Randomised Language Modelling for Statistical Machine Translation

David Talbot and Miles Osborne
School of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh, EH8 9LW, UK
d.r.talbot@sms.ed.ac.uk, miles@inf.ed.ac.uk

Abstract

A Bloom filter (BF) is a randomised data structure for set membership queries. Its space requirements are significantly below lossless information-theoretic lower bounds but it produces false positives with some quantifiable probability. Here we explore the use of BFs for language modelling in statistical machine translation.

We show how a BF containing n-grams can enable us to use much larger corpora and higher-order models complementing a conventional n-gram LM within an SMT system. We also consider (i) how to include approximate frequency information efficiently within a BF and (ii) how to reduce the error rate of these models by first checking for lower-order sub-sequences in candidate n-grams. Our solutions in both cases retain the one-sided error guarantees of the BF while taking advantage of the Zipf-like distribution of word frequencies to reduce the space requirements.

1 Introduction

Language modelling (LM) is a crucial component in statistical machine translation (SMT). Standard n-gram language models assign probabilities to translation hypotheses in the target language, typically as smoothed trigram models, e.g. (Chiang, 2005). Although it is well-known that higher-order LMs and models trained on additional monolingual corpora can yield better translation performance, the challenges in deploying large LMs are not trivial. Increasing the order of an n-gram model can result in an exponential increase in the number of parameters; for corpora such as the English Gigaword corpus, for instance, there are 300 million distinct trigrams and over 1.2 billion 5-grams. Since a LM may be queried millions of times per sentence, it should ideally reside locally in memory to avoid time-consuming remote or disk-based look-ups.

Against this background, we consider a radically different approach to language modelling: instead of explicitly storing all distinct n-grams, we store a randomised representation. In particular, we show that the *Bloom filter* (Bloom (1970); BF), a simple space-efficient randomised data structure for representing sets, may be used to represent statistics from larger corpora and for higher-order n-grams to complement a conventional smoothed trigram model within an SMT decoder. [1]

The space requirements of a Bloom filter are quite spectacular, falling significantly below information-theoretic error-free lower bounds while query times are constant. This efficiency, however, comes at the price of *false positives*: the filter may erroneously report that an item not in the set is a member. False negatives, on the other hand, will never occur: the error is said to be *one-sided*.

In this paper, we show that a Bloom filter can be used effectively for language modelling within an SMT decoder and present the *log-frequency Bloom filter*, an extension of