Smoothing and Data Selection in Large SMT Systems

Holger Schwenk

LIUM, University of Le Mans, France

Holger.Schwenk@lium.univ-lemans.fr

May 14, 2008
Plan

- Introduction and motivation
- NIST task
- Baseline architecture
- Data selection/emphasizing
  - language modeling
  - translation models
- Smoothing techniques
  - language modeling
- Perspectives
Statistical Machine Translation

- All knowledge is automatically extracted from representative data:
  - bitexts: existing human supplied translations (100k–200M)
  - monolingual data: used for the LM, usually journals or WEB data (10M–10G)
- Estimate probability distributions from this data:
  - phrase table with various scores
  - $n$-gram language model
Probability estimation

- Relative frequency
  - high variance, low bias
  - overestimation of rare events
  - no generalization to unseen events

- Some kind of smoothing is needed
  - common practice in language modeling
  - but not (yet) frequently used for the translation model
  - some work has shown possible improvements for instance [Foster et al, EMNLP’06]
Introduction

Data selection/emphasizing

- Data often comes from a large variety of sources
  - in- versus out-of-domain
  - old versus recent sources
  - high quality human versus approximate translations
  - ...

- Large variations in size
- It seems suboptimal to mix all these data sources and to use them uniformly

⇒ How to weight the data sources in function of their relevance to the task?
NIST Open MT evaluation

- yearly evaluations performed by NIST since 2001
- focus on translation from Mandarin and Arabic to English
- large amounts of training data available:
  - 175M words of bitexts and 3.5G of newspaper texts
  - considerable computational resources are needed
  - approaches that achieved improvements on smaller task may not help anymore or be too expensive to apply
- carefully selected test data with four high quality human translations

⇒ NIST evaluations have played a key role to advance the field by providing a common test bed and infrastructure to compare the most promising approaches
Bitexts

- Various small corpora (9.1M words)
- Development data from previous evaluations (2M words)
- ISI automatically aligned data (35M words)
- UN corpus (130M words)

⇒ phrase-table with 228M entries (6.2G gzipped)

Monolingual data

- English part of bitexts (175M words)
- Gigaword corpus of newspaper texts (3.2G words)
- Parts of Google n-grams (139M out of 1T n-grams)

⇒ 4-gram back-off LM with 264M 4-grams, file size of 5.5GB
System Architecture

Design decisions of the system

- Pure statistical system without usage of linguistic knowledge (yet)
- Validate system architecture and algorithms that did work well on small (IWSLT) and medium sized tasks (Europarl)
- Build a state-of-the-art system based on open-source
- Single system without system combination
- Careful use of available data
  - do we need quality or quantity?
  - reasonably compact representation of the data
System Architecture Overview

- Parallel corpus
- Monolingual corpus
- Phrase extraction
- SRILM
- CSLM
- Phrase table
- 4g LM
- 5g CSLM
- Moses
- 1000 bests
- LM rescoring
- Trg
- Condor
- BLEU

only 14 feature functions
- translation model (4)
- lex. reordering (7)
- LM (1)
- penalties (2)

decode optimized with MERT

2nd pass optimization
Data Selection in the LM

Data selection

- Merge all data and build one LM
  → important but small data is outvoted by large corpora
- LM combination:
  + select common word list
  + train individual LM on each subcorpus
  + linear combination:
    \[
    P_{LM}(w_3|w_1w_2) = \sum_i \lambda_i P_{LM_i}(w_3|w_1w_2)
    \]
- log-linear: each LM is a feature function among others
  \[
  P = \sum_j \log P_j + \sum_i \lambda_i \log P_{LM_i}(w_3|w_1w_2)
  \]
## Data Selection in the LM

### Theoretical comparison

<table>
<thead>
<tr>
<th>Probabilities:</th>
<th>linear</th>
<th>log-linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>criterion:</td>
<td>added</td>
<td>multiplied</td>
</tr>
<tr>
<td>optimisation:</td>
<td>perplexity</td>
<td>BLEU</td>
</tr>
<tr>
<td># of models:</td>
<td>can be merged</td>
<td>numerical</td>
</tr>
</tbody>
</table>

- added
- multiplied
- optimisation: perplexity
- EM
- # of models: can be merged into one
- as much as submodels
Data Selection in the LM

Experimental comparison

• Combining europarl and news-commentary LMs:

![Graph showing perplexity and BLEU scores against interpolation coefficient.](image)

• Experimental comparison is not always clear
• Linear combination is usually as good and much easier to realize
Data Selection in the LM

Example: NIST task

- bitexts: 175M
  - Gale translations (1.1M words)
  - development data from previous years (0.9M words)
  - various news wire data (8.1M words)
  - automatically extracted parallel texts from ISI (35M words)
  - UN data (130M words)
- Gigaword newspaper corpus: 3.4G
  - divided into 7 subsets to keep estimation tractable
- Google n-grams: 1T
  - selected subset of 139M 4-grams

