Joint Optimization for Machine Translation System Combination

Xiaodong He
Microsoft Research
One Microsoft Way, Redmond, WA
xiaohe@microsoft.com

Kristina Toutanova
Microsoft Research
One Microsoft Way, Redmond, WA
kristout@microsoft.com

Abstract

System combination has emerged as a powerful method for machine translation (MT). This paper pursues a joint optimization strategy for combining outputs from multiple MT systems, where word alignment, ordering, and lexical selection decisions are made jointly according to a set of feature functions combined in a single log-linear model. The decoding algorithm is described in detail and a set of new features that support this joint decoding approach is proposed. The approach is evaluated in comparison to state-of-the-art confusion-network-based system combination methods using equivalent features and shown to outperform them significantly.

1 Introduction

System combination for machine translation (MT) has emerged as a powerful method of combining the strengths of multiple MT systems and achieving results which surpass those of each individual system (e.g. Bangalore, et. al., 2001, Matusov, et. al., 2006, Rosti, et. al., 2007a). Most state-of-the-art system combination methods are based on constructing a confusion network (CN) from several input translation hypotheses, and choosing the best output from the CN based on several scoring functions (e.g. Rosti et. al., 2007a, He et. al., 2008, Matusov et al. 2008). Confusion networks allow word-level system combination, which was shown to outperform sentence re-ranking methods and phrase-level combination (Rosti, et. al. 2007a).

We will review confusion-network-based system combination with the help of the examples in Figures 1 and 2. Figure 1 shows translation hypotheses from three Chinese-to-English MT systems. The general idea is to combine hypotheses in a representation such as the ones in Figure 2, where for each word position there is a set of possible words, shown in columns.[1] The final output is determined by choosing one word from each column, which can be a real word or the empty word ε. For example, the CN in Figure 2a) can generate eight distinct sequences of words, including e.g. "she bought the Jeep" and "she bought the SUV Jeep". The choice is performed to maximize a scoring function using a set of features and a log-linear model (Matusov, et. al 2006, Rosti, et al. 2007a).

We can view a confusion network as an ordered sequence of columns (*correspondence sets*). Each word from each input hypothesis belongs to exactly one correspondence set. Each correspondence set contains at most one word from each input hypothesis and contributes exactly one of its words (including the possible ε) to the final output. Final words are output in the order of correspondence sets. In order to construct such a representation, we need to solve the following two sub-problems: arrange words from all input hypotheses into correspondence sets (alignment problem) and order correspondence sets (ordering problem). After constructing the confusion network we need to solve a third sub-problem: decide which words to output from each correspondence set (lexical choice problem).

In current state-of-the-art approaches, the construction of the confusion network is performed as follows: first, a backbone hypothesis is selected. The backbone hypothesis determines the order of words in the final system output, and guides word-level alignments for construction of columns of possible words at each position. Let us assume that for our example in Figure 1, the second hypothesis is selected as a backbone. All other hypotheses are aligned to the backbone such that these alignments are one-to-one; empty words are inserted where necessary to make one-to-one

[1] This representation is alternative to directed acyclic graph representations of confusion networks.

alignment possible. Words in all hypotheses are sorted by the position of the backbone word they align to and the confusion network is determined.

It is clear that the quality of selection of the backbone and alignments has a large impact on the performance, because the word order is determined by the backbone, and the set of possible words at each position is determined by alignment. Since the space of possible alignments is extremely large, approximate and heuristic techniques have been employed to derive them. In pair-wise alignment, each hypothesis is aligned to the backbone in turn, with separate processing to combine the multiple alignments. Several models have been used for pair-wise alignment, starting with TER and proceeding with more sophisticated techniques such as HMM models, ITG, and IHMM (Rosti et. al 2007a, Matusov et al 2008, Krakos et al. 2008, He et al. 2008). A major problem with such methods is that each hypothesis is aligned to the backbone independently, leading to sub-optimal behavior. For example, suppose that we use a state-of-the-art word alignment model for pairs of hypotheses, such as the IHMM. Figure 1 shows likely alignment links between every pair of hypotheses. If Hypothesis 1 is aligned to Hypothesis 2 (the backbone), *Jeep* is likely to align to *SUV* because they express similar Chinese content. Hypothesis 3 is separately aligned to the backbone and since the alignment is constrained to be one-to-one, *SUV* is aligned to *SUV* and *Jeep* to an empty word which is inserted after *SUV*. The network in Figure 2a) is the result of this process. An undesirable property of this CN is that the two instances of Jeep are placed in separate columns and cannot vote to reinforce each other.

