Scalable Modified Kneser-Ney Language Model Estimation

Kenneth Heafield Ivan Pouzyrevsky Jonathan H. Clark
Philipp Koehn

University of Edinburgh, Carnegie Mellon, Yandex

6 August, 2013

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

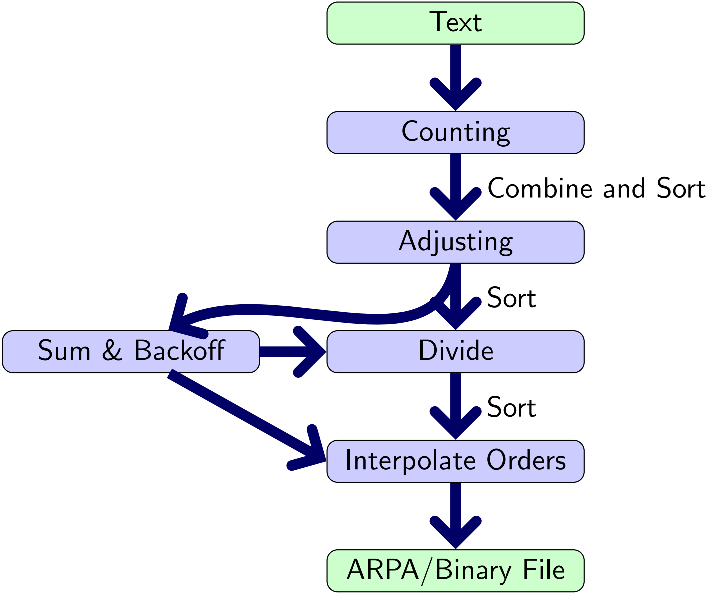
This Work

- 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

Outline

1. Estimation Pipeline
2. Streaming and Sorting
3. Experiments


Text
Counting
Combine and Sort
Adjusting
Sort
Sum & Backoff
Divide
Sort
Interpolate Orders
ARPA/Binary File

Counting

<s> Australia is one of

↓

3-gram	**Count**
<s> Australia is	1
Australia is one	1
is one of	1

Combine in a hash table, spill to merge sort.

Adjusting

Adjusted counts are:

Trigrams Same as counts.

Others Number of unique words to the left.

Adjusting

Adjusted counts are:

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

Suffix Sorted Input

3	2	1	Count
are	one	of	1
is	one	of	5
are	two	of	3

→

Output

1-gram	Adjusted
of	2

Output

2-gram	Adjusted
one of	2
two of	1

Calculating Discounts

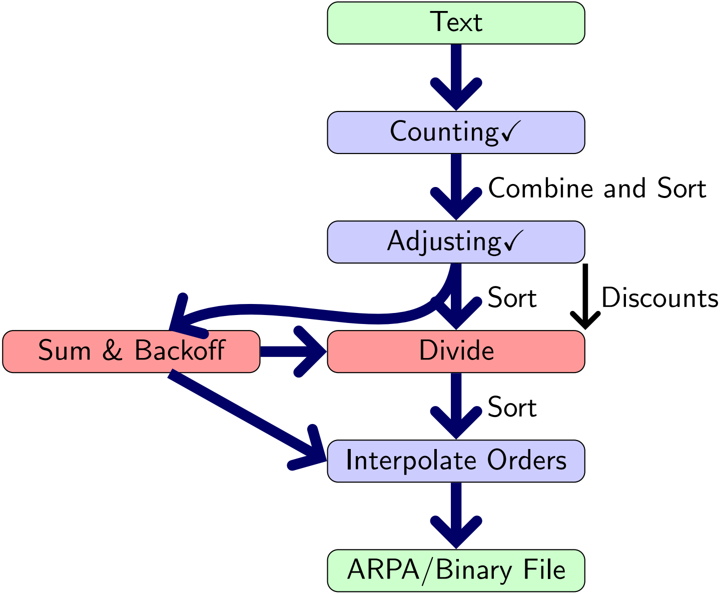
Count singletons, doubletons, tripletons, and quadrupletons for each order.

