Efficient Language Modeling Algorithms with Applications to Statistical Machine Translation

Kenneth Heafield

September 20, 2013
CPU and RAM Costs Matter

“had to favor speed over performance” [Moreau et al, 2013]

“could not test whether this result holds in a large scale evaluation” [Durrani et al, 2013]
CPU and RAM Costs Matter

“had to favor speed over performance” [Moreau et al, 2013]

“could not test whether this result holds in a large scale evaluation” [Durrani et al, 2013]

0.5–1.9% BLEU gain from English Gigaword

[Koehn et al, 2012]
This Work

Some of the thesis is already used for speech [Kim et al, 2012; Si et al, 2013].
Some of the thesis is already used for speech [Kim et al, 2012; Si et al, 2013].
Some of the thesis is already used for speech [Kim et al, 2012; Si et al, 2013].
Optical Character Recognition

Written Text

Character Model

Text Corpus

Language Model

Query

Estimate

This Work

Commercial OCR

Lattice Rescoring

Text

Image

Numen (2013) is using most of this thesis for OCR.
Language Models Are Expensive

Store a sparse set of 121 billion $n$-grams $\implies$ RAM
Language Models Are Expensive

Store a sparse set of 121 billion $n$-grams $\implies$ RAM

Millions of probability queries per sentence $\implies$ CPU
Language Models Are Expensive

Store a sparse set of 121 billion $n$-grams $\Rightarrow$ RAM

Millions of probability queries per sentence $\Rightarrow$ CPU

Probability does not multiply when strings are concatenated:

\[ p(\text{saw the man}) \neq p(\text{saw})p(\text{the man}) \]

$\Rightarrow$ Search is hard $\Rightarrow$ CPU
Much of the CPU and RAM cost is due to the language model. Researchers routinely compromise quality due to these costs.
Costs Due To Language Models

Estimation from text
Probability queries
Search when the objective includes log probability
Results Preview

**Speed RAM**

- Estimation from Text: 7.1x 0.07x
- Raw Queries: 2.4x 0.57x
- Decoding: 3.2–10.0x 0.85x

Decoding performance includes ≈1.15x speedup from raw queries.
Baseline: SRILM and cube pruning (more later).
### Outline

<table>
<thead>
<tr>
<th></th>
<th>Speed</th>
<th>RAM</th>
<th>Published</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Estimation from Text</td>
<td>7.1x</td>
<td>0.07x</td>
</tr>
<tr>
<td>2</td>
<td>Raw Queries</td>
<td>2.4x</td>
<td>0.57x</td>
</tr>
<tr>
<td>3</td>
<td>Decoding</td>
<td>3.2–10.0x</td>
<td>0.85x</td>
</tr>
</tbody>
</table>

Decoding performance includes $\approx 1.15x$ speedup from raw queries.
Estimating LMs is Costly

MIT RAM
SRI RAM, time
IRST RAM, time, approximation
Berkeley RAM, time, approximation
Estimating LMs is Costly

- **MIT** RAM
- **SRI** RAM, time
- **IRST** RAM, time, approximation
- **Berkeley** RAM, time, approximation
- **Microsoft** Delay some computation to query time
- **Google** 100–1500 machines, optional stupid backoff
Estimating LMs is Costly

MIT  RAM
SRI  RAM, time
IRST RAM, time, approximation
Berkeley RAM, time, approximation

Microsoft  Delay some computation to query time
Google  100–1500 machines, optional stupid backoff

“When, oh when, will there be an alternative?”
Implz Features

- Disk-based streaming and sorting
- User-specified RAM
- Fast
- Interpolated modified Kneser-Ney

7.7% of SRI’s RAM, 14% of SRI’s wall time
Adjusted counts are:

**Trigrams** Same as counts.

**Others** Number of unique words to the left.
Adjusted counts are:

- **Trigrams** Same as counts.
- **Others** Number of unique words to the left.

**Suffix Sorted**

<table>
<thead>
<tr>
<th>Input</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>are one of</td>
<td>1</td>
</tr>
<tr>
<td>is one of</td>
<td>5</td>
</tr>
<tr>
<td>are two of</td>
<td>3</td>
</tr>
</tbody>
</table>

**Output**

1-gram Adjusted

<table>
<thead>
<tr>
<th>Output</th>
<th>Adjusted</th>
</tr>
</thead>
<tbody>
<tr>
<td>of</td>
<td>2</td>
</tr>
</tbody>
</table>

2-gram Adjusted

<table>
<thead>
<tr>
<th>Output</th>
<th>Adjusted</th>
</tr>
</thead>
<tbody>
<tr>
<td>one of</td>
<td>2</td>
</tr>
<tr>
<td>two of</td>
<td>1</td>
</tr>
</tbody>
</table>
Streaming Framework

Memory is divided into blocks. Blocks are recycled.

- Lazily Merge Input
- Adjust Counts
- Sort Block
- Write to Disk

Prepare for next step.
Adjusted Counts Detail

Lazily merge counts in suffix order

Adjust counts

Sort each block for the next step

Write to disk

Each vertex is a thread $\Rightarrow$ Simultaneous disk and CPU.
Experiment: Toolkit Comparison

Task: Build an unpruned 5-gram LM
Data: Subset of English ClueWeb09 (webpages)
Machine: 64 GB RAM
Output Format: Binary (or ARPA when faster)

IRST disk: 3-way split. Peak RAM of any one process (as if run serially).
Berkeley: Binary search for minimum JVM memory.
This Work 3.9G

SRI compact

SRI disk

SRI

MIT

IRST

IRST disk

Berkeley

RAM (GB)

Tokens (millions)
This Work 3.9G

IRST disk

SRI compact

SRI disk

IRST

MIT

Wall time (hours)

Tokens (millions)
Scaling

<table>
<thead>
<tr>
<th>This Work</th>
<th>Tokens</th>
<th>Smoothing</th>
<th>Machines</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>126 billion</td>
<td>Kneser-Ney</td>
<td>1</td>
<td>2.8</td>
</tr>
</tbody>
</table>

