Abstract

The MIT-LL/AFRL MT system is a statistical phrase-based translation system that implements many modern SMT training and decoding techniques. Our system was designed with the long-term goal of dealing with corrupted ASR input and limited amounts of training data for speech-to-speech MT applications. This paper will discuss the architecture of the MIT-LL/AFRL MT system, improvements over our 2005 system, and experiments with manual and ASR transcription data that were run as part of the IWSLT-2006 evaluation campaign.

1. Introduction

In recent years, the development of statistical methods for machine translation has made usable MT a real possibility. Specifically, advances in methods to:

- Extract word alignments from parallel corpora [1][2]
- Learn and model the translation of phrases [3] [4]
- Decode and Rescore Test data [8] [9]

These advances have helped to dramatically increase the quality of MT output. Our 2006 IWSLT system extends these methods and work we did in 2005 [10].

In subsequent sections, we will discuss the details of the translation system including our alignment and language models and methods we’ve implemented for optimization and decoding. Specifically, we will highlight improvements and changes made to:

1. Better utilize the larger 2006 training set
2. Coverage of Italian and Japanese
3. Enhance the coverage of extracted phrases

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.
Fig. 2: A Factor-based Consistency-Checking Model

Factored translation models extend standard phrase-based statistical models by representing words as vectors of factors. This representation can be used to decompose words into constituent parts (e.g., lemma + affix) for the purpose of modeling them separately, or as generalizing words into larger linguistic “classes” (e.g., part-of-speech). From a factored representation, it is possible to train standard statistical models that are then combined using standard log-linear assumptions in which feature functions of the form \( h_{FACTOR_k}(e_1, ..., f_1) \) represent translation likelihoods that are specific to factor \( k \) and special generation features \( h_{gen}(FACTOR_k(e_i), FACTOR_{l}(e_i)) \) that represent the likelihood of generating \( FACTOR_k \) from \( FACTOR_l \).

Because we did not have access to analysis tools in Chinese during the IWSLT evaluation, we chose to create models using automatically derived word classes (as generated by mkcls). In our experiments words are represented both by their surface form and by their associated word classes.

Using this representation we trained two different models:

- **Consistency-Checking Model** – Translate source surface forms to target, generate word classes for each target, then apply a class-based LM.
- **A Parallel Translation Model** – Translate both source surface forms and word-classes to target word/class pairs, then apply a class-based LM.

These models are shown schematically in Figure 2 and Figure 3, respectively. We note that the parallel approach is quite similar to the alignment template model proposed in [15] with an additional surface-to-surface form translation model. These models were not applied in time for official submission to the 2006 evaluation, but in post-evaluation experiments we found these models to be quite helpful.

5. Improved Decoding

For the 2006 evaluation we used a combination of two decoders: our in-house decoder mtdecoder and the moses decoder developed as part of the 2006 JHU summer workshop. For most experiments, both decoders performed on par with each other (though we generally used our own decoder for minimum error rate training, because of its speed). For factored experiments, we used moses. With both decoders we found it advantageous to use 4-gram and 5-gram language models in decoding. Our official submissions for Chinese, Japanese and Italian use 4-gram Interpo-
Reordering Constraints

- Revisit Reordering Constraints [Zens 03]

Disallowed

(k = 2)

Source

安 共 少 钱
calm money
small calm money with

Target

Disallowed

Figure 4: An example of a disallowed reordering using IBM constraints

Figure 5: An example of a disallowed reordering using ITG constraints

lated Kneser-Ney models trained using the SRI Language Modeling Toolkit [16] [17] [18].

Using our decoder we implemented three types of reordering constraints, revisiting work done in [19] with the IWSLT-2006 data. We explored both ITG [20] and IBM constraints, and the results shown in Section 7 indicate that different reordering constraints don’t decrease the BLEU score significantly in most language pairs while reducing decoding time by 20-50%. Both constraints disallow certain reordering configurations. Figures 5 and 4 offer examples of these configurations. Details of these experiments are described in [21].

6. Rescoring N-best Lists

As in 2005, we employ minimum error rate training to optimize model scaling factors for both decoding and rescoring features. In this year’s evaluation, we added 5-gram rescoring language models and 6-gram class-based rescoring language models after decoding. After the evaluation we added sentence length posterior features for rescoring. A full list of the feature functions used in our system is shown in Table 1.