↓

Chen and Goodman

↓

discount$_n$


Text
Counting✓
Combine and Sort
Adjusting✓
Sort
Discounts
Sum & Backoff
Divide
Sort
Interpolate Orders
ARPA/Binary File

Discounting and Normalization

Save mass for unseen events

$$\text{pseudo}(w_n|w_1^{n-1}) = \frac{\text{adjusted}(w_1^n) - \text{discount}_n(\text{adjusted}(w_1^n))}{\sum_x \text{adjusted}(w_1^{n-1}x)}$$

Normalize

Discounting and Normalization

Save mass for unseen events

$$\text{pseudo}(w_n | w_1^{n-1}) = \frac{\text{adjusted}(w_1^n) - \text{discount}_n(\text{adjusted}(w_1^n))}{\sum_x \text{adjusted}(w_1^{n-1} x)}$$

Normalize

Context Sorted Input

2	1	3	Adjusted
are	one	of	1
are	one	that	2
is	one	of	5

→

Output

3-gram	Pseudo
are one of	0.26
are one that	0.47
is one of	0.62

Denominator Looks Ahead

Save mass for unseen events

$$\text{pseudo}(w_n | w_1^{n-1}) = \frac{\text{adjusted}(w_1^n) - \text{discount}_n(\text{adjusted}(w_1^n))}{\sum_x \text{adjusted}(w_1^{n-1}x)}$$

Normalize

Context Sorted Input

2 1	3	Adjusted
are one	of	**1**
are one	that	**2**
is one	of	5

→

Output

3-gram	Pseudo
are one of	**0.26**
are one that	0.47
is one of	0.62

Two Threads

Sum Thread

2	1	3	Adjusted
are	one	of	1
are	one	that	2
is	one	of	5

Reads ahead and sums

sum=3

Divide Thread

2	1	3	Adjusted
are	one	of	1
are	one	that	2
is	one	of	5

Reads behind to normalize

Computing Backoffs

Backoffs are penalties for unseen events.

Bin the entries "are one x" by their adjusted counts

$$\begin{aligned} \text{continue(are one)} = (&\text{number with adjusted count 1,} \\ &\ldots\text{adjusted count 2,} \\ &\ldots\text{adjusted count} \geq 3) \end{aligned}$$

Computing Backoffs

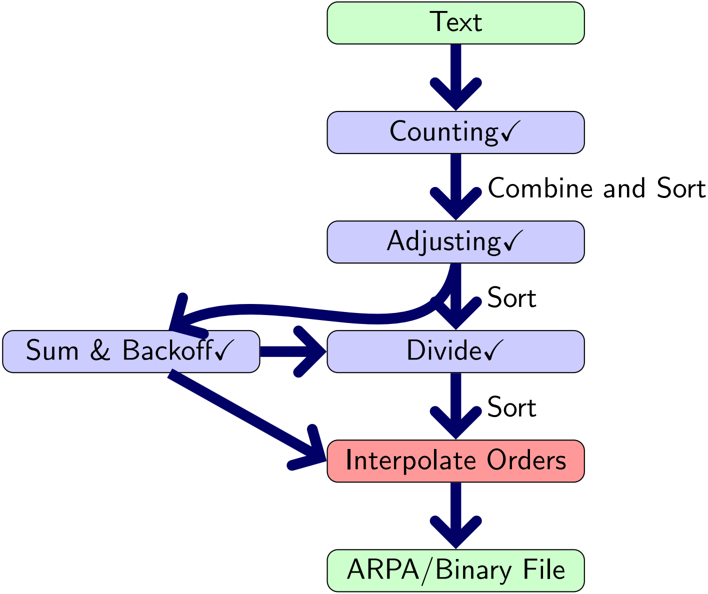
Backoffs are penalties for unseen events.

Bin the entries "are one x" by their adjusted counts

$$\text{continue(are one)} = (\text{number with adjusted count } 1, \\ \ldots\text{adjusted count } 2, \\ \ldots\text{adjusted count } \geq 3)$$

Compute backoff in the sum thread

$$\text{backoff(are one)} = \frac{\text{continue(are one)} \cdot \text{discount}_3}{\sum_x \text{adjusted(are one } x)}$$


Text
Counting✓
Combine and Sort
Adjusting✓
Sort
Sum & Backoff✓
Divide✓
Sort
Interpolate Orders
ARPA/Binary File

Interpolate Orders

Interpolate unigrams with the uniform distribution.

$$p(\text{of}) = \text{pseudo}(\text{of}) + \text{backoff}(\epsilon)\frac{1}{|\text{vocabulary}|}$$

Interpolate Orders

Interpolate unigrams with the uniform distribution,

$$p(\text{of}) = \text{pseudo}(\text{of}) + \text{backoff}(\epsilon)\frac{1}{|\text{vocabulary}|}$$