Counts

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>This Work 126B</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Pruned Google 1T</td>
<td>14m</td>
<td>3,775m</td>
<td>17,629m</td>
<td>39,919m</td>
<td>59,794m</td>
</tr>
</tbody>
</table>

(This work used a machine with 140 GB RAM and a RAID5 array.)
# Scaling

<table>
<thead>
<tr>
<th>This Work</th>
<th>Tokens</th>
<th>Smoothing</th>
<th>Machines</th>
<th>Days</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>126 billion</td>
<td>Kneser-Ney</td>
<td>1</td>
<td>2.8</td>
<td>2013</td>
</tr>
<tr>
<td>Google</td>
<td>31 billion</td>
<td>Kneser-Ney</td>
<td>400</td>
<td>2</td>
<td>2007</td>
</tr>
<tr>
<td>Google</td>
<td>1800 billion</td>
<td>Stupid</td>
<td>1500</td>
<td>1</td>
<td>2007</td>
</tr>
</tbody>
</table>

Counts

<table>
<thead>
<tr>
<th>This Work 126B</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Pruned Google 1T</th>
</tr>
</thead>
<tbody>
<tr>
<td>393m</td>
<td>3,775m</td>
<td>17,629m</td>
<td>39,919m</td>
<td>59,794m</td>
<td>14m</td>
<td>315m</td>
</tr>
</tbody>
</table>

(This work used a machine with 140 GB RAM and a RAID5 array.)
<table>
<thead>
<tr>
<th></th>
<th>Czech–English</th>
<th></th>
<th>French–English</th>
<th></th>
<th>Spanish–English</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rank</td>
<td>BLEU</td>
<td>Rank</td>
<td>BLEU</td>
<td>Rank</td>
</tr>
<tr>
<td>This Work</td>
<td>1</td>
<td>28.16</td>
<td>1</td>
<td>33.37</td>
<td>1</td>
</tr>
<tr>
<td>Google</td>
<td>2–3</td>
<td>27.11</td>
<td>2–3</td>
<td>32.62</td>
<td>2</td>
</tr>
<tr>
<td>Baseline</td>
<td>3–5</td>
<td>27.38</td>
<td>2–3</td>
<td>32.57</td>
<td>3–5</td>
</tr>
</tbody>
</table>
Future Work on Estimation

- Pruning
- Linearly interpolate separately trained models → SRI’s ARPA output is misleading.
- More smoothing methods
- Parallelization by data splitting
Outline

1. Estimation from Text  
   Speed: 7.1x  
   RAM: 0.07x  
   Published: ACL 2013

2. Raw Queries  
   Speed: 2.4x  
   RAM: 0.57x  
   Published: WMT 2011

3. Decoding  
   Speed: 3.2–10.0x  
   RAM: 0.85x  
   Published: IWSLT 2011, EMNLP 2012, NAACL 2013

Decoding performance includes ≈ 1.15x speedup from raw queries.

Baseline: SRILM and cube pruning (more later).
Answer language model queries using less time and memory.

\[
\begin{align*}
\log p(\text{iran} \mid <s>) &= -3.33437 \\
\log p(\text{is} \mid <s> \text{ iran}) &= -1.05931 \\
\log p(\text{one} \mid <s> \text{ iran is}) &= -1.80743 \\
\log p(\text{of} \mid <s> \text{ iran is one}) &= -0.03705 \\
\log p(\text{the} \mid \text{ iran is one of}) &= -0.08317 \\
\log p(\text{few} \mid \text{ is one of the}) &= -1.20788
\end{align*}
\]
# Example Language Model

<table>
<thead>
<tr>
<th>Unigrams</th>
<th>Bigrams</th>
<th>Trigrams</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Words</strong></td>
<td><strong>Words</strong></td>
<td><strong>Words</strong></td>
</tr>
<tr>
<td>&lt;s&gt;</td>
<td>&lt;s&gt; iran</td>
<td>&lt;s&gt; iran is</td>
</tr>
<tr>
<td>-∞</td>
<td>-3.3</td>
<td>-1.1</td>
</tr>
<tr>
<td>-2.0</td>
<td>-1.2</td>
<td></td>
</tr>
<tr>
<td>iran</td>
<td>iran is</td>
<td>iran is one</td>
</tr>
<tr>
<td>-4.1</td>
<td>-1.7</td>
<td>-2.0</td>
</tr>
<tr>
<td>-0.8</td>
<td>-0.4</td>
<td>-0.9</td>
</tr>
<tr>
<td>is</td>
<td>is one</td>
<td>is one of</td>
</tr>
<tr>
<td>-2.5</td>
<td>-2.0</td>
<td>-1.4</td>
</tr>
<tr>
<td>-1.4</td>
<td>-0.9</td>
<td>-0.6</td>
</tr>
<tr>
<td>one</td>
<td>one of</td>
<td></td>
</tr>
<tr>
<td>-3.3</td>
<td>-1.4</td>
<td>-0.3</td>
</tr>
<tr>
<td>-0.9</td>
<td>-0.6</td>
<td></td>
</tr>
<tr>
<td>of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Example Queries

<table>
<thead>
<tr>
<th>Unigrams</th>
<th>Bigrams</th>
<th>Trigrams</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Words</strong></td>
<td><strong>log p</strong></td>
<td><strong>log b</strong></td>
</tr>
<tr>
<td>&lt;s&gt;</td>
<td>−∞</td>
<td>−2.0</td>
</tr>
<tr>
<td>iran</td>
<td>−4.1</td>
<td>−0.8</td>
</tr>
<tr>
<td>is</td>
<td>−2.5</td>
<td>−1.4</td>
</tr>
<tr>
<td>one</td>
<td>−3.3</td>
<td>−0.9</td>
</tr>
<tr>
<td>of</td>
<td>−2.5</td>
<td>−1.1</td>
</tr>
</tbody>
</table>

**Query:** <s> iran is

\[ \log p(is \mid <s> \text{ iran}) = −1.1 \]