We approximate sentence length posteriors from the n-best list as:

\[ P(L|f_1...f) \approx \sum_{\{e \mid |e|-L\}} P(e_1...L|f_1...j) \]  \hspace{1cm} (1)

Similarly IBM model 1 scores can be computed for each n-best list entry:

\[ P_{ibm1} \approx \frac{1}{(I+1)^J} \prod_{j=1}^{J} \sum_{i=1}^{L} p(f_j|e_i) \]  \hspace{1cm} (2)

7. Development Experiments

In preparation for the arrival of the official evaluation data, we conducted experiments with our system using dev4 in each of the language pairs. For these experiments we set aside dev1 for minimum error rate training.

7.1. Segmentation and Alignment

For different language pairs we employ different segmentation techniques. We use basic word segmentation for Italian, combining phrases extracted from IBM model 5 alignments with CLA alignments. For Japanese, we found it optimal to use word segmentation with character segmentation backoff with CLA alignments. In this configuration, words that were unseen in training (OOV) are broken into constituent characters then translated using character phrases. In the Chinese case, we use both word and character segmentation. From both, we compute both CLA and IBM model 5 alignments and extract phrases that are then normalized to character segmentation when aggregating counts.

Tables 2, 3 and 4 show a summary of results for various configurations of segmentation and alignment.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character segmented</td>
<td>21.24</td>
</tr>
<tr>
<td>Word segmented</td>
<td>21.01</td>
</tr>
<tr>
<td>Char+Word segmented</td>
<td>21.21</td>
</tr>
<tr>
<td>Char+Word segmented + CLA</td>
<td>22.18</td>
</tr>
</tbody>
</table>

Table 2: Segmentation/alignment results for Chinese (dev4)

7.2. Rescoring

In addition to standard features that we use during decoding, we introduce a number of additional features for rescoring n-best lists generated by our decoder (or moses). For the 2006 evaluation we tried a number of new features, including longer context LMs (text and class-based), IBM model 1, unigram posteriors and sentence length posteriors. Empir-
Table 3: Segmentation/alignment results for Japanese (dev4)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Segmented</td>
<td>23.63</td>
</tr>
<tr>
<td>Word Segmented + Character Backoff</td>
<td>23.82</td>
</tr>
<tr>
<td>Word Segmented + CLA</td>
<td>23.34</td>
</tr>
<tr>
<td>Word Segmented + Character Backoff + CLA</td>
<td>24.28</td>
</tr>
</tbody>
</table>

Table 4: Segmentation/alignment results for Italian (dev4)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Segmented</td>
<td>35.13</td>
</tr>
<tr>
<td>Word Segmented + CLA</td>
<td>37.40</td>
</tr>
</tbody>
</table>

Table 5: Rescoring results for all languages (dev4)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Chinese</th>
<th>Japanese</th>
<th>Italian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline 4-gram Decode</td>
<td>21.59</td>
<td>21.92</td>
<td>36.92</td>
</tr>
<tr>
<td>w/5-gram rescore LM</td>
<td>21.55</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>w/6-gram class-based LM</td>
<td>21.52</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>w/Model 1</td>
<td>21.86</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>w/Sent. Length Posterior</td>
<td>22.10</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>ALL Features</td>
<td>22.10</td>
<td>24.28</td>
<td>37.40</td>
</tr>
</tbody>
</table>

7.3. Pre/Post-Processing

During the evaluation, we explored different pre and post-processing options to optimize this year’s official evaluation criterion (mixed-case, with punctuation, no source punctuation provided). We tried two different methods of producing target punctuation: 1) training asymmetric models by removing source punctuation from train and development corpora, and 2) repunctuating source sentence in the supplied development and test corpora.

