



Smoothed Bloom Filter Language Models

Saving Space by Flipping Coins

David Talbot and Miles Osborne

School of Informatics
University of Edinburgh

First Machine Translation Marathon (16th April 2007)



Outline

1 Motivation

- Scaling Language Modelling
- Problems with Lossless Representations
- Lossy Representations

2 Bloom Filter Language Models

- The Bloom Filter
- Extending the BF for Language Modelling

3 Experiments



Outline

1 Motivation

- Scaling Language Modelling
- Problems with Lossless Representations
- Lossy Representations

2 Bloom Filter Language Models

- The Bloom Filter
- Extending the BF for Language Modelling

3 Experiments



Outline

1 Motivation

- Scaling Language Modelling
- Problems with Lossless Representations
- Lossy Representations

2 Bloom Filter Language Models

- The Bloom Filter
- Extending the BF for Language Modelling

3 Experiments



Outline

1 Motivation

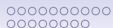
- Scaling Language Modelling
- Problems with Lossless Representations
- Lossy Representations

2 Bloom Filter Language Models

- The Bloom Filter
- Extending the BF for Language Modelling

3 Experiments





Language Modelling Challenges

- Good modelling
 - N-grams ($n = 8?$)
- Good estimation
 - Millions / billions / trillions of words
 - Good estimators (e.g., Witten-Bell, Kneser-Ney)
- Small memory footprint
- Low computational complexity





Language Modelling Challenges

- Good modelling
 - N-grams ($n = 8?$)
- Good estimation
 - Millions / billions / trillions of words
 - Good estimators (e.g., Witten-Bell, Kneser-Ney)
- Small memory footprint
- Low computational complexity





Language Modelling Challenges

- Good modelling
 - N-grams ($n = 8?$)
- Good estimation
 - Millions / billions / trillions of words
 - Good estimators (e.g., Witten-Bell, Kneser-Ney)
- Small memory footprint
- Low computational complexity





Language Modelling Challenges

- Good modelling
 - N-grams ($n = 8?$)
- Good estimation
 - Millions / billions / trillions of words
 - Good estimators (e.g., Witten-Bell, Kneser-Ney)
- Small memory footprint
- Low computational complexity



A Curse of Dimensionality - and Large Corpora

- Size of N -gram event space increases exponentially

$$|\mathcal{U}_N| = |\text{vocab}|^N$$

- Set of observed N -grams n increases more slowly

$$n \ll |\text{vocab}| \times 50^{N-1}$$

- These are *very* different quantities





Some Corpus Statistics

Corpus	Gigaword	Europarl	GW Apriori	EP Apriori
1-gms	281K	61K	281K	61K
2-gms	5,441K	127K	78,961,000K	3,721,000K
3-gms	274,844K	467K	etc.	
4-gms	599,383K	815K		
5-gms	842,297K	1,028K		



Outline

1 Motivation

- Scaling Language Modelling
- Problems with Lossless Representations
- Lossy Representations

2 Bloom Filter Language Models

- The Bloom Filter
- Extending the BF for Language Modelling

3 Experiments





Information-based Space Lower Bound

Statement

$\log_2 \binom{|\mathcal{U}|}{n}$ bits are needed to represent n items from a Universe \mathcal{U}

Why

- There are $\binom{|\mathcal{U}|}{n}$ distinct sets of size n in \mathcal{U}
- A distinct code must be assigned to each such set
- $\log_2(x)$ bits are needed to represent x distinct codes

Problem

Any lossless representation scales with $|\mathcal{U}|$ (this is not good)



Outline

1 Motivation

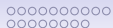
- Scaling Language Modelling
- Problems with Lossless Representations
- Lossy Representations

2 Bloom Filter Language Models

- The Bloom Filter
- Extending the BF for Language Modelling

3 Experiments





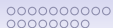
All Language Models are Approximate

- Model assumptions are *approximate*
- Not using all available data is *approximate*
- Model reduction - pruning, clustering etc. - is *approximate*
- Parameter estimates are *approximate*

Bloom filters

Are also approximate but may reduce the above approximations





All Language Models are Approximate

- Model assumptions are *approximate*
- Not using all available data is *approximate*
- Model reduction - pruning, clustering etc. - is *approximate*
- Parameter estimates are *approximate*

Bloom filters

Are also approximate but may reduce the above approximations



Outline

1 Motivation

- Scaling Language Modelling
- Problems with Lossless Representations
- Lossy Representations

2 Bloom Filter Language Models

- The Bloom Filter
- Extending the BF for Language Modelling

3 Experiments



Representing a Set via Hashing

Problem

Represent a set \mathcal{S} of size n drawn from \mathcal{U} where $n \ll |\mathcal{U}|$

Solution

Bloom Filter uses a bitarray of size m and k hash functions

To Train:

- Hash each item k times setting corresponding bits in m

To Test:

- Hash a candidate k times, if all bits set report *member* else *non-member*





Representing a Set via Hashing

Bloom filter (cont.)