**Query:** iran is of

\[
\begin{align*}
\log p(of) &= −2.5 \\
\log b(is) &= −1.4 \\
\log b(iran \text{ is}) &= −0.4 \\
\log p(of \mid \text{ iran is}) &= −4.3
\end{align*}
\]
## Trie Based

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMU-Cambridge</td>
<td>Early implementation</td>
</tr>
<tr>
<td>SRI</td>
<td>Popular, considered fast, high-memory</td>
</tr>
<tr>
<td>IRST</td>
<td>Smaller than SRI, single-threaded</td>
</tr>
<tr>
<td>MIT</td>
<td>Batch querying</td>
</tr>
<tr>
<td>TPT</td>
<td>Memory locality</td>
</tr>
<tr>
<td>Joshua</td>
<td>Java, “not as scalable as the SRILM” [Li et al]</td>
</tr>
<tr>
<td>Berkeley</td>
<td>Java</td>
</tr>
</tbody>
</table>

Lossy Low-Memory
- Rand Bloom maps
- Shef Minimal perfect hashing
- Google Minimal perfect hashing, larger than Shef
### Trie Based

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMU-Cambridge</td>
<td>Early implementation</td>
</tr>
<tr>
<td>SRI</td>
<td>Popular, considered fast, high-memory</td>
</tr>
<tr>
<td>IRST</td>
<td>Smaller than SRI, single-threaded</td>
</tr>
<tr>
<td>MIT</td>
<td>Batch querying</td>
</tr>
<tr>
<td>TPT</td>
<td>Memory locality</td>
</tr>
<tr>
<td>Joshua</td>
<td>Java, “not as scalable as the SRILM” [Li et al]</td>
</tr>
<tr>
<td>Berkeley</td>
<td>Java</td>
</tr>
</tbody>
</table>

### Lossy Low-Memory

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rand</td>
<td>Bloom maps</td>
</tr>
<tr>
<td>Shef</td>
<td>Minimal perfect hashing</td>
</tr>
<tr>
<td>Google</td>
<td>Minimal perfect hashing, larger than Shef</td>
</tr>
</tbody>
</table>
KenLM Features

- Faster than all baselines
- Lowest lossless memory
- Multithreaded
- Quick loading via memory mapping
- Easier to compile
Probing Fast.

Trie Small. But still fast.
Probing

Hash every $n$-gram to a 64-bit integer. Ignore collisions. Store $n$-grams in custom linear probing hash tables.

Fastest, 24 bytes/$n$-gram (still less than SRI).
Trie

Reverse $n$-grams, arrange in a trie.

Smaller than most, faster than all but probing.
Optimizing the Trie

**CPU** Interpolation search instead of binary search [Yehoshua et al, 1978]

**RAM** Pack at the bit level i.e. log p has no sign bit
Options to Save More Memory

Cluster floats into $2^f$ bins, store $f$ bits/float.

Chop bits from integer sequences, store offsets.

[Whittaker and Raj, 2001; Raj and Whittaker, 2003]
Experiment: Raw Queries

Task: Score the English Gigaword corpus
Model: 5-gram Europarl + deduped news crawl 2011

Queries/ms: Excludes loading and file reading time
Loaded RAM: Resident after loading
Peak RAM: Peak virtual after scoring
Raw Queries: Exact Variants

- **SRI Compact**
- **IRST Inverted**
- **IRST Loaded**
- **Peak**

This Work Probing

This Work Trie

This Work Chop

- **IRST Inverted**
- **IRST**
- **SRI Compact**
- **MIT**
- **Loaded**
- **Peak**

- Memory (GB)
- Queries/ms

- **Intro**
- **Estimation**
- **Queries**
- **Decoding**
- **State**
- **Score Estimates**
- **Search**
Experiment: Translate 3003 Sentences

Task  WMT 2011 French–English baseline
Decoder  Moses
Model  5-gram Europarl+News LM (same as before)
Formalism  Phrase-based from Europarl

Time  Total wall time, including loading
Memory  Total resident memory after decoding
New: Yasuhara et al, EMNLP 2013

“An Efficient Language Model Using Double-Array Structures”

- 19% less RAM and 4-9% faster than this work’s probing method.
- More RAM than the trie method.
- 4 days to build a data structure with 936 million $n$-grams.
WMT 2013 Adoption: Any Task

This Work 17

SRI 22

IRST 4

Intro  Estimation  Queries  Decoding  State  Score Estimates  Search

13  1  2  1  0  1  8
Decoding performance includes $\approx 1.15x$ speedup from raw queries.
Baseline: SRILM and cube pruning (more later).
Parsing-Based MT is Slow

26 CPU hours to translate 3000 sentences

Le garçon a vu l’homme avec un télescope
Decoding Example: Parse with SCFG

S: S

X: NP

Le garçon

X: VP

a vu

X: VP

l’homme

X: PP

avec un télescope
Decoding Example: Read Target Side

Le garçon
The boy
A boy

a vu
seen
saw
view

l’homme
man
the man
some men

avec un télescope
with the telescope
to an telescope
with a telescope
Decoding Example: One Constituent

\[ S:S \]

\[ X:NP \]

\[ X:VP \]

\[ X:VP \]

\[ X:PP \]

\[ X:NП \]

\[ X:V \]

\[ Lе gаrсоn \]

The boy

A boy

\[ a \ vу \]

seen

saw

view

\[ l'homme \]

man

the man

some men

\[ avec \ uн \ télescope \]

with the telescope
to an telescope
with a telescope
a vu l'homme

Hyp
seen
saw
view

Hyp
man
the man
some men
Hyp: seen
saw
view

X: VP

Hyp: seen man
saw man
view man

Hyp: seen the man
saw the man
view the man

Hyp: seen some men
saw some men
view some men

Hypothesis

a vu l’homme

Intro  Estimation  Queries  Decoding  State  Score Estimates  Search

60
a vu l’homme

Hypothesis | Score
--- | ---
seen man | -8.8
seen the man | -7.6
seen some men | -9.5
saw man | -8.3
saw the man | -6.9
saw some men | -8.5
view man | -8.5
view the man | -8.9
view some men | -10.8
$a \ vu \ l'homme$

