Joint Decoding with Multiple Translation Models

Yang Liu and **Haitao Mi** and **Yang Feng** and **Qun Liu**
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
{yliu,htmi,fengyang,liuqun}@ict.ac.cn

Abstract

Current SMT systems usually decode with single translation models and cannot benefit from the strengths of other models in decoding phase. We instead propose *joint decoding*, a method that combines multiple translation models in one decoder. Our joint decoder draws connections among multiple models by integrating the translation hypergraphs they produce individually. Therefore, one model can share translations and even derivations with other models. Comparable to the state-of-the-art system combination technique, joint decoding achieves an absolute improvement of 1.5 BLEU points over individual decoding.

1 Introduction

System combination aims to find consensus translations among different machine translation systems. It proves that such consensus translations are usually better than the output of individual systems (Frederking and Nirenburg, 1994).

Recent several years have witnessed the rapid development of system combination methods based on confusion networks (e.g., (Rosti et al., 2007; He et al., 2008)), which show state-of-the-art performance in MT benchmarks. A confusion network consists of a sequence of sets of candidate words. Each candidate word is associated with a score. The optimal consensus translation can be obtained by selecting one word from each set of candidates to maximizing the overall score. While it is easy and efficient to manipulate strings, current methods usually have no access to most information available in decoding phase, which might be useful for obtaining further improvements.

In this paper, we propose a framework for combining multiple translation models directly in decoding phase. [1] Based on max-translation decoding and max-derivation decoding used in conventional *individual* decoders (Section 2), we go further to develop a *joint* decoder that integrates multiple models on a firm basis:

- Structuring the search space of each model as a *translation hypergraph* (Section 3.1), our joint decoder packs individual translation hypergraphs together by merging nodes that have identical partial translations (Section 3.2). Although such *translation-level combination* will not produce new translations, it does change the way of selecting promising candidates.
- Two models could even share derivations with each other if they produce the same structures on the target side (Section 3.3), which we refer to as *derivation-level combination*. This method enlarges the search space by allowing for mixing different types of translation rules within one derivation.
- As multiple derivations are used for finding optimal translations, we extend the minimum error rate training (MERT) algorithm (Och, 2003) to tune feature weights with respect to BLEU score for max-translation decoding (Section 4).

We evaluated our joint decoder that integrated a hierarchical phrase-based model (Chiang, 2005; Chiang, 2007) and a tree-to-string model (Liu et al., 2006) on the NIST 2005 Chinese-English testset. Experimental results show that joint decod-

[1] It might be controversial to use the term "model", which usually has a very precise definition in the field. Some researchers prefer to saying "phrase-based approaches" or "phrase-based systems". On the other hand, other authors (e.g., (Och and Ney, 2004; Koehn et al., 2003; Chiang, 2007)) do use the expression "phrase-based models". In this paper, we use the term "model" to emphasize that we integrate different approaches directly in decoding phase rather than post-processing system outputs.

$$
\begin{aligned}
S &\rightarrow \langle X_1, X_1 \rangle \\
X &\rightarrow \langle \textit{fabiao } X_1, \textit{give a } X_1 \rangle \\
X &\rightarrow \langle \textit{yanjiang}, \textit{talk} \rangle
\end{aligned}
$$

Figure 1: A derivation composed of SCFG rules that translates a Chinese sentence "*fabiao yanjiang*" into an English sentence "*give a talk*".

ing with multiple models achieves an absolute improvement of 1.5 BLEU points over individual decoding with single models (Section 5).

2 Background

Statistical machine translation is a decision problem where we need decide on the best of target sentence matching a source sentence. The process of searching for the best translation is conventionally called *decoding*, which usually involves sequences of decisions that translate a source sentence into a target sentence step by step.

For example, Figure 1 shows a sequence of SCFG rules (Chiang, 2005; Chiang, 2007) that translates a Chinese sentence "*fabiao yanjiang*" into an English sentence "*give a talk*". Such sequence of decisions is called a *derivation*. In phrase-based models, a decision can be translating a source phrase into a target phrase or reordering the target phrases. In syntax-based models, decisions usually correspond to transduction rules. Often, there are many derivations that are distinct yet produce the same translation.

