The MIT-LL/AFRL IWSLT-2007 MT System

Wade Shen, Brian Delaney†

MIT Lincoln Laboratory
Information Systems and Technology Group
244 Wood St.
Lexington, MA 02420, USA
{swade,bdelaney}@ll.mit.edu

Tim Anderson, Ray Slyh

Air Force Research Laboratory
Human Effectiveness Directorate
2255 H St.
Wright-Patterson AFB, OH 45433
{Timothy.Anderson,Raymond.Slyh}@wpafb.af.mil

Abstract

The MIT-LL/AFRL MT system implements a standard phrase-based, statistical translation model. It incorporates a number of extensions that improve performance for speech-based translation. During this evaluation our efforts focused on the rapid porting of our SMT system to a new language (Arabic) and novel approaches to translation from speech input.

This paper discusses the architecture of the MIT-LL/AFRL MT system, improvements over our 2006 system, and experiments we ran during the IWSLT-2007 evaluation. Specifically, we focus on 1) experiments comparing the performance of confusion network decoding and direct lattice decoding techniques for machine translation of speech, 2) the application of lightweight morphology for Arabic MT preprocessing and 3) improved confusion network decoding.

1. Introduction

During the evaluation campaign for the 2007 International Workshop on Spoken Language Translation (IWSLT-2007) our experimental efforts centered on 1) the improvement of statistical MT when processing speech input and 2) robust methods for coping with minimal training data.

In this paper we describe improvements over our baseline system. Refer to [5] for more detail regarding the implementation of these stages of processing. The conditions of this year's evaluation have changed as have the data sets and languages for which we've submitted system. As such, the processing stages described in 2 have been adapted accordingly.

The remainder of this paper is structured as follows. In Section 2, we discuss the baseline system and minor improvements to this standard statistical MT architecture that we incorporate. In sections 3, 4 and 5 we describe improvements to our baseline system including the design and implementation of a new, lattice-based decoder for speech input and the implementation of light morphological analysis for Arabic MT preprocessing.

Section 6 shows results from a variety of development experiments using the provided data sets. In these experiments, we compare the performance of different ASR-based MT strategies using the same baseline system on both the Arabic and Italian speech translation tasks. We also report on experiments comparing the light Arabic morphological processing we employed to more sophisticated methods and their relative performance impact on MT quality.

1.1. IWSLT-2007 Data Usage

We submitted systems for Chinese, Arabic and Italian-to-English language pairs. In each case, we used only data supplied by the evaluation (and in Italian, the named-entity list) for each language pair for training and optimization. From these data, we extract word/character alignments. These alignments are then expanded using slightly modified versions of standard heuristics. This process is described in detail in Section 2. Phrases are then extracted and counted, and the resulting phrase table is then used for decoding and rescoring. Language models are trained using the English side of each language pair.

Using development bitexts separated from the training set, we then employ a minimum error rate training process to optimize model parameters with a held-out development set. These trained parameters and models can then be applied to test data during decoding and rescoring phases of the translation process.

2. Baseline System

Our baseline system implements a fairly standard SMT architecture allowing for training of a variety of word alignment types and rescoring models. It has been applied successfully to a number of different translation tasks in prior work, including prior IWSLT evaluations. The training/decoding procedure for our system is outlined in Table 1. Details of this system are described in [5].

†This work is sponsored by the Air Force Research Laboratory under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Training Process
• Word and character segment training corpus • Compute GIZA++ and Competitive Linking Alignments (CLA) for segmented data [6] [7] • Extract phrases for all variants of the training corpus • Split word-segmented phrases into characters • Combine phrase counts and normalize • Train language models from the training corpus • Train TrueCase models • Train source language repunctuation models
Decoding/Rescoring Process
• Decode input sentences use base models • Add rescoring features (e.g. IBM model-1 score, etc.) • Merge n-best lists (if input is ASR n-best) • Rerank n-best list entries

Table 1: *Training/decoding structure*

2.1. Phrase Table Training

To maximize phrase table coverage we combine multiple word and character alignment strategies, extending the method described in [6]. For all language pairs, we combine alignments from IBM model 5 (see [1] and [8]) and alignments extracted using the competitive linking algorithm (CLA) described in [7]. Phrases were extracted from both types of alignments and combined in one phrase table. This was done by summing counts of phrases extracted from alignment types before computing the relative frequency used in the our phrase tables.

