

Language Modeling

Kenneth Heafield
heafield@cs.cmu.edu

University of Edinburgh & Carnegie Mellon

September 7, 2011

MT Marathon 2011

Language Models

AP

>



Public domain photo from the New York Zoological Society.

What is a Language Model?

A probability distribution p over strings (usually sentences).

Examples

$$\log p(\langle s \rangle \text{ iran is one of the } \mathbf{few} \text{ countries } \langle /s \rangle) = -11.7990$$

$$\log p(\langle s \rangle \text{ iran is one of the } \mathbf{many} \text{ countries } \langle /s \rangle) = -12.9443$$

$$\log p(\langle s \rangle \text{ our advice on how to choose } \langle /s \rangle) = -16.0042$$

Language Model Applications

Biasing Towards Fluency

$$\text{output}^* = \underset{\text{output}}{\operatorname{argmax}} \mathbf{p}(\text{output}) p(\text{output}|\text{input})$$

- Machine translation
- Speech recognition
- Optical character recognition

Analyzing Text

$\mathbf{p}(\text{input})$

- Information retrieval
- Essay grading
- Language detection

Language Model Desiderata

Assign high probability to natural text.
Larger $p(\text{corpus})$ is better.

$$\log_2 \text{Perplexity} = -\frac{1}{|\text{corpus}|} \log_2 p(\text{corpus})$$

Unseen Data

Use separate data to train and evaluate models.

Estimating p

First Try

- 1 Obtain a corpus.
- 2 Count how many times each sentence occurs.
- 3 Apply maximum likelihood: $p(s) \propto \text{count}(s)$.

Problem

New sentences have 0 probability $\implies p(\text{unseen corpus}) = 0$.

Smoothing

Estimate the probability of unseen events.
When data is sparse, generalize.

First Type of Smoothing

Break a sentence up using independence assumptions.

Types of Language Models

Model how a sentence is generated.

- ***N*-gram Markov model**
- Hidden Markov model
- Decision trees
- Statistical parser probability [Schwartz et al 2011]

N-Gram Markov Model: Applying the Chain Rule

$$\begin{array}{rcl}
 \log p(\langle s \rangle) & \rightarrow \text{iran} &) \\
 \log p(\langle s \rangle \text{ iran}) & \rightarrow \text{is} &) \\
 \log p(\langle s \rangle \text{ iran is}) & \rightarrow \text{one} &) \\
 \log p(\langle s \rangle \text{ iran is one}) & \rightarrow \text{of} &) \\
 \log p(\langle s \rangle \text{ iran is one of}) & \rightarrow \text{the} &) \\
 \log p(\langle s \rangle \text{ iran is one of the}) & \rightarrow \text{few} &) \\
 \log p(\langle s \rangle \text{ iran is one of the few}) & \rightarrow \text{countries}) &) \\
 \log p(\langle s \rangle \text{ iran is one of the few countries}) & \rightarrow . &) \\
 + \log p(\langle s \rangle \text{ iran is one of the few countries} .) & \rightarrow \langle /s \rangle &) \\
 \hline
 = \log p(\langle s \rangle \text{ iran is one of the few countries} . \langle /s \rangle) & &)
 \end{array}$$

N-Gram Markov Model: Applying the Chain Rule

$$\begin{array}{l}
 \log p(\langle s \rangle \rightarrow \text{iran}) = -3.33437 \\
 \log p(\langle s \rangle \text{ iran} \rightarrow \text{is}) = -1.05931 \\
 \log p(\langle s \rangle \text{ iran is} \rightarrow \text{one}) = -1.80743 \\
 \log p(\langle s \rangle \text{ iran is one} \rightarrow \text{of}) = -0.03705 \\
 \log p(\langle s \rangle \text{ iran is one of} \rightarrow \text{the}) = \text{???} \\
 \log p(\langle s \rangle \text{ iran is one of the} \rightarrow \text{few}) = \text{???} \\
 \log p(\langle s \rangle \text{ iran is one of the few} \rightarrow \text{countries}) = \text{???} \\
 \log p(\langle s \rangle \text{ iran is one of the few countries} \rightarrow \text{.}) = \text{???} \\
 + \log p(\langle s \rangle \text{ iran is one of the few countries.} \rightarrow \langle /s \rangle) = \text{???} \\
 \hline
 = \log p(\langle s \rangle \text{ iran is one of the few countries.} \langle /s \rangle) = \text{???}
 \end{array}$$

