System Combination Using Discriminative Cross-Adaptation

Jacob Devlin and **Antti-Veikko I. Rosti** and
Shankar Ananthakrishnan and **Spyros Matsoukas** *
Raytheon BBN Technologies, 10 Moulton St, Cambridge, MA 02138, USA
{jdevlin,arosti,sanantha,smatsouk}@bbn.com

Abstract

Cross-adaptation (CA) based methods of machine translation (MT) system combination work by adapting the decoding step of a baseline system using information from alternate systems. Generally, the required information is very deep, such as a full decoding forest. In this paper, we describe a method of cross-adaptation based system combination which only requires the final output from each alternate system. This is achieved by adding a discriminatively weighted n-gram confidence feature to our decoder. In order to optimize the confidence weight of each system, we present a novel procedure called *non-linear Expected-BLEU optimization* that can be used to optimize arbitrary non-linear parameters for any decoding feature. We also describe a method for explicitly creating an adapted system that is *dissimilar* from each particular input system, which we have found to be useful in combination. Although our new method does not outperform a state-of-the-art confusion network (CN) based combination system on its own, we obtain statistically significant gains of 0.21-0.45 BLEU when the CA output is used as an additional system in CN combination.

This work was supported by DARPA/I2O Contract No. HR0011-06-C-0022 under the GALE program (Approved for Public Release, Distribution Unlimited). The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

1 Introduction

In general terms, *system combination* is the task of using multiple machine translation (MT) systems to produce an output that is better than any input system could produce by itself. The method for doing this is largely shaped by the information available: some require an actual MT decoder for each system (Li et al., 2009), some require a full forest of derivations (DeNero et al., 2010), and some only require the final output translations from each system (Rosti et al., 2009). Our research was done as part of a project where multiple sites develop their decoding systems independently, and then these outputs are combined together to produce a "team" output. Because these systems widely differ in structure, the only information we have available for system combination is an n-best list of hypotheses. No source-side correspondences are available, and some systems are only capable of producing a 1-best list.

Additionally, in this procedure, we assume that a strong baseline decoding system is available for each condition where system combination is performed. The general principle behind this procedure is to run a baseline decoding system with the output *adapted* towards the output of the other systems. In this case, we perform the adaptation by effectively increasing the probability of language model n-grams that are seen in other systems' outputs, as well as including translation rules extracted from the other systems' outputs. The relative weight of each system is estimated discriminatively using a novel extension to the Expected-BLEU optimization procedure which allows for the tuning of non-linear feature weights.

Currently, we use a state-of-the-art confusion network based procedure for system combination

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 667–675,
Chiang Mai, Thailand, November 8 – 13, 2011.

based on (Rosti et al., 2009). Although our new method *does* produce a significant gain over the best single system, it does not perform as well as our confusion network decoding on any condition that we tried. However, this new procedure was not designed to replace our existing method, but rather to complement it. When we used the output of the adaptation-based combination as an additional system in our standard confusion network decoding, we obtained a moderate gain in Arabic web, and a smaller gain on Arabic newswire.

2 Related Work

Existing research in MT system combination can generally be divided into two major approaches: confusion networks and cross-adaptation.[1] Confusion network based methods align the various input hypotheses against one another to form the confusion network, and then generate the most likely path through this network to produce a combined 1-best (Bangalore et al., 2001). However, as mentioned previously, we already have state-of-the-art confusion network system in place, which is based on (Rosti et al., 2007; Rosti et al., 2009). It is known that combining the outputs of multiple system combination procedures can produce further gains. Therefore, our goal was to design a *complementary* combination system which could be used in conjunction with our current system to produce better results than either could independently.

In automated speech recognition (ASR), generalized forms of adaptation have widely been used for a number of years (Rozzi and Stern, 1991) and more recently, cross-adaptation has been used as a method of ASR system combination (Stüker et al., 2006; Gales et al., 2007). In machine translation, a number of cross-adaptation based combination methods have been developed under a multitude of names. However, nearly all of these methods require rich information about each system which is not available for our task. Collaborative decoding (Li et al., 2009) requires that each system's decoder be run multiple times in an iterative fashion. DeNero's model combination (DeNero et al., 2010) requires a full decoding forest from each system. Joint optimization (He and Toutanova, 2009) and hybrid decoding (Cui et al., 2010) require that the input models[2] for each system be available to the combination decoder.