Blunsom et al. (2008) present a latent variable model that describes the relationship between translation and derivation clearly. Given a source sentence $\mathbf{f}$, the probability of a target sentence $\mathbf{e}$ being its translation is the sum over all possible derivations:

$$Pr(\mathbf{e}|\mathbf{f}) = \sum_{\mathbf{d} \in \Delta(\mathbf{e},\mathbf{f})} Pr(\mathbf{d}, \mathbf{e}|\mathbf{f}) \quad (1)$$

where $\Delta(\mathbf{e}, \mathbf{f})$ is the set of all possible derivations that translate $\mathbf{f}$ into $\mathbf{e}$ and $\mathbf{d}$ is one such derivation.

They use a log-linear model to define the conditional probability of a derivation $\mathbf{d}$ and corresponding translation $\mathbf{e}$ conditioned on a source sentence $\mathbf{f}$:

$$Pr(\mathbf{d}, \mathbf{e}|\mathbf{f}) = \frac{\exp \sum_m \lambda_m h_m(\mathbf{d}, \mathbf{e}, \mathbf{f})}{Z(\mathbf{f})} \quad (2)$$

where h_m is a feature function, λ_m is the associated feature weight, and $Z(\mathbf{f})$ is a constant for normalization:

$$Z(\mathbf{f}) = \sum_{\mathbf{e}} \sum_{\mathbf{d} \in \Delta(\mathbf{e},\mathbf{f})} \exp \sum_m \lambda_m h_m(\mathbf{d}, \mathbf{e}, \mathbf{f}) \quad (3)$$

A feature value is usually decomposed as the product of decision probabilities: [2]

$$h(\mathbf{d}, \mathbf{e}, \mathbf{f}) = \prod_{d \in \mathbf{d}} p(d) \quad (4)$$

where d is a decision in the derivation $\mathbf{d}$.

Although originally proposed for supporting large sets of non-independent and overlapping features, the latent variable model is actually a more general form of conventional linear model (Och and Ney, 2002).

Accordingly, decoding for the latent variable model can be formalized as

$$\hat{\mathbf{e}} = \operatorname*{argmax}_{\mathbf{e}} \left\{ \sum_{\mathbf{d} \in \Delta(\mathbf{e},\mathbf{f})} \exp \sum_m \lambda_m h_m(\mathbf{d}, \mathbf{e}, \mathbf{f}) \right\} \quad (5)$$

where $Z(\mathbf{f})$ is not needed in decoding because it is independent of $\mathbf{e}$.

Most SMT systems approximate the summation over all possible derivations by using 1-best derivation for efficiency. They search for the 1-best derivation and take its target yield as the best translation:

$$\hat{\mathbf{e}} \approx \operatorname*{argmax}_{\mathbf{e},\mathbf{d}} \left\{ \sum_m \lambda_m h_m(\mathbf{d}, \mathbf{e}, \mathbf{f}) \right\} \quad (6)$$

We refer to Eq. (5) as *max-translation decoding* and Eq. (6) as *max-derivation decoding*, which are first termed by Blunsom et al. (2008).

By now, most current SMT systems, adopting either max-derivation decoding or max-translation decoding, have only used single models in decoding phase. We refer to them as *individual decoders*. In the following section, we will present a new method called *joint decoding* that includes multiple models in one decoder.

3 Joint Decoding

There are two major challenges for combining multiple models directly in decoding phase. First, they rely on different kinds of knowledge sources

[2] There are also features independent of derivations, such as language model and word penalty.