Interpolate bigrams with unigrams, etc.

$$p(\text{of}|\text{one}) = \text{pseudo}(\text{of}|\text{one}) + \text{backoff}(\text{one})p(\text{of})$$

Interpolate Orders

Interpolate unigrams with the uniform distribution,

$$p(\text{of}) = \text{pseudo}(\text{of}) + \text{backoff}(\epsilon)\frac{1}{|\text{vocabulary}|}$$

Interpolate bigrams with unigrams, etc.

$$p(\text{of}|\text{one}) = \text{pseudo}(\text{of}|\text{one}) + \text{backoff}(\text{one})p(\text{of})$$

Suffix Lexicographic Sorted Input

n-gram	pseudo	interpolation weight
of	0.1	backoff(ϵ) = 0.1
one of	0.2	backoff(one) = 0.3
are one of	0.4	backoff(are one) = 0.2

→

Output

n-gram	*p*
of	0.110
one of	0.233
are one of	0.447

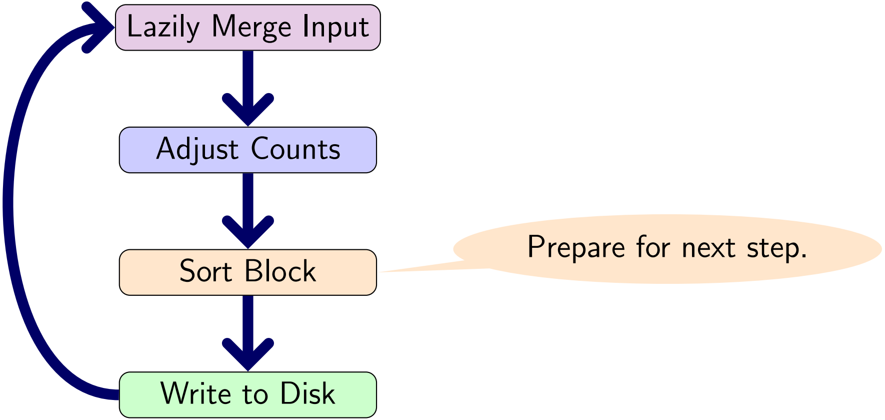
Summary

Compute interpolated modified Kneser-Ney without pruning in
Four streaming passes and three sorts.

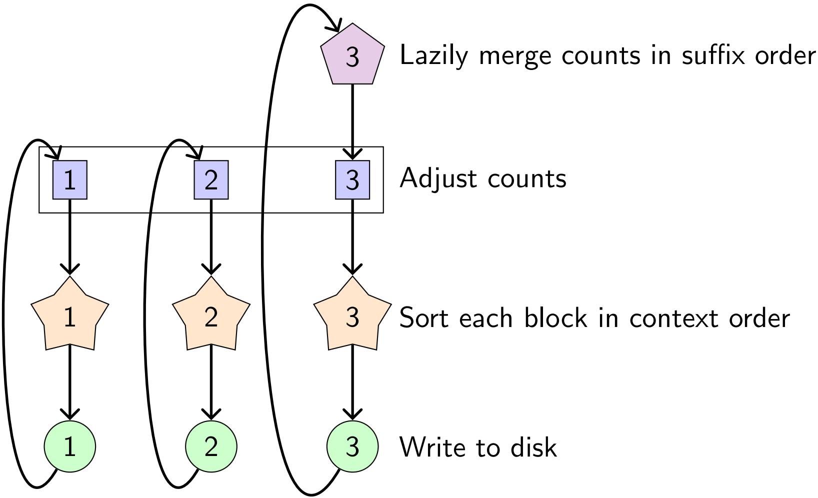
How do we make this efficient?

Streaming Framework

Memory is divided into blocks. Blocks are recycled.



