity*. Among all the trees yielding the *oracle string*, we define the one with the best baseline model score as the *oracle tree*.

Oracle extraction on a hypergraph is a nontrivial task because computing the similarity of any one hypothesis requires information scattered over many items in the hypergraph, and the exponentially large number of hypotheses makes a brute-force linear search intractable. Therefore, efficient algorithms that can exploit the structure of the hypergraph are required. We present an efficient oracle extraction algorithm in Li and Khudanpur (2009), which involves two key ideas. Firstly, we view the oracle extraction as a bottom-up *model scoring* process on a hypergraph, where the *model* is an n-gram model "trained" on the reference translation(s) and the *score* is the BLEU value. This algorithm, however, requires maintaining a separate dynamic programming state for each distinguished sequence of "state" words and the number of such sequences can be huge, making the search very slow. Secondly, therefore, we present a novel look-ahead technique, called *equivalent oracle-state maintenance*, to merge multiple states that are equivalent for similarity computation. Our experiments show that the *equivalent oracle-state maintenance* technique significantly speeds up (more than 40 times) the oracle extraction.

5 Supporting Algorithms for Scalable Discriminative Training

In this section, we present several supporting algorithms in the hypergraph-based discriminative training framework, to make it *scalable* to big training sets and to a large number of features.

5.1 Hypergraph Pruning

Saving the hypergraphs on the disk requires a lot of storage space and also incurs I/O overheads. Therefore, we adopt a hypergraph pruning algorithm, which substantially reduces the required disk space without degrading the hypergraph quality too much. Specifically, we run an *inside-outside* algorithm to compute the Viterbi inside score $\beta(v)$ and the Viterbi outside score $\alpha(v)$ for each node v, and then compute the *merit* $\alpha\beta(e)$ for each hyperedge e. A hyperedge e gets pruned if its merit is worse than the score of the best derivation in the hypergraph by a relative threshold p. A node v gets pruned if all its incoming hyperedges get pruned. We use a development set to find the optimal p in terms of the tradeoff between disk space and hypergraph quality (as measured by oracle BLEU). Our algorithm is similar to that in Huang (2008) for monolingual hypergraphs.

5.2 Feature Selection

The perceptron algorithm (see Figure 2) itself can be thought as a *feature selection* algorithm as it incrementally adds features into the model whenever they are activated. Due to this, however, the model size can grow arbitrary large, which imposes difficulty in our SMT task as the number of possible features is enormous. For example, the number of phrase pairs extractable from the parallel corpora can easily run into tens of millions. On the other hand, one can employ a dedicated feature selection algorithm and constrain the perceptron algorithm such that it updates only on the selected features.

We propose to use only the features that are present in the *test* sets. Specifically, before the discriminative training, we traverse all the hypergraphs in the test sets and collect all the features present in these hypergraphs.[4] This method is extremely simple and computationally cheap. In machine learning terminology, it is called *directed ad hoc inference*, inference that occupies an intermediate position between *inductive-deductive* inference and *transductive* inference (Vapnik, 2006). Intuitively, we can view this as an intermediate learning paradigm between the regular supervised training and the unsupervised clustering (e.g., k-nearest neighbors).

5.3 Data Selection

We can present all the training examples (i.e., hypergraphs together with the corresponding oracle trees) to the discriminative training algorithm. Alternatively, we can carry out a data selection procedure that selects a subset of these examples. Data selection is useful for speeding up the training procedure as most training algorithms require computational time proportional to the size of training data. It also has potential to improve the model quality. We adopted the data selection procedure proposed by Li and Khudanpur (2008).

[4]Of course, this does not peek at the reference translations.

6 Feature Functions

Baseline Model Feature: The baseline feature $\Phi_0(x, y)$ is the score assigned to y by the baseline SMT system, which is a linear combination of several models, with the relative weights among these models obtained via a minimum error rate training procedure (Och et al., 2003).

Language Model Features: The *count* of each n-gram in the English yield of the derivation tree y constitutes a feature.

Translation Model Features: The *count* of each *rule* in the derivation tree y constitutes a feature. The rule can be any flat/hierarchical phrase in Hiero.

Boundary n-gram Features: In Hiero, a *hierarchical* rule combines small translation units into a larger one. The combination creates n-grams that cross the boundary between the terminal words in the hierarchical rule and the LM state words in the antecedent items. Some of these boundary n-grams may never appear in monolingual corpora including the English side of the bitext. Clearly, the boundary n-gram features form a *strict subset* of the general n-gram features defined before. For example, all the n-grams inside a *flat* rule and all the unigrams do not belong to the set of boundary n-grams.

7 Experimental Results

We report results using an open source MT toolkit, called **Joshua** (Li et al., 2009a), which implements Hiero (Chiang, 2007).