Incremental alignment methods have been proposed to relax the independence assumption of pair-wise alignment (Rosti et al. 2008, Li et al. 2009). Such methods align hypotheses to a partially constructed CN in some order. For example, if in such method, Hypothesis 3 is first aligned to the backbone, followed by Hypothesis 1, we are likely to arrive at the CN in Figure 2b) in which the two instances of Jeep are aligned. However, if Hypothesis 1 is aligned to the backbone first, we would still get the CN in Figure 2a). Notice that the desirable output "*She bought the Jeep SUV*" cannot be generated from either of the confusion networks because a re-reordering of columns would be required.

A common characteristic of CN-based approaches is that the order of words (backbone) and the alignment of words (correspondence sets) are decided as greedy steps independently of the lexical choice for the final output. The backbone and alignment are optimized according to auxiliary scoring functions and heuristics which may or may not be optimal with respect to producing CNs leading to good translations. In some recent approaches, these assumptions are relaxed to allow each input hypothesis as a backbone. Each backbone produces a separate CN and the decision of which CN to choose is taken at a later decoding stage, but this still restricts the possible orders and alignments greatly (Rosti et al. 2008, Matusov et al. 2008).

In this paper, we present a joint optimization method for system combination. In this method, the alignment, ordering and lexical selection sub-problems are solved jointly in a single decoding framework based on a log-linear model.

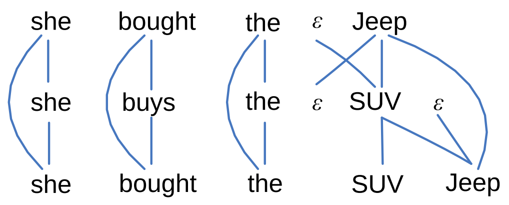

Figure 1. Three MT system hypotheses with pair-wise alignments.

she	bought	the	Jeep	ε
she	buys	the	SUV	ε
she	bought	the	SUV	Jeep

a) Confusion network with pair-wise alignment.

she	bought	the	ε	Jeep
she	buys	the	SUV	ε
she	bought	the	SUV	Jeep

b) Confusion network with incremental alignment.

Figure 2. Correspondence sets of confusion networks under pair-wise and incremental alignment, using the second hypothesis as a backbone.

2 Related Work

There has been a large body of work on MT system combination. Among confusion-network-based algorithms, most relevant to our work are state-of-the-art methods for constructing word alignments (correspondence sets) and methods for improving the selection of a backbone hypothesis. We have already reviewed such work in the introduction and will note relation to specific models throughout the paper as we discuss specifics of our scoring functions.

In confusion network algorithms which use pair-wise (or incremental) word-level alignment algorithms for correspondence set construction, problems of converting many-to-many alignments and handling multiple insertions and deletions need to be addressed. Prior work has used a number of heuristics to deal with these problems (Matusov, et. al., 2006, He et al 08). Some work has made such decisions in a more principled fashion by computing model-based scores (Matusov et al. 2008), but still special-purpose algorithms and heuristics are needed and a single alignment is fixed.

In our approach, no heuristics are used to convert alignments and no concept of a backbone is used. Instead, the globally highest scoring combination of alignment, order, and lexical choice is selected (subject to search error).

Other than confusion-network-based algorithms, work most closely related to ours is the method of MT system combination proposed in (Jayaraman and Lavie 2005), which we will refer to as J&L. Like our method, this approach performs word-level system combination and is not limited to following the word order of a single backbone hypothesis; it also allows more flexibility in the selection of correspondence sets during decoding, compared to a confusion-network-based approach. Even though their algorithm and ours are broadly similar, there are several important differences.

Firstly, the J&L approach is based on pair-wise alignments between words in different hypotheses, which are hard and do not have associated probabilities. Every word in every hypothesis is aligned to at most one word from each of the remaining hypotheses. Thus there is no uncertainty about which words should belong to the correspondence set of an aligned word w, once that word is selected to extend a partial hypothesis during search. If words do not have corresponding matching words in some hypotheses, heuristic matching to currently unused words is attempted.

In contrast, our algorithm is based on the definition of a joint scoring model, which takes into account alignment uncertainty and combines information from word-level alignment models, ordering and lexical selection models, to address the three sub-problems of word-level system combination. In addition to the language model and word-voting features used by the J&L model, we incorporate features which measure alignment confidence via word-level alignment models and features which evaluate re-ordering via distortion models with respect to original hypotheses. While the J&L search algorithm incorporates a number of special-purpose heuristics to address phenomena of unused words lagging behind the last used words, the goal in our work is to minimize heuristics and perform search to jointly optimize the assignment of hidden variables (ordered correspondence sets) and observed output variables (words in final translations).

Finally, the J&L method has not been evaluated in comparison to confusion-network-based methods to study the impact of performing joint decoding for the three sub-problems.