Hyp | Score
---|---
seen | -3.8
saw | -4.0
view | -4.0

l'homme

Hyp | Score
---|---
man | -3.6
the man | -4.3
some men | -6.3

Hypothesis | Score
---|---
saw the man | -6.9
seen the man | -7.6
saw man | -8.3
saw some men | -8.5
view man | -8.5
seen man | -8.8
view the man | -8.9
seen some men | -9.5
view some men | -10.8
a vu l’homme

Hypothesis Score
saw the man -6.9
seen the man -7.6
saw man -8.3
saw some men -8.5
view man -8.5
seen man -8.8
view the man -8.9
seen some men -9.5
view some men -10.8

Scores do not sum
Pruning is Approximate
Appending Strings

Hypotheses are built by string concatenation. Language model probability changes when this is done:

\[
\frac{p(\text{saw the man})}{p(\text{saw})p(\text{the man})} = \frac{p(\text{the} | \text{saw})p(\text{man} | \text{saw the})}{p(\text{the})p(\text{man} | \text{the})}
\]
Appending Strings

Hypotheses are built by string concatenation. Language model probability changes when this is done:

\[
\frac{p(\text{saw the man})}{p(\text{saw})p(\text{the man})} = \frac{p(\text{the } | \text{ saw})p(\text{man } | \text{ saw the})}{p(\text{the}) p(\text{man } | \text{ the})}
\]

Log probability is part of the score

\(\implies\) Scores do not sum
\(\implies\) Local decisions may not be globally optimal
\(\implies\) Search is hard.
<table>
<thead>
<tr>
<th>Outline</th>
<th>Speed</th>
<th>RAM</th>
<th>Published</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Estimation from Text</td>
<td>7.1x</td>
<td>0.07x</td>
<td>ACL 2013</td>
</tr>
<tr>
<td>2 Raw Queries</td>
<td>2.4x</td>
<td>0.57x</td>
<td>WMT 2011</td>
</tr>
<tr>
<td>3 Decoding</td>
<td>3.2–10.0x</td>
<td>0.85x</td>
<td></td>
</tr>
<tr>
<td>1 State and Recombination</td>
<td></td>
<td></td>
<td>IWSLT 2011</td>
</tr>
<tr>
<td>2 Score Estimates</td>
<td></td>
<td></td>
<td>EMNLP 2012</td>
</tr>
<tr>
<td>3 New Search Algorithm</td>
<td></td>
<td></td>
<td>NAACL 2013</td>
</tr>
</tbody>
</table>
Appending Strings

Hypotheses are built by string concatenation. Language model probability changes when this is done:

\[ c(\text{saw } \bullet \text{ the man}) = \frac{p(\text{saw the man})}{p(\text{saw})p(\text{the man})} = \frac{p(\text{the } | \text{ saw})p(\text{man } | \text{ saw the})}{p(\text{the})p(\text{man } | \text{ the})} \]

What words does correction \( c \) examine?
Markov Assumption

A 5-gram language model uses up to 4 words of context:

\[ p(\text{man} \mid <s> \text{ the boy saw the}) = p(\text{man} \mid \text{the boy saw the}) \]

\[ \Rightarrow \]

Correction \( c \) examines up to 4 words from each string:

\[ c(<s> \leftarrow \text{the boy saw the} \bullet \text{man with a telescope} \rightarrow .) \]

Right State

End of state

Left State

End of state
Hypotheses have State

\[ X : X \]

Left State
- countries that maintain diplomatic relations with North Korea.
- countries that maintain diplomatic ties with North Korea.
- countries that maintain some diplomatic ties with North Korea.

Right State
- relations ties with North Korea.
- ties with North Korea.
- diplomatic ties with North Korea.

The decoder may recombine hypotheses with equal state.
Smaller state \( \Rightarrow \) More recombination \( \Rightarrow \) Reason over more hypotheses at once \( \Rightarrow \) Improved time-accuracy tradeoff.
The decoder may recombine hypotheses with equal state.
State Controls Recombination

The decoder may recombine hypotheses with equal state.

Smaller state

⇒ More recombination

⇒ Reason over more hypotheses at once

⇒ Improved time-accuracy tradeoff.
Efficiently Minimizing State

Li et al [2008]  Criterion for state minimization. “Inefficient implementation”

This Work (IWSLT 2011)  Repurpose log probability sign bit. Use existing lookups. Encode state to make queries faster.
Efficiently Minimizing State

Li et al [2008] Criterion for state minimization. “Inefficient implementation”

This Work (IWSLT 2011) Repurpose log probability sign bit. Use existing lookups. Encode state to make queries faster.

11% faster than right state minimization alone. On hierarchical Chinese–English with beam size 1000.
Outline

1 Estimation from Text 7.1x 0.07x ACL 2013
2 Raw Queries 2.4x 0.57x WMT 2011
3 Decoding 3.2–10.0x 0.85x
   1 State and Recombination IWSLT 2011
   2 Score Estimates EMNLP 2012
   3 New Search Algorithm NAACL 2013
Baseline: How to Score a Fragment

\[
\begin{align*}
\log p_5(\text{is}) &= -2.63 \\
\log p_5(\text{one} | \text{is}) &= -2.03 \\
\log p_5(\text{of} | \text{is one}) &= -0.24 \\
\log p_5(\text{the} | \text{is one of}) &= -0.47 \\
\log p_5(\text{few} | \text{is one of the}) &= -1.26 \\
\end{align*}
\]

\[
= \log p_5(\text{is one of the few}) = -6.62
\]
The Problem: Lower Order Entries

5-Gram Model: \( \log p_5(\text{is}) = -2.63 \)

Unigram Model: \( \log p_1(\text{is}) = -2.30 \)

Same training data.
The Problem: Lower Order Entries

5-Gram Model: \( \log p_5(\text{is}) = -2.63 \)
Unigram Model: \( \log p_1(\text{is}) = -2.30 \)
Same training data.

