Unsupervised Machine Translation

- Given: two monolingual corpora on source/target (not sentence-aligned)
  - No parallel corpora, no seed lexicon
  - Target language model (LM): trained beforehand
  - To train: word lexicon model \( p(f|e) \)
  - Task assumption: 1-1 monotone word alignment

  ![Source words](f_1 \ldots f_n) \quad \text{and} \quad \text{Target words} \quad e_1 \ldots e_n

- Computationally infeasible to consider phrases and reorderings
- Data preparation / Task setup
  1. Learn word alignments of a parallel corpus
  2. Reorder/Source words to make the alignment 1-1 monotonic
  3. Divide the corpus into two parts:

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st part</td>
<td>Training data Reference (only for evaluation)</td>
</tr>
<tr>
<td>2nd part</td>
<td>-</td>
</tr>
</tbody>
</table>

Baseline Framework

- Hidden Markov model (HMM)
  \[ p(e_i^N, f_i^N) = \prod_{n=1}^{N} p(e_{n|e_{n-1}}) p(f_{n|e_n}; \theta) \]
- Training: expectation-maximization (EM) algorithm
  \[ L(\theta) = \sum_{e_i^N} p(e_i^N, f_i^N) \]
  - Latent variable: target sentence \( e_i^N \)
  - E-step: compute posteriors \( p_{\theta}(e_i^N | f_i^N) \) (forward-backward algorithm)
  - M-step: update lexicon table \( \theta_{f_i e} \)
- This work: first attempt at 100k-lexicon scenarios

Sparse Lexicon

- Problem 1: full table \( \theta_{f_i e} \) is too large to fit in memory
  - How can we represent the lexicon efficiently?
- Solution: filter out unlikely entries for each iteration
  1. Select the lexicon entries with a high probability (threshold \( \tau \))
  \[ \mathcal{F}(e) = \{ f | \theta_{f_i e} \geq \tau \} \]
  2. Renormalize over the selected entries, setting other entries to zero
  \[ p_{\theta}(f|e) = \begin{cases} \frac{\theta_{f_i e}}{\sum_{f' \in \mathcal{F}(e)} \theta_{f' e}} & \text{if } f \in \mathcal{F}(e) \\ 0 & \text{otherwise} \end{cases} \]
  3. Smooth with a uniform back-off model \( p_{\text{bo}}(f) \)
  \[ p(f|e) = \lambda \cdot p_{\theta}(f|e) + (1 - \lambda) \cdot p_{\text{bo}}(f) \]
  - Enforces multinomial sparsity throughout the training
  - Reduces the model size on the fly
- Results on EUTRANS es-en (no pruning)

<table>
<thead>
<tr>
<th>Lexicon</th>
<th>( \tau )</th>
<th>Accuracy [%]</th>
<th>Memory [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>-</td>
<td>70.2</td>
<td>100</td>
</tr>
<tr>
<td>Sparse</td>
<td>0.005</td>
<td>69.0</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td>0.002</td>
<td>72.3</td>
<td>5.1</td>
</tr>
<tr>
<td></td>
<td>0.0001</td>
<td>70.1</td>
<td>9.1</td>
</tr>
</tbody>
</table>
- Outperforms full table by setting \( \tau \) properly
- Greatly reduces the memory usage

Initialization Using Word Classes

- Problem 2: harsh pruning is inevitable for large hypothesis lattices
  - EM algorithm does not converge properly
  - How can we stabilize the training?
- Solution: learn an initial lexicon on word class vocabulary
  1. Estimate word-class mappings on both sides \( (C_{\text{src}}, C_{\text{tgt}}) \)
     - Exchange algorithm, e.g. mkcls tool
  2. Map each word in the corpus to its class
  \[ f \mapsto C_{\text{src}}(f) \quad e \mapsto C_{\text{tgt}}(e) \]
  3. Train a class-to-class full lexicon \( p_{\theta}(C_{\text{src}}(f) | C_{\text{tgt}}(e)) \)
  4. Convert \( p_{\theta} \) to a word lexicon score by mapping each class back to its member words (not normalized yet)
  \[ \forall (f, e) \quad \theta_{f e} := p_{\theta}(C_{\text{src}}(f) | C_{\text{tgt}}(e)) \]
  5. Apply the thresholding and renormalization to 4 (sparse lexicon)
- Class vocabulary \( C_{\text{src}} \) - word vocabulary: marginal increase in memory/time
- Results on EUTRANS es-en (pruning with beam size 10)

<table>
<thead>
<tr>
<th>Initialization</th>
<th>Accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>63.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word Classes</th>
<th>#Classes</th>
<th>Class LM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25</td>
<td>2-gram 67.4</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>2-gram 69.1</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>2-gram 72.1</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>3-gram 76.0</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>4-gram 76.2</td>
</tr>
</tbody>
</table>
- More performance gain with:
  - larger number of classes
  - better class LMs

Large Vocabulary Experiments

- Corpus statistics

<table>
<thead>
<tr>
<th>Task</th>
<th>Source (Input)</th>
<th>Target (LM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EUROPARL</td>
<td>Running Words</td>
<td>2.7M</td>
</tr>
<tr>
<td>en-es</td>
<td>Vocabulary</td>
<td>32k</td>
</tr>
<tr>
<td>IWSLT 2014</td>
<td>Running Words</td>
<td>2.8M</td>
</tr>
<tr>
<td>ro-en</td>
<td>Vocabulary</td>
<td>99k</td>
</tr>
</tbody>
</table>
- Results

<table>
<thead>
<tr>
<th>Task</th>
<th>Supervised</th>
<th>Unsupervised</th>
<th>Memory [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>en-es</td>
<td>77.5</td>
<td>54.2</td>
<td>0.06</td>
</tr>
<tr>
<td>ro-en</td>
<td>72.3</td>
<td>32.2</td>
<td>0.03</td>
</tr>
</tbody>
</table>
- Significantly high accuracy with \(< 0.1\% \) memory
- Conventional decipherment methods are not applicable

Conclusion and Outlook

- First promising results in 100k-lexicon unsupervised machine translation
  - Sparse lexicon = no memory bottleneck + effective model structure
  - Initialization using word classes = robust training + performance boost
- Outlook
  - Incorporating local reorderings
  - Neural network lexicon models
  - Using more training data and more powerful LMs