- False positives occur with quantifiable probability
- Size and false positive rate *independent* of $|\mathcal{U}|$ (in theory)
- No false negative - i.e., *one-sided error*

E.g. 7.2 bits per item \rightarrow false positive rate ≈ 0.03





Using a Bloom Filter

0 0 0 0 0 0 0 0 0 0 0 0 0

a b c d

Corpus

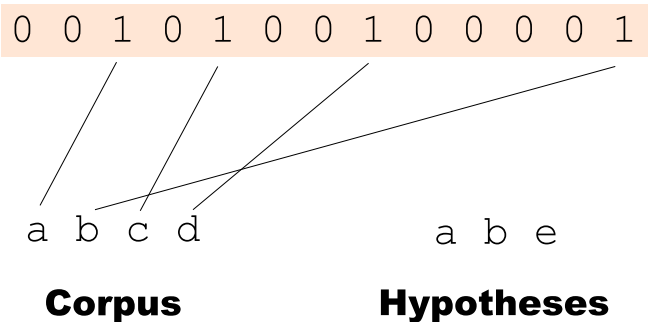
a b e

Hypotheses



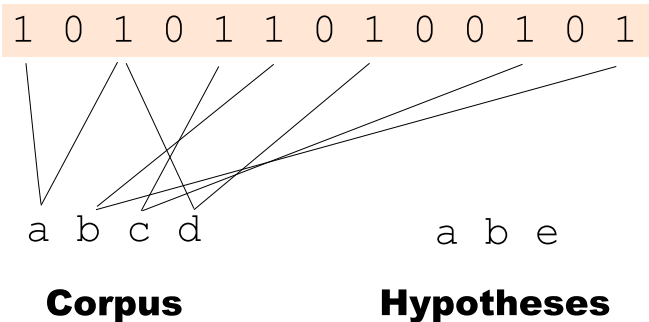


Using a Bloom Filter



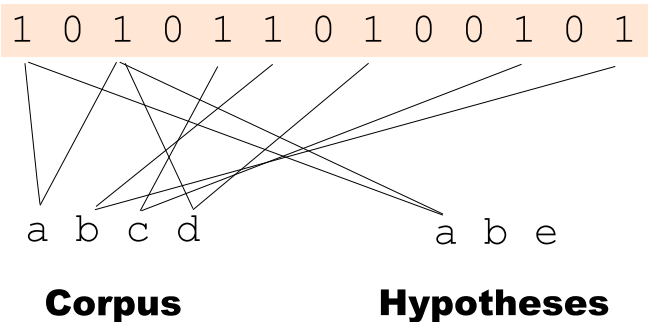


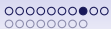
Using a Bloom Filter



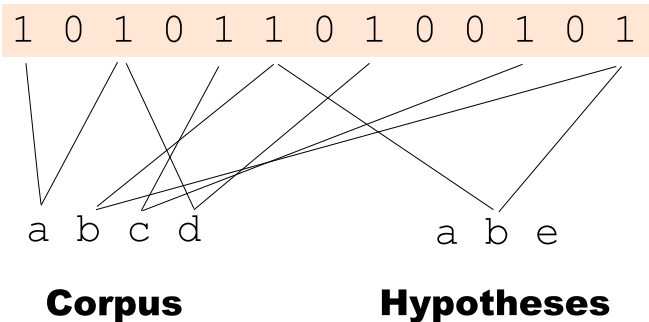


Using a Bloom Filter



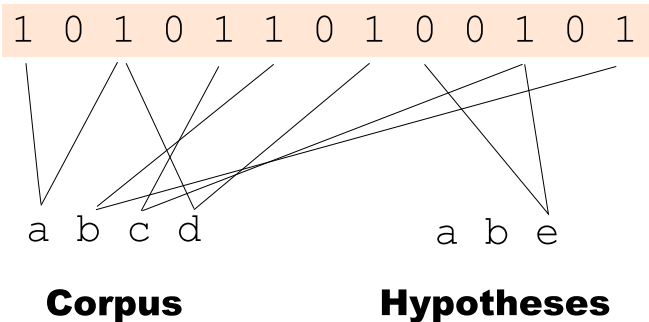


Using a Bloom Filter





Using a Bloom Filter



Optimising a Bloom Filter

How many hash functions?