Crego et al. (2010) describes a method for LM-based combination which is most similar to the work presented in this paper, but their method does not discriminatively estimate weights for an arbitrary number of input systems.[3] In fact, they only present results where their main system is adapted using the output of a *single* other systems. By contrast, we present results where our main system is adapted by 7-14 other systems, and the weight for each of these is estimated discriminatively. As far as we know, there has been no previous work in developing a method of cross-adaptation which (a) requires only the final output from each system, and (b) discriminatively estimates the adaptation weight for each system. The latter is highly desirable when combining a large number of systems of varying quality, as with our task.

In order to discriminatively estimate the adaptation weight of each input system, we use a highly-scalable MT-specific optimization procedure called *Expected-BLEU* (Devlin, 2009; Rosti et al., 2010). The Expected-BLEU procedure is very similar to the Co-BLEU metric (Pauls et al., 2009), and is more distantly related to earlier work such as Tromble's linear approximation of BLEU (Tromble et al., 2008) and Smith's minimum risk annealing (Smith and Eisner, 2006). In this paper, we describe a novel extension to Expected-BLEU that allows for the optimization of arbitrary non-linear feature parameters. This allows for any feature parameters that are differentiable with respect to the feature score to be optimized alongside the normal log-linear decoding weights during the standard optimization procedure.

3 Description of MT System

Our baseline machine translation system, which is also used to perform the cross-adaptation, is a state-of-the-art hierarchical decoder based on (Shen et al., 2008) and (Chiang, 2007). Bottom-up chart parsing is performed to produce a shared forest of derivations, and possible path through the

[1] There is a third approach, known as "hypothesis selection," which simply selects the best input hypotheses as the "combined" output, based on some features (Hildebrand and Vogel, 2008). However, there has been comparatively little research on this method due to its simplicity and lack of room for improvement.

[2] Such as the set of translation rules and the language model.

[3] The *adaptation weight* represents how much each system should influence the adaptation process.

forest defines one hypothesis h.[4] The decoder uses a log-linear translation model, so the score of hypothesis h is defined as:

$$S_h(\vec{w}) = \sum_{i=1}^{m} w_i \sum_{r \in R(h)} F_{ri} \quad (1)$$

where $R(h)$ is the set of translation rules that make up hypothesis h, m is the number of features, F_{ri} is the score of the i^{th} feature in rule r, and w_i is the weight of feature i. This weight vector is optimized discriminatively to maximize BLEU score on a tuning set, using the Expected-BLEU optimization procedure.

Our decoder uses all of the standard statistical MT features, such as:

- $P(T|S)$ = forward rule translation
- $P(S|T)$ = backward rule translation
- $LS(T|S)$ = lexical smoothing
- $P(q_j|q_{j-1}, ...)$ = language model

Additionally, we use approximately 50,000 sparse, binary-valued features which model specific events such as "Is the bigram 'united states' seen in the target side of the translation rule?" These features do cause some amount of overfitting on the tuning set, but we have not found this to be harmful to the test sets. However, this also causes a certain amount of variability on the tune set results, so minor variations in BLEU score on the tuning set from condition to condition are to be expected.

The cross adaptation models described below are used as standard log-linear feature scores, and the weights are optimized jointly with the normal decoding features.

4 Discriminative Model Adaptation

In this section, we describe how the cross-adaptation is actually implemented in our system. We perform the adaptation by defining additional decoding features which affect both the language model and the translation model, in order to make the MT output appear more like the output of the other systems.

4.1 Discriminative Language Model Adaptation

We perform language model adaptation by effectively increasing the language model probability of n-grams that are seen in the other systems' outputs. Since we use a 3-gram decoder, we adapt the probability of all 1-grams, 2-grams, and 3-grams.

In the past, Snover et al. (2008) performed language model adaptation using simple linear interpolation:

$$P(q_j|q_{j-1}, ...) = \sum_{i=1}^{K} v_i P_i(q_j|q_{j-1}, ...) \quad (2)$$

where K is the number of models, $q = (q_j, q_{j-1}, ...)$ is the n-gram in question, and v_i is the weight for model i, constrained so that $\sum_i v_i = 1$. This formula replaces the standard language model probability in the log-linear decoding model. When this feature was used by Snover, it combined the standard language model with a test-sentence-specific language model that was trained on several hundred documents. However, in the case of system combination, we are adapting towards a much smaller amount of data, so $P(q_j|q_{j-1}, ..., q_{j-n+1})$ will not be well estimated. Therefore, we use a modified formula which uses a binary function $\delta(q)$ to indicate the presence or absence of the n-gram q. This formula is used as an additional feature in the log-linear decoding model:[5]