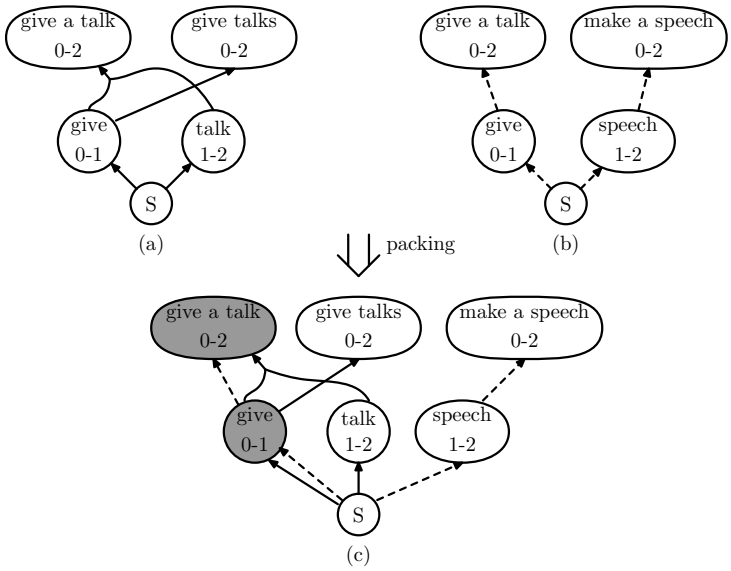

Figure 2: (a) A translation hypergraph produced by one model; (b) a translation hypergraph produced by another model; (c) the packed translation hypergraph based on (a) and (b). Solid and dashed lines denote the translation rules of the two models, respectively. Shaded nodes occur in both (a) and (b), indicating that the two models produce the same translations.

and thus need to collect different information during decoding. For example, taking a source parse as input, a tree-to-string decoder (e.g., (Liu et al., 2006)) pattern-matches the source parse with tree-to-string rules and produces a string on the target side. On the contrary, a string-to-tree decoder (e.g., (Galley et al., 2006; Shen et al., 2008)) is a parser that applies string-to-tree rules to obtain a target parse for the source string. As a result, the hypothesis structures of the two models are fundamentally different.

Second, translation models differ in decoding algorithms. Depending on the generating order of a target sentence, we distinguish between two major categories: *left-to-right* and *bottom-up*. Decoders that use rules with flat structures (e.g., phrase pairs) usually generate target sentences from left to right while those using rules with hierarchical structures (e.g., SCFG rules) often run in a bottom-up style.

In response to the two challenges, we first argue that the search space of an arbitrary model can be structured as a translation hypergraph, which makes each model connectable to others (Section 3.1). Then, we show that a packed translation hypergraph that integrates the hypergraphs of individual models can be generated in a bottom-up topological order, either integrated at the translation level (Section 3.2) or the derivation level (Section 3.3).

3.1 Translation Hypergraph

Despite the diversity of translation models, they all have to produce partial translations for substrings of input sentences. Therefore, we represent the search space of a translation model as a structure called *translation hypergraph*.

Figure 2(a) demonstrates a translation hypergraph for one model, for example, a hierarchical phrase-based model. A *node* in a hypergraph denotes a partial translation for a source substring, except for the starting node "S". For example, given the example source sentence

$$_0\ \textit{fabiao}\ _1\ \textit{yanjiang}\ _2$$

the node $\langle$"*give talks*"$, [0, 2]\rangle$ in Figure 2(a) denotes that "*give talks*" is one translation of the source string $f_1^2 =$ "*fabiao yanjiang*".

The *hyperedges* between nodes denote the decision steps that produce head nodes from tail nodes. For example, the incoming hyperedge of the node $\langle$"*give talks*"$, [0, 2]\rangle$ could correspond to an SCFG rule:

$$X \rightarrow \langle X_1\ \textit{yanjiang}, X_1\ \textit{talks}\rangle$$

Each hyperedge is associated with a number of weights, which are the feature values of the corresponding translation rules. A path of hyperedges constitutes a derivation.

Hypergraph	Decoding
node	translation
hyperedge	rule
path	derivation

Table 1: Correspondence between translation hypergraph and decoding.