False positive probability: $f = (1 - p)^k$

where $p = (1 - \frac{1}{m})^{kn}$ is the probability that a bit is still zero

f is minimized for: $k^* = \frac{m}{n} \ln(2)$

Previous Example: $m = 13, n = 4$

With $k = 1$ the false positive rate was $\frac{4}{13} \approx 0.30$

With $k = 2$ the false positive rate was $(\frac{7}{13})^2 \approx 0.28$

Asymptotically setting half the bits is optimal

Outline

1 Motivation

- Scaling Language Modelling
- Problems with Lossless Representations
- Lossy Representations

2 Bloom Filter Language Models

- The Bloom Filter
- Extending the BF for Language Modelling

3 Experiments



Storing Corpus Statistics

Problem

Bloom filters are *not* an associative data structure

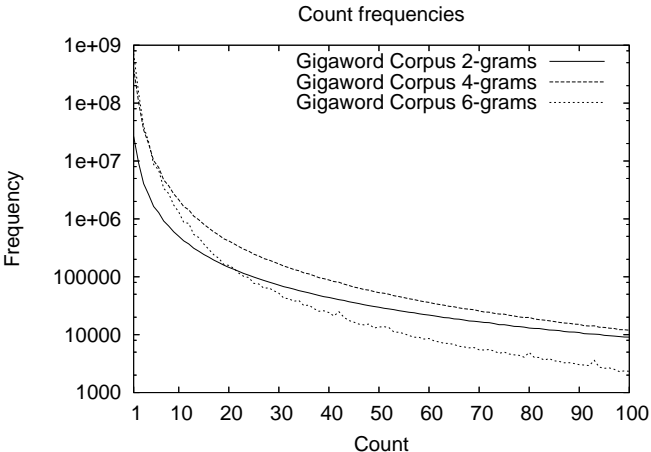
Possible Solutions

- 1 Append each N -gram in set by its count
 - False positive rate will increase by factor $|MAXCOUNT|$
 - Error will no longer be *one-sided*
- 2 Replace each bit by a counter
 - Space increased by factor $\log(|MAXCOUNT|)$
 - Most counters will be set to 1 or 2





Storing Corpus Statistics



Storing Corpus Statistics

Our Solution

Store each N -gram $1 + \lfloor \log(count) \rfloor$ times

Log Frequency Bloom filter

- Store each N -gram appended by an integer j

$$1 \geq j \geq 1 + \lfloor \log(count) \rfloor$$

- Query an N -gram's frequency by appending an integer $j = 1$ and incrementing until hitting a 0

Estimation errors decay exponentially: $f(d) = f^d$ for $d > 0$



Converting Corpus Frequencies to a Set

Raw Counts

the cat	15
the hat	3
the mat	1
the eggs	1
the bacon	1

Quant Counts

	4
	2
→	1
	1
	1

Transformed Set

{the cat_1, the cat_2, the cat_3,
the cat_4, the hat_1, the hat_2,
the mat_1, the eggs_1, the bacon_1}



Storing Related Events

Language Model Statistics

- Witten-Bell: N -gram and suffix counts
- Kneser-Ney: N -gram, prefix, suffix and infix counts

Proxy Events

- Use existence of one event to *infer* a related event
e.g. presence of $N - 1$ -gram implies suffix count ≥ 1

Savings for Witten-Bell

- No need to store singleton suffix counts
- Reduced set $\approx \frac{2}{3}$ size of complete set





Reducing Effective Error Rate

Actual Error Rate

Errors only occur for non-members (i.e. one-sided error)

$$err = Pr(x \notin Corpus | x \in Hypothesis) \times f$$

Can we increase the *a priori* membership probability?