$$F(q, \vec{v}) = \log(\epsilon + \frac{\sum_{i=1}^{K} e^{v_i} G_i(q)}{\epsilon + \sum_{i=1}^{K} e^{v_i}}) \quad (3)$$

$$G_i(q) = \frac{\sum_{j=1}^{H_i} b_{ij} \delta_{ij}(q)}{\sum_{j=1}^{H_i} b_{ij}} \quad (4)$$

where K is the number of systems, ϵ is a small positive value fixed at 10^{-8}, and v_i is the discriminatively estimated weight for system i.[6] If an n-gram is seen in exactly *zero* system outputs, it does *not* receive a score of $\log(\epsilon)$, but instead receives a score of $\log(1)$ and triggers an additional binary feature. This additional feature acts as a discriminative backoff log-probability for unseen n-grams, and generally optimizes to a large negative value in the range of -10 to -15.

The function $G_i(q)$ represents the "count" of n-gram q in system i's n-best list. In it, H_i is the number of hypotheses in system i, $\delta_{ij}(q)$ is a binary "occurrence function" that returns 1 if n-gram q is seen in hypothesis ij, and b_{ij} is a

[4] In this case, *hypothesis* refers to a specific path through the shared forest, rather than a specific output string.

[5] Therefore, the standard language model probability remains unchanged.

[6] ϵ is used to prevent $\log()$ underflow or divide-by-zero errors, since v_i is unbounded.

hypothesis-specific positive weighting which assigns mass to each hypothesis as a function of its rank. Obviously, if q is seen in every hypothesis of system i, $G_i(q) = 1$. Note that $\delta_{ij}(q)$ does not return the actual *count* of q, because we never want $G_i(q)$ or $F(q, v)$ to return a value over 1 (or $\log(1)$).

Theoretically, b_{ij} could be estimated discriminatively, but for simplicity we use the following formula which assigns mass to each hypothesis based on its rank in the n-best list:

$$b_{ij} = H_i - j + 1 \quad (5)$$

As an example, if system i has a 5-best output, and n-gram q is seen in hypotheses 1 and 4, then $G_i(q) = \frac{5+2}{5+4+3+2+1} = 0.467$.

Equation 3 can be thought of as the log-confidence that n-gram q is consistent with the other systems' outputs. Note that $F(q, \vec{v})$ is continuous and differentiable with respect to the weights $\vec{v}$, and $F(q, \vec{v})$ returns a value in the range $[\log(\epsilon), 0]$ for all values of $\vec{v}$. These properties make it possible to discriminatively estimate $\vec{v}$ within our standard optimization framework, which is discussed in Section 5.

4.2 Discriminative Translation Model Adaptation

We extract adapted translation rules using the method described in (Snover et al., 2008). Essentially, for each source phrase s, we consider every target phrase t from each of the input system hypotheses as a possible translation of s.[7] We limit the maximum length of s and t to 3 and 5, respectively. Clearly, this produces many extraneous rules. We prune these rules only keeping the top 20 most likely target translations of each s, sorted by the "noisy-or" lexical smoothing score between s and t (Zens and Ney, 2004). This lexical smoothing score is used as an additional decoding feature.

The formula for the discriminative adapted rule confidence is exactly the same as Equation 3:

$$H(r, \vec{z}) = \log(\epsilon + \frac{\sum_{i=1}^{K} e^{z_i} G_i(r)}{\epsilon + \sum_{i=1}^{K} e^{z_i}}) \quad (6)$$

$$G_i(r) = \frac{\sum_{j=1}^{H_i} b_{ij} \delta_{ij}(r)}{\sum_{j=1}^{H_i} b_{ij}} \quad (7)$$

where r is the adapted rule and $\vec{z}$ are the discriminative system weights. The "occurrence function" $\delta_{ij}(r)$ function returns 1 when the rule r's target phrase $\vec{t}$ is seen *anywhere* in hypothesis ij.

[7] This adaptation performed independently for each test sentence.

5 Parameter Optimization

In this section, we describe a novel procedure for discriminatively optimizing the non-linear parameters $\vec{v}$ and $\vec{z}$ used in Equations 3 and 6. These parameters are jointly optimized alongside the standard log-linear decoding weights to directly maximize BLEU score on a tuning set. In order to perform this optimization, we use a modified version of *maximum BLEU* tuning, called *Expected-BLEU* (Rosti et al., 2010). For the benefit of the reader, we will first give a brief overview of maximum BLEU and Expected-BLEU optimization. Afterwards, we will describe a novel extension to Expected-BLEU which allows for the optimization of non-linear feature parameters.