3 Notation

Before elaborating the models and decoding algorithms, we first clarify the notation that will be used in the paper.

We denote by $\boldsymbol{H} = \{h_1, \dots, h_N\}$ the set of hypotheses from multiple MT systems, where h_i is the hypothesis from the i-th system and h_i is a word sequence $w_{i,1}, \dots, w_{i,L(i)}$ with length $L(i)$. For simplicity, we assume that each system contributes only its 1-best hypothesis for combination. Accordingly, the i-th hypothesis h_i will be associated with a weight $W(i)$ which is the weight of the i-th system. In the scenario that N-best lists are available from individual systems for combination, the weight of each hypothesis can be computed based on its rank in the N-best list (Rosti et. al. 2007a).

Like in CN-based system combination, we construct a set of ordered correspondence sets (CS) from input hypotheses, and select one word from each CS to form the final output. A CS is defined as a set of (possibly empty) words, one from each hypothesis, that implicitly align to each other and that contributes exactly one of its words to the final output. A valid complete set of CS includes each non-empty word from each hypothesis in exactly one CS. As opposed to CN-based algorithms, our ordered correspondence sets are constructed during a joint decoding process which performs lexical selection at the same time.

To facilitate the presentation of our features, we define notation for ordered CS. A sequence of correspondence sets is denoted by $C = CS_1, \dots, CS_m$. Each correspondence set is specified by listing the positions of each of the words in the CS in their respective input hypotheses. Each input hypothesis is assumed to have one special empty word ε at position 0. A CS is denoted by $CS(l_1, \ldots, l_N) = \{w_{1,l_1}, \ldots, w_{N,l_N}\}$, where w_{i,l_i} is the l_i-th word in the i-th hypothesis and the word position vector $v = [l_1, \ldots, l_N]^T$ specifies the position of each word in its original hypothesis. Correspondingly, word w_{i,l_i} has the same weight $W(i)$ as its original hypothesis h_i. As an example, the last two correspondence sets specified by the CN in Figure 2a) would be specified as $CS_4 = CS(4,4,4) = \{Jeep, SUV, SUV\}$ and $CS_5 = CS(0,0,5) = \{\varepsilon, \varepsilon, Jeep\}$.

As opposed to the CS defined in a conventional CN, words that have the same surface form but come from different hypotheses are not collapsed to be one single candidate since they have different original word positions. We need to trace each of them separately during the decoding process.

4 A Joint Optimization Framework For System Combination

The joint decoding framework chooses optimal output according to the following log-linear model:

$$w^* = \underset{w \in \boldsymbol{W}, O \in \boldsymbol{O}, C \in \boldsymbol{C}}{\operatorname{argmax}} \; exp\left\{\sum_{i=1}^{F} \alpha_i \cdot f_i(w, O, C, \boldsymbol{H})\right\}$$

where we denote by $\boldsymbol{C}$ the set of all possible valid arrangements of CS, $\boldsymbol{O}$ the set of all possible orders of CS, $\boldsymbol{W}$ the set of all possible word sequences, consisting of words from the input hypotheses. $\{f_i(w, O, C, \boldsymbol{H})\}$ are the features and $\{\alpha_i\}$ are the feature weights in the log-linear model, respectively.

4.1 Features

A set of features are used in this framework. Each of them models one or more of the alignment, ordering, and lexical selection sub-problems. Features are defined as follows.

Word posterior model:

The word posterior feature is the same as the one proposed by Rosti et. al. (2007a). i.e.,

$$f_{wp}(w, O, C, \boldsymbol{H}) = \sum_{m=1}^{M} log\big(P(w_m | CS_m)\big)$$

where the posterior of a single word in a CS is computed based on a weighted voting score:

$$P(w_{i,l_i} | CS) = P\left(w_{i,l_i} \middle| CS(l_1, \ldots, l_N)\right) = \sum_{k=1}^{N} W(k)\, \delta(w_{k,l_k} = w_{i,l_i})$$

and M is the number of CS generated. Note that M may be larger than the length of the output word sequence w since some CS may generate empty words.