Adjusted Counts Detail

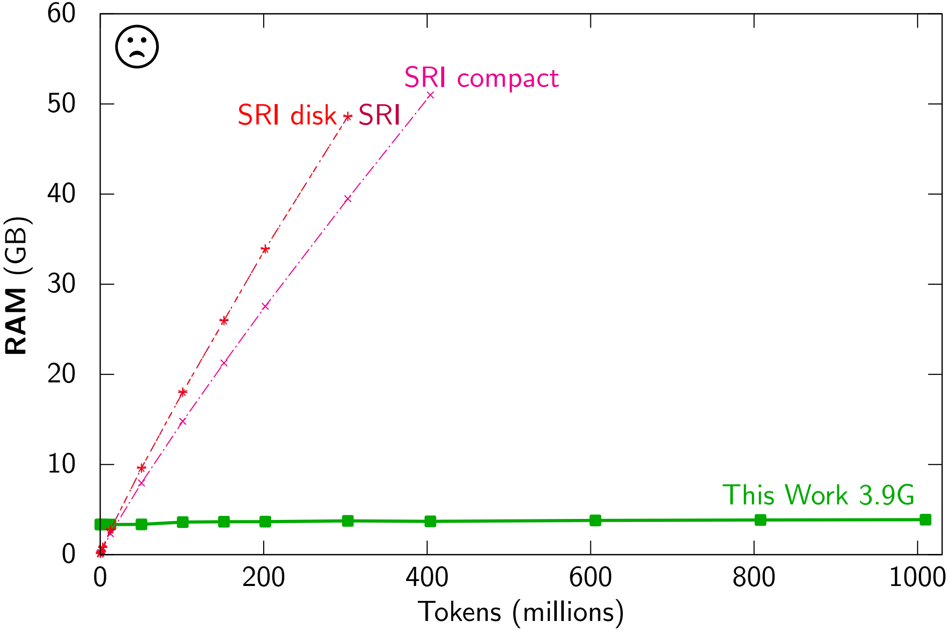


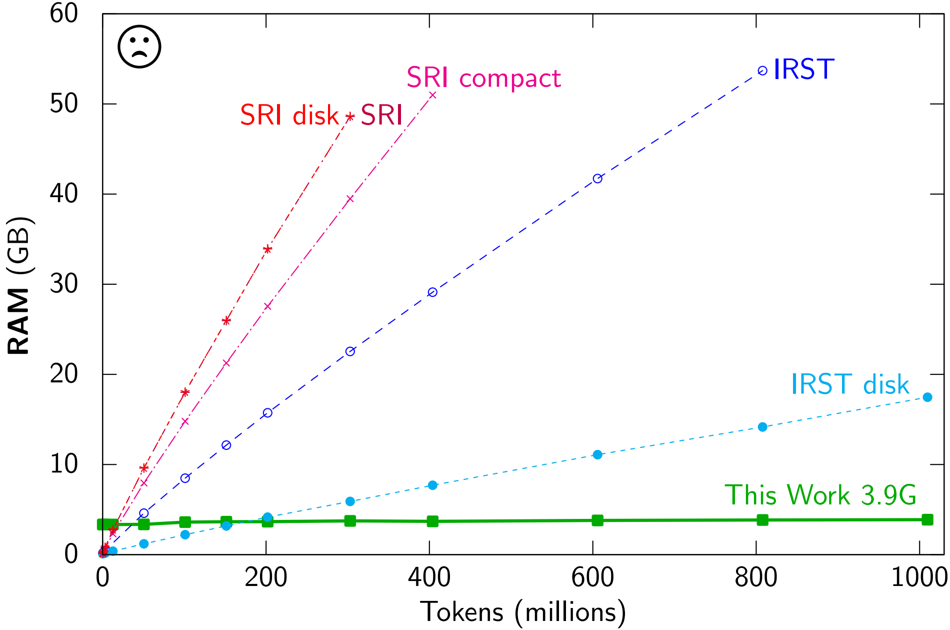
Each vertex is a thread $\Longrightarrow$ Simultaneous disk and CPU.