Markov Assumption

$$\begin{array}{rcl}
 \log p(\langle s \rangle & \rightarrow \text{iran} &) = -3.33437 \\
 \log p(\langle s \rangle \text{iran} & \rightarrow \text{is} &) = -1.05931 \\
 \log p(\langle s \rangle \text{iran is} & \rightarrow \text{one} &) = -1.80743 \\
 \log p(\langle s \rangle \text{iran is one} & \rightarrow \text{of} &) = -0.03705 \\
 \log p(\langle s \rangle \text{iran is one of} & \rightarrow \text{the} &) = -0.08317 \\
 \log p(\langle s \rangle \text{iran is one of the} & \rightarrow \text{few} &) = -1.20788 \\
 \log p(\langle s \rangle \text{iran is one of the few} & \rightarrow \text{countries}) = -1.62030 \\
 \log p(\langle s \rangle \text{iran is one of the few countries} & \rightarrow . &) = -2.60261 \\
 + \log p(\langle s \rangle \text{iran is one of the few countries} & \rightarrow \langle /s \rangle &) = -0.04688 \\
 \hline
 = \log p(\langle s \rangle \text{iran is one of the few countries} & \langle /s \rangle &) = -11.79900
 \end{array}$$

The Markov Assumption

Close words are better predictors than far words.

Doubts

- Grammatical structure
- Topical coherence
- Words tend to repeat

Estimating an N -Gram Model

First Try

$$p(\langle s \rangle \text{ iran is } \rightarrow \text{one}) = \frac{\text{count}(\langle s \rangle \text{ iran is one})}{\text{count}(\langle s \rangle \text{ iran is})}$$

This maximizes probability of the training data.

Problem

Is $p(\langle s \rangle \text{ iran is } \rightarrow \text{of}) = 0$ because it was not seen?

Backing off in an N -Gram Model

Use the longest matching history.
Charge a backoff penalty for unused history.

$$\log p(\langle s \rangle \text{ iran} \rightarrow \text{is}) = \log p(\langle s \rangle \text{ iran} \rightarrow \text{is})$$

$$\log p(\text{iran is} \rightarrow \text{of}) = \log p(\text{of}) + \text{backoff}(\text{iran is}) + \text{backoff}(\text{is})$$

Backing off in an N -Gram Model

Use the longest matching history.
Charge a backoff penalty for unused history.

$$\log p(\langle s \rangle \text{ iran} \rightarrow \text{is}) = \log p(\langle s \rangle \text{ iran} \rightarrow \text{is})$$

$$\log p(\text{iran is} \rightarrow \text{of}) = \log p(\text{of}) + \text{backoff}(\text{iran is}) + \text{backoff}(\text{is})$$

What this implies

- Keep all n -grams up to length N
- Every n -gram has a probability
- For $n < N$, every n -gram has a backoff penalty

Example Language Model

Unigrams

Words	$\log p$	Back
<s>	$-\infty$	-2.0
iran	-4.1	-0.8
is	-2.5	-1.4
one	-3.3	-0.9
of	-2.5	-1.1

Bigrams

Words	$\log p$	Back
<s> iran	-3.3	-1.2
iran is	-1.7	-0.4
is one	-2.0	-0.9
one of	-1.4	-0.6

Trigrams

Words	$\log p$
<s> iran is	-1.1
iran is one	-2.0
is one of	-0.3

Example Queries

Unigrams

Words	$\log p$	Back
<s>	$-\infty$	-2.0
iran	-4.1	-0.8
is	-2.5	-1.4
one	-3.3	-0.9
of	-2.5	-1.1

Bigrams

Words	$\log p$	Back
<s> iran	-3.3	-1.2
iran is	-1.7	-0.4
is one	-2.0	-0.9
one of	-1.4	-0.6

Trigrams

Words	$\log p$
<s> iran is	-1.1
iran is one	-2.0
is one of	-0.3

Query: <s> iran is

$$\log p(\langle s \rangle \text{ iran} \rightarrow \text{is}) = -1.1$$

Query: iran is of

$\log p(\text{of})$	-2.5
Backoff(is)	-1.4
Backoff(iran is)	+ -0.4
<hr/>	
$\log p(\text{iran is} \rightarrow \text{of})$	= -4.3

Summary

Two kinds of smoothing: the Markov assumption and backing off.

Next: Where do the probability and backoff come from?
(Hint: more smoothing)

Where do p and backoff come from?

Count n -grams in a large monolingual corpus, apply smoothing.

Smoothing Strategies

- Add a constant to each count (including 0)
- Good-Turing
- Witten-Bell
- Kneser-Ney
- **Modified Kneser-Ney**

Modified Kneser-Ney

Typically the best on unseen data and BLEU score.

- 1 Count n -grams in corpus
- 2 Adjust counts
- 3 Compute discounted probabilities
- 4 Optional: Interpolate models across orders
- 5 Compute backoffs from probabilities

Adjusted Counts

An n -gram is consulted only if an $n + 1$ -gram was not found.