⇒ total of 12 submodels
Data Selection in the LM

Result summary

<table>
<thead>
<tr>
<th>corpus</th>
<th>train #words</th>
<th>LM size</th>
<th>Px dev06 all</th>
<th>Nwire</th>
<th>WEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>bitexts pooled</td>
<td>175M</td>
<td>666M</td>
<td>189.3</td>
<td>145.7</td>
<td>351.3</td>
</tr>
<tr>
<td>idem w/o UN</td>
<td>45M</td>
<td>278M</td>
<td>183.0</td>
<td>140.2</td>
<td>343.7</td>
</tr>
<tr>
<td>bitexts ipol</td>
<td>175M</td>
<td>309M</td>
<td>161.7</td>
<td>131.0</td>
<td>266.2</td>
</tr>
<tr>
<td>+ GigaWord</td>
<td>3.4G</td>
<td>3.7G</td>
<td>128.1</td>
<td>104.7</td>
<td>206.5</td>
</tr>
<tr>
<td>+ Google</td>
<td>(1T)</td>
<td>5.5G</td>
<td>114.5</td>
<td>99.0</td>
<td>161.7</td>
</tr>
</tbody>
</table>

- Pooled LM is better without the UN data!
- It’s very important to consider the heterogeneous data in the bitexts, in particular for the WEB part
- Google n-grams achieve decrease of 11%, mainly on WEB
Data Selection in the TM

How to account for the heterogeneous data?

- multiple phrase tables
- linear interpolation of separately trained phrase tables
- some kind of discriminative training
Data Selection in the TM

Multiple phrase tables

- build a phrase table per source and provide multiple tables to Moses
- log-linear combination
- MERT training should weight correctly the different models
- but each table provides 5 scores
  → high dimensional optimisation problem
    (even worse when we also consider lexical reordering)
    - Unrealistic for more than three models
- alignments risk to be suboptimal for small corpora
- contradictory experimental results
Data Selection in the TM

Linear interpolation of separately trained phrase tables

- motivated by the procedure used for LMs
- how to judge the quality of a phrase-table without running a full system (something equivalent to perplexity)?
- how to estimate the coefficients?
- merging into one phrase table is not obvious
- alignments risk to be suboptimal for small corpora

⇒ often only one phrase table is estimated on the pooled data
Data Selection in the TM

ISI automatically extracted parallel data

- found pseudo parallel data in the English and Arabic Gigaword corpus
- algorithm [Munteanu & Marcu, CL 2005]:
  - consider time window, word dictionary, IBM1 alignments, max entropy classifier, ...
- 1.1M sentences were extracted (35M words)
- confidence scores are provided
Data Selection in the TM

How to best use the ISI automatically aligned bitexts?

- Keep only sentences with a confidence score superior to a threshold
- Initial experiments with Gale manual translations only:

⇒ Gain of 2 points BLEU when not all ISI data is used
Data Selection in the TM

Result summary (LM trained on all bitexts + Gigaword)

<table>
<thead>
<tr>
<th>Bitext</th>
<th>#words</th>
<th>Dev06</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gale+nw</td>
<td>9M</td>
<td>43.02</td>
</tr>
<tr>
<td>Gale+nw+ISI</td>
<td>35M</td>
<td>45.09</td>
</tr>
<tr>
<td>Gale+nw+ISI+dev</td>
<td>36M</td>
<td>45.38</td>
</tr>
<tr>
<td>Gale+nw+ISI+dev+un</td>
<td>165M</td>
<td>45.98</td>
</tr>
</tbody>
</table>

- Filtered ISI automatic texts are pretty useful
- Adding old Dev data gives 0.3 improvement
  → Pretty good result with core bitexts of 36M words only
- Only +0.6 BLEU with 129M words of UN data
  → High quality in-domain data seems to be more important than large amounts of general data
Continuous Space LM

Theoretical drawbacks of back-off LM:

- Words are represented in a high-dimensional discrete space
- Probability distributions are not smooth functions
- Any change of the word indices can result in an arbitrary change of LM probability

⇒ True generalization is difficult to obtain

Main idea [Bengio, NIPS’01]:

- Project word indices onto a continuous space and use a probability estimator operating on this space
- Probability functions are smooth functions and better generalization can be expected
CSLM - Probability Calculation

- Outputs = LM posterior probabilities of all words: 
  \[ P(w_j = i | h_j) \quad \forall i \in [1, N] \]
- Context \( h_j \) = sequence of \( n-1 \) points in this space

\[
\begin{align*}
\text{Outputs} &= \text{LM posterior probabilities of all words:} \\
&= P(w_j = i | h_j) \quad \forall i \in [1, N] \\
\text{Context} \ h_j &= \text{sequence of} \ n-1 \text{ points in this space}
\end{align*}
\]
CSLM - Training