More formally, a hypergraph (Klein and Manning., 2001; Huang and Chiang, 2005) is a tuple $\langle V, E, \mathbf{R} \rangle$, where V is a set of nodes, E is a set of hyperedges, and $\mathbf{R}$ is a set of weights. For a given source sentence $\mathbf{f} = f_1^n = f_1 \dots f_n$, each node $v \in V$ is in the form of $\langle t, [i, j] \rangle$, which denotes the recognition of t as one translation of the source substring spanning from i through j (that is, $f_{i+1} \dots f_j$). Each hyperedge $e \in E$ is a tuple $e = \langle tails(e), head(e), w(e) \rangle$, where $head(e) \in V$ is the consequent node in the deductive step, $tails(e) \in V^*$ is the list of antecedent nodes, and $w(e)$ is a weight function from $\mathbf{R}^{|tails(e)|}$ to $\mathbf{R}$.

As a general representation, a translation hypergraph is capable of characterizing the search space of an arbitrary translation model. Furthermore, it offers a graphic interpretation of decoding process. A node in a hypergraph denotes a translation, a hyperedge denotes a decision step, and a path of hyperedges denotes a derivation. A translation hypergraph is formally a semiring as the weight of a path is the product of hyperedge weights and the weight of a node is the sum of path weights. While max-derivation decoding only retains the single best path at each node, max-translation decoding sums up all incoming paths. Table 1 summarizes the relationship between translation hypergraph and decoding.

3.2 Translation-Level Combination

The conventional interpretation of Eq. (1) is that the probability of a translation is the sum over all possible derivations coming from the *same* model. Alternatively, we interpret Eq. (1) as that the derivations could come from *different* models.[3] This forms the theoretical basis of joint decoding.

Although the information inside a derivation differs widely among translation models, the beginning and end points (i.e., $\mathbf{f}$ and $\mathbf{e}$, respectively) must be identical. For example, a tree-to-string model first parses $\mathbf{f}$ to obtain a source tree $T(\mathbf{f})$ and then transforms $T(\mathbf{f})$ to the target sentence $\mathbf{e}$. Conversely, a string-to-tree model first parses $\mathbf{f}$ into a target tree $T(\mathbf{e})$ and then takes the surface string $\mathbf{e}$ as the translation. Despite different inside, their derivations must begin with $\mathbf{f}$ and end with $\mathbf{e}$.

This situation remains the same for derivations between a source substring f_i^j and its partial translation t during joint decoding:

$$Pr(t|f_i^j) = \sum_{\mathbf{d} \in \Delta(t, f_i^j)} Pr(\mathbf{d}, t|f_i^j) \quad (7)$$

where $\mathbf{d}$ might come from multiple models. In other words, derivations from multiple models could be brought together for computing the probability of one partial translation.

Graphically speaking, joint decoding creates a *packed* translation hypergraph that combines individual hypergraphs by merging nodes that have identical translations. For example, Figure 2 (a) and (b) demonstrate two translation hypergraphs generated by two models respectively and Figure 2 (c) is the resulting packed hypergraph. The solid lines denote the hyperedges of the first model and the dashed lines denote those of the second model. The shaded nodes are shared by both models. Therefore, the two models are combined at the *translation level*. Intuitively, shared nodes should be favored in decoding because they offer consensus translations among different models.

Now the question is how to decode with multiple models jointly in just one decoder. We believe that both left-to-right and bottom-up strategies can be used for joint decoding. Although phrase-based decoders usually produce translations from left to right, they can adopt bottom-up decoding in principle. Xiong et al. (2006) develop a bottom-up decoder for BTG (Wu, 1997) that uses only phrase pairs. They treat reordering of phrases as a binary classification problem. On the other hand, it is possible for syntax-based models to decode from left to right. Watanabe et al. (2006) propose left-to-right target generation for hierarchical phrase-based translation. Although left-to-right decoding might enable a more efficient use of language models and hopefully produce better translations, we adopt bottom-up decoding in this paper just for convenience.