7.1 Data Resources

We compile a parallel dataset (about 4M sentence pairs) consisting of corpora distributed by LDC for the NIST Chinese-English MT evaluation, from which we select about 1M sentences pairs (about 28M words in each language) as the parallel training data using a sampling method based on the n-gram matches between training and test sets in the foreign side. The language model training data consists of a 130M words subset of the English Gigaword (LDC2007T07) and the English side of the parallel corpora.

7.2 Baselines and Discriminative Training Data

One could train a single baseline SMT system on the entire parallel corpus, and decode all the source language sentences using this system, for discriminative training. However, this may lead to a significant mismatch during actual test conditions. In particular, the hypergraphs generated on the *training* data will have better translation quality than on *unseen* test data. To avoid this pitfall, we partition the parallel data into 30 disjoint sections, and train *30 baseline SMT systems*. To decode sentences from a particular section, we use a baseline system that excludes that section. We carry out this jack-knife training for both the TM and LM training.

For each baseline system, we use the GIZA toolkit (Och and Ney, 2000), a suffix-array architecture (Lopez, 2007), the SRILM toolkit (Stolcke, 2002), and minimum error rate training (MERT) (Och et al., 2003) to obtain word-alignments, translation models, language models, and the optimal weights for combining these models, respectively. The baseline LM is a *trigram*[5] LM with modified Kneser-Ney smoothing (Chen and Goodman, 1998). The MERT is performed on the NIST MT'03 evaluation data set.

7.3 Goodness of the Oracle Translations

Table 1 reports the BLEU scores of the oracle translations on four NIST evaluation data sets on which we will report performance later. This table sets an upper bound on improvements possible using the discriminative model. In general, the oracle BLEU score in a hypergraph is much better than that in a 500 *unique* n-best strings (corresponding to thousands of distinct derivation trees). This shows that a hypergraph provides a much better basis for reranking than an n-best. Moreover, the oracle BLEU score increases very slowly along with the increase of n for the n-best derivation trees. Huang (2008) observed a similar trend for monolingual parsing.

7.4 Goodness of Hypergraph Pruning

Figure 4 shows the tradeoff between disk-space (in terms of number of hyperedges needed to be stored) and hypergraph quality (in terms of the oracle BLEU score) on a dev set. Clearly, as the threshold p increases, less number of hyperedges are pruned, and the oracle BLEU score decreases less. We choose $p = 7$ for the remaining experiments.

[5] We use a trigram LM because the decoding of the whole training set (about 1M sentences) is too computationally expensive if we use a larger order LM.

Task	**1-best**	**Oracle**	
		500-unique-best	hypergraph
MT'03	35.0	40.7	47.1
MT'04	35.7	44.0	52.8
MT'05	32.6	41.2	51.8
MT'06	28.3	35.1	37.8

Table 1: Oracle and 1-best BLEU Scores (4-gram BLEU with four references) on MT Sets

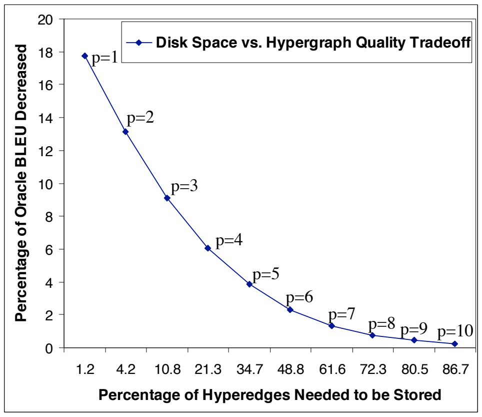

Figure 4: Tradeoff between Disk Space and Hypergraph Quality by varying threshold p.

7.5 Results on NIST MT Benchmark Tests

For the experiments reported here, we perform data selection as described in Li and Khudanpur (2008), and select 610K sentence-pairs from about 1M for training various perceptron models. The optimal β for the baseline feature is found using MT04 as a tuning set.

Table 2 reports the BLEU results for one baseline system, two discriminative *n-best* reranking systems, and five discriminative *hypergraph* reranking systems.

In n-best reranking, we consider only the regular language model features.[6] Clearly, the n-best discriminative training is able to improve the baseline significantly. Also, the "WFS" (with feature selection) model[7], containing about 160K features, achieves comparable performance to "NFS" (no feature selection) that contains about 1444K features.

[6]We consider only unigram and bigram for language model features. Adding trigram features does not lead to much improvement. Roark et al. (2007) observed a similar trend for a speech recognition task

[7]Though the feature selection algorithm in Section 5.2 is originally proposed to make the hypergraph-based training feasible, we use it in the n-best case as well, for comparison.