In Kneser-Ney, lower orders have adjusted counts.
Build One Model For Each Order

<table>
<thead>
<tr>
<th>Expression</th>
<th>Baseline</th>
<th>Lower</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \log , p_5(\text{is}) )</td>
<td>(-2.63)</td>
<td>(-2.30)</td>
</tr>
<tr>
<td>( \log , p_5(\text{one} \mid \text{is}) )</td>
<td>(-2.03)</td>
<td>(-1.92)</td>
</tr>
<tr>
<td>( \log , p_5(\text{of} \mid \text{is one}) )</td>
<td>(-0.24)</td>
<td>(-0.08)</td>
</tr>
<tr>
<td>( \log , p_5(\text{the} \mid \text{is one of}) )</td>
<td>(-0.47)</td>
<td>(-0.21)</td>
</tr>
<tr>
<td>( \log , p_5(\text{few} \mid \text{is one of the}) )</td>
<td>(-1.26)</td>
<td>(-1.26)</td>
</tr>
<tr>
<td>( \log , p_5(\text{is one of the few}) )</td>
<td>(-6.62)</td>
<td>(-5.77)</td>
</tr>
</tbody>
</table>
Storing Lower Order Models

One extra float per entry, except for longest order.

**Unigrams**

<table>
<thead>
<tr>
<th>Words</th>
<th>$\log p_5$</th>
<th>$\log b_5$</th>
<th>$\log p_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>australia</td>
<td>−3.9</td>
<td>−0.6</td>
<td>−3.6</td>
</tr>
<tr>
<td>is</td>
<td>−2.6</td>
<td>−1.5</td>
<td>−2.3</td>
</tr>
<tr>
<td>one</td>
<td>−3.4</td>
<td>−1.0</td>
<td>−2.9</td>
</tr>
<tr>
<td>of</td>
<td>−2.5</td>
<td>−1.1</td>
<td>−1.7</td>
</tr>
</tbody>
</table>

**No need for backoff** $b_1$

If backoff occurs, use of $p_5$ is appropriate.
One extra float per entry, except for longest order.

**Unigrams**

<table>
<thead>
<tr>
<th>Words</th>
<th>$\log p_5$</th>
<th>$\log b_5$</th>
<th>$\log p_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>australia</td>
<td>$-3.9$</td>
<td>$-0.6$</td>
<td>$-3.6$</td>
</tr>
<tr>
<td>is</td>
<td>$-2.6$</td>
<td>$-1.5$</td>
<td>$-2.3$</td>
</tr>
<tr>
<td>one</td>
<td>$-3.4$</td>
<td>$-1.0$</td>
<td>$-2.9$</td>
</tr>
<tr>
<td>of</td>
<td>$-2.5$</td>
<td>$-1.1$</td>
<td>$-1.7$</td>
</tr>
</tbody>
</table>

No need for backoff $b_1$

If backoff occurs, use of $p_5$ is appropriate.

Related: store upper bounds [Carter et al., also EMNLP 2012].
So far  Better estimates but more memory.
Next    Worse estimates with less memory.
Assume backoff all the way to unigrams.

\[ q(\text{is one of}) = p(\text{is one of}) b(\text{is one of}) b(\text{one of}) b(\text{of}) \]

**Sentence Scores Are Unchanged**

\[ q(<s> \cdots </s>) = p(<s> \cdots </s>) \]

because \( b(\cdots </s>) = 1 \)
Assume backoff all the way to unigrams.

\[ q(\text{is one of}) = p(\text{is one of}) b(\text{is one of}) b(\text{one of}) b(\text{of}) \]

**Sentence Scores Are Unchanged**

\[ q(<s> \cdots </s>) = p(<s> \cdots </s>) \]

because \( b(\cdots </s>) = 1 \)

**Telescoping**

\[ q(\text{is}) = p(\text{is}) b(\text{is}) \]

\[ q(\text{one} \mid \text{is}) = p(\text{one} \mid \text{is}) \frac{b(\text{is one}) b(\text{one})}{b(\text{is})} \]
### Unigrams

<table>
<thead>
<tr>
<th>Words</th>
<th>$\log p_5$</th>
<th>$\log b_5$</th>
<th>Unigrams</th>
<th>$\log q_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>australia</td>
<td>$-3.9$</td>
<td>$-0.6$</td>
<td>australia</td>
<td>$-4.5$</td>
</tr>
<tr>
<td>is</td>
<td>$-2.6$</td>
<td>$-1.5$</td>
<td>is</td>
<td>$-4.1$</td>
</tr>
<tr>
<td>one</td>
<td>$-3.4$</td>
<td>$-1.0$</td>
<td>one</td>
<td>$-4.4$</td>
</tr>
<tr>
<td>of</td>
<td>$-2.5$</td>
<td>$-1.1$</td>
<td>of</td>
<td>$-3.6$</td>
</tr>
</tbody>
</table>

One less float per entry, except for longest order.

Backoff smoothing with RAM comparable to stupid backoff’s counts. Includes Kneser-Ney.
Experiments

Task  WMT 2011 German-English
Decoder  Moses with probing LM + state minimization
LM  5-gram from Europarl, news commentary, and news
Grammar  Target-syntax and hierarchical systems
Parser  Collins
Hierarchical Model Score

Average model score vs. CPU seconds/sentence for different models:

- Lower Order
- Lower Order + Pessimistic
- Baseline
- Pessimistic

The graph shows the trade-off between model score and computational cost, with different lines representing different model configurations.
Effect of adding or removing a float per entry.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Baseline (MB)</th>
<th>Change (MB)</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probing</td>
<td>4,072</td>
<td>517</td>
<td>13%</td>
</tr>
<tr>
<td>Trie</td>
<td>2,647</td>
<td>506</td>
<td>19%</td>
</tr>
<tr>
<td>8-bit quantized trie</td>
<td>1,236</td>
<td>140</td>
<td>11%</td>
</tr>
<tr>
<td>8-bit minimal perfect hash</td>
<td>540</td>
<td>140</td>
<td>26%</td>
</tr>
</tbody>
</table>
### Outline