Additionally, for Chinese-to-English translation, both word and character segmentation were used for training CLA and GIZA alignment models. Phrases were then extracted from all four alignments and combined. Word segmented phrases were resegmented into characters before counting.

2.2. Language Model Training

During the training process we built n-gram language models for use in decoding/rescoring, TrueCasing and repunctuation. In all cases, the SRI Language Modeling Toolkit [9] was used to create interpolated Knesser-Ney LMs. Additional class-based language model were also trained for rescoring. All models were trained with the English side of IWSLT parallel texts that were supplied.

2.3. Optimization and Rescoring

Our translation model assumes a log-linear combination phrase translation models, language models, etc.

$$\log P(\vec{e}|\vec{f}) \propto \sum_{\forall r} \lambda_r h_r(\vec{f}, \vec{e})$$

To optimize system performance we training scaling factors, λ_r, for both decoding and rescoring features so as to minimize an objective error criterion. This is done using a standard Powell-like grid search using a development set [2] [3]. In this year's evaluation, we used an error criterion that combines NIST and BLEU scores:

$$Error(s) = k - (100.0BLEU(s) + 1.0NIST(s))$$

A full list of the independent model parameters that we used in our system is shown in Table 2.

Decoding Features
$P(f\|e)$
$P(e\|f)$
$LexW(f\|e)$
$LexW(e\|f)$
Phrase Penalty
Lexical Backoff
Word Penalty
Distortion
$\hat{P}(e)$ – 4-gram language model
Rescoring Features
$\hat{P}_{rescore}(e)$ – 5-gram LM
$\hat{P}_{class}(e)$ – 6-gram class-based LM
$P_{Model1}(f\|e)$ – IBM model 1 translation probabilities

Table 2: *Independent models used in log-linear combination*

3. Direct ASR Lattice Decoding

In this section, we describe our weighted finite state transducer (FST) decoder. Finite state transducers provide a useful framework for natural language processing applications as the implementation details of graph optimization and search are handled through a software library that operates on a common state machine representation [11]. However, using FST technology for machine translation does present some technical hurdles. The FST composition operation combines individual model scores such that it is difficult to extract these scores for weight optimization. Additionally, the inputs paths to intermediate models can sometimes be lost during composition, making them difficult to retrieve later. For example, after the application of all necessary models for translation, it is possible to retrieve either the unordered input words/phrases or the re-ordered input words/phrases but not both. Finally, distortion must be limited as the total number of combinations allowed must be computed up front.

Finite state transducers have been used successfully in machine translation applications, and our decoder is based on some of these systems. We use the MIT FST toolkit in our implementation [12]. In [14], a phrased-based FST system was shown to perform better than a word based system. In [13], the authors implemented a phrase-based FST translation system. The system showed gains when the input lattice acoustic and language model scores were included in the minimum error rate training. An FST system based on

alignment templates is presented in [15]. Our implementation borrows the distortion model used in this system. A multi-layered FST decoding approach is presented in [16]. Significant gains in speed over the `pharaoh` decoder [4] are shown.

Our decoder takes as input a finite state acceptor which represents either a single input sentence or the output from a speech recognizer. The best translation can be described as the best path through the transducer given by:

$$E = I \circ P \circ D \circ T \circ L \quad (1)$$

where $\circ$ represents the composition operation. I represents the input acceptor and P is the phrase segmentation transducer. The transducer given by D performs phrase permutations. T and L are the translation and language models, respectively. In the following sections, we describe each model in detail.

3.1. Input Lattice Processing

The input acceptor, I, is a representation of the source language input. For the 1-best case, it is simply a linear automaton with a single path. For ASR input, it consists of a pruned and pre-processed weighted ASR lattice. The input lattice may require some language specific pre-processing such as character/word segmentation for Chinese. This is first performed on the original ASR input lattice. Next, the input lattice in SLF format is converted to the FST format using weights for the acoustic model, language model, and word penalty. All noise/garbage words are translated to epsilon/null links. A punctuation mark is hypothesized after each word only if the word-punctuation bigram occurred in the source language training set. This helps to ensure that source phrases are not missed due to punctuation in the translation model. The resulting transducer with punctuation is then rescored with a 4-gram language model trained on the source language bitext. Finally, pruning is performed such that the final transducer contains only the top-N entries. The last step is necessary to control memory usage during decoding. Empirical results show that the decoder rarely visits paths which occur deep in the n-best list. Figure 1 shows an example of an input speech acceptor with punctuation added. Next, the input transducer, I, is composed with a phrase segmentation transducer, P, which maps input source words to phrases taken from the phrase table.