Lower order probabilities condition on backing off.

Adjusted Counts

An n -gram is consulted only if an $n + 1$ -gram was not found.



Lower order probabilities condition on backing off.

Lower order ($n < N$)

$\text{adjusted}(w_1^n) = \text{number of unique extensions} = |\{w_1^n v \in \text{corpus}\}|$

Highest order (N)

$\text{adjusted}(w_1^N) = \text{count}(w_1^N)$

Discounting

Make space for unobserved n -grams.

$$p_{\text{disc}}(w_1^{n-1} \rightarrow w_n) = \frac{\text{adjusted}(w_1^n) - D(\text{adjusted}(w_1^n))}{\text{adjusted}(w_1^{n-1})}$$

D is different for counts 0, 1, 2, and any count ≥ 3 .

Interpolation

If $p(is)$ is high, then $p(\langle s \rangle \text{ iran} \rightarrow is)$ probably is too.

$$\begin{aligned} p(\langle s \rangle \text{ iran} \rightarrow is) &= \lambda_0 \frac{1}{|\text{vocabulary}|} \\ &+ \lambda_1 p_{\text{disc}}(is) \\ &+ \lambda_2 p_{\text{disc}}(\text{iran} \rightarrow is) \\ &+ \lambda_3 p_{\text{disc}}(\langle s \rangle \text{ iran} \rightarrow is) \end{aligned}$$

The λ_i are weights and complicated to estimate.

Backoff Weights

Backoff is charged when an n -gram is not found:

$$p(\langle s \rangle \text{ iran} \rightarrow \text{of}) = p(\text{of}) + \text{Backoff}(\langle s \rangle \text{ iran}) + \text{Backoff}(\text{iran})$$

This penalty depends on how strongly predicts following words:

$$\text{Backoff}(\langle s \rangle \text{ iran}) = \frac{1 - \sum_{x \text{ extends } \langle s \rangle \text{ iran}} p(\langle s \rangle \text{ iran} \rightarrow x)}{1 - \sum_{x \text{ extends } \langle s \rangle \text{ iran}} p(\text{iran} \rightarrow x)}$$

Unknown Words

We expect to see new words.

As Implemented in SRILM

Default Probability is 0. Moses thresholds to e^{-100} .

-unk Give some leftover probability to unknown words.

Empirical Use part of the corpus to estimate the probability.

Andreas Stolcke, SRILM's lead author, recommends empirical.

My Recommendation

Use another feature that counts unknown words.

Let MERT figure out the unknown word penalty.

Modified Kneser-Ney Summary

Several Types of Smoothing

- Independence assumption: the Markov model
- Backing off
- Discounting probabilities
- Interpolation
- Unknown words

Optional (not covered)

Word classes: parts of speech or a class for numbers

Outline

- 1 Estimating
 - Modified Kneser-Ney
- 2 Applying
 - Optimizing Backoff
 - State
 - Data Structures
 - Probing
 - Trie
 - Chop
 - Results
 - Perplexity
 - Translation

(The part that Moses calls.)

Toolkits

Downloadable Baselines

- SRI** Popular and considered fast but high-memory
- IRST** Open source, low-memory, single-threaded
- Rand** Low-memory lossy compression
- MIT** Mostly estimates models but also does queries

Papers Without Code

- TPT** Better memory locality
- Sheffield** Lossy compression techniques

Toolkits

Downloadable Baselines

- SRI** Popular and considered fast but high-memory
- IRST** Open source, low-memory, single-threaded
- Rand** Low-memory lossy compression
- MIT** Mostly estimates models but also does queries

Papers Without Code

- TPT** Better memory locality
- Sheffield** Lossy compression techniques

Released Later

- Berkeley** Java; slow and high memory

Why I Wrote KenLM

Decoding takes too long

- Answer queries quickly
- Load quickly with memory mapping
- Thread-safe

Why I Wrote KenLM

Decoding takes too long

- Answer queries quickly
- Load quickly with memory mapping
- Thread-safe

Bigger models

- Conserve memory

Why I Wrote KenLM

Decoding takes too long

- Answer queries quickly
- Load quickly with memory mapping
- Thread-safe

Bigger models

- Conserve memory

SRI doesn't compile

- Distribute and compile with decoders

Example Language Model

Unigrams

Words	$\log p$	Back
<s>	$-\infty$	-2.0
iran	-4.1	-0.8
is	-2.5	-1.4
one	-3.3	-0.9
of	-2.5	-1.1