Bi-gram voting model:

The second feature we used is a bi-gram voting feature proposed by Zhao and He (2009), i.e., for each bi-gram $\langle w_i, w_{i+1}\rangle$, a weighted position-independent voting score is computed:

$$P(\langle w_i, w_{i+1}\rangle | \boldsymbol{H}) = \sum_{k=1}^{N} W(k)\, \delta(\langle w_i, w_{i+1}\rangle \in h_i)$$

And the global bi-gram voting feature is defined as:

$$f_{bgv}(w, O, C, \boldsymbol{H}) = \sum_{i=1}^{|w|-1} log\big(P(\langle w_i, w_{i+1}\rangle | \boldsymbol{H})\big)$$

Distortion model:

Unlike in the conventional CN-based system combination, flexible orders of CS are allowed in this joint decoding framework. In order to model the distortion of different orderings, a distortion model between two CS is defined as follows:

First we define the distortion cost between two words at a single hypothesis. Similarly to the distortion penalty in the conventional phrase-based decoder (Koehn 2004b), the distortion cost of jumping from a word at position i to another word at position j, $d(i,j)$, is proportional to the distance between i and j, e.g., $|i\text{-}j|$. Then, the distortion cost of jumping from one CS, which has a position vector recording the original position of each word in that CS, to another CS is a weighted sum of single-hypothesis-based distortion costs:

$$d(CS_m, CS_{m+1}) = \sum_{k=1}^{N} W(k) \cdot |l_{m,k} - l_{m+1,k}|$$

where $l_{m,k}$ and $l_{m+1,k}$ are the k-th element of the word position vector of CS_m and CS_{m+1}, respectively. For the purpose of computing the distortion feature, the position of an empty word is taken to be the same as the position of the last visited non-empty word from the same hypothesis.

The overall ordering feature can then be computed based on $d(CS_m, CS_{m+1})$:

$$f_{dis}(w, O, C, \boldsymbol{H}) = -\sum_{m=1}^{M-1} d(CS_m, CS_{m+1})$$

It is worth noting that this is not the only feature modeling the re-ordering behavior. Under the joint decoding framework, other features such as the language model and bi-gram voting affect the ordering as well.

Alignment model:

Each CS consists of a set of words, one from each hypothesis, that are implicitly aligned to each other. Therefore, a valid complete set of CS defines the word alignment among different hypotheses. In this paper, we derive an alignment score of a CS based on alignment scores of word pairs in that CS. To compute scores for word pairs, we perform pair-wise hypothesis alignment using the indirect HMM (He et al. 2008) for every pair of input hypotheses. Note that this involves a total of *N* by (*N*-1)/2 bi-directional hypothesis alignments. The alignment score for a pair of words w_{j,l_j} and w_{k,l_k} is defined as the average of posterior probabilities of alignment links in both directions and is thus direction independent:

$$p\left(w_{j,l_j}, w_{k,l_k}\right) = \frac{1}{2}\left(p(a_{l_j} = l_k | h_j, h_k) + p(a_{l_k} = l_j | h_k, h_j)\right)$$

If one of the two words is ε, the posterior of aligning word ε to state j is computed as suggested by Liang et al. (2006), i.e.,

$$p\left(a_0 = l_j | h_k, h_j\right) = \prod_{i=1}^{L(k)} \left(1 - p\left(a_i = l_j | h_k, h_j\right)\right)$$

And $p(a_{l_j} = 0 | h_j, h_k)$ can be computed by the HMM directly.

If both words are ε, then a pre-defined $p_{\varepsilon\varepsilon}$ is assigned, i.e., $p\left(a_0 = 0 | h_k, h_j\right) = p_{\varepsilon\varepsilon}$, where $p_{\varepsilon\varepsilon}$ can be optimized on a held-out validation set.

For a CS of words, if we set the *j*-th word as an anchor word, the probability that all other words align to that word is:

$$p(j|CS) = \prod_{\substack{k=1 \\ k \neq j}}^{N} p\left(w_{j,l_j}, w_{k,l_k}\right)$$

The alignment score of the whole CS is a weighted sum of the logarithm of the above alignment probabilities, i.e.,

$$S_{aln}(CS) = \sum_{j=1}^{N} W(j) \, log(P(j|CS))$$

and the global alignment score is computed as:

$$f_{aln}(w, O, C, \boldsymbol{H}) = \sum_{m=1}^{M} S_{aln}(CS_m)$$

Entropy model:

In general, it is preferable to align the same word from different hypotheses into a common CS. Therefore, we use entropy to model the purity of a CS. The entropy of a CS is defined as:

$$Ent(CS) = Ent(CS(l_1, \ldots, l_N)) = \sum_{i=1}^{N'} P\left(w_{i,l_i} | CS\right) log P\left(w_{i,l_i} | CS\right)$$

where the sum is taken over all distinct words in the CS. Then the global entropy score is computed as:

$$f_{ent}(w, O, C, \boldsymbol{H}) = \sum_{m=1}^{M} Ent(CS_m)$$

Other features used in our log-linear model include the count of real words |*w*|, a n-gram language model, and the count *M* of CS sets.

These features address one or more of the three sub-problems of MT system combination. By performing joint decoding with all these features working together, we hope to derive better decisions on alignment, ordering and lexical selection.