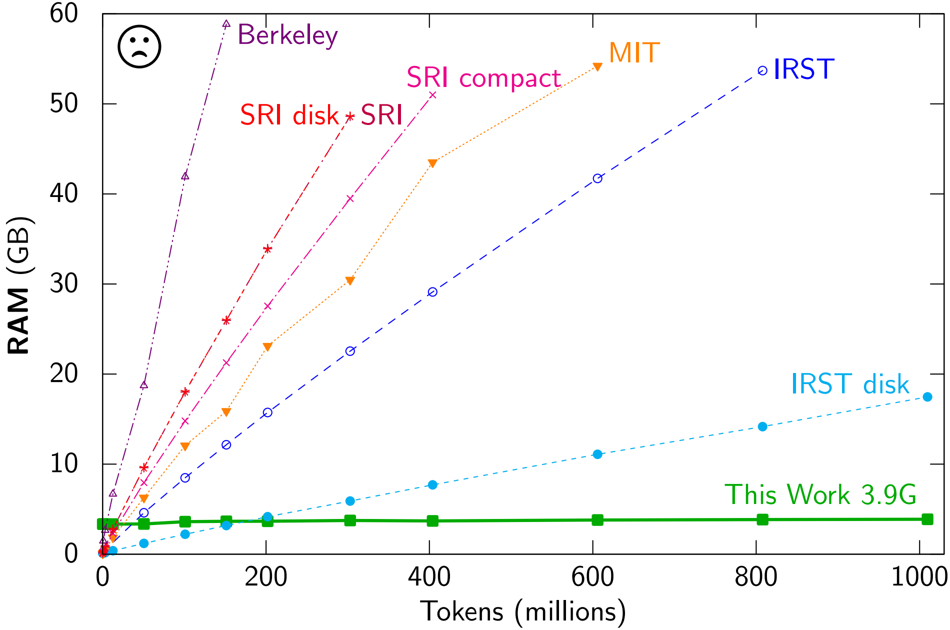
Experiment: Toolkit Comparison

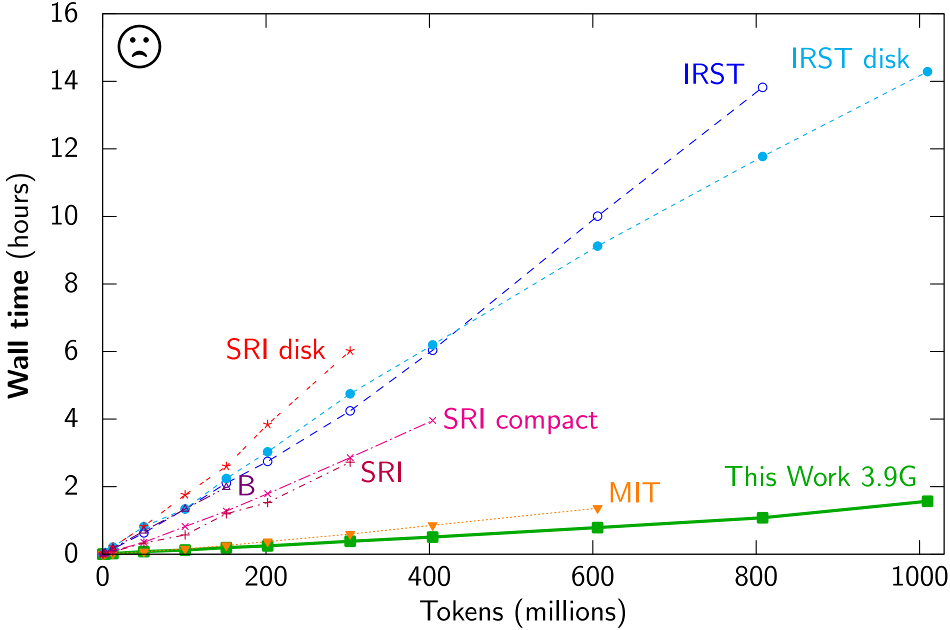
Task	Build an unpruned 5-gram language model
Data	Subset of English ClueWeb09 (webpages)
Machine	64 GB RAM
Output Format	Binary (or ARPA when faster)

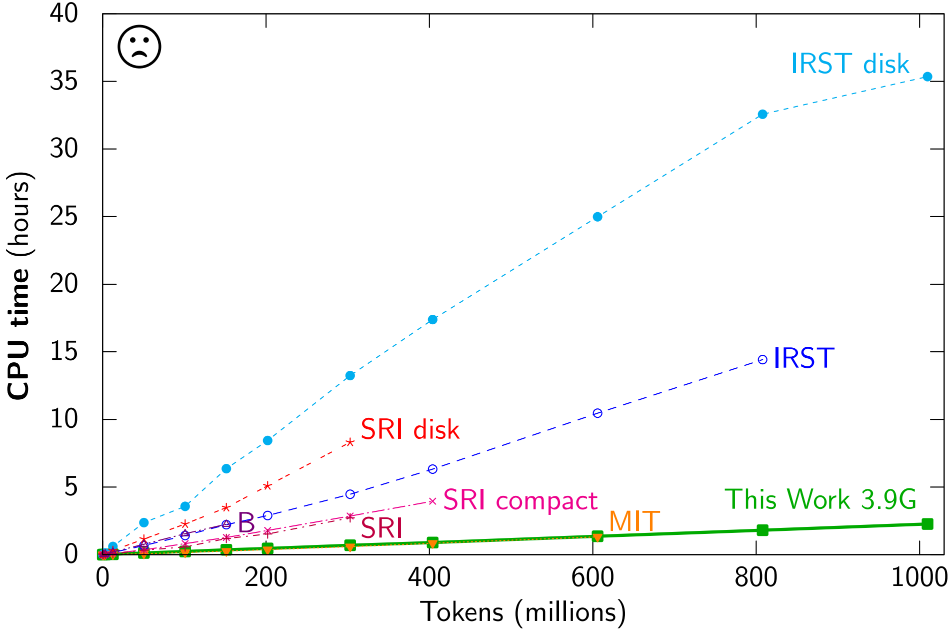
IRST disk: 3-way split. Peak RAM of any one process (as if run serially).
Berkeley: Binary search for minimum JVM memory.