Bigrams

Words	$\log p$	Back
<s> iran	-3.3	-1.2
iran is	-1.7	-0.4
is one	-2.0	-0.9
one of	-1.4	-0.6

Trigrams

Words	$\log p$
<s> iran is	-1.1
iran is one	-2.0
is one of	-0.3

Example Queries

Unigrams

Words	$\log p$	Back
<s>	$-\infty$	-2.0
iran	-4.1	-0.8
is	-2.5	-1.4
one	-3.3	-0.9
of	-2.5	-1.1

Bigrams

Words	$\log p$	Back
<s> iran	-3.3	-1.2
iran is	-1.7	-0.4
is one	-2.0	-0.9
one of	-1.4	-0.6

Trigrams

Words	$\log p$
<s> iran is	-1.1
iran is one	-2.0
is one of	-0.3

Query: <s> iran is

$$\log p(\langle s \rangle \text{ iran} \rightarrow \text{is}) = -1.1$$

Query: iran is of

$\log p(\text{of})$	-2.5
Backoff(is)	-1.4
Backoff(iran is)	+ -0.4
<hr/>	
$\log p(\text{iran is} \rightarrow \text{of})$	= -4.3

Lookups Performed by Queries

<s> iran is

Lookup

- 1 is
- 2 iran is
- 3 <s> iran is

Score

$$\log p(\langle s \rangle \text{ iran} \rightarrow \text{is}) = -1.1$$

iran is of

Lookup

- 1 of
- 2 is of (not found)
- 3 is
- 4 iran is

Score

$\log p(\text{of})$	-2.5
$\text{Backoff}(\text{is})$	-1.4
$\text{Backoff}(\text{iran is})$	+ -0.4
<hr/>	
$\log p(\text{iran is} \rightarrow \text{of})$	= -4.3

Lookups Performed by Queries

<s> iran is

Lookup

- 1 is
- 2 iran is
- 3 <s> iran is

Score

$$\log p(\langle s \rangle \text{ iran} \rightarrow \text{is}) = -1.1$$

iran is of

Lookup

- 1 of
- 2 is of (not found)
- 3 is
- 4 iran is

Score

$\log p(\text{of})$	-2.5
Backoff(is)	-1.4
Backoff(iran is)	+ -0.4
<hr/>	
$\log p(\text{iran is} \rightarrow \text{of})$	= -4.3

Lookups Performed by Queries

<s> iran is

Lookup

- 1 is
- 2 iran is
- 3 <s> iran is

State

Backoff(is)
Backoff(iran is)

iran is of

Lookup

- 1 of
- 2 is of (not found)
- ~~3 is~~
- ~~4 iran is~~

Score

$$\log p(\langle s \rangle \text{ iran} \rightarrow \text{is}) = -1.1$$

Score

$\log p(\text{of})$	-2.5
Backoff(is)	-1.4
Backoff(iran is)	+ -0.4
$\log p(\text{iran is} \rightarrow \text{of})$	= -4.3

Stateful Query Pattern

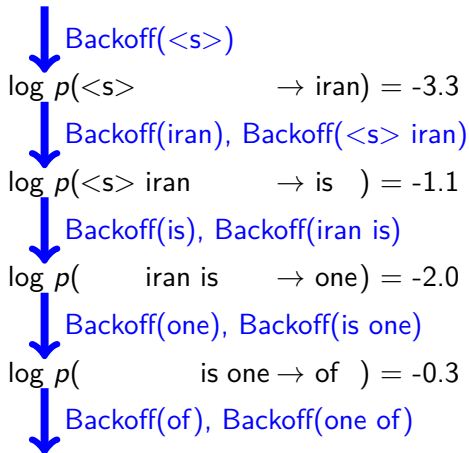
$$\log p(\langle s \rangle \rightarrow \text{iran}) = -3.3$$

$$\log p(\langle s \rangle \text{ iran} \rightarrow \text{is}) = -1.1$$

$$\log p(\text{iran is} \rightarrow \text{one}) = -2.0$$

$$\log p(\text{is one} \rightarrow \text{of}) = -0.3$$

Stateful Query Pattern



Data Structures

Probing Fast. Uses hash tables.