<table>
<thead>
<tr>
<th></th>
<th>Speed</th>
<th>RAM</th>
<th>Published</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Estimation from Text</td>
<td>7.1x</td>
<td>0.07x</td>
</tr>
<tr>
<td>2</td>
<td>Raw Queries</td>
<td>2.4x</td>
<td>0.57x</td>
</tr>
<tr>
<td>3</td>
<td>Decoding</td>
<td>3.2–10.0x</td>
<td>0.85x</td>
</tr>
</tbody>
</table>

#### 3 Decoding

1. State and Recombination
2. Score Estimates
3. **New Search Algorithm**

- IWSLT 2011
- EMNLP 2012
- NAACL 2013

Decoding performance includes ≈1.15x speedup from raw queries.

Baseline: SRILM and cube pruning (more later).

---

Intro

Estimation

Queries

Decoding

State

Score Estimates

Search
Pruning is Approximate
## Beam Search [Lowerre, 1976; Chiang, 2005]

<table>
<thead>
<tr>
<th>Verb</th>
<th>phrase</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>seen</td>
<td>man</td>
<td>-3.6</td>
</tr>
<tr>
<td></td>
<td>the man</td>
<td>-4.3</td>
</tr>
<tr>
<td></td>
<td>some men</td>
<td>-6.3</td>
</tr>
<tr>
<td></td>
<td>seen man</td>
<td>-8.8</td>
</tr>
<tr>
<td></td>
<td>seen the man</td>
<td>-7.6</td>
</tr>
<tr>
<td></td>
<td>seen some men</td>
<td>-9.5</td>
</tr>
<tr>
<td></td>
<td>saw man</td>
<td>-8.3</td>
</tr>
<tr>
<td></td>
<td>saw the man</td>
<td>-6.9</td>
</tr>
<tr>
<td></td>
<td>saw some men</td>
<td>-8.5</td>
</tr>
<tr>
<td></td>
<td>view man</td>
<td>-8.5</td>
</tr>
<tr>
<td></td>
<td>view the man</td>
<td>-8.9</td>
</tr>
<tr>
<td></td>
<td>view some men</td>
<td>-10.8</td>
</tr>
</tbody>
</table>
Baseline: Cube Pruning [Chiang, 2007]

man $-3.6$  the man $-4.3$  some men $-6.3$

seen $-3.8$  Queue
saw $-4.0$
view $-4.0$

Queue
Hypothesis  Sum
$\rightarrow$seen man $-3.8 - 3.6 = -7.4$
Baseline: Cube Pruning [Chiang, 2007]

<table>
<thead>
<tr>
<th></th>
<th>man</th>
<th>the man</th>
<th>some men</th>
</tr>
</thead>
<tbody>
<tr>
<td>seen</td>
<td>−3.8</td>
<td>−4.3</td>
<td>−6.3</td>
</tr>
<tr>
<td>seen man</td>
<td>−8.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>saw</td>
<td>−4.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>view</td>
<td>−4.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Queue

Hypothesis

<table>
<thead>
<tr>
<th></th>
<th>saw man</th>
<th>seen the man</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>−4.0 − 3.6 = −7.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>−3.8 − 4.3 = −8.1</td>
<td></td>
</tr>
</tbody>
</table>
Baseline: Cube Pruning [Chiang, 2007]

<table>
<thead>
<tr>
<th>Word</th>
<th>Score</th>
<th>Hypothesis</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>seen</td>
<td>−3.8</td>
<td>seen man</td>
<td>−8.8</td>
</tr>
<tr>
<td>saw</td>
<td>−4.0</td>
<td>saw man</td>
<td>−8.3</td>
</tr>
<tr>
<td>view</td>
<td>−4.0</td>
<td>Queue</td>
<td>−8.5</td>
</tr>
<tr>
<td>man</td>
<td>−3.6</td>
<td>Queue</td>
<td>−7.6</td>
</tr>
<tr>
<td>the man</td>
<td>−4.3</td>
<td>Queue</td>
<td>−8.1</td>
</tr>
<tr>
<td>some men</td>
<td>−6.3</td>
<td></td>
<td>−8.3</td>
</tr>
</tbody>
</table>
Problem With Cube Pruning

Hypothesis

countries that

country

Hypothesis

is a countries that

are a countries which

are a countries which

No notion that “a countries” is bad.
Problem With Cube Pruning

Hypothesis is a countries that are a countries which country

Hypothesis is a countries that are a countries which

No notion that “a countries” is bad.

Idea: group by outermost words.
Example Hypotheses

**Left State**
- Countries that do not maintain diplomatic relations with North Korea.
- Countries that have an embassy in DPR Korea.
- Country that maintains some diplomatic ties in North Korea.
- Nations which have some diplomatic ties with DPR Korea.
- Country that maintains some diplomatic ties with DPR Korea.

**Right State**
- Countries that maintain diplomatic relations with North Korea.
- Countries that have an embassy in DPR Korea.
- Country in North Korea.
- Country with DPR Korea.
- Country with DPR Korea.
## Example Hypotheses

<table>
<thead>
<tr>
<th>Left State</th>
<th>Right State</th>
</tr>
</thead>
<tbody>
<tr>
<td>(countries that ⧫ ⦃ ⧬ with North Korea .)</td>
<td></td>
</tr>
<tr>
<td>(nations which has ⧫ ⦃ ⧬ with DPR Korea .)</td>
<td></td>
</tr>
<tr>
<td>(countries that have ⧫ ⦃ ⧬ DPR Korea .)</td>
<td></td>
</tr>
<tr>
<td>(country ⧫ ⦃ ⧬ in North Korea .)</td>
<td></td>
</tr>
<tr>
<td>(country ⧫ ⦃ ⧬ with DPR Korea .)</td>
<td></td>
</tr>
</tbody>
</table>

- ⧫ Left state is completely present.