3.2. Distortion

The distortion model is inspired by the technique in [15]. The exponential number of source word permutations does not allow for arbitrary word movement prior to phrase segmentation. A compromise is to perform swapping at the phrase level after phrase segmentation. This can still allow for long distance reordering with reasonable decoding times. The distortion penalty, expressed as a log probability, is calculated

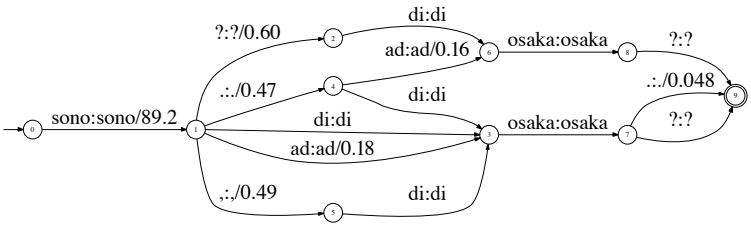

Figure 1: *Example input source language acceptor with punctuation.*

as follows:

$$D_{pen} = -(len(phraseA) + len(phraseB)) \times W_{dist} \quad (2)$$

where $len()$ denotes the length of the phrase in words, and W_{dist} is the weight applied to the distortion model. Longer phrase swaps can be discouraged by increasing the value of W_{dist}.

Initial experiments showed that a single phrase swap was not enough for language pairs with very different sentence structures. We simply apply the phrase swapping transducer twice to provide longer distance reordering permutations. However, this results in a duplication of paths and an increase in lattice size as the original phrase order is duplicated with higher path cost.

3.3. Translation Model

The translation model transducer, T, takes source phrases as input and produces target words as output. The transducer arc weights are calculated as a weighted sum of the translation model weights, including source phrase and target word penalties. No language model is applied at this time. Initially the model is built to accept only one source phrase before terminating with a final node. At this point, the model can be optimized, which can result in a reduction in the number of states by factor of 6.

After the reordered source phrase transducer is composed with the translation model, an intermediate pruning step is performed. This pruning step uses a wider beam threshold and removes any high-cost translation and distortion related paths before the application of the language model.

3.4. Language Model

The language model is a weighted finite state acceptor built from a standard back-off n-gram model with Knesser-Ney smoothing. Some additional software translates the LM into an FST representation and optimization is performed. We use a 4-gram model for decoding the IWSLT dev and test sets, but the FST architecture allows for arbitrary n-gram length subject to memory constraints.

The language model is applied to the intermediate transducer created by $I \circ P \circ D \circ T$. Due to the large number

of path expansions with higher order language models, this step uses a Viterbi search with beam and histogram pruning as opposed to a full composition operation.

3.5. Out of Vocabulary Words

During the composition of $C = A \circ B$, if the output alphabet of A is not a subset of the input alphabet of B, then some or all input paths of A will not be present in C. More simply, out-of-vocabulary source language words that are not in our phrase table will result in transducers with incomplete paths.

An OOV word is detected by comparing the input alphabet of the phrase segmentation transducer with the output alphabet of the input transducer I. Any words in I which are not found in P are handled in a separate FST, O, which simply passes them through with a fixed OOV penalty. The union of O with P and T is taken and Kleene-* closure is applied:

$$\hat{P} = Closure(P \cup O) \quad (3)$$

$$\hat{T} = Closure(T \cup O) \quad (4)$$

The resulting transducers $\hat{P}$ and $\hat{T}$ can now accept any number of source words, including the detected OOV words.

For the language model, we train an open-class LM with an `<unk>` tag. Then we search the network for all `<unk>` tags and add a parallel link with equal weight that passes the OOV word to the output. The resulting models can now accept all words present in the source acceptor I.