5.1 Expected-BLEU Optimization

Standard maximum BLEU optimization attempts to find the set of weights $\vec{w}$ that maximizes the 1-best BLEU score over an n-best list, with BLEU (Papineni et al., 2002) defined as:

$$BLEU(\vec{w}) = \left(\prod_{m=1}^{4} \frac{\sum_{i=1}^{N} c_{ib_i(\vec{w})}^{(m)}}{\sum_{i=1}^{N} t_{ib_i(\vec{w})}^{(m)}} \right)^{1/4} \cdot \theta \left(1 - \frac{\sum_i r_{ib_i(\vec{w})}}{\sum_{i=1}^{N} h_{ib_i(\vec{w})}} \right) \quad (8)$$

$$b_i(\vec{w}) = \text{argmax}_{h \in H_i}(S_h(\vec{w})) \quad (9)$$

where $\theta(x) = \min(1.0, e^x)$, N is the number of test sentences, b_i is the 1-best hypothesis of the i^{th} sentence selected using weights $\vec{w}$, and $\{c_{b_i}^{(1)}, c_{b_i}^{(2)}, ...h_{b_i}\}$ are the 10 pre-computed BLEU statistics for hypothesis b_i.[8] Crucially, the selection of 1-best hypotheses with respect to $\vec{w}$ is *discrete*, and therefore the max BLEU function is non-differentiable. Because of this, a line search algorithm such as Powell's method (Powell, 1964) must be used to optimize the weights, which does not perform well when more than few dozen weights are optimized simultaneously.

Expected-BLEU optimization seeks to approximate max BLEU using a continuous objective function. The advantage of this is that it quickly

[8] $c^{(m)}$ is the number of matching m-grams, $t^{(m)}$ is the number of total hypothesis m-grams, r is the reference length, h is the hypothesis length, and $\theta(x)$ is a "brevity penalty" which penalizes short hypotheses.

converges even when tens of thousands of weights are estimated simultaneously. Therefore, instead of discretely selecting a 1-best hypothesis from each of the m test sentences, the BLEU statistics are summed over *every* hypothesis weighted by its corresponding posterior probability. Note that we compute the expectation of each of the 10 BLEU statistics *independently*, so the Expected-BLEU formula is not technically equivalent to the "expected value of BLEU." However, in practice this has not been an issue.

The posterior probability for hypothesis ij is simply the normalized decoding probability:

$$p_{ij}(\vec{w}) = \frac{e^{\gamma S_{ij}(\vec{w})}}{\sum_{k=1}^{n} e^{\gamma S_{ik}(\vec{w})}} \quad (10)$$

The free parameter γ controls the shape of the distribution, with a higher γ shifting more mass towards the 1-best hypothesis.

As a final step, we replace Equation 8's "brevity penalty" $\theta(x) = \min(1, e^x)$ with a differentiable approximate over the range $0.9 \leq x \leq 1.1$. The function is defined as:

$$\phi(x) = \frac{e^x - 1}{e^{1000x} + 1} + 1 \quad (11)$$

The final Expected-BLEU objective function is then:

$$ExpBLEU(\vec{w}) = \left(\prod_{m=1}^{4} \frac{\sum_i \sum_j p_{ij} c_{ij}^{(m)}}{\sum_i \sum_j p_{ij} t_{ij}^{(m)}} \right)^{1/4} \cdot \phi\left(1 - \frac{\sum_i \sum_j p_{ij} r_{ij}}{\sum_i \sum_j p_{ij} h_{ij}}\right) \quad (12)$$

where p_{ij} is short for $p_{ij}(\vec{w})$, as defined in Equation 10. The values $\{c_{ij}^{(m)}, ...\}$ are the same precomputed BLEU statistics as in Equation 8.

The differentiation of this function with respect to $\vec{w}$ can be performed in a fairly small number of steps using basic calculus, the details of which are provided in (Devlin, 2009). The functions $ExpBLEU(\vec{w})$ and $\frac{dExpBLEU}{dw}$ are used with LBFGS (Liu and Nocedal, 1989) to perform n-best based parameter optimization. We use a standard iterative optimization procedure: (1) Decode tuning set with initial weights $\vec{w}$ and generate n-best list, (2) Optimize $\vec{w}$ on n-best list, (3) Repeat (1) and (2) until convergence, (4) Decode validation set.