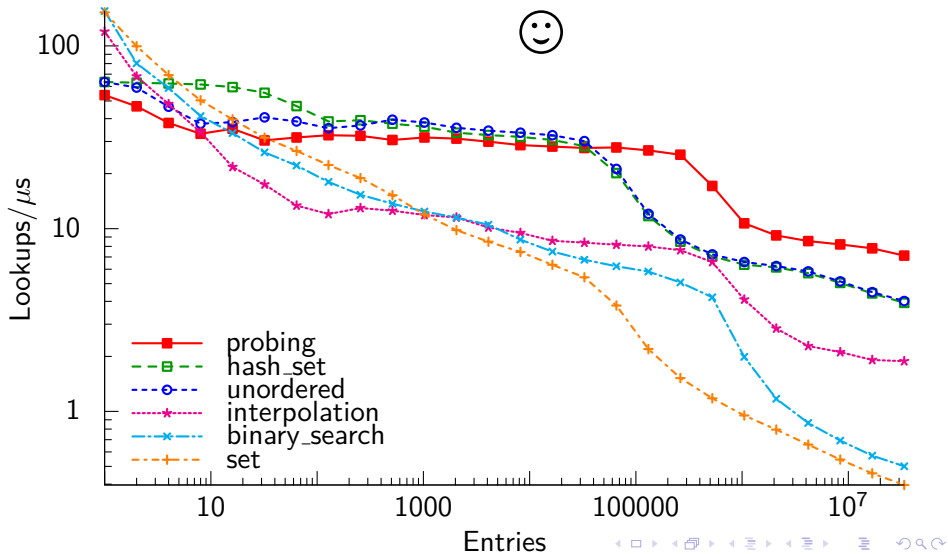
Trie Small. Uses sorted arrays.

Chop Smaller. Trie with compressed pointers.

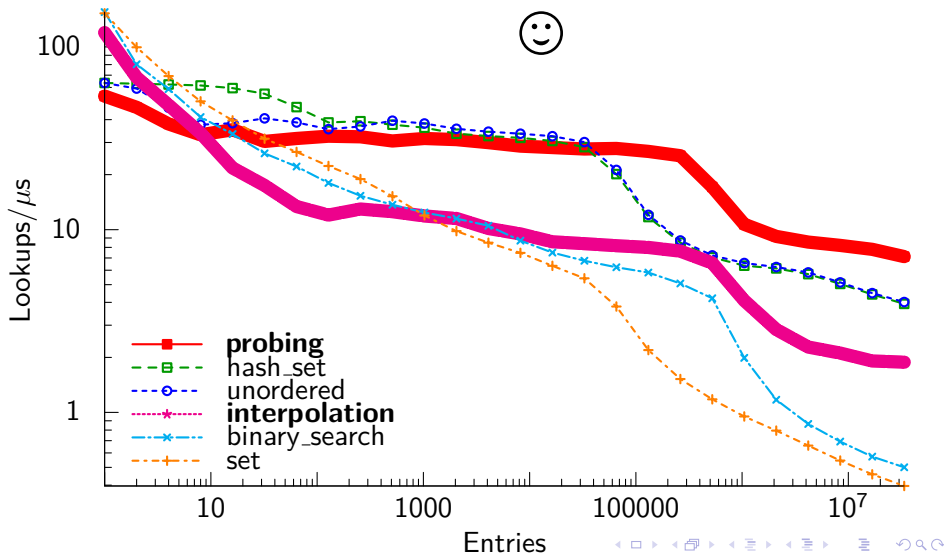
Key Subproblem

Sparse lookup: efficiently retrieve values for sparse keys

Sparse Lookup Speed



Sparse Lookup Speed



Linear Probing Hash Table

Store 64-bit hashes and ignore collisions.

Words	Bigrams Hash	log p	Back
<s> iran	0xf0ae9c2442c6920e	-3.3	-1.2
iran is	0x959e48455f4a2e90	-1.7	-0.4
is one	0x186a7caef34acf16	-2.0	-0.9
one of	0xac66610314db8dac	-1.4	-0.6

Linear Probing Hash Table

- 1.5 buckets/entry (so buckets = 6).
- Ideal bucket = hash mod buckets.
- Resolve bucket collisions using the next free bucket.

Words	Ideal	Bigrams Hash	$\log p$	Back
iran is	0	0x959e48455f4a2e90	-1.7	-0.4
		0x0	0	0
is one	2	0x186a7caef34acf16	-2.0	-0.9
		0xac66610314db8dac	-1.4	-0.6
one of	2	0xf0ae9c2442c6920e	-3.3	-1.2
		0x0	0	0
<s> iran	4	0xf0ae9c2442c6920e	-3.3	-1.2
		0x0	0	0

Array

Probing Data Structure

Unigrams

Words	$\log p$	Back
<s>	$-\infty$	-2.0
iran	-4.1	-0.8
is	-2.5	-1.4
one	-3.3	-0.9
of	-2.5	-1.1

Array

Bigrams

Words	$\log p$	Back
<s> iran	-3.3	-1.2
iran is	-1.7	-0.4
is one	-2.0	-0.9
one of	-1.4	-0.6

Probing Hash Table

Trigrams

Words	$\log p$
<s> iran is	-1.1
iran is one	-2.0
is one of	-0.3

Probing Hash Table

Probing Hash Table Summary

Hash tables are fast. But memory is 24 bytes/entry.