☹
SRI compact
SRI disk
SRI
This Work 3.9G
RAM (GB)
Tokens (millions)
0
10
20
30
40
50
60
0
200
400
600
800
1000


☹
SRI compact
IRST
SRI disk
SRI
IRST disk
This Work 3.9G
RAM (GB)
Tokens (millions)
0
10
20
30
40
50
60
0
200
400
600
800
1000


Berkeley
MIT
SRI compact
IRST
SRI disk
SRI
IRST disk
This Work 3.9G
RAM (GB)
Tokens (millions)
0
10
20
30
40
50
60
200
400
600
800
1000


☹
IRST
IRST disk
SRI disk
SRI compact
SRI
B
MIT
This Work 3.9G
Wall time (hours)
Tokens (millions)
0
2
4
6
8
10
12
14
16
0
200
400
600
800
1000


☹
CPU time (hours)
Tokens (millions)
0
5
10
15
20
25
30
35
40
0
200
400
600
800
1000
IRST disk
IRST
SRI disk
SRI compact
SRI
B
MIT
This Work 3.9G

Scaling

	Tokens	Smoothing	Machines	Days
This Work	126 billion	Kneser-Ney	**1**	2.8

Counts

	1	**2**	**3**	**4**	**5**
This Work 126B	393m	3,775m	17,629m	39,919m	59,794m
Pruned Google 1T	14m	315m	977m	1,313m	1,176m

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

Scaling

	Tokens	Smoothing	Machines	Days	Year
This Work	126 billion	Kneser-Ney	**1**	2.8	2013
Google	31 billion	Kneser-Ney	400	2	2007
Google	230 billion	Kneser-Ney	?	?	2013
Google	1800 billion	Stupid	1500	1	2007

Counts

	1	**2**	**3**	**4**	**5**
This Work 126B	393m	3,775m	17,629m	39,919m	59,794m
Pruned Google 1T	14m	315m	977m	1,313m	1,176m

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

WMT 2013 Results

1. Compress the big LM to 676 GB
2. Decode with 1 TB RAM
3. Make three WMT submissions

	Czech–English		**French–English**		**Spanish–English**	
	Rank	BLEU	Rank	BLEU	Rank	BLEU
This Work	1	28.16	1	33.37	1	32.55
Google	2–3	27.11	2–3	32.62	2	33.65
Baseline	3–5	27.38	2–3	32.57	3–5	31.76

Rankings?

Pairwise significant above baseline

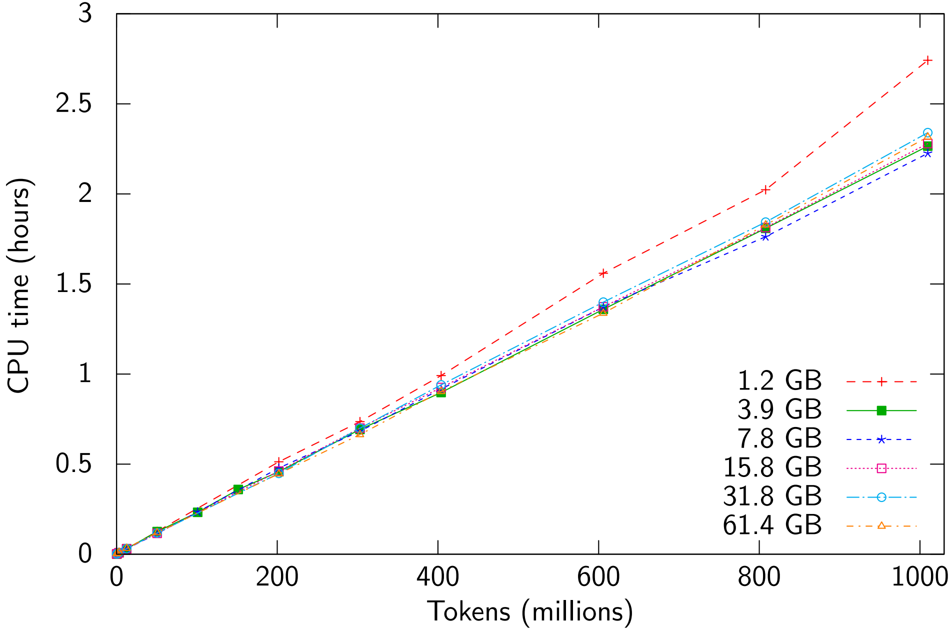
Build language models with user-specified RAM