In order to prevent over-fitting, we add a standard $L2$-norm regularization term to our objective function:

$$Obj(\vec{w}) = ExpBLEU(\vec{w}) - \alpha||\vec{w} - \vec{w}'||^2 \quad (13)$$

where $\vec{w}'$ is the initial weight vector at the current iteration of optimization, and α is the regularization term, fixed at 10^{-5}.

5.2 Non-Linear Feature Optimization

The previous section describes the optimization procedure for standard linear decoding weights, but it can also be extended to optimize non-linear weights, as in Equation 3 and Equation 6. We simply modify Equation 1 so that F_i is a function of parameters $\vec{v}_i$:

$$S_h(\vec{w}, \vec{v}) = \sum_{i=1}^{m} w_i \sum_{r \in R_h} F_{ri}(\vec{v}_i) \quad (14)$$

For each non-linear feature type, we must implement $\frac{dF_{ri}}{d\vec{v}_i}$. Then, we can "generically" compute $\frac{dExpBLEU}{dF_{ri}}$ inside of the optimizer and apply the chain rule to compute:

$$\frac{dExpBLEU}{d\vec{v}_i} = \sum_h \sum_{r \in R_h} \frac{dExpBLEU}{dF_{ri}} \frac{dF_{ri}}{d\vec{v}_i} \quad (15)$$

where R_h is the set of rules associated with hypothesis h. This information is usually not available to the optimizer, so this functionality must be added.

The non-linear weights $\vec{v}$ are jointly optimized with the standard decoding weights $\vec{w}$ using the function $ExpBLEU(\vec{w}, \vec{v})$.

5.3 Dissimilarity Optimization

Because the ultimate goal of the cross-adaptation combination is to use it as an additional system in confusion network decoding, it would be beneficial if we could ensure that it contains complementary information. In the past, we have seen that that system combination performs best when the input systems have similar performance but are very different from one another. We model this difference or *dissimilarity* between two sentences using the TER metric, which measures the normalized number of "edits" required to transform the one sentence into another (Snover et al., 2006). Normally, TER is computed on an MT hypothesis against its reference translation. In this case, we instead measured the TER of the cross-adaptation

output against the external input system hypotheses.[9]

We attempted to explicitly increase TER dissimilarity by discriminatively optimizing against the input system hypotheses.[10] In other words, we used the input hypotheses as reference translation and optimized in the *opposite* direction that we normally would. We added this as a linear term in our existing Expected-BLEU objective function. Note that Expected-BLEU is computed against the reference translations and Expected TER is computed against the external input systems:

$$\begin{aligned} Obj(\vec{w},\vec{v}) &= ExpBLEU_{Ref}(\vec{w},\vec{v}) \\ &+0.1{\cdot}ExpTER_{ExSys}(\vec{w},\vec{v}) \\ &-\alpha||\vec{w}-\vec{w}'||^2 \end{aligned} \quad (16)$$

The dissimilarity term is weighted at 0.1 in order to prevent the BLEU score from degrading by a significant amount. Note that it is not contradictory to adapt *towards* the input hypotheses while simultaneously optimizing *against* those same hypotheses, because the adaptation is done at the n-gram level and TER is computed on the sentence level. We want to use words and phrases from the input systems, but we don't want the final sentence to be too similar to any one particular input hypothesis.

6 Experiments

The input to our system combination procedure consists of output from 14 different machine translation systems developed independently at 5 different sites. Of these, 7 were "internal" systems developed at our site, while 7 were "external" systems developed at the 4 outside sites. The 7 internal systems all used the same hierarchical decoder and feature set described in Section 3, but varied by source tokenization and method of word alignment. The 7 external systems include a phrasal system, two hierarchical systems, a syntax system, a tree-to-string system, a string-to-tree system, and a hand-crafted rule based system. We will present results on Arabic-to-English web and newswire.

Our parallel training data and development sets consist of publicly available LDC/NIST data, as well as data specific to DARPA's Global Autonomous Language Exploitation (GALE) project. The publicly available training data consists of 3.3 million words of newswire/treebank LDC-released data as well as 118 million words of LDC-released UN data. The GALE-only training data consists of 46 million words of LDC-released data plus 30 million words released by Sakhr Software. The monolingual LM training consists of 4 billion words from the GigaWord corpus and 4 billion words from various other sources such as Google News and New York Times. We use a 3-gram LM for decoding and 5-gram LM for rescoring.

