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
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       | Count |
|---------------|-------------|-------|
| are one of    | 1           |       |
| is one of     | 5           |       |
| are two of    | 3           |       |

Adjustments:

1-gram

| Output | Adjusted |
|--------|----------|
| of     | 2        |

2-gram

| Output      | Adjusted |
|-------------|----------|
| one of      | 2        |
| two of      | 1        |
Calculating Discounts

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

Chen and Goodman

\[ \text{discount}_n \]
Discounting and Normalization

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

- Save mass for unseen events
- Normalize
Discounting and Normalization

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

Save mass for unseen events

Normalize

| Context Sorted | Input | Output |
|---------------|-------|--------|
| 2 1 | 3 | Adjusted | 3-gram | Pseudo |
| are one | of | 1 | are one of | 0.26 |
| are one | that | 2 | are one that | 0.47 |
| is one | of | 5 | is one of | 0.62 |
Denominator Looks Ahead

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

Save mass for unseen events

Normalize

| Context | Sorted Input | Adjusted |
|---------|--------------|---------|
| 2 1 3   | are one      | 1       |
|         | of that      | 2       |
|         | 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 | Adjusted |
|------------|----------|
| 2 1        | 3        |
| are one    | of       |
| are one    | that     |
| is one     | of       |
| 1          | 2        |
| 2          | 5        |

Reads ahead and sums

| Divide Thread | Adjusted |
|---------------|----------|
| 2 1           | 3        |
| are one       | of       |
| are one       | that     |
| is one        | of       |
| 1            | 2        |
| 2            | 5        |

Reads behind to normalize

sum = 3
Computing Backoffs
Backoffs are penalties for unseen events.

Bin the entries “are one x” by their adjusted counts

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

Backoffs are penalties for unseen events.

Bin the entries “are one x” by their adjusted counts

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

Compute backoff in the sum thread

\[
\text{backoff(are one)} = \frac{\text{continue(are one)} \cdot \text{discount}_3}{\sum_x \text{adjusted(are one x)}}
\]
Interpolate unigrams with the uniform distribution.

\[ p(\text{of}) = \text{pseudo(\text{of})} + \text{backoff}(\epsilon) \frac{1}{|\text{vocabulary}|} \]
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 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 | Output |
|---|---|---|
| \(n\)-gram | pseudo | interpolation weight | \(n\)-gram | \(p\) |
| of | 0.1 | backoff(\(\epsilon\)) = 0.1 | of | 0.110 |
| one of | 0.2 | backoff(one) = 0.3 | one of | 0.233 |
| are one of | 0.4 | backoff(are one) = 0.2 | are one of | 0.447 |
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.

- 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 in context order

Write to disk

Each vertex is a thread $\implies$ 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.
This Work 3.9G

SRI disk

SRI compact

SRI compact

Tokens (millions)

RAM (GB)
This Work 3.9G

IRST disk

SRI compact

SRI disk

IRST

Berkeley

RAM (GB)

Tokens (millions)
This Work 3.9G

Estimating

Streaming and Sorting

Experiments
Wall time (hours)

Tokens (millions)

This Work 3.9G

IRST

IRST disk

MIT

SRI compact

SRI disk

B
Estimating

Streaming and Sorting

Experiments
Scaling

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

Counts

|           | 1  | 2  | 3  | 4  | 5  |
|-----------|----|----|----|----|----|
| This Work | 393m | 3,775m | 17,629m | 39,919m | 59,794m |
| Pruned Google | 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**  | 393m| 3,775m| 17,629m| 39,919m| 59,794m|
| **Pruned Google** | 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
Calculating Discounts

Summary statistics are collected while adjusting counts:
\[ s_n(a) = \text{number of } n\text{-grams with adjusted count } a. \]
Calculating Discounts

Summary statistics are collected while adjusting counts:
\[ s_n(a) = \text{number of } n\text{-grams with adjusted count } a. \]

Chen and Goodman

\[
\text{discount}_n(a) = a - \frac{(a + 1)s_n(1)s_n(a + 1)}{(s_n(1) + 2s_n(2))s_n(a)}
\]
Calculating Discounts

Summary statistics are collected while adjusting counts:

\[ s_n(a) = \text{number of } n\text{-grams with adjusted count } a. \]

Chen and Goodman discount

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

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