Next: Saving memory with Trie.

Trie Uses Sorted Arrays

Sort in suffix order.

Unigrams

Words	$\log p$	Back
<s>	$-\infty$	-2.0
iran	-4.1	-0.8
is	-2.5	-1.4
one	-3.3	-0.9
of	-2.5	-1.1

Bigrams

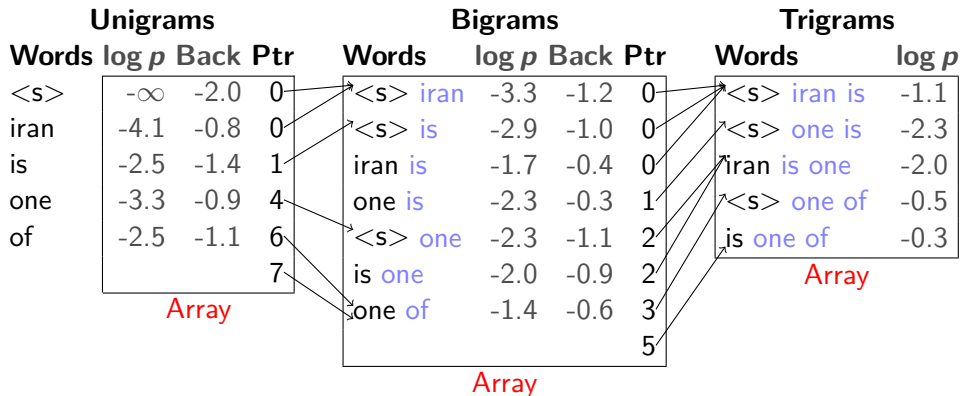
Words	$\log p$	Back
<s> iran	-3.3	-1.2
iran is	-1.7	-0.4
one is	-2.3	-0.3
<s> one	-2.3	-1.1
is one	-2.0	-0.9
one of	-1.4	-0.6

Trigrams

Words	$\log p$
<s> iran is	-1.1
<s> one is	-2.3
iran is one	-2.0
<s> one of	-0.5
is one of	-0.3

Trie

Sort in suffix order. Encode **suffix** using pointers.



Interpolation Search In Trie

Each trie node is a sorted array.

Bigrams: * is

Words log p Back Ptr

<s> is	-2.9	-1.0	0
iran is	-1.7	-0.4	0
one is	-2.3	-0.3	1

Interpolation Search $O(\log \log n)$

$$pivot = |A| \frac{key - A.first}{A.last - A.first}$$

Binary Search: $O(\log n)$

$$pivot = \frac{|A|}{2}$$

Saving Memory with Trie

Bit-Level Packing


Store word index and pointer using the minimum number of bits.

Optional Quantization

Cluster floats into 2^q bins, store q bits/float (same as IRSTLM).

Chop: Compress Trie Pointers

		Bigrams		
Words		$\log p$	Back	Ptr
<s>	iran	-3.3	-1.2	0
iran	is	-1.7	-0.4	0
one	is	-2.3	-0.3	1
<s>	one	-2.3	-1.1	2
is	one	-2.0	-0.9	2
one	of	-1.4	-0.6	3
				5



Increasing

Chop: Compress Trie Pointers

Offset	Ptr	Binary
0	0	000
1	0	000
2	1	001
3	2	010
4	2	010
5	3	011
6	5	101

Raj and Whittaker (2003)

Chop: Compress Trie Pointers

Offset	Ptr	Binary
--------	-----	--------

0	0	000
1	0	000
2	1	001
3	2	010
4	2	010
5	3	011
6	5	101

Chopped	Offset
---------	--------

1	6
---	---

Raj and Whittaker (2003)

Chop: Compress Trie Pointers

Offset	Ptr	Binary
--------	-----	--------

0	0	000
1	0	000
2	1	001
3	2	010
4	2	010
5	3	011
6	5	101

Chopped	Offset
---------	--------

01	3
10	6

Raj and Whittaker (2003)

Trie/Chop Summary

Save memory: bit packing, quantization, and pointer compression.

Outline

- 1 Estimating
 - Modified Kneser-Ney
- 2 Applying
 - Optimizing Backoff
 - State
 - Data Structures
 - Probing
 - Trie
 - Chop
 - Results
 - Perplexity
 - Translation

Perplexity Task

Score the English Gigaword corpus.