Our development sets were constructed using the NIST MT04, MT05, MT06, and MT08 data sets, as well as the GALE Phase1-Phase4 development/evaluation sets. We use one tuning set, referred to as "Tune," to optimize both the cross-adaptation and confusion network based systems. Our validation set is referred to as "Test." [11]

Our cross-adaptation system uses the features described in Section 4 to actually perform the adaptation, but is otherwise identical to our baseline system, referred to as "Best Single System" or "Internal Best."

Because we use a large number of discriminative features in our baseline MT system, there is a moderate-to-significant over-fitting effect when optimizing on any new set. However, in the past we have found that even a large amount of over-fitting (e.g., 3-4 BLEU points) on the tuning set does *not* have a negative affect on the test set results. Here, we see less than 1.0 BLEU over-fitting during cross-adaptation and less than 0.5 BLEU on the final combination, so we did not take any steps to mitigate it. The proper solution would likely be to use separate sets to optimize the cross-adaptation and the confusion network. The descriptions of the "Tune" and "Test" sets are shown in Table 1.

	Tune		*Test*	
	# sents	# refs	# sents	# refs
ara nw	5456	2.1	1986	1.4
ara web	5454	2.3	2276	2.4

Table 1: "Number of sentences" and "average number of references per sentence" for the development sets.

[9] The "external" input systems are those that were developed at outside sites. Since our internal system is used to perform the cross-adaptation, we do not perform dissimilarity optimization against its baseline output.

[10] Alternately, we could optimize against the confusion network baseline output, but we found that this did not perform well.

[11] The input systems were optimized on a third set, which is not used here.

Since it is not practical to present the scores on all input systems, Table 2 shows BLEU scores for the best internal system as well as the top three external systems. In both cases, the best internal system outperforms all of the external systems. It is interesting to note that the best internal system outperforms the top two external systems by a greater margin on Arabic web than Arabic newswire.

	ara nw		**ara web**	
	Tune	*Test*	*Tune*	*Test*
	BLEU	BLEU	BLEU	BLEU
Internal Best	48.48	45.00	39.77	41.44
External 1st	47.74	44.35	38.27	40.20
External 2nd	47.71	44.20	37.49	39.01
External 3rd	44.84	42.29	36.77	38.54

Table 2: Comparison of best internal system vs. top three external systems. Here, "Tune" is a valid test set, since none of the *input* systems were optimized on it.

One final detail to note is that on Arabic web we performed adaptation using all 14 input systems, while on Arabic newswire we only used the 7 external systems. The reason is that on newswire we encountered an optimization issue where the weights for the internal systems would receive a large value during the first few iterations of tuning, which would cause the optimization to converge at a sub-optimal local maximum. However, even when we manually finessed the optimization, we did not see a gain on cross-adaptation from using all 14 systems on newswire. Because the weights are estimated discriminatively, it should theoretically never be harmful to include additional systems,[12] so we plan to experiment with different types of regularization to solve this optimization issue. On Arabic web, this issue did not occur, so we were able to use all 14 systems without any "manual finessing."

6.1 Cross-Adaptation Results

Tables 3 and 4 show the effect of using the cross-adaptation features from Equations 3 and 6. We use separate weights for each n-gram order as well as the adapted rules, which results in $4K$ total weights, where K is the number of systems.[13]

On Arabic web, our optimized cross-adaptation system (*CA*) gains 3.1 BLEU over our best single system (*BSS*), and gets within 0.4 BLEU of our confusion network baseline (*CN*). When the cross-adaptation output is used as an additional system during confusion network decoding, we see a gain of 0.37 BLEU (*CN w/ CA*). Using the dissimilarity cross-adaptation as a second additional system helps slightly more, bringing the total gain to 0.45 BLEU (*CN w/ CA+DCA*). In both cases, the gain is statistically significant.

On Arabic newswire, the optimized cross-adaptation system gains 2.3 BLEU over the best single system, but performs 0.8 BLEU worse than our confusion network baseline. Using the cross-adaptation output as an additional system yields no gain, while using the dissimilarity optimized cross-adaptation output as an additional system yields a minor gain of 0.2 BLEU. However, the gain on BLEU is statistically significant.