Figure 3 demonstrates the search algorithm of our joint decoder. The input is a source language sentence f_1^n, and a set of translation models M

[3] The same for all $\mathbf{d}$ occurrences in Section 2. For example, $\Delta(\mathbf{e}, \mathbf{f})$ might include derivations from various models now. Note that we still use Z for normalization.

```
1: procedure JOINTDECODING(f_1^n, M)
2:     G ← ∅
3:     for l ← 1 ... n do
4:         for all i, j s.t. j − i = l do
5:             for all m ∈ M do
6:                 ADD(G, i, j, m)
7:             end for
8:             PRUNE(G, i, j)
9:         end for
10:    end for
11: end procedure
```

Figure 3: Search algorithm for joint decoding.

(line 1). After initializing the translation hypergraph G (line 2), the decoder runs in a bottom-up style, adding nodes for each span $[i, j]$ and for each model m. For each span $[i, j]$ (lines 3-5), the procedure ADD(G, i, j, m) add nodes generated by the model m to the hypergraph G (line 6). Each model searches for partial translations independently: it uses its own knowledge sources and visits its own antecedent nodes, just running like a bottom-up individual decoder. After all models finishes adding nodes for span $[i, j]$, the procedure PRUNE(G, i, j) merges identical nodes and removes less promising nodes to control the search space (line 8). The pruning strategy is similar to that of individual decoders, except that we require there must exist at least one node for each model to ensure further inference.

Although translation-level combination will not offer new translations as compared to single models, it changes the way of selecting promising candidates in a combined search space and might potentially produce better translations than individual decoding.

3.3 Derivation-Level Combination

In translation-level combination, different models interact with each other only at the nodes. The derivations of one model are unaccessible to other models. However, if two models produce the same structures on the target side, it is possible to combine two models within one derivation, which we refer to as *derivation-level combination*.

For example, although different on the source side, both hierarchical phrase-based and tree-to-string models produce strings of terminals and nonterminals on the target side. Figure 4 shows a derivation composed of both hierarchical phrase pairs and tree-to-string rules. Hierarchical phrase pairs are used for translating smaller units and tree-to-string rules for bigger ones. It is appealing to combine them in such a way because the hierarchical phrase-based model provides excellent rule coverage while the tree-to-string model offers linguistically motivated non-local reordering. Similarly, Blunsom and Osborne (2008) use both hierarchical phrase pairs and tree-to-string rules in decoding, where source parse trees serve as conditioning context rather than hard constraints.

$$\begin{aligned} \text{IP}(x_1\text{:VV}, x_2\text{:NN}) &\rightarrow x_1\ x_2 \\ X &\rightarrow \langle \textit{fabiao, give} \rangle \\ X &\rightarrow \langle \textit{yanjiang, a talk} \rangle \end{aligned}$$

Figure 4: A derivation composed of both SCFG and tree-to-string rules.

Depending on the target side output, we distinguish between *string-targeted* and *tree-targeted* models. String-targeted models include phrase-based, hierarchical phrase-based, and tree-to-string models. Tree-targeted models include string-to-tree and tree-to-tree models. All models can be combined at the translation level. Models that share with same target output structure can be further combined at the derivation level.

The joint decoder usually runs as max-translation decoding because multiple derivations from various models are used. However, if all models involved belong to the same category, a joint decoder can also adopt the max-derivation fashion because all nodes and hyperedges are accessible now (Section 5.2).

Allowing derivations for comprising rules from different models and integrating their strengths, derivation-level combination could hopefully produce new and better translations as compared with single models.

4 Extended Minimum Error Rate Training

Minimum error rate training (Och, 2003) is widely used to optimize feature weights for a linear model (Och and Ney, 2002). The key idea of MERT is to tune one feature weight to minimize error rate each time while keep others fixed. Therefore, each

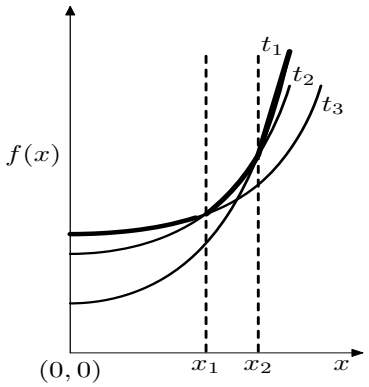

Figure 5: Calculation of critical intersections.

candidate translation can be represented as a line:

$$f(x) = a \times x + b \tag{8}$$

where a is the feature value of current dimension, x is the feature weight being tuned, and b is the dotproduct of other dimensions. The intersection of two lines is where the candidate translation will change. Instead of computing all intersections, Och (2003) only computes *critical* intersections where highest-score translations will change. This method reduces the computational overhead significantly.

