eppex: Epochal Phrase Table Extraction for Statistical Machine Translation

Česlav Przywara, Ondřej Bojar
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University in Prague
Outline

- Intro + motivation
- Implementation
  - approximate frequency counting
- Experiments
- Conclusions and future work
Phrase table construction

- Input: parallel corpus + word alignments + phrase extraction algorithm (symmetrisation heuristics)

- Output: phrase table
  
  epochal extraction ||| epochální extrakce  
  p(f|e) lex(f|e) p(e|f) lex(e|f) ...  

- direct and inverse translation probabilities
  
  - p(e|f) = C(e,f) / C(f)  
  - p(f|e) = C(e,f) / C(e)

- lexical weights
  
  - lex(f|e), lex(e|f)

- ...
Phrase table construction in Moses

- Substeps of steps 5 and 6 of `train-model.perl`
  - **phrase extraction** – produces direct and reverse phrase table halves (with word alignments, no scores yet)
  - **gzipping, sorting** and **scoring** of the **direct** table
  - **gzipping, sorting** and **scoring** of the **reverse** table
  - **sorting** of the scored reverse table
  - **consolidation** of the scored direct and reverse tables
  - **gzipping** of the consolidated phrase table

- Optional post-processing:
  - **significance filtering**
Motivation

- phrase table construction is time consuming
  - temporary data are read/written to disk
  - phrase tables size ~ usually several GB or even more
- phrase table quality is not strictly determined by its size
  - *significance filtering* – Johnson et al. (2007)
- more and more physical memory is available
  - laptops ~ 4 GB
  - computational clusters ~ 16 GB (and more) per node
Our inspiration:
- Goyal et al. (2009) used approximate frequency counting for Language Modeling

Our current status:
- extraction of phrase pairs with on the fly filtration implemented via Lossy Counting

Our ultimate goal:
- in-memory phrase table construction (with on-the-fly filtration)
Manku and Motwani (2002)
- approximate frequency counts over stream of data
user defines two parameters: error $\epsilon$ and support $s$ (such that $\epsilon \ll s$
algorithm guarantees ($N =$ number of instances):
- all items whose true frequency exceeds $sN$ are output
- no item whose true frequency is less than $(s-\epsilon)N$ is output
- estimated frequencies are less than the true frequencies by at most $\epsilon N$
- the space used by the algorithm is $O(1/\epsilon \times \log(\epsilon N))$
Lossy Counting algorithm (2)

- input data ~ stream of items conceptually divided into epochs of size $w = \lceil 1/\varepsilon \rceil$
  - $T$ – current epoch ID
- internally maintains database $D$ of triples $(e, f, \Delta)$
  - $e$ – element, $f$ – est. frequency, $\Delta$ – max. error
- new item $e$ arrives
  - if $e$ in $D$: increment $f$ by one
  - otherwise: insert new triple $(e, 1, T-1)$
- pruning at the end of each epoch ($N \equiv 0 \mod w$)
  - remove all triples where $f + \Delta \leq T$
At any time the Lossy Counting algorithm can be asked to produce a list of elements with $f \geq (s - \varepsilon)N$

- such elements satisfies the aforementioned guarantees
- in practice an alternative is also to output all items that survived the pruning so far
eppex implementation

- drop-in alternative to *extract* component from *phrase-extract* toolkit
  - fully compatible input/output format
- written in C++
  - strings stored as C-strings in memory pools (*Boost library*)
  - internally all strings represented by **4-byte** integers
  - Lossy Counting implemented as generic template
- comes with *counter* utility
Usage

Syntax:

```bash
eppex tgt src align extract \ lossy-counter [lossy-counter-2 [lossy-counter-3 [ ... ]]] \ [orientation [--model [wbe|phrase|hier]-[msd|mslr|mono]]]
```

Lossy Counter specification:

- `phrase-pair-length: error: support`
  - 1:0:0 2-4:2e-7:8e-7
  - no pruning of phrase pairs of length 1
  - phrase pairs of length 2-4 stored by one LC with $\epsilon = 2 \times 10^{-7}$ and $s = 8 \times 10^{-7}$
    - 1:0:0 2:2e-7:8e-7 3:2e-7:8e-7 4:2e-7:8e-7
  - similar as above, but phrase pairs of length 2-4 stored in `separate` counters
Usage (in Moses)

- train-model.perl
  - --eppex="1:0:0 2-4:2e-7:8e-7"

- experiment.perl (EMS)
  - config: [TRAINING] > training-options
Experiments environment

- All experiments run on the same machine
  - 64-bit Ubuntu 10.04 server edition
  - 2 Core4 AMD Opteron 2.8 GHz processors
  - 32 GB RAM
  - all input and output files read from and written to a locally mounted disk
Experiments - dataset

- Training data: CzEng corpus with a few additions
  - 8.4M sentence pairs
  - 107.2M English and 93.2M Czech tokens
  - exact setup: Mareček et al. (2011), system "cu-bojar"
- Tuning and testing data: WMT 2011 Translation Task
Experiments – scenarios