5 Joint Decoding

5.1 Core algorithm

Decoding is based on a beam search algorithm similar to that of the phrase-based MT decoder (Koehn 2004b). The input is a set of translation hypotheses to be combined, and the final output sentence is generated left to right. Figure 3 illustrates the decoding process, using the example input hypotheses from Figure 1. Each decoding state represents a partial sequence of correspondence sets covering some of the words in the input hypotheses and a sequence of words selected from the CS to form a partial output hypothesis. The initial decoding state has an empty sequence of CS and an empty output sequence. A state corresponds to a complete output candidate if its CS covers all input words.

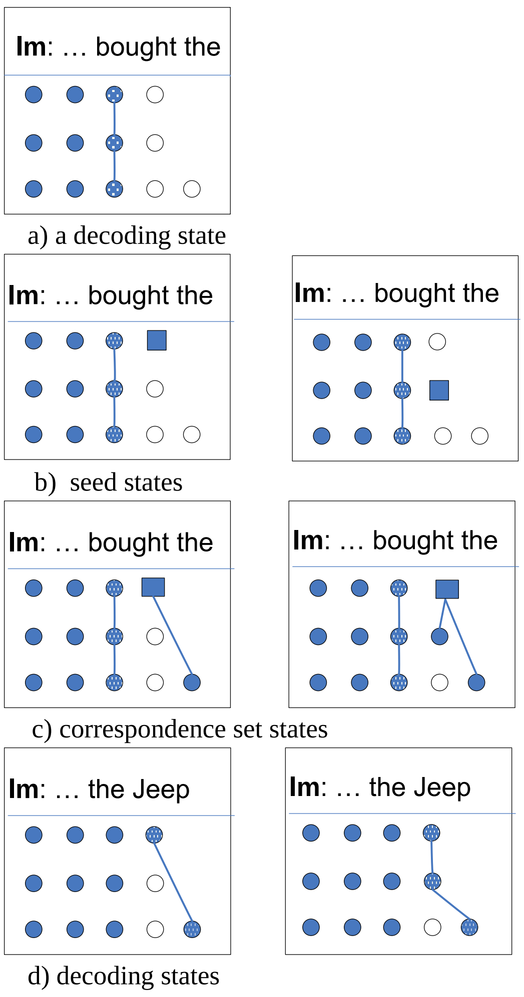

Figure 3. Illustration of the decoding process.

In practice, because the features over hypotheses can be decomposed, we do not need to encode all of this information in a decoding state. It suffices to store a few attributes. They include positions of words from each input hypothesis that have been visited, the last two non-empty words generated (if a tri-gram LM is used), and an "end position vector (EPV)" recording positions of words in the last CS, which were just visited. In the figure, the visited words are shown with filled circles and the EPV is shown with a dotted pattern in the filled circles. Words specified by the EPV are implicitly aligned. In the state in Figure 3 a) the first three words of each hypothesis have been visited, the third word of each hypothesis is the last word visited (in the EPV), and the last two words produced are "*bought the*". The states also record the decoding score accumulated so far and an estimated future score to cover words that have not been visited yet (not shown).

The expansion from one decoding state to a set of new decoding states is illustrated in Figure 3. The expansion is done in three steps with the help of intermediate states. Starting from a decoding state as shown in Figure 3a), first a set of "seed states" as shown in Figure 3b) are generated. Each seed state represents a choice of one of unvisited words, called a "seed word" which is selected and marked as visited. For example, the word *Jeep* from the first hypothesis and the word *SUV* from the second hypothesis are selected in the two seed states shown in Figure 3b), respectively. These seed states further expand into a set of "CS states" as shown in Figure 3c). I.e., a CS is formed by picking one word from each of the other hypotheses which is unvisited and has a valid alignment link to the seed word. Figure 3c) shows two CS states expanded from the first seed state of Figure 3b), using *Jeep* from the first hypothesis as a seed word. In one of them the empty word from the second hypothesis is chosen, and in the other, the word *SUV* is chosen. Both are allowed by the alignments illustrated in Figure 1. Finally, each CS state generates one or more complete decoding states, in which a word is chosen from the current CS and the EPV vector is advanced to reflect the last newly visited words. Figure 3d) shows two such states, descending from the corresponding CS states in 3c). After one more expansion the state in 3d) on the left can generate the translation "*She bought the Jeep SUV*", which cannot be produced by either confusion network in Figure 2.

5.2 Pruning

The full search space of joint decoding is a product of the alignment, ordering, and lexical selection spaces. Its size is exponential in the length of the sentence and the number of hypotheses involved in combination. Therefore, pruning techniques are necessary to reduce the search space.