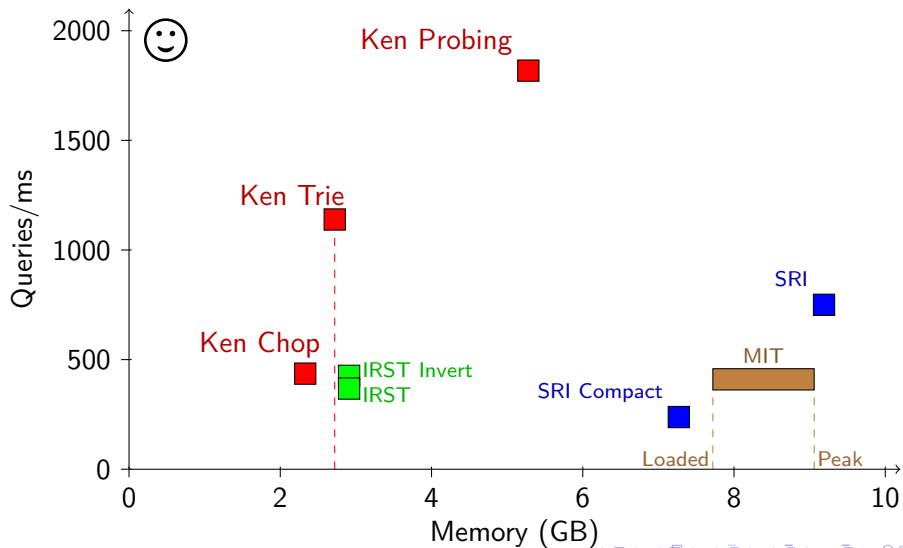
Model

SRILM 5-gram from Europarl + De-duplicated News Crawl

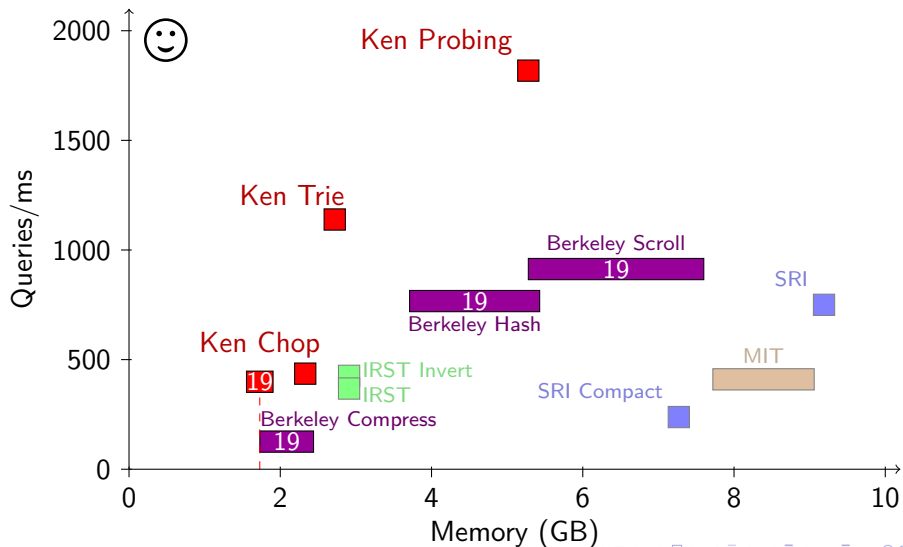
Measurements

Queries/ms	Excludes loading and file reading time
Loaded Memory	Resident after loading
Peak Memory	Peak virtual after scoring

Perplexity Task: Exact Models



Perplexity Task: Berkeley Always Quantizes to 19 bits



Translation Task

Translate 3003 sentences using Moses.