We also provide the results on two additional test conditions for Arabic web. The condition *BSS*

	ara web			
	Tune		*Test*	
	BLEU	TER	BLEU	TER
BSS	39.77	47.81	41.44	46.69
BSS w/ "Tune"	42.41	46.39	41.69	46.61
CA, No Opt	43.32	45.55	43.95	44.75
CA	43.52	45.27	44.58	44.49
DCA	41.88	46.44	43.15	45.48
CN	43.10	45.52	45.00	44.56
CN w/ CA	43.70	45.20	45.37*	44.46
CN w/ DCA	43.47	45.27	45.12	44.27*
CN w/ CA+DCA	43.76	45.23	45.45*	44.36*

Table 3: Combination results on Arabic web using 14 input system. * indicates that the system is significantly better than *CN* using a 95% confidence interval, as defined in (Koehn, 2004). Significance is only shown on the Test set. **BSS** = Best single system. **BSS w/ "Tune"** = Tuning on "Tune" using only standard features, instead of the normal decoding tuning set. **CN** = Confusion network baseline. **CA, No Opt** = Cross-adaptation, fixing all of the non-linear system weight to $a_i = 0$, but the standard linear feature weights optimized as normal. **CA** = Cross-adaptation, allowing the non-linear system weights to optimize. **CN w/ CA** = Using the output of *CA* as an additional input in *CN*. **DCA** = Cross adaptation with dissimilarity optimization. **CN w/ DCA** = Using the output of *DCA* as an additional input in *CN*. **CN w/ CA+DCA** = Using the output of both *CA* and *DCA* as an additional input in *CN*.

[12] It should never harm the results on the tuning set, although it could be harmful on the test set.

[13] We estimate a separate set of weights for (1) unigrams, (2) bigrams, (3) trigrams, and (4) adapted rules.

	ara nw			
	Tune		*Test*	
	BLEU	TER	BLEU	TER
BSS	48.48	38.59	45.00	38.51
CA	51.54	36.37	47.30	36.72
DCA	49.94	37.33	46.27	37.35
CN	51.67	35.87	48.07	35.87
CN w/ CA	52.03	35.91	48.05	35.87
CN w/ DCA	51.81	35.95	48.27*	35.77
CN w/ CA+DCA	52.06	35.84	48.28*	35.71

Table 4: Combination results on Arabic newswire using 7 input systems. Conditions have same meaning as in Table 3.

w/ "Tune" shows the results of optimizing on the system combination tuning set, as opposed to the standard tuning set which all of the input systems were optimized on. We can see that the gain on "Test" is very small, meaning that this difference was not an issue. For the condition *CA, No Opt* we set all of the adaptation weights to a fixed value of $a_i = 0$, so all systems receive an equal "vote" in the adaptation features.[14] As expected, this has a detrimental effect on the results, losing 0.6 BLEU compared to *CA*.

6.2 Dissimilarity Optimization Results

The previous tables demonstrate that although dissimilarity optimization performs worse than standard cross-adaptation, it is still beneficial to use it as an additional system in the confusion network decoding. Table 5 shows how dissimilar the *DCA* output is from the input systems compared to *CA*. The *DisTER* score is computed on the MT output of each condition against the 7 external input systems. We see that on newswire the *DCA* output is 4.5 TER points more dissimilar than the *CA* output, while on web it is 3.1 TER more dissimilar. At the same time, both *DCA* conditions gains 1.0-1.5 BLEU points over the best single system.

	ara nw		ara web	
	Test		*Test*	
	BLEU	DisTER	BLEU	DisTER
BSS	45.00	23.16	41.44	29.65
CA	47.30	15.63	44.58	22.38
DCA	46.27	20.17	43.15	25.50

Table 5: **BSS** = Best single system. **CA** = Cross-adaptation. **DCA** = Cross-adaptation with dissimilarity optimization. **BLEU** is computed against the reference translations, while **DisTER** is computed against the input systems.

[14] Recall that the true weight is e^{a_i}

7 Conclusions and Future Work

In this paper, we presented a novel method of cross-adaptation based system combination which obtains statistically significant BLEU gains over best single system. The advantages of this method are that it can be implemented using only simple decoding features, and that it requires just an n-best list from the input systems, as opposed to alternate cross-adaptation methods that require deeper information. Although this new method does not perform as well as our existing confusion network based combination, we showed that it is beneficial when used as additional system in the confusion network decoding.

We also showed that it is possible to explicitly create a system with complementary information by using *dissimilarity optimization*, where the TER score between the cross adaptation output and the input systems is used as part of the optimization objective function. Although this method of optimization degrades the BLEU score compared to standard cross-adaptation, we showed that it is useful to use this output as a second additional system during confusion network decoding.