First we will prune down the alignment space. Instead of allowing any alignment link between arbitrary words of two hypotheses, only links that have alignment score higher than a threshold are allowed, plus links in the union of the Viterbi alignments in both directions. In order to prevent the garbage collection problem where many words align to a rare word at the other side (Moore, 2004), we further impose the limit that if one word is aligned to more than *T* words, these links are sorted by their alignment score and only the top *T* links are kept. Meanwhile, alignments between a real word and ε are always allowed.

We then prune down the ordering space by limiting the expansion of new states. Only states that are *adjacent* to their preceding states are created. Two states are called *adjacent* if their EPVs are adjacent, i.e., given the EPV of the preceding state *m* as $\left[l_{m,1},\dots,l_{m,N}\right]^T$ and the EPV of the next state *m*+1 as $\left[l_{m+1,1},\dots,l_{m+1,N}\right]^T$, if at least at one dimension *k*, $l_{m+1,k} = l_{m,k}+1$, then these two states are *adjacent*. When checking the adjacency of two states, the position of an empty word is taken to be the same as the position of the last visited non-empty word from the same hypothesis.

The number of possible CS states expanded from a decoding state is exponential in the number of hypotheses. In decoding, these CS states are sorted by their alignment scores and only the top *K* CS states are kept.

The search space can be further pruned down by the widely used technique of path recombination and by best-first pruning.

Path recombination is a risk-free pruning method. Two paths can be recombined if they agree on a) words from each hypothesis that have been visited so far, b) the last two real words generated, and c) their EPVs. In such case, we only need to keep the path with the higher score.

Best-first pruning can help to reduce the search space even further. In the decoding process we compare paths that have generated the same number of words (both real and empty words) and only keep a certain number of most promising paths. Pruning is based on an estimated overall score of each path, which is the sum of the decoding score accumulated so far and an estimated future score to cover the words that have not been visited. Next we discuss the future score computation.

5.3 Computing the future score

In order to estimate the future cost of an unfinished path, we treat the unvisited words of one input hypothesis as a backbone, and apply a greedy search for alignment based on it; i.e., for each word of this backbone, the most likely words (based on the alignment link scores) from other hypotheses, one word from each hypothesis, are collected to form a CS. These CS are ordered according to the word order of the backbone and form a CN. Then, a light decoding process with a search beam of size one is applied to decode this CN and find the approximate future path, with future feature scores computed during the decoding process. If there are leftover words not included in this CN, they are treated in the way described in section 5.4. Additionally, caching techniques are applied to speed up the computation of future scores further.

Given the method discussed above, we can estimate a future score based on each input hypothesis, and the final future score is estimated as the best of these hypothesis-dependent scores.

5.4 Dealing with leftover input words

At a certain point a path will reach the end, i.e., no more states can be generated from it according to the state expansion requirement. Then it is marked as a finished path. However, sometimes the state may contain a few input words that have not been visited. An example of this situation is the second state in Figure 3d). The word *SUV* in the third input hypothesis is left unvisited and it cannot be selected next because there is no adjacent state that can be generated. For such cases, we need to compute an extra score of covering these leftover words. Our approach is to create a state that produces the same output translation, but also covers all remaining words. For each leftover word, we create a pseudo CS that contains just that word plus ε's from all other hypotheses, and let it output ε. Moreover, that CS is inserted at a place such that no extra distortion cost is incurred. Figure 4 shows an example using the second state in Figure 3d). The last two words from the first two MT hypotheses "*the Jeep*" and "*the SUV*" align to the third and fifth words of the third hypothesis "*the Jeep*"; the word $w_{3,4}$ from the third hypothesis is left unvisited. The original path has two CS and one left-over word $w_{3,4}$. It is expanded to have three CS, with a pseudo CS inserted between the two CS.

It is worth noting that the new inserted pseudo CS will not affect the word count feature and contextually dependent feature scores such as the LM and bi-gram voting, since it only generates an empty word. Moreover, it will not affect the distortion score either. For example, as shown in Figure 4, the distortion cost of jumping from word $w_{2,3}$ to ε_2 and then to $w_{2,4}$ is the same as the cost of jumping from $w_{2,3}$ to $w_{2,4}$ given the way we assign position to empty word and the fact that the distortion cost is proportional to the difference between word positions.

Scores of other features for this pseudo CS such as word posterior (of ε), alignment score, CS entropy, and CS count are all local scores and can be computed easily. Unlike future scores which are approximate, the score computed in this process is exact. Adding this extra score to the existing score accumulated in the final state gives the complete score of this finished path. When all paths are finished, the one with the best complete score is returned as the final output sentence.