3.6. Minimum Error Rate Training

Similar to other decoders, we are able to perform minimum error rate training using n-best lists output from the decoder. However, due to the score combination and loss of intermediate input paths using FSTs, we need to take some extra steps to retrieve individual model scores. Table 3 shows the various model weights that are trained during minimum error rate training. Our system uses some custom software to retrieve each of these individual model scores from an n-best entry.

Input lattice parameters	
$P(X\|f)$	Source acoustic model
$P(f)$	Source language model
$W_{pen}(f)$	Source word insertion penalty
$P_{punc}(f)$	Source language model w/punctuation
OOV_{pen}	Out-of-vocabulary penalty

Table 3: *Additional model parameters used in FST optimization.*

4. Light Morphology for Arabic MT

The Arabic language makes use of rich morphology to encode syntactic and semantic information. For standard statistical models of MT, the rich variation of morphological forms present in Arabic limits phrase learning to inflected forms seen during training. When training data is sparse, this can severely limit the performance of an SMT system (see 6.3). As the training data for the IWSLT evaluations is extremely limited, we performed morphological preprocessing prior to training translation models and testing.

In addition to morphological features that are part of the grammar of the "standard" written language, often called Modern Standard Arabic (MSA), inconsistencies arise from differing conventions in various regions of the Arabic speaking world or an individual's proficiency with MSA. Inconsistencies due to an individual's proficiency can be considerable as the native language of Arabic speakers is almost always a colloquial dialect of Arabic rather than MSA [17, 18].

For the purpose of improving MT quality we addressed, at least to some degree, (1) inconsistencies in the use of the Arabic character *hamza;* (2) inconsistencies in the marking of *tanween;* (3) the attachment as proclitics of certain conjunctions, prepositions, and the definite article; and (4) the attachment as enclitics of direct object pronouns and possessive adjective pronouns [19, 20, 21]. The separation of proclitics and enclitics can be performed by Arabic tokenizers such as the support vector machine (SVM) tokenizer described in [19].[1] However, the approach taken here involves only lightweight stemming specifically designed to improve word alignment and phrase training. The process, which we call AP5, incorporates five simple stemming stages.

4.1. AP5 Process Details

The first step in the preprocessing addressed some inconsistencies in the use of the Arabic character *hamza* (ء) when used with the Arabic character *alef* (ا) at the beginning of words. In this step, when the forms إ, آ, أ, and إ were encountered at the beginning of a word, they were collapsed into the single form ا following the convention used by NIST in ASR evaluations [2].

The second step in the preprocessing addressed inconsistencies in marking *tanween,* which denote the indefiniteness of nouns [20, 21]. There are three *tanween* markings that denote the three grammatical cases of nominative, accusative, and genitive. While there are rules and conventions regarding their use, one can still find them inconsistently applied. In this step, all *tanween* markings were removed.

The third step separated the proclitics و ("wa," → "and") and ال ("al," the definite article) from the rest of the words to which they were attached. The conjunction و can precede ال as in والسيارة ("wa-al-sayArah," → "and the car"), which would be separated as و ال سيارة. Again, the separation was performed by a simple rule; there was no morphological analysis performed to determine whether leading و and ال were really proclitics or just the leading parts of words. Thus, words such as وصل ("waSala," → "he arrived") would be incorrectly separated. For statistical MT preprocessing, such false alarm errors could potentially be repaired by the MT system during translation.

[1] The first version of this tool was formerly available at: `http://www-nlp.stanford.edu/~mdiab`.

[2] Tools available at: `http://www.nist.gov/speech/tools/`

The fourth step separated the proclitics ب ("bi," → "by/with"), ف ("fa," → "then"), ك ("ka," → "as"), and ل ("li," → "to"), as well as their individual combinations with a following ال. The case of ل followed by ال is written in MSA without the intervening *alef* as لل ("lil"); however, when separated, the *alef* of the definite article was restored. As with the separation of و and ال, some words can be incorrectly separated by the regular expression substitutions.

The final step separated the enclitic pronouns from the words to which they were attached. First, when separating enclitic pronouns, some ambiguities can be created. For example the second person singular masculine attached pronoun "ka" is written as ك, which is indistinguishable orthographically from the proclitic ك (see step 4) when it is stemmed. To prevent these ambiguities, each enclitic pronoun is marked with the suffix `post` to disambiguate them.