Using Monotonicity of N -gram Event Space

- If a unigram x tests false, then a bigram xy *cannot* be a member
- More generally, $freq(xy) \leq \min\{freq(x), freq(y)\}$



An Example

Interpolated Witten-Bell BF-LM

$$P_{wb}(w_i | w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} P_{ml}(w_i | w_{i-n+1}^{i-1}) \\ + (1 - \lambda_{w_{i-n+1}^{i-1}}) P_{wb}(w_i | w_{i-n+2}^{i-1})$$

where λ_x is defined via,

$$1 - \lambda_x = \frac{\text{count}(x)}{\text{suffix}(x) + \text{count}(x)},$$

- Start from lowest order event (i.e. unigram)
- Bound numerator in ml term by count of denominator
- Bound suffix count by its token frequency
- Truncate computation if ml denominator is zero



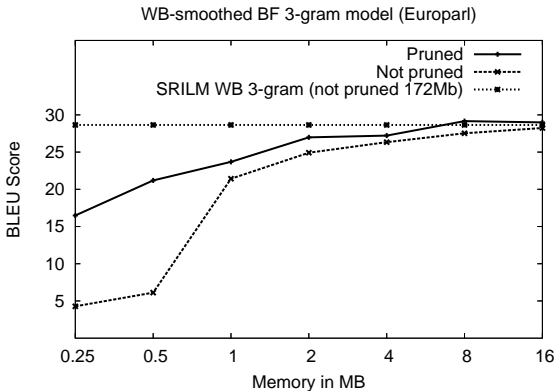
Baseline Models Europarl Witten-Bell

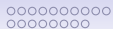
n	Pruned	Types	Mem.	Gzip'd	BLEU
3	No	5.9M	172Mb	51Mb	28.95
3	Yes	2.4M	64Mb	21Mb	28.96
4	No	14.1M	477Mb	129Mb	28.99
4	Yes	3.5M	102Mb	33Mb	29.41
5	No	24.2M	924Mb	238Mb	29.38
5	Yes	4.2M	131Mb	38Mb	29.60