Unfortunately, minimum error rate training cannot be directly used to optimize feature weights of max-translation decoding because Eq. (5) is not a linear model. However, if we also tune one dimension each time and keep other dimensions fixed, we obtain a monotonic curve as follows:

$$f(x) = \sum_{k=1}^{K} \mathrm{e}^{a_k \times x + b_k} \tag{9}$$

where K is the number of derivations for a candidate translation, a_k is the feature value of current dimension on the kth derivation and b_k is the dotproduct of other dimensions on the kth derivation. If we restrict that a_k is always non-negative, the curve shown in Eq. (9) will be a monotonically increasing function. Therefore, it is possible to extend the MERT algorithm to handle situations where multiple derivations are taken into account for decoding.

The key difference is the calculation of critical intersections. The major challenge is that two curves might have multiple intersections while two lines have at most one intersection. Fortunately, as the curve is monotonically increasing, we need only to find the *leftmost* intersection of a curve with other curves that have greater values after the intersection as a candidate critical intersection.

Figure 5 demonstrates three curves: t_1, t_2, and t_3. Suppose that the left bound of x is 0, we compute the function values for t_1, t_2, and t_3 at $x = 0$ and find that t_3 has the greatest value. As a result, we choose $x = 0$ as the first critical intersection. Then, we compute the leftmost intersections of t_3 with t_1 and t_2 and choose the intersection closest to $x = 0$, that is x_1, as our new critical intersection. Similarly, we start from x_1 and find x_2 as the next critical intersection. This iteration continues until it reaches the right bound. The bold curve denotes the translations we will choose over different ranges. For example, we will always choose t_2 for the range $[x_1, x_2]$.

To compute the leftmost intersection of two curves, we divide the range from current critical intersection to the right bound into many bins (i.e., smaller ranges) and search the bins one by one from left to right. We assume that there is at most one intersection in each bin. As a result, we can use the Bisection method for finding the intersection in each bin. The search process ends immediately once an intersection is found.

We divide max-translation decoding into three phases: (1) build the translation hypergraphs, (2) generate n-best translations, and (3) generate n'-best derivations. We apply Algorithm 3 of Huang and Chiang (2005) for n-best list generation. Extended MERT runs on n-best translations plus n'-best derivations to optimize the feature weights. Note that feature weights of various models are tuned jointly in extended MERT.

5 Experiments

5.1 Data Preparation

Our experiments were on Chinese-to-English translation. We used the FBIS corpus (6.9M + 8.9M words) as the training corpus. For language model, we used the SRI Language Modeling Toolkit (Stolcke, 2002) to train a 4-gram model on the Xinhua portion of GIGAWORD corpus. We used the NIST 2002 MT Evaluation test set as our development set, and used the NIST 2005 test set as test set. We evaluated the translation quality using *case-insensitive* BLEU metric (Papineni et al., 2002).

Our joint decoder included two models. The

Model	Combination	Max-derivation		Max-translation	
		Time	BLEU	Time	BLEU
hierarchical	N/A	40.53	30.11	44.87	29.82
tree-to-string	N/A	6.13	27.23	6.69	27.11
both	translation	N/A	N/A	55.89	30.79
	derivation	48.45	31.63	54.91	31.49

Table 2: Comparison of individual decoding and joint decoding on average decoding time (seconds/sentence) and BLEU score (case-insensitive).

first model was the hierarchical phrase-based model (Chiang, 2005; Chiang, 2007). We obtained word alignments of training data by first running GIZA++ (Och and Ney, 2003) and then applying the refinement rule "grow-diag-final-and" (Koehn et al., 2003). About 2.6M hierarchical phrase pairs extracted from the training corpus were used on the test set.