5. Improved Confusion Network Decoding

In preparation for this year's evaluation, we made a number of enhancements to the confusion network decoding process that we applied to last year's evaluation [24]. Due to limited time in 2006, we were not able to directly optimize source ASR parameters for our evaluation systems. This year's evaluation system fully optimizes all source input parameters.

In the process, we modified the confusion network processing within the `moses` decoder to handle additional scores from input ASR lattices. This allowed for separate scaling factor optimization for source language acoustic and language model scores. Acoustic and language model scores are kept when processing the lattice into confusion network form using the SRI `lattice-tool` [9]. During minimum error rate training, we use these scores in place of the ASR posterior probability, (eliminating the need to set a fixed posterior scaling).

Additionally, we experimented with source confusion network repunctuation of input confusion networks. For text-only systems that needed source repunctuation (Italian and Arabic) we create a confusion network for each sentence by inserting a column of possible punctuation marks between each source word. For ASR input systems we compared the strategy used in [24] with a full repunctuation of the input confusion network. For a given column of the input confusion network k, this process requires summation of posterior probabilities for all paths of length $n-1$ leading to the column k.

6. Experiments

In preparation for the 2007 evaluation, we ran a number of experiments to evaluate the performance of 1) different speech decoding strategies, 2) the use of light morphology for Arabic and 3) the use of skip language models for Chinese. These experiments and the development data configurations are described below.

6.1. Corpus Description

For this year's evaluation we made trained models using only the provided data set. For optimization and tuning we used different combinations of development sets for different languages and different input types. For Italian translation, as `dev5b` was significantly different than prior evaluation data sets, we split this set into two halves, `dev5bp1` (the first 500 sentences of `dev5`) and `dev5bp2` (the remainder) for use in optimization and testing respectively.

6.2. Comparison Lattice and Confusion Network Decoding Strategies

We ran experiments in the text-only condition using confusion network repunctuation. The results of these experiments are shown in Table 4 using the first 500 sentences of `dev5b` (designated as `dev5bp1`) for optimization and testing against the remainder of (`dev5bp2`). As shown, allowing ambiguity in source repunctuation improves performance. Similarly, for ASR input, full repunctuation slightly outperforms the 1-best strategy we employed during last year's evaluation.

Condition	*Repunct Method*	*BLEU*
IE Text	1-best	18.60
IE Text	Full Conf-Net	19.44
IE ASR	1-best	18.00
IE ASR	Full Conf-Net	18.20

Table 4: *BLEU Scores for different repunctuation strategies.*

We also ran a number of experiments optimizing for both source language model and acoustic model scores in place of the ASR posterior. As shown in Table 5 this improves performance for all language pairs, though most prominently in Arabic.

Language (dev/test)	*Source Features*	*BLEU*
CE (`dev4/dev5`)	ASR Posterior	18.17
CE (`dev4/dev5`)	src LM + AM	18.30
AE (`dev4/dev5`)	ASR Posterior	21.77
AE (`dev4/dev5`)	src LM + AM	22.92
IE (`dev5bp1/dev5bp2`)	ASR Posterior	17.93
IE (`dev5bp1/dev5bp2`)	src LM + AM	18.20

Table 5: *BLEU Scores for different source language optimization features.*

Table 6 shows the results comparing the performance of the confusion network and the FST-based lattice decoders. Both systems perform comparably and, in Arabic, the lattice-based decoder showed significant gains. We expect that, with further refinement, this system could be improved.

	BLEU	
Language Pair	*Lattice*	*Conf Net*
CE	17.76	**18.3**
AE	**23.94**	22.92
IE (opt: `dev4`, test: `dev5`)	**30.65**	30.45
IE (opt: `dev5bp1`, test: `dev5bp2`)	17.23	**18.20**

Table 6: *BLEU scores on* `dev5` *after optimization on* `dev4` *with FST decoder.*

Morphological Processing	*BLEU*
None (baseline)	55.40
Steps 1 + 2: Hamza and Tanween normalization	55.93
+ Step 3: `wa-al` proclitic stemming	57.62
+ Step 4: Proclitic stemming II	57.52
+ Step 5: enclitic pronoun stemming	58.73

Table 7: *Impact of different AP5 processing steps.*

6.3. Arabic Morphology Experiments

We conducted experiments comparing AP5 processing against our baseline system (no morphological processing) and against the tokenization algorithm described in [19]. In all cases, a 4-gram LM is used during decoding along with two rescoring language models (7-gram class + 6-gram word). Additionally IBM model-1 scores are applied during rescoring. Results for these experiments are described below on `dev3` using `dev2` for optimization.