In the future, we plan to use the dissimilarity optimization procedure to produce multiple input systems which are explicitly optimized to be different from one another. We already know that it is beneficial to combine multiple systems that use the same decoder/feature set but vary by tokenization/alignment/etc. If we can discriminatively optimize these systems so that they have higher pair wise TER scores without harming their BLEU scores, it may be possible to obtain a larger gain during combination.

Additionally, we presented a highly-scalable, robust method for optimizing arbitrary non-linear feature parameters alongside the standard log-linear decoding weights. We have already used this method to explore many types of new features, such as using a neural net based language model and discriminatively optimizing sentence-level confidence weights on the training data. We plan to further refine our optimization procedure to use additional regularization and normalization, so that very high-dimensional non-linear feature sets can be used without any issues.

References

S. Bangalore, G. Bordel, and G. Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *ASRU*, pages 351–354.

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

J.M. Crego, A. Max, and F. Yvon. 2010. Local lexical adaptation in machine translation through triangulation: SMT helping SMT. In *COLING*, pages 232–240.

L. Cui, D. Zhang, M. Li, M. Zhou, and T. Zhao. 2010. Hybrid decoding: decoding with partial hypotheses combination over multiple SMT systems. In *COLING*, pages 214–222.

J. DeNero, S. Kumar, C. Chelba, and F. Och. 2010. Model combination for machine translation. In *NAACL-HLT*, pages 975–983.

J. Devlin. 2009. Lexical features for statistical machine translation. Master's thesis, University of Maryland.

M.J.F. Gales, X. Liu, R. Sinha, P.C. Woodland, K. Yu, S. Matsoukas, T. Ng, K. Nguyen, L. Nguyen, J.-L. Gauvain, L. Lamel, and A. Messaoudi. 2007. Speech recognition system combination for machine translation. In *ICASSP*, pages 1277–1280.

X. He and K. Toutanova. 2009. Joint optimization for machine translation system combination. In *EMNLP*, pages 1202–1211.

A.S. Hildebrand and S. Vogel. 2008. Combination of machine translation systems via hypothesis selection from combined n-best lists. In *AMTA*, pages 254–261.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395.

M. Li, N. Duan, D. Zhang, C.-H. Li, and M. Zhou. 2009. Collaborative decoding: partial hypothesis re-ranking using translation consensus between decoders. In *ACL-IJCNLP*, pages 585–592.

D.C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

A. Pauls, J. DeNero, and D. Klein. 2009. Consensus training for consensus decoding in machine translation. In *EMNLP*, pages 1418–1427.

M.J.D. Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 7(2):155–162.

A.-V.I. Rosti, S. Matsoukas, and R. Schwartz. 2007. Improved word-level system combination for machine translation. In *ACL*, pages 312–319.

A.-V.I. Rosti, B. Zhang, S. Matsoukas, and R. Schwartz. 2009. Incremental hypothesis alignment with flexible matching for building confusion networks: BBN system description for WMT09 system combination task. In *WMT*, pages 61–65.

A.-V.I. Rosti, B. Zhang, S. Matsoukas, and R. Schwartz. 2010. BBN system description for WMT10 system combination task. In *WMT/MetricsMATR*, pages 321–326.

W.A. Rozzi and R.M. Stern. 1991. Speaker adaptation in continuous speech recognition via estimation of correlated mean vectors. In *ICASSP*, pages 865–868.

L. Shen, J. Xu, and R. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *ACL-HLT*, pages 577–585.

D.A. Smith and J. Eisner. 2006. Minimum risk annealing for training log-linear models. In *ACL-COLING*, pages 787–794.

M. Snover, B. Dorr, R. Schwartz, L. Micciula, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*, pages 223–231.

M. Snover, B. Dorr, and R. Schwartz. 2008. Language and translation model adaptation using comparable corpora. In *EMNLP*, pages 857–866.

S. Stüker, C. Fügen, S. Burger, and M. Wölfel. 2006. Cross-system adaptation and combination for continuous speech recognition. In *INTERSPEECH*, pages 521–524.

R. Tromble, S. Kumar, F. Och, and W. Macherey. 2008. Lattice minimum Bayes-risk decoding for statistical machine translation. In *EMNLP*, pages 620–629.

R. Zens and H. Ney. 2004. Improvements in phrase-based statistical machine translation. In *HLT-NAACL*, pages 257–264.