Another model was the tree-to-string model (Liu et al., 2006; Liu et al., 2007). Based on the same word-aligned training corpus, we ran a Chinese parser on the source side to obtain 1-best parses. For 15,157 sentences we failed to obtain 1-best parses. Therefore, only 93.7% of the training corpus were used by the tree-to-string model. About 578K tree-to-string rules extracted from the training corpus were used on the test set.

5.2 Individual Decoding Vs. Joint Decoding

Table 2 shows the results of comparing individual decoding and joint decoding on the test set. With conventional max-derivation decoding, the hierarchical phrase-based model achieved a BLEU score of 30.11 on the test set, with an average decoding time of 40.53 seconds/sentence. We found that accounting for all possible derivations in max-translation decoding resulted in a small negative effect on BLEU score (from 30.11 to 29.82), even though the feature weights were tuned with respect to BLEU score. One possible reason is that we only used n-best derivations instead of all possible derivations for minimum error rate training.

Max-derivation decoding with the tree-to-string model yielded much lower BLEU score (i.e., 27.23) than the hierarchical phrase-based model. One reason is that the tree-to-string model fails to capture a large amount of linguistically unmotivated mappings due to syntactic constraints. Another reason is that the tree-to-string model only used part of the training data because of parsing failure. Similarly, accounting for all possible derivations in max-translation decoding failed to bring benefits for the tree-to-string model (from 27.23 to 27.11).

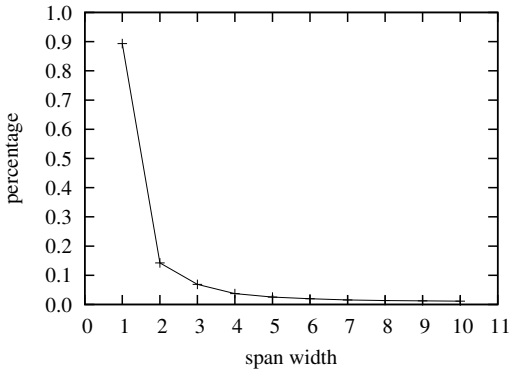

Figure 6: Node sharing in max-translation decoding with varying span widths. We retain at most 100 nodes for each source substring for each model.

When combining the two models at the translation level, the joint decoder achieved a BLEU score of 30.79 that outperformed the best result (i.e., 30.11) of individual decoding significantly ($p < 0.05$). This suggests that accounting for all possible derivations from multiple models will help discriminate among candidate translations.

Figure 6 demonstrates the percentages of nodes shared by the two models over various span widths in packed translation hypergraphs during max-translation decoding. For one-word source strings, 89.33% nodes in the hypergrpah were shared by both models. With the increase of span width, the percentage decreased dramatically due to the diversity of the two models. However, there still exist nodes shared by two models even for source substrings that contain 33 words.

When combining the two models at the derivation level using max-derivation decoding, the joint decoder achieved a BLEU score of 31.63 that outperformed the best result (i.e., 30.11) of individ-

Method	Model	BLEU
individual decoding	hierarchical	30.11
	tree-to-string	27.23
system combination	both	31.50
joint decoding	both	31.63

Table 3: Comparison of individual decoding, system combination, and joint decoding.

ual decoding significantly ($p < 0.01$). This improvement resulted from the mixture of hierarchical phrase pairs and tree-to-string rules. To produce the result, the joint decoder made use of 8,114 hierarchical phrase pairs learned from training data, 6,800 glue rules connecting partial translations monotonically, and 16,554 tree-to-string rules. While tree-to-string rules offer linguistically motivated non-local reordering during decoding, hierarchical phrase pairs ensure good rule coverage. Max-translation decoding still failed to surpass max-derivation decoding in this case.

5.3 Comparison with System Combination

We re-implemented a state-of-the-art system combination method (Rosti et al., 2007). As shown in Table 3, taking the translations of the two individual decoders as input, the system combination method achieved a BLEU score of 31.50, slightly lower than that of joint decoding. But this difference is not significant statistically.