To produce mixed-case output, we applied implemented an HMM-based truecasing model as proposed in [22]:

\[
\hat{w}^{*1\ldots j} = \arg \max_{w1\ldots j} P(w1\ldots j | s1\ldots j) \tag{3}
\]

\[
\hat{w} = \arg \max_{w1\ldots j} P(s1\ldots j | w1\ldots j) \tag{4}
\]

\[
\hat{w} = P(w1\ldots j) \tag{5}
\]

where a standard, interpolated language model approximation is used as in:

\[
\hat{P}(w1\ldots j) \approx \prod_{k=1}^{j} P(w_k|w_{k-1} \ldots w_{k-n+1}) \tag{6}
\]

and an approximate table of conditional emission probabilities is represented by:

\[
\hat{P}(s1\ldots j | w1\ldots j) \approx \prod_{k=1}^{j} P(s_k|w_k) \tag{7}
\]

Where \(w^{*1\ldots k}\) is the maximum likelihood TrueCased output sequence and \(s_{1\ldots j}\) is the corresponding lower-case input. As shown in Table 6, automatic repunctuation of the input source is beneficial in performance terms. Similarly, small gains can be had by choosing the appropriate language model order for TrueCasing.

7.4. Factored Models

After the official evaluation deadline, we ran a number of experiments to explore the performance of the factored models described in Section 4. Our experiments focus on a baseline Chinese-to-English system trained using only word segmentation and optimized as described above. Due to time constraints, we did not perform the rescoring described in Section 7.2. With this configuration, our baseline system achieve a BLEU score of 19.60 on dev4 with the official evaluation criteria.

We ran experiments with both Consistency Checking models using a class-based language model, and Parallel...
Translation models using both class-based translation and language models. As shown in Figure 6, both factored approaches achieve substantial gains, though the Consistency Checking model (shown as Class-LM) is consistently better than both the baseline and the Parallel Translation model (shown as Class Trans+LM). This approach equals the performance of our best rescoring model on dev4 despite starting from a worse baseline.

We have seen that limitations in the current implementation of moses may cause search errors in our parallel translation models. Despite current limitations, our parallel models offer some advantage.

7.5. Decoder Reordering Constraints

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Chinese</th>
<th>Japanese</th>
<th>Italian</th>
</tr>
</thead>
<tbody>
<tr>
<td>free</td>
<td>20.32/3509.5</td>
<td>22.35/3309.7</td>
<td>35.85/90.6</td>
</tr>
<tr>
<td>IBM</td>
<td>19.85/2961.0</td>
<td>21.46/2969.3</td>
<td>35.52/36.2</td>
</tr>
<tr>
<td>ITG</td>
<td>19.85/2961.0</td>
<td>21.37/1868.7</td>
<td>35.52/36.2</td>
</tr>
</tbody>
</table>

Table 7: Performance of different reordering constraints (dev4)

Although we did not use ITG or IBM reordering constraints in our official submissions, some development experiments with these constraints did yield gains. Unfortunately, these gains were not consistent across dev sets. Table 7 shows the performance of different reordering constraints in contrast to our baseline configuration, free reordering, in which all possible reorderings are allowed within a fixed window (in our default configuration this is set to 10).

Gains in processing time are quite apparent. 20-60% improvement in speed can be had with minimal BLEU score impact using these reordering constraints. More detailed experiments with these constraint can be found in [21].

8. Evaluation Results and Analysis

<table>
<thead>
<tr>
<th>Text Input</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ch (Read)</td>
<td>Jp</td>
</tr>
<tr>
<td>Ch (Spon.)</td>
<td>It</td>
</tr>
<tr>
<td>Opt. (dev4)</td>
<td>21.57</td>
</tr>
<tr>
<td>10-best, Opt (dev4)</td>
<td>17.42</td>
</tr>
<tr>
<td>1-best, Opt (dev4)</td>
<td>18.46</td>
</tr>
</tbody>
</table>

Table 8: Overall performance of submitted systems with text input (test-2006)

primary submissions were optimized using dev4. These submissions processed 1-best ASR input and reference transcription. Our secondary submissions decoded 10-best from the ASR lattice, merging MT n-best lists and rescoring with ASR features as described in [10].

Reruning our system using the 2005 train/dev/test paradigm, we found that our system gained over 4 BLEU points (8.7% relative improvement) with respect to our previous best.

Our next steps include further development of our in-house decoder and experiments with factored models using better baselines and better search methods.

9. Acknowledgements

We’d like to thank Eric Chapla, Johnn Luu, Tyler Pierce for providing language support in Japanese and Chinese respectively. 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.

10. References