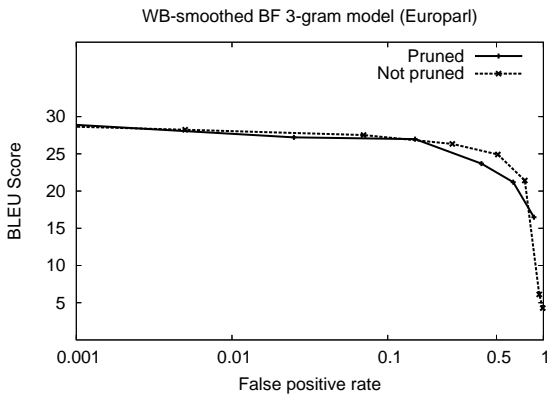


Witten-Bell BF 3-gram Europarl



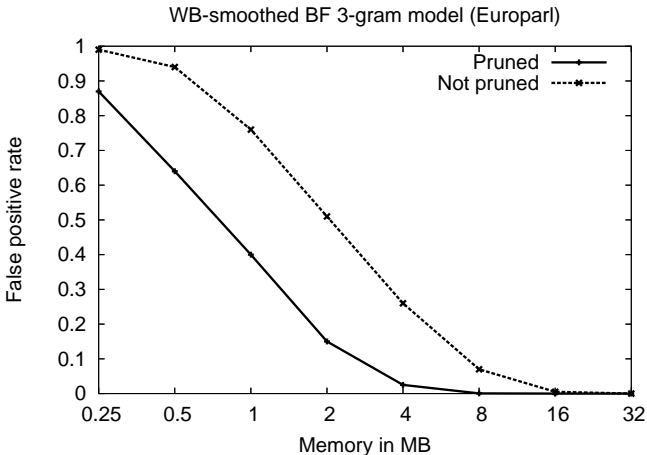


Witten-Bell BF 3-gram Europarl



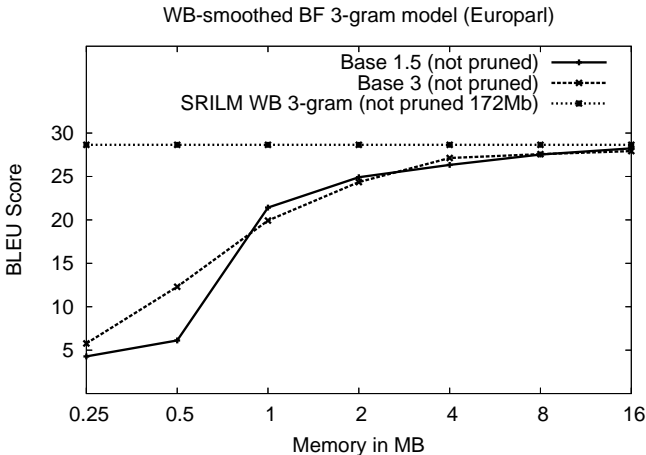


Witten-Bell BF 3-gram Europarl



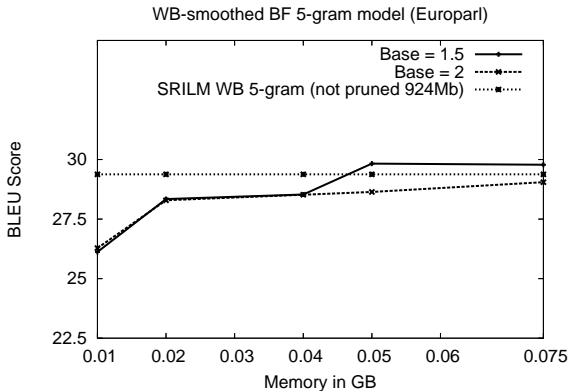


Witten-Bell BF 3-gram Europarl





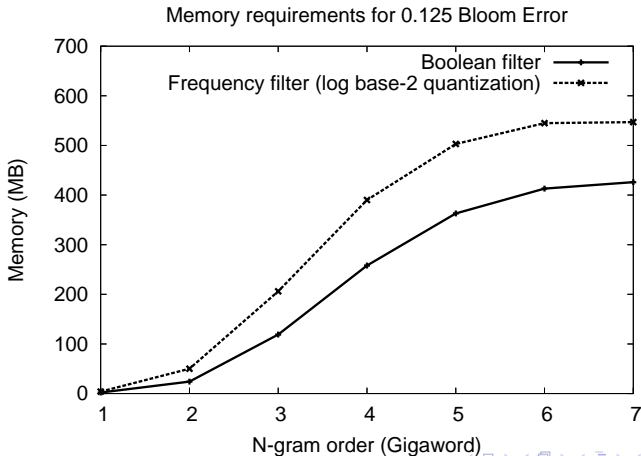
Witten-Bell BF 5-gram Europarl





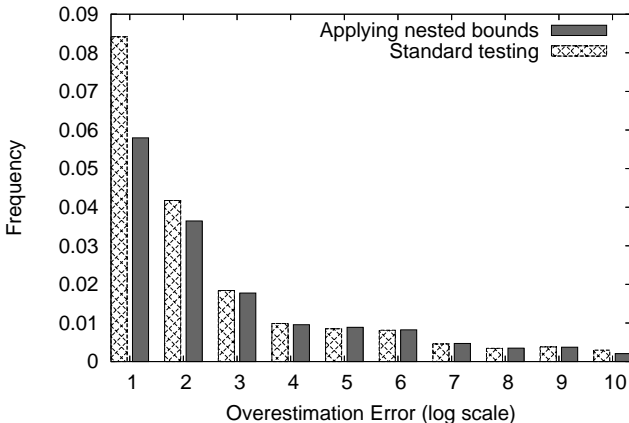
Log Frequency Scheme for Corpus Statistics

- Set increases by *less than 2* when storing frequencies



Applying Nested Bounds

Frequency Estimation Errors WB 3-gms BF (Europarl, 16Mb)





Summary

- Bloom filters can be used effectively for language modelling **below information-theoretic lower bounds**
- **11 - 15 bits** per N -gram seems like enough
- Future Work
 - Reducing computation in the log frequency BF scheme
 - Hybrid models - e.g. explicit 1,2-grams + BF 3,4,5-grams
 - Other NLP applications of log frequency BF framework



Some References



R. Motwani and P. Raghavan.

Randomized Algorithms.

Cambridge University Press, 1995.



B. Bloom.

Space/time tradeoffs in hash coding with allowable errors.

Communications of the ACM, 13:422–426, 1970.



A. Broder and M. Mitzenmacher.

Network Applications of Bloom filters: A Survey.

Internet Mathematics, 1(4):485–509, 2005.



Thanks

- Thanks for listening!
- Thanks to all the Moses Team!