- Backprop training, cross-entropy error
  \[ E = \sum_{i=1}^{N} d_i \log p_i \]
  + weight decay
  \[ \Rightarrow \text{NN minimizes perplexity on training data} \]
- continuous word codes are also learned (random initialization)
Continuous Space LM

Some details (Computer Speech and Language, pp 492–518, 2007)

- Projection and estimation is done with a multi-layer neural network
- Still an \(n\)-gram approach
- But LM probability for any \(n\)-gram can be calculated without backing off
- Usually trained on the same data than the back-off LM using a resampling algorithm
- Efficient implementation is very important
- Used in second pass as an additional feature function
- Quite succesful in several tasks and languages
CSLM - Training

Training Procedure

- Same training data than back-off LM (bibtexts + Giga)
- Resample algorithm (HLT/EMNLP’05 paper)
- Shortlist of length 8k
- Trained several networks with different context sizes
- Interpolated with 4-gram back-off LM

Incorporation into MT System

- $n$-best list rescoring
- Feature function coefficients are again optimized
### Result summary - perplexities

<table>
<thead>
<tr>
<th>corpus</th>
<th>train #words</th>
<th>LM size</th>
<th>Px dev06 all</th>
<th>Px dev06 Nwire</th>
<th>Px dev06 WEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>bitexts pooled</td>
<td>175M</td>
<td>666M</td>
<td>189.3</td>
<td>145.7</td>
<td>351.3</td>
</tr>
<tr>
<td>idem w/o UN</td>
<td>45M</td>
<td>278M</td>
<td>183.0</td>
<td>140.2</td>
<td>343.7</td>
</tr>
<tr>
<td>bitexts ipol</td>
<td>175M</td>
<td>309M</td>
<td>161.7</td>
<td>131.0</td>
<td>266.2</td>
</tr>
<tr>
<td>+ GigaWord</td>
<td>3.4G</td>
<td>3.7G</td>
<td>128.1</td>
<td>104.7</td>
<td>206.5</td>
</tr>
<tr>
<td>+ Google</td>
<td>(1T)</td>
<td>5.5G</td>
<td>114.5</td>
<td>99.0</td>
<td>161.7</td>
</tr>
<tr>
<td>+ CSLM</td>
<td>3.4G</td>
<td>+1G</td>
<td>98.3</td>
<td>85.3</td>
<td>137.4</td>
</tr>
</tbody>
</table>

- It seems to be very important to consider the heterogeneous data in the bitexts, in particular for the WEB part
- Google n-grams achieve decrease of 11%, mainly on WEB
- CSLM gives 14% improvement on top of this large LM
Smothing and Data Selection in Large SMT Systems

H. Schwenk

Introduction
Task
Architecture Overview
Data selection
LM
TM
CSLM
Architecture Results
Conclusions

**Result summary - BLEU scores**

<table>
<thead>
<tr>
<th>System</th>
<th>All</th>
<th>Dev06</th>
<th>Eval08</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>43.99</td>
<td>46.84</td>
<td>34.51</td>
</tr>
<tr>
<td>beam tuning</td>
<td>44.40</td>
<td>47.27</td>
<td>34.90</td>
</tr>
<tr>
<td>+ Google LM</td>
<td>44.70</td>
<td>47.22</td>
<td>36.11</td>
</tr>
<tr>
<td>+ CSLM</td>
<td>45.96</td>
<td>48.56</td>
<td>36.69</td>
</tr>
<tr>
<td>Dev06</td>
<td></td>
<td></td>
<td>41.69</td>
</tr>
<tr>
<td>Eval08</td>
<td></td>
<td></td>
<td>42.13</td>
</tr>
<tr>
<td>All</td>
<td></td>
<td></td>
<td>41.90</td>
</tr>
<tr>
<td>All</td>
<td></td>
<td></td>
<td>42.98</td>
</tr>
</tbody>
</table>

- Tuning of beam affects both subsets
- Filtered Google LM mainly improves BLEU on WEB data
- CSLM gives overall improvement of 1.1 BLEU on test data on top of the completely tuned system
Conclusion and Perspectives

Conclusion

- Data selection/emphasizing is very important
- There is a common practice for LM:
  - train individual models,
  - optimize perplexity with EM procedure
  - linear interpolation + merge into one model
    → apply this procedure consequently
- but there is no satisfactory straight-forward procedure for the translation model
⇒ Research in this direction is needed
Conclusion and Perspectives

Conclusion

- Automatically aligned data can be very helpful
- But it must be carefully selected
- Using too much can actually hurt

⇒ Continue to explore the usage of “found bitext”

- Nice result with CSLM: careful smoothing and good generalisation is important even with large amounts of training data

⇒ Can we do something similar with the translation model?
Conclusion and Perspectives

Perspectives

- Phrase-based translation models are still too simple:
  - data emphasizing is difficult
  - no smoothing
  - bad generalization to unseen phrases (singular $\rightarrow$ plural)

- Possible research directions
  - factored representations of translation and language model
  - continuous space translation model
  - discriminative approaches