- ⦃ Stands for elided words

- ⧬ Right state is completely present.
Reveal Common Words in Each Group

(countries that \( \diamond \) Korea .)

\((\varepsilon \diamond \varepsilon)\) \rightarrow (nations which has \( \vdash \diamond \vdash \) with DPR Korea .)

(country \( \vdash \diamond \) Korea .)
Alternate Sides Until Tree is Full

\[(\epsilon \diamond \epsilon) \rightarrow (countries \ that \ have \ \dashv \diamond \dashv \ with \ DPR \ Korea)\]

\[(\epsilon \diamond \epsilon) \rightarrow (countries \ that \ \diamond \ Korea)\]

\[(\epsilon \diamond \epsilon) \rightarrow (nations \ which \ has \ \dashv \diamond \dashv \ with \ DPR \ Korea)\]

\[(\epsilon \diamond \epsilon) \rightarrow (country \ \dashv \diamond \dashv \ in \ North \ Korea)\]

\[(\epsilon \diamond \epsilon) \rightarrow (country \ \dashv \diamond \dashv \ with \ DPR \ Korea)\]
Using Rules

\[ X:NP1 \quad \text{turns into} \quad X:V1 \quad \text{the} \quad X:N2 \]

\[ (\epsilon \diamond \epsilon) \quad \text{turns into} \quad (\epsilon \diamond \epsilon) \]

\[ X:V1 \quad X:N2 \]
Exploring and Backtracking

Does the LM like “is a (countries that Korea .) </s>”? 

Yes Try more detail.

No Consider alternatives.
Exploring and Backtracking

Does the LM like “is a (countries that ◇ Korea .) </s>”?

Yes  Try more detail.
No   Consider alternatives.

Formally: priority queue containing breadcrumbs.
Split and Leave Breadcrumbs

(countries that $\vdash \lozenge \vdash$ with North Korea.)

(countries that $\lozenge$ Korea.)

(countries that have $\vdash \lozenge \vdash$ DPR Korea.)

($\epsilon \lozenge \epsilon$) $\rightarrow$ (nations which has $\vdash \lozenge \vdash$ with DPR Korea.)

(countries that have $\vdash \lozenge \vdash$ with DPR Korea.)

(country $\vdash \lozenge \vdash$ in North Korea.)

(country $\vdash \lozenge \vdash$ in North Korea.)

(country $\vdash \lozenge \vdash$ with DPR Korea.)

(country $\vdash \lozenge \vdash$ with DPR Korea.)
Split and Leave Breadcrumbs

\[(\epsilon \diamond \epsilon) \rightarrow (nations \ which \ has \ \neg \diamond \neg \ with \ DPR \ Korea .)\]

\[(\epsilon \diamond \epsilon) \rightarrow (countries \ that \ have \ \neg \diamond \neg \ with \ DPR \ Korea .)\]

\[(\epsilon \diamond \epsilon) \rightarrow (countries \ that \ can \ \neg \diamond \neg \ with \ North \ Korea .)\]

\[(\epsilon \diamond \epsilon) \rightarrow (countries \ which \ have \ \neg \diamond \neg \ with \ DPR \ Korea .)\]

\[(\epsilon \diamond \epsilon) \rightarrow (countries \ that \ have \ \neg \diamond \neg \ in \ North \ Korea .)\]

\[(\epsilon \diamond \epsilon) \rightarrow (countries \ that \ have \ \neg \diamond \neg \ in \ North \ Korea .)\]

\[(\epsilon \diamond \epsilon) \rightarrow (countries \ that \ have \ \neg \diamond \neg \ with \ DPR \ Korea .)\]
The queue entry

is a \((\epsilon \bowtie \epsilon) \</s>\)

splits into

Zeroth Child “is a (countries that \diamond Korea .) \</s>”

Other Children “is a \((\epsilon \diamond \epsilon)[1+] \</s>”

Children except the zeroth.
A priority queue contains competing entries:

\[
\begin{align*}
&\text{is a (countries that } \diamond \text{ Korea .)} \text{ </s> } \\
&(\varepsilon \diamond \varepsilon) \text{ the } (\varepsilon \diamond \varepsilon) \\
&\text{is a } (\varepsilon \diamond \varepsilon)[1+] \text{ </s>} \\
\end{align*}
\]

The algorithm pops the top entry, splits a non-terminal, and pushes.
A priority queue contains competing entries:

\[
\begin{align*}
&\text{is a (countries that } \diamond \text{ Korea .} \text{)} \langle /s \rangle \\
&(\epsilon \diamond \epsilon) \text{ the } (\epsilon \diamond \epsilon) \\
&\text{is a } (\epsilon \diamond \epsilon)[1+] \langle /s \rangle
\end{align*}
\]

The algorithm pops the top entry, splits a non-terminal, and pushes.

Next: Scoring queue entries
Scores come from the best descendant:

\[
\begin{align*}
\text{Score}(\epsilon \diamond \epsilon) &= \\
\text{Score(countries that } &\not\vdash \diamond \vdash \text{ with North Korea .}) \\
\geq \\
\text{Score}(\epsilon \diamond \epsilon)[1+] &= \\
\text{Score(nations which has } &\not\vdash \diamond \vdash \text{ with DPR Korea .})
\end{align*}
\]
is a $\epsilon \diamond \epsilon$ </s> is a (countries that $\diamond$ Korea .) </s>

$p(\text{is})$
$p(\text{a} \mid \text{is})$
$p(\text{countries})$
$p(\text{that} \mid \text{countries})$
$p(</s>)$

$p(\text{is})$
$p(\text{a} \mid \text{is})$
$p(\text{countries} \mid \text{is a})$
$p(\text{that} \mid \text{is a countries})$
$p(</s> \mid \text{Korea .})$
Summary: Processing a Constituent

1. **Initialize**: Push rules onto a priority queue.

2. **Best-First Loop**:
   1. Pop the top entry.
   2. If it’s complete, add to the beam. Otherwise, split and push.