6.3.1. AP5 vs. No Morphological Processing

The preprocessing steps of AP5 discussed in the previous section were developed and tested one at a time, and each succeeding step improved the BLEU score over that obtained using all of the preceding steps. Nevertheless, there are a number of interactions that can occur between the steps, and additional experiments are needed to determine the effects of reordering and modifying some of these steps. After tuning, the results for each stage are shown in Table 7.

Interestingly, as we would expect from such a simple morphological analysis system, the current ordering of the morphological processing operations leads to several incorrect splits due to interactions across steps. For example, in البالكونة ("al-bAlkUnah," → "the balcony"), the leading ال is separated in step three, thereby exposing بال for incorrect separation in step four. Despite this, MT performance on the whole is improved significantly over the baseline.

6.3.2. AP5 vs. SVM-based Tokenization for MT

Additional experiments were conducted to compare the performance of AP5 against the tokenization step of Version 1 of the SVM-based part-of-speech tagger and base phrase chunker of [19]. The SVM-based algorithm of [19] was trained on data from the Arabic Penn Treebank 1 (Version 2.0),[3]. As such, it may be significantly mismatched to data from the IWSLT evaluation set. We compared the performance of this tokenization process against AP5 for machine translation.

Stemming Applied	*BLEU*
None (baseline)	55.40
AP5	58.73
SVM-based [19]	55.65

Table 8: *AP5 processing vs. SVM-based tokenization*

In [19], their SVM-based tokenizer was compared to a tokenizer based on rules (similar to those of AP5) as well as to a rule-based system with a dictionary of pretokenized forms. Their SVM tokenizer outperformed, in terms of tokenization accuracy, the other two tokenizers when tested on other Arabic Treebank data ($F_{\beta=1} = 99.12$ versus $F_{\beta=1} = 88.62$ for the rule-based system and $F_{\beta=1} = 93.71$ for the rule- and dictionary-based tokenizer). We expect that AP5 would likely perform worse than the either the dictionary or SVM-based systems.

That said, in our experiments AP5 was found to perform better in terms of BLEU score than the SVM-based tokenizer, though both systems improved the baseline system. Table 8 shows the results of these experiments.

This may be caused by differences between IWSLT-style data and Treebank-style data, or a lack of consistency in tokenization for word alignment and phrase extraction. Other related studies have shown that better tokenization does not necessarily improve MT peformance [22] and the effect here may be related.

6.4. Chinese Text-only Experiments

In this year's Chinese to English system we started with the baseline system configuration we used for last year's system. As this year's evaluations differed from 2006, our model training procedures were adapted accordingly. Decoding was done using the `moses` decoder with a 4-gram language model and n-best lists of 100 sentences were rescored with a 5-gram language model and a 6-gram class-based language model. Results were TrueCased with a 4-gram TrueCaser trained on the training data.

Our primary system was optimized using development sets 1 and 2 (evaluation sets from CSTAR 2003 and IWSLT 2004). TrueCased results on dev3 were 58.53 BLEU in this configuration.

It is worth noting that optimization from different random starts yielded significantly different results. We ran a set of baseline experiments with six consecutive optimization and test cycles. The results of this were highly variable: 57.66, 57.91, 58.12, 57.87, 57.58, and 58.53 (mean: 57.70, confidence interval[95%]: $\pm$ 0.41). Similar results

[3]See: `http://www.ldc.upenn.edu`

Sk	d12+sk	d12-sk	lm+d2+sk	lm+d2-sk
2	57.40 ± 0.40	57.74 ± 0.39	57.78 ± 0.26	57.88 ± 0.09
3	57.35 ± 0.47	57.60 ± 0.38	**58.05** ± 0.21	57.79 ± 0.40
4	57.15 ± 0.19	56.96 ± 0.33	57.51 ± 0.24	57.69 ± 0.25
5	57.00 ± 0.50	57.75 ± 0.40	57.71 ± 0.26	**58.00** ± 0.21
6	57.04 ± 0.23	57.62 ± 0.53	57.73 ± 0.22	57.61 ± 0.25

Table 9: *Chinese* dev3 *results with and without skip language models*

were reproduced using an independent implementation (from the moses toolkit) of the Powell-based minimum error rate training procedure.