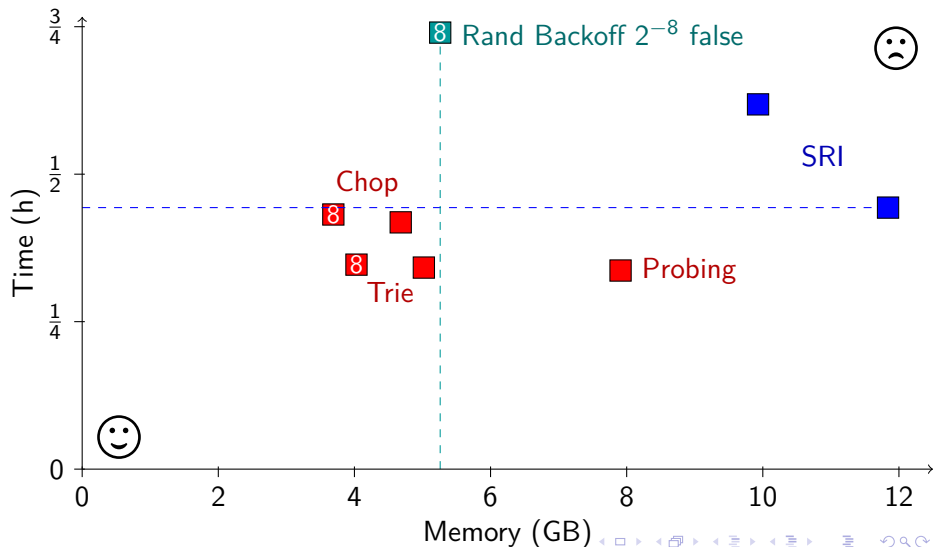
System

WMT 2011 French-English baseline, Europarl+News LM

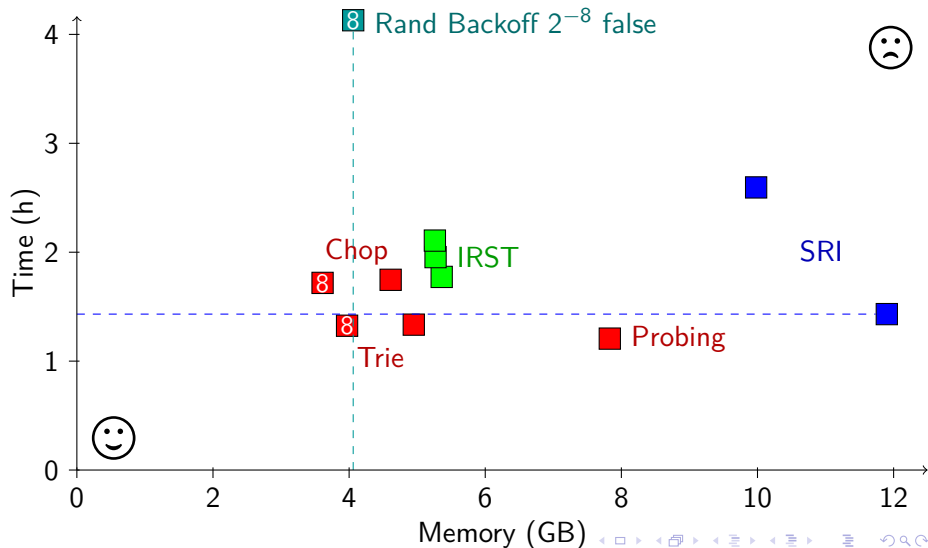
Measurements

Time	Total wall time, including loading
Memory	Total resident memory after decoding

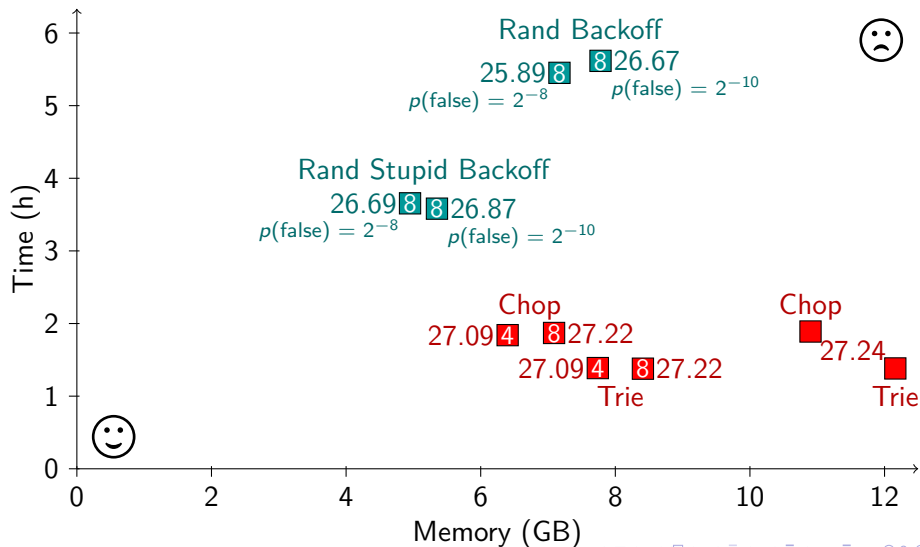
Moses Benchmarks: 8 Threads



Moses Benchmarks: Single Threaded



Comparison to RandLM (Unpruned Model, One Thread)



Conclusion

Maximize speed and accuracy subject to memory.

Probing > Trie > Chop > RandLM Stupid
for both speed and memory.

Distributed with decoders:

Moses	8 0 5 file
cdec	KLanguageModel
Joshua	use_kenlm=true

kheafield.com/code/kenlm/