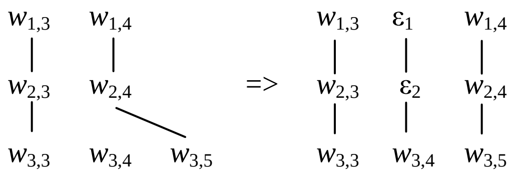

Figure 4. Expanding a leftover word to a pseudo correspondence set.

6 Evaluation

6.1 Experimental conditions

For the joint decoding method, the threshold for alignment-score-based pruning is set to 0.25 and the maximum number of words that can align to the same word is limited to 3. We call this the *standard setting*. The joint decoding approach is evaluated on the Chinese-to-English (C2E) test set of the 2008 NIST Open MT Evaluation (NIST 2008). Results are reported in case insensitive BLEU score in percentages (Papineni et. al., 2002).

The NIST MT08 C2E test set contains 691 and 666 sentences of data from two genres, newswire and web-data, respectively. Each test sentence has four references provided by human translators. Individual systems in our experiments belong to the official submissions of the MT08 C2E constraint-training track. Each submission provides 1-best translation of the whole test set. In order to train feature weights, the original test set is divided into two parts, called the dev and test set, respectively. The dev set consists of the first half of both newswire and web-data, and the test set consists of the second half of data of both genres.

There are 20 individual systems available. We ranked them by their BLEU score results on the dev set and picked the top five systems, excluding systems ranked 5th and 6th since they are subsets of the first entry (NIST 2008). Performance of these systems on the dev and test sets is shown in Table 1.

The baselines include a pair-wise hypothesis alignment approach using the indirect HMM (IHMM) proposed by He et al. (2008), and an incremental hypothesis alignment approach using the incremental HMM (IncHMM) proposed by Li et al. (2009). The lexical translation model used to compute the semantic similarity is estimated from two million parallel sentence-pairs selected from the training corpus of MT08. The backbone for the IHMM-based approach is selected based on Minimum Bayes Risk (MBR) using a BLEU-based loss function. The various parameters of the IHMM and the IncHMM are tuned on the dev set. The same IHMM is used to compute the alignment feature score for the joint decoding approach.

The final combination output can be obtained by decoding the CN with a set of features. The features used for the baseline systems are the same as the features used by the joint decoding approach. Some of these features are constant across decoding hypotheses and can be ignored. The non-constant features are word posterior, bi-gram voting, language model score, and word count. They are computed in the same way as for the joint decoding approach.

System weights and feature weights are trained together using Powell's search for the IHMM-based approach. Then the same system weights are applied to both IncHMM and Joint Decoding -based approaches, and the feature weights of them are trained using the max-BLEU training method proposed by Och (2003) and refined by Moore and Quirk (2008).

Table 1: Performance of individual systems on the dev and test set

System ID	dev	test
System A	32.88	31.81
System B	32.82	32.03
System C	32.16	31.87
System D	31.40	31.32
System E	27.44	27.67

6.2 Comparison against baselines

Table 2 lists the BLEU scores achieved by the two baselines and the joint decoding approach. Both baselines surpass the best individual system significantly. However, the gain of incremental HMM over IHMM is smaller than that reported in Li et al. (2009). One possible reason of such discrepancy could be that fewer hypotheses are used for combination in this experiment compared to that of Li et al. (2009), so the performance difference between them is narrowed accordingly. Despite that, the proposed joint decoding method outperforms both IHMM and IncHMM baselines significantly.

Table 2: Comparison between the joint decoding approach and the two baselines

method	dev	test
IHMM	36.91	35.85
IncHMM	37.32	36.38
Joint Decoding	37.94	37.20*

* The gains of Joint Decoding over IHMM and IncHMM are both with a statistical significance level > 99%, measured based on the paired bootstrap re-sampling method (Koehn 2004a)

6.3 Comparison of alignment pruning

The effect of alignment pruning is also studied. We tested with limiting the allowable links to just those that in the union of bi-directional Viterbi alignments.

The results are presented in Table 3. Compared to the standard setting, allowing only links in the union of the bi-directional Viterbi alignments causes slight performance degradation. On the other hand, it still outperforms the IHMM baseline by a fair margin. This is because the joint decoding approach is effectively resolving the ambiguous 1-to-many alignments and deciding proper places to insert empty words during decoding.