Additional experiments were conducted to investigate the potential benefit of using skip language models in the rescoring process [23]. Skip language models with skips ranging from 2 to 6 were investigated as either a replacement for or in addition to the 5-gram language model normally used in our rescoring process. Two systems were tested with skip language models:

- d12 – The baseline system optimized on dev sets 1 and 2.
- lm+d2 – The baseline system in which dev2 references augment the target language model training, only dev set 1 used in optimization and n-best lists of 1000 were used.

Table 9 shows the results for each model configuration when a skip language model is used in addition to a rescoring language model (+sk) and when it replaces the standard rescoring language model (-sk). Each cell of this table shows the average score from six optimization runs with error bars (at 95% confidence).

The results indicate that replacing the rescoring language model with a skip language model in general does better than adding it as an additional rescoring model. System lm+d2 with skip of 5 (replacing the rescoring language model) and skip of 3 (adding the skip language model) yields BLEU scores that show significant improvement over the baseline.

These results indicate that a skip language model may provide some benefit when used in the rescoring process. Future experiments will investigate how well the results of this system and that of our baseline configuration would fuse.

7. Evaluation Results

7.1. System Configurations

As the source input conditions differed for different language pairs, we tailored our systems appropriately. Table 10 shows the configuration of our systems for each language pair and input type.

Two different repunctuation strategies were employed for Arabic speech input. The aren-asr-2 system makes use of phrase tables trained with source punctuation removed. In contrast, the aren-asr-1 system uses a punctuated phrase table and repunctuates the source ASR lattice.

Text Input		
Language	*System*	*BLEU*
Chinese	czen-primary	**36.31**
Arabic	aren-primary	**45.53**
Arabic	aren-secondary	47.41
Italian	iten-primary	**28.42**

Table 11: *Overall performance of submitted systems with text input on* test-2007. (primary systems in **bold**)

ASR Input	*BLEU*	
Configuration	*Ar (Speech)*	*It (Speech)*
Confusion Net Decoder	42.93	**25.00**
Lattice Decoder	**44.29**	22.78

Table 12: *Overall performance of submitted systems with ASR input* test-2006. (primary systems in **bold**)

Tables 11 and 12 show our official submissions to the 2007 IWSLT evaluation. Official primary submissions results are shown in bold.

8. Acknowledgements

We'd like to thank Tyler Pierce for providing language support in Chinese. We would also like to thank the staff of the Information Systems and Technology group at MIT Lincoln Lab for making machines available for this evaluation effort.

9. References

[1] Brown, P., Della Pietra, V., Della Pietra, S. and Mercer, R. "The Mathematics of Statistical Machine Translation: Parameter Estimation," Computational Linguistics 19(2):263–311, 1993.

[2] Och, F. J. and Ney, H., "Discriminative Training and Maximum Entropy Models for Statistical Machine Translation," In ACL 2002: Proc. of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 295-302, Philadelphia, PA, July 2002.

[3] Och, F. J., "Minimum Error Rate Training for Statistical Machine Translation," In ACL 2003: Proc. of the Association for Computational Linguistics, Japan, Sapporo, 2003.

[4] Koehn, P., "Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models," In Proceedings of the Association of Machine Translation in the Americas (AMTA), Washington, DC, 2004.