3. **Finalize**: Convert the beam to a tree (lazily).
Summary: Processing a Constituent

1. **Initialize:** Push rules onto a priority queue.

2. **Best-First Loop:**
   1. Pop the top entry.
   2. If it’s complete, add to the beam. Otherwise, split and push.

3. **Finalize:** Convert the beam to a tree (lazily).

Process constituents in bottom-up order (like cube pruning).
Coarse-to-Fine  [Zhang et al, 2008; Petrov et al, 2008]

Decode multiple times, each with more detail:

- LM order
- Word classes
Coarse-to-Fine [Zhang et al, 2008; Petrov et al, 2008]

Decode multiple times, each with more detail:
- LM order
- Word classes

Key Difference

Coarse-to-Fine | Lock-step refinement
-----|----------------------
This Work     | Locally refine on demand

Future Work
- Use this work for each decoding pass
- Word classes for this work
Exact Algorithms

- Weighted finite state transducers [Iglesias et al, 2011]
- Integer linear programming [Rush et al, 2011]
- Later: upper bounds and LM refinement [Aziz et al, WMT 2013]

Currently intractable for large MT (7 hours for a 7-word sentence)

⇒ Used as first pass of approximate coarse-to-fine.
Exact Algorithms

- Weighted finite state transducers [Iglesias et al, 2011]
- Integer linear programming [Rush et al, 2011]
- Later: upper bounds and LM refinement [Aziz et al, WMT 2013]

Currently intractable for large MT (7 hours for a 7-word sentence)

⇒ Used as first pass of approximate coarse-to-fine.

Key Difference

Approximation based on average-case scores before expanding hypotheses.
Experiment

Task  WMT 2011 German-English
Built  [Koehn et al, 2011]
Model  Hierarchical
Decoder  Moses
Baseline  Queries + State + Rest Costs
Moses Hierarchical

This work
Cube pruning
Additive cube pruning
This work
Cube pruning
Additive cube pruning
Summary

Optimized the entire LM pipeline from estimation to search.
Summary

Optimized the entire LM pipeline from estimation to search.

Comparison

<table>
<thead>
<tr>
<th>Task</th>
<th>WMT 2011 German-English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built</td>
<td>[Koehn et al, 2011]</td>
</tr>
<tr>
<td>Model</td>
<td>Hierarchical</td>
</tr>
<tr>
<td>Decoder</td>
<td>Moses</td>
</tr>
</tbody>
</table>
Average model score

CPU seconds/sentence

-101.7
-101.6
-101.5
-101.4

-101.7
-101.6
-101.5
-101.4

Cube pruning with SRILM
Queries and State
+Rest Costs
+Search
## Decoder Support

<table>
<thead>
<tr>
<th>Feature</th>
<th>Moses</th>
<th>cdec</th>
<th>Joshua</th>
<th>Jane</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Queries</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>State</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rest Costs</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Search</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[kheafield.com/code]
Questions?
Keep only words that might form cross-hypothesis $n$-grams.

**Left State [Joshua]**

For any word $w$, does the model contain:

- $w$ countries that maintain some  $\times$
- $w$ countries that maintain  $\times$
- $w$ countries that  $\checkmark$

$\Rightarrow$ Left state minimizes to “countries that” $\land$.
Full State Minimization

Keep only words that might form cross-hypothesis $n$-grams.

**Left State [Joshua]**

For any word $w$, does the model contain:

- $w$ countries that maintain some $\times$
- $w$ countries that maintain $\times$
- $w$ countries that $\checkmark$

$\implies$ Left state minimizes to “countries that” $\vdash$.

**Right State [SRI, Rand, Joshua]**

For any word $w$, does the model contain:

- with North Korea . $w \checkmark$

$\implies$ Right state minimizes to $\vdash$ “with North Korea .”
## Related Work on State

<table>
<thead>
<tr>
<th>Joshua</th>
<th>Left and right but “inefficient implementation”</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRI</td>
<td>Right only, additional lookups</td>
</tr>
</tbody>
</table>

**This Work**
- Repurpose memory, existing lookups
- Also: encode state to make queries faster
## Experimental Setup

<table>
<thead>
<tr>
<th>Task</th>
<th>NIST Chinese–English [Koehn, 2011]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM</td>
<td>Xinhua and AFP from English Gigaword 4 + Parallel Data</td>
</tr>
<tr>
<td>Grammar</td>
<td>Hierarchical</td>
</tr>
<tr>
<td>Decoder</td>
<td>Moses with cube pruning and faster raw queries</td>
</tr>
</tbody>
</table>
Experimental Setup

<table>
<thead>
<tr>
<th>Task</th>
<th>NIST Chinese–English [Koehn, 2011]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM</td>
<td>Xinhua and AFP from English Gigaword 4 + Parallel Data</td>
</tr>
<tr>
<td>Grammar</td>
<td>Hierarchical</td>
</tr>
<tr>
<td>Decoder</td>
<td>Moses with cube pruning and faster raw queries</td>
</tr>
</tbody>
</table>

11% faster
Experiments: Systems

**Hierarchical with Moses [Koehn, 2012]**
- German–English also ported to cdec, Joshua, and Jane
- English–German
- Chinese–English

**Target Syntax with Moses [Koehn, 2012]**
- German–English
- English–German

**Tree-to-Tree with cdec [Ammar et al, 2013]**
- French–English
Experiments: Systems and Scenarios

<table>
<thead>
<tr>
<th>Method</th>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchical with Moses</td>
<td>German–English,</td>
</tr>
<tr>
<td></td>
<td>English–German,</td>
</tr>
<tr>
<td></td>
<td>Chinese–English</td>
</tr>
<tr>
<td>Target Syntax with Moses</td>
<td>German–English,</td>
</tr>
<tr>
<td></td>
<td>English–German</td>
</tr>
<tr>
<td>Tree-to-Tree with cdec</td>
<td>French–English</td>
</tr>
</tbody>
</table>

Baseline and improved rest costs, 2–3 flavors of cube pruning.