Table 3: Comparison between different settings of alignment pruning

Setting	Test
standard settings	37.20
union of Viterbi	36.88

6.4 Comparison of ordering constraints

In order to investigate the effect of allowing flexible word ordering, we conducted experiments using different constraints on the ordering of CS in the decoding process. In the first case, we restrict the order of CS to follow the word order of a backbone, which is one of the input hypotheses selected by MBR-BLEU. In the second case, the order of CS is constrained to follow the word order of at least one of the input hypotheses. As shown in Table 4, in comparison to the standard setting that allows backbone-free word ordering, the constrained settings did not lead to significant performance degradation. This indicates that most of the gain due to the joint decoding approach comes from the joint optimization of alignment and word selection. It is possible, though, that if we lift the CS *adjacency* constraint during search, we might derive more benefit from flexible word ordering.

Table 4: Effect of ordering constraints

Setting	test
standard settings	37.20
monotone w.r.t. backbone	37.22
monotone w.r.t. any hyp.	37.12

7 Discussion

This paper proposed a joint optimization approach for word-level combination of translation hypotheses from multiple machine translation systems. Unlike conventional confusion-network-based methods, alignments between words from different hypotheses are not pre-determined and flexible word orderings are allowed. Decisions on word alignment between hypotheses, word ordering, and the lexical choice of the final output are made jointly according to a set of features in the decoding process. A new set of features to model alignment and re-ordering behavior is also proposed. The method is evaluated against state-of-the-art baselines on the NIST MT08 C2E task. The joint decoding approach is shown to outperform baselines significantly.

Because of the complexity of search, a challenge for our approach is combining a large number of input hypotheses. When N-best hypotheses from the same system are added, it is possible to pre-compute and fix the one-to-one word alignment among the same-system hypotheses; such pre-computation is reasonable given our observation that the disagreement among hypotheses from different systems is larger than that among hypotheses from the same system. This will reduce the alignment search space to be the same as that for 1-best case. We plan to study this setting in future work.

To further improve the performance of our approach we see the biggest opportunity in developing better estimates of future scores and incorporating additional features. Beside potential performance improvement, they may help on more effective pruning and speed up the overall decoding process as well.

References

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proceedings of IEEE ASRU*.

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 *Proceedings of EMNLP*.

Shyamsundar Jayaraman and Alon Lavie. 2005. Multi-engine machine translation guided by explicit word matching. In *Proceedings of EAMT*.

Damianos Karakos, Jason Eisner, Sanjeev Khudanpur, and Markus Dreyer 2008. Machine Translation System Combination using ITG-based Alignments. In *Proceedings of ACL*.

Philipp Koehn, 2004a, Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of EMNLP*.

Philipp Koehn. 2004b. Pharaoh: A Beam Search Decoder For Phrase Based Statistical Machine Translation Models. In *Proceedings of AMTA*.

Chi-Ho Li, Xiaodong He, Yupeng Liu and Ning Xi, 2009. Incremental HMM Alignment for MT System Combination. In *Proceedings of ACL.*

Percy Liang, Ben Taskar, and Dan Klein. 2006. Personal Communication

Evgeny Matusov, Nicola Ueffing and Hermann Ney. 2006. Computing Consensus Translation from Multiple Machine Translation Systems using Enhanced Hypothesis Alignment. In *Proceedings of EACL*.

Evgeny Matusov, Gregor Leusch, Rafael E. Banchs, Nicola Bertoldi, Daniel Déchelotte, Marcello Federico, Muntsin Kolss, Young-Suk Lee, José B. Mariño, Matthias Paulik, Salim Roukos, Holger Schwenk, and Hermann Ney. 2008. System combination for machine translation of spoken and written language. *IEEE transactions on audio speech and language processing* 16(7).

Robert C. Moore and Chris Quirk. 2008. Random Restarts in Minimum Error Rate Training for Statistical Machine Translation, In *Proceedings of COLING*

Robert C. Moore. 2004. Improving IBM Word Alignment Model 1, In *Proceedings of ACL.*

NIST 2008. The NIST Open Machine Translation Evaluation.www.nist.gov/speech/tests/mt/2008/doc/

Franz J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward and Wei- Jing Zhu 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL.*

Antti-Veikko I. Rosti, Bing Xiang, Spyros Matsoukas, Richard Schwartz, Necip Fazil Ayan, and Bonnie J. Dorr. 2007a. Combining outputs from multiple machine translation systems. In *Proceedings of NAACL-HLT*.

Antti-Veikko I. Rosti, Spyros Matsoukas, and Richard Schwartz 2007b. Improved Word-level System Combination for Machine Translation. In *Proceedings of ACL*.

Antti-Veikko I. Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz 2008. Incremental Hypothesis Alignment for Building Confusion Networks with Application to Machine Translation System Combination. In *Proceedings of the 3rd ACL Workshop on SMT*.

Yong Zhao and Xiaodong He, 2009. Using N-gram based Features for Machine Translation System Combination. In *Proceedings of NAACL-HLT*