[5] Shen, W., Delaney, B. and Anderson, T., "The MIT-LL/AFRL IWSLT-2006 MT System," In Proc. Of the

System	*Language*	*Decoder Type*	*Source Repunctation*	*Light Morph.*	*Rescoring*
`czen-primary`	Chinese	`Text`	N/A	no	6g class + 5g skip-LM
`aren-primary`	Arabic	`Text`	N/A	yes	7g class + 6g rescore + model1
`aren-secondary`	Arabic	`Text`	N/A	yes	none
`aren-asr-1`	Arabic	`Lattice`	yes	yes	none
`aren-asr-2`	Arabic	`ConfNet`	N/A	no	7g class + 6g rescore + model1
`iten-primary`	Italian	`Text`	yes	no	6g class + 5g rescore
`iten-asr-1`	Italian	`ConfNet`	yes	no	6g class + 5g rescore
`iten-asr-2`	Italian	`Lattice`	yes	no	none

Table 10: *System Configurations for different language pairs.*

International Workshop on Spoken Language Translation, Kyoto, Japan, 2006.

[6] Chen, B. et al, "The ITC-irst SMT System for IWSLT-2005," In Proc. Of the International Workshop on Spoken Language Translation, Pittsburgh, PA, 2005.

[7] Melamed, D., "Models of Translational Equivalence among Words," In Computational Linguistics, vol. 26, no. 2, pp. 221-249, 2000.

[8] Al-Onaizan, Y., Curin, J., Jahr, M., Knight, K., Lafferty, J., Melamed, I.D., Och, F.J., Purdy, D., Smith, N.A., Yarowsky, D., "Statistical machine translation: Final report," In Proceedings of the Summer Workshop on Language Engineering at JHU, Baltimore, MD 1999.

[9] Stolcke, A., "SRILM - An Extensible Language Modeling Toolkit," In Proceedings of the International Conference on Spoken Language Processing, Denver, CO, 2002.

[10] Philipp Koehn et al, "Moses: Open Source Toolkit for Statistical Machine Translation," Annual Meeting of the Association for Computational Linguistics (ACL), Prague, Czech Republic, June 2007.

[11] M. Mohri, "Finite-State Transducers in Language and Speech Processing," In Computational Linguistics, vol 23, no 2, pp 269-311, 1997.

[12] L. Hetherington, "The MIT Finite-State Transducer Toolkit for Speech and Language Processing," In Proc. Intl. Conf. on Spoken Language Processing, Jeju, Korea, 2004.

[13] E. Matusov, H. Ney and R. Schulster, "Phrase-Based Translation of Speech Recognizer Word Lattices Using Loglinear Model Combination," In Proc. Automatic Speech Recognition and Understanding Workshop (ASRU), San Juan, Puerto Rico, 2005.

[14] A. Perez, M.I. Torres, and F. Casacuberta, "Speech Translation with Phrase Based Stochastic Finite-State," In Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing, Toulouse, France, 2007.

[15] S. Kumar and W. Byrne, "A Weighted Finite State Transducer Implementation of the Alignment Template Model for Statistical Machine Translation," In Proc. of 2003 Conf. of the NAACL, Edmonton, Canada, 2003.

[16] B. Zhou, L. Besacier and Y. Gao, "On Efficient Coupling of ASR and SMT for Speech Translation," In Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing, Toulouse, France, 2007.

[17] C. Holes, "Modern Arabic: Structures, Functions, and Varieties," Georgetown University Press: Washington, D.C., 2004.

[18] M. C. Bateson, "Arabic Language Handbook," Georgetown University Press: Washington, D.C., 2003.

[19] M. Diab, K. Hacioglu, and D. Jurafsky, "Automatic tagging of Arabic text: From raw text to base phrase chunks," in Proceedings of HLT-NAACL, (Boston MA), May 2004.

[20] E. Badawi, M. G. Carter, and A. Gully, "Modern Written Arabic: A Comprehensive Grammar," Routledge: London, 2004.

[21] J. Mace, "Arabic Grammar: A Reference Guide," Edinburgh University Press: Edinburgh, 1998.

[22] P. Koehn and K. Knight, "Empirical Methods for Compound Splitting," In Proc. of the European Assocation for Computational Linguistics, Budapest, Hungary, 2003.

[23] D. Guthrie et al, "A Closer Look at Skip-gram Modelling," In Proceedings of the Language Resources and Evaluation Conference (LREC), pp. 1222-1225, Genoa, Italy, 2006.

[24] W. Shen, R. Zens, N. Bertoldi, and M. Federico, "The JHU Workshop IWSLT-2006 MT System," In Proc. Of the International Workshop on Spoken Language Translation, Kyoto, Japan, 2006.