5.4 Individual Training Vs. Joint Training

Table 4 shows the effects of individual training and joint training. By individual, we mean that the two models are trained independently. We concatenate and normalize their feature weights for the joint decoder. By joint, we mean that they are trained together by the extended MERT algorithm. We found that joint training outperformed individual training significantly for both max-derivation decoding and max-translation decoding.

Training	Max-derivation	Max-translation
individual	30.70	29.95
joint	31.63	30.79

Table 4: Comparison of individual training and joint training.

6 Related Work

System combination has benefited various NLP tasks in recent years, such as products-of-experts (e.g., (Smith and Eisner, 2005)) and ensemble-based parsing (e.g., (Henderson and Brill, 1999)). In machine translation, confusion-network based combination techniques (e.g., (Rosti et al., 2007; He et al., 2008)) have achieved the state-of-the-art performance in MT evaluations. From a different perspective, we try to combine different approaches directly in decoding phase by using hypergraphs. While system combination techniques manipulate only the final translations of each system, our method opens the possibility of exploiting much more information.

Blunsom et al. (2008) first distinguish between max-derivation decoding and max-translation decoding explicitly. They show that max-translation decoding outperforms max-derivation decoding for the latent variable model. While they train the parameters using a maximum *a posteriori* estimator, we extend the MERT algorithm (Och, 2003) to take the evaluation metric into account.

Hypergraphs have been successfully used in parsing (Klein and Manning., 2001; Huang and Chiang, 2005; Huang, 2008) and machine translation (Huang and Chiang, 2007; Mi et al., 2008; Mi and Huang, 2008). Both Mi et al. (2008) and Blunsom et al. (2008) use a translation hypergraph to represent search space. The difference is that their hypergraphs are specifically designed for the forest-based tree-to-string model and the hierarchical phrase-based model, respectively, while ours is more general and can be applied to arbitrary models.

7 Conclusion

We have presented a framework for including multiple translation models in one decoder. Representing search space as a translation hypergraph, individual models are accessible to others via sharing nodes and even hyperedges. As our decoder accounts for multiple derivations, we extend the MERT algorithm to tune feature weights with respect to BLEU score for max-translation decoding. In the future, we plan to optimize feature weights for max-translation decoding directly on the entire packed translation hypergraph rather than on n-best derivations, following the lattice-based MERT (Macherey et al., 2008).

Acknowledgement

The authors were supported by National Natural Science Foundation of China, Contracts 60873167 and 60736014, and 863 State Key Project No. 2006AA010108. Part of this work was done while Yang Liu was visiting the SMT group led by Stephan Vogel at CMU. We thank the anonymous reviewers for their insightful comments. We are also grateful to Yajuan Lü, Liang Huang, Nguyen Bach, Andreas Zollmann, Vamshi Ambati, and Kevin Gimpel for their helpful feedback.

References

Phil Blunsom and Mile Osborne. 2008. Probabilistic inference for machine translation. In *Proc. of EMNLP08*.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. of ACL08*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL05*.

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

Robert Frederking and Sergei Nirenburg. 1994. Three heads are better than one. In *Proc. of ANLP94*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of ACL06*.

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-HMM-based hypothesis alignment for combining outputs from machine translation systems. In *Proc. of EMNLP08*.

John C. Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proc. of EMNLP99*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proc. of IWPT05*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of ACL07*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL08*.

Dan Klein and Christopher D. Manning. 2001. Parsing and hypergraphs. In *Proc. of ACL08*.

Phillip Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL03*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL06*.

Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *Proc. of ACL07*.

Wolfgang Macherey, Franz J. Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proc. of EMNLP08*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proc. of EMNLP08*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. of ACL08*.

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL02*.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1).

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4).

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL03*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL02*.

Antti-Veikko Rosti, Spyros Matsoukas, and Richard Schwartz. 2007. Improved word-level system combination for machine translation. In *Proc. of ACL07*.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL08*.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL05*.

Andreas Stolcke. 2002. Srilm - an extension language model modeling toolkit. In *Proc. of ICSLP02*.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *Proc. of ACL06*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proc. of ACL06*.