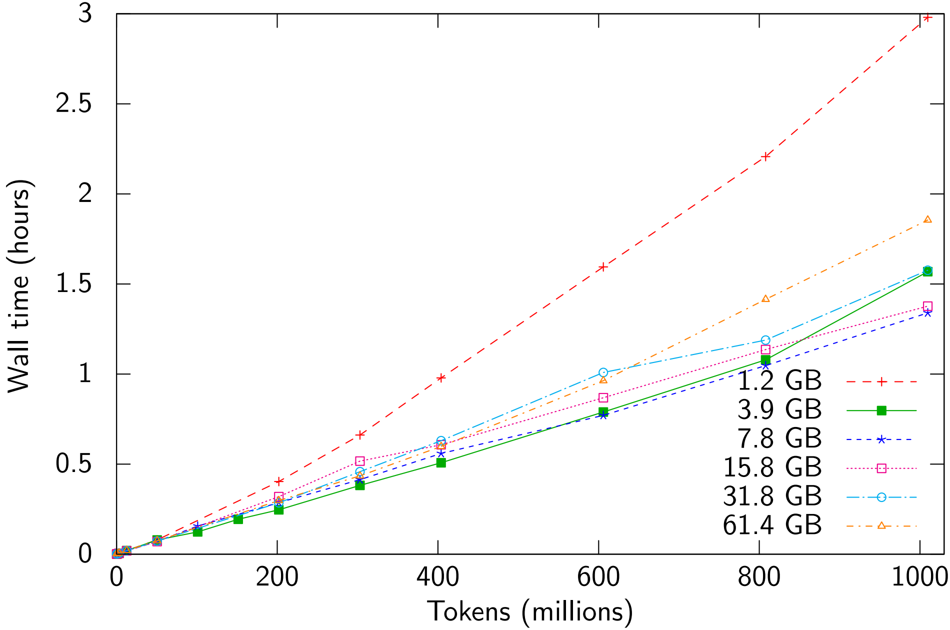
kheafield.com/code/kenlm/

bin/lmplz -o 5 -S 10G <text >arpa

Future Work

- Interpolating models trained on separate data
- Pruning
- CommonCrawl corpus


CPU time (hours)
Tokens (millions)
0
0.5
1
1.5
2
2.5
3
0
200
400
600
800
1000
1.2 GB
3.9 GB
7.8 GB
15.8 GB
31.8 GB
61.4 GB


Wall time (hours)
Tokens (millions)
0
0.5
1
1.5
2
2.5
3
0
200
400
600
800
1000
1.2 GB
3.9 GB
7.8 GB
15.8 GB
31.8 GB
61.4 GB

Calculating Discounts

Summary statistics are collected while adjusting counts:
$s_n(a)$ = number of n-grams with adjusted count a.

Calculating Discounts

Summary statistics are collected while adjusting counts:
$s_n(a)$ = number of n-grams with adjusted count a.

$$\mathrm{discount}_n(a) = a - \frac{(a+1)s_n(1)s_n(a+1)}{(s_n(1)+2s_n(2))s_n(a)}$$

Chen and Goodman

Calculating Discounts

Summary statistics are collected while adjusting counts:
$s_n(a)$ = number of n-grams with adjusted count a.

$$\text{discount}_n(a) = a - \frac{(a+1)s_n(1)s_n(a+1)}{(s_n(1)+2s_n(2))s_n(a)}$$

Chen and Goodman

Use $\text{discount}_n(3)$ for counts above 3.