System		MT04	MT05	MT06
Baseline	Joshua	35.7	32.6	28.3
N-best	NFS	**36.6**	33.2	**29.3**
	WFS	36.5	**33.5**	29.2
Hyper-graph	LM	35.9	33.0	28.2
	BLM	35.9	33.0	28.4
	TM	**36.1**	**33.2**	**28.7**
	TM+LM	36.0	33.1	28.6
	TM+BLM	**36.1**	33.0	28.6

Table 2: BLEU Scores for MT Sets under Various Models. The MT04 set is used for tuning some hyper-parameters in the reranking model.

LM	TM	BLM
1556K	1852K	1513K

Table 3: Number of Features Collected from the Hypergraphs for the Test-sets

In the hypergraph-based discriminative reranking, we always carry out feature selection. Table 3 shows the number of features extracted from the test-sets (MT03-MT06), where BLM means the boundary n-gram language model feature set. As clear in Table 2, while the hypergraph-based discriminative training outperforms the baseline on all three test sets, the improvement in BLEU is slightly less than that from n-best based training. This may be due to overfitting in hypergraph based training as it yields many more features than n-best based training. Also, all our features so far are "local" and Huang (2008) also observes that forest-based re-ranking in *monolingual parsing* does not outperform n-best based re-ranking when only local features are used. We propose to investigate these issues in the near future.

8 Conclusions

In this paper, we present a *scalable* discriminative training framework for statistical machine translation. The framework can be used both for *n-best* and *hypergraph-based* reranking. In the context of hypergraph-based reranking, we employ an oracle extraction algorithm to efficiently extract an oracle trees from a hypergraph. To make the hypergraph-based method scalable, we adopt several efficient algorithms: hypergraph-pruning, data selection, and feature selection. We show that both n-*best* and *hypergraph-based* reranking improve over a full-scale state-of-the-art hierarchical machine translation system (Chiang, 2007). However, the *hypergraph-based* reranking under-performs the n-best reranking, which is quite surprising and the reasons remain to be investigated.

Our framework forms a solid basis for feature engineering and incorporating better machine learning methods in statistical machine translation.

References

Phil Blunsom, Trevor Cohn and Miles Osborne. 2008. A Discriminative Latent Variable Model for Statistical Machine Translation. *In Proceedings of ACL 2008.*

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201-228.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. *In Proceedings of EMNLP 2008.*

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. *In Proceedings of EMNLP 2002.*

Liang Huang. 2008. Forest Reranking: Discriminative Parsing with Non-Local Features. *In Proceedings of ACL 2008.*

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese and Omar Zaidan. 2009a. Joshua: An Open Source Toolkit for Parsing-based Machine Translation. *In Proceedings of WMT 2009.*

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009b. Variational Decoding for Statistical Machine Translation. *In Proceedings of ACL 2009.*

Zhifei Li and Jason Eisner. 2009. First- and Second-order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests. *In Proceedings of EMNLP 2009.*

Zhifei Li and Sanjeev Khudanpur. 2008. Large-scale Discriminative n-gram Language Models for Statistical Machine Translation. *In Proceedings of AMTA 2008.*

Zhifei Li and Sanjeev Khudanpur. 2009. Efficient Extraction of Oracle-best Translations from Hypergraphs. *In Proceedings of NAACL 2009.*

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

Adam Lopez. 2007. Hierarchical Phrase-Based Translation with Suffix Arrays. *In Proceedings of EMNLP-CoNLL 2007.*

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. *In Proceedings of ACL 2003.*

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. *In Proceedings of ACL 2000.*

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. *In Proceedings of ACL 2002.*

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech and Language*, 21(2):373-392.

Libin Shen, Anoop Sarkar and Franz Josef Och. 2004. Discriminative Reranking for Machine Translation. *In Proceedings of HLT/NAACL 2004.*

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. *In Proceedings of ACL 2006.*

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. *In Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 901-904.

Charles Sutton, Michael Sindelar, and Andrew McCallum. 2006. Reducing Weight Undertraining in Structured Discriminative Learning. *In Proceedings of HLT-NAACL 2006.*

Roy W. Tromble and Shankar Kumar and Franz Och and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation. *In Proceedings of EMNLP 2008.*

Vladimir Vapnik. 2006. Estimation of Dependencies Based on Empirical Data (Second Edition). Springer, 2006.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online Large-Margin Training for Statistical Machine Translation. *In Proceedings of EMNLP-CoNLL 2007.*

P.C. Woodland and D. Povey. 2002. Large scale discriminative training of hidden Markov models for speech recognition. *Computer Speech and Language*, 16(1):25-47.

Richard Zens, Sasa Hasan, and Hermann Ney. 2007. A Systematic Comparison of Training Criteria for Statistical Machine Translation. *In Proceedings of EMNLP-CoNLL 2007.*