- baseline (default approach)
- baseline + sigfilter
  - -l a-e → all 1-1-1 phrase pairs kept in
  - -l a+e → all 1-1-1 phrase pairs removed
  - -n 30 → top n pairs kept (sorted by forward probability)
- eppex 1-in
  - all phrase pairs of length 1–3 kept in
- eppex 1-out
  - all single-occurring phrase pairs removed
<table>
<thead>
<tr>
<th>Experiment</th>
<th>Number of phr. pairs</th>
<th>Gzipped file size</th>
<th>BLEU on wmt10</th>
<th>BLEU on wmt11</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>153.6 M</td>
<td>3.68 GB</td>
<td>17.36</td>
<td>18.22</td>
</tr>
<tr>
<td>sigfilter 30</td>
<td>137.0 M</td>
<td>3.36 GB</td>
<td>17.48</td>
<td>18.13</td>
</tr>
<tr>
<td>sigfilter a-e</td>
<td>92.4 M</td>
<td>2.39 GB</td>
<td>17.23</td>
<td>17.87</td>
</tr>
<tr>
<td>eppex 1-in</td>
<td>57.1 M</td>
<td>1.28 GB</td>
<td>17.60</td>
<td>18.10</td>
</tr>
<tr>
<td>sigfilter a+e</td>
<td>35.0 M</td>
<td>0.86 GB</td>
<td>17.31</td>
<td>17.99</td>
</tr>
<tr>
<td>eppex 1-out</td>
<td>14.4 M</td>
<td>0.33 GB</td>
<td>17.23</td>
<td>17.94</td>
</tr>
</tbody>
</table>
### Experiments – wallclock time

<table>
<thead>
<tr>
<th>Step</th>
<th>baseline</th>
<th>eppex 1-in</th>
<th>eppex 1-out</th>
</tr>
</thead>
<tbody>
<tr>
<td>phr-ext</td>
<td>1152</td>
<td>4360</td>
<td>4361</td>
</tr>
<tr>
<td>gzip</td>
<td>1303</td>
<td>502</td>
<td>246</td>
</tr>
<tr>
<td>sort</td>
<td>5101</td>
<td>1632</td>
<td>1131</td>
</tr>
<tr>
<td>score</td>
<td>20417</td>
<td>7433</td>
<td>712</td>
</tr>
<tr>
<td>sort-inv</td>
<td>1569</td>
<td>129</td>
<td>22</td>
</tr>
<tr>
<td>cons</td>
<td>1361</td>
<td>269</td>
<td>66</td>
</tr>
<tr>
<td>pt-gzip</td>
<td>881</td>
<td>259</td>
<td>65</td>
</tr>
<tr>
<td>TOTAL</td>
<td>31784</td>
<td>14584</td>
<td>6603</td>
</tr>
<tr>
<td>(hh:mm:ss)</td>
<td><strong>8:49:44</strong></td>
<td><strong>4:03:04</strong></td>
<td><strong>1:50:03</strong></td>
</tr>
</tbody>
</table>
## Experiments – sigfilter wallclock time

<table>
<thead>
<tr>
<th></th>
<th>-l a+e</th>
<th>-l a-e</th>
<th>-n 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td></td>
<td></td>
<td>31784</td>
</tr>
<tr>
<td>sigfiltering</td>
<td>18248</td>
<td>18449</td>
<td>1141</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>50032</td>
<td>50233</td>
<td>32925</td>
</tr>
<tr>
<td>Experiment</td>
<td>VM peak</td>
<td>in step</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>baseline</td>
<td>1.1 GB</td>
<td>scoring-e2f</td>
<td></td>
</tr>
<tr>
<td>sigfilter 30</td>
<td>1.1 GB</td>
<td>scoring-e2f</td>
<td></td>
</tr>
<tr>
<td>sigfilter a-e</td>
<td>5.4 GB</td>
<td>sigfilter</td>
<td></td>
</tr>
<tr>
<td>eppex 1-in</td>
<td>19.2 GB</td>
<td>phr-ext</td>
<td></td>
</tr>
<tr>
<td>sigfilter a+e</td>
<td>5.4 GB</td>
<td>sigfilter</td>
<td></td>
</tr>
<tr>
<td>eppex 1-out</td>
<td>16.7 GB</td>
<td>phr-ext</td>
<td></td>
</tr>
</tbody>
</table>
## Old vs. new scorer – wallclock time

<table>
<thead>
<tr>
<th>Step</th>
<th>Baseline (old)</th>
<th>Baseline (new)</th>
</tr>
</thead>
<tbody>
<tr>
<td>phr-ext</td>
<td>1152</td>
<td>1272</td>
</tr>
<tr>
<td>gzip</td>
<td>1303</td>
<td>1354</td>
</tr>
<tr>
<td>sort</td>
<td>5101</td>
<td>4599</td>
</tr>
<tr>
<td>score</td>
<td>20417</td>
<td>7470</td>
</tr>
<tr>
<td>sort-inv</td>
<td>1569</td>
<td>1383</td>
</tr>
<tr>
<td>cons</td>
<td>1361</td>
<td>1419</td>
</tr>
<tr>
<td>pt-gzip</td>
<td>881</td>
<td>849</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>31784</strong></td>
<td><strong>18346</strong></td>
</tr>
<tr>
<td>(hh:mm:ss)</td>
<td><strong>8:49:44</strong></td>
<td><strong>5:05:46</strong></td>
</tr>
</tbody>
</table>
Conclusions

- bulk of phrase pairs to be scored can be significantly reduced
  - 3.68 GB → 1.28 GB
- translation quality can be preserved (BLEU)
  - wmt10: 17.36 → 17.60
  - wmt11: 18.22 → 18.10
- significant RAM requirements
  - 1.1 GB → 19.2 GB
  - not for laptop use...
Future work

- further optimization of memory usage
- integration with memscore – Hardmeier (2010)
- confrontation with larger corpora (Fr-En)

(Ondřej would like me to)

- compare eppex and suffix arrays approach used for incremental training


Questions?

- Any comments and suggestions are appreciated!
  - ceslav@przywara.cz
  - bojar@ufal.mff.cuni.cz
  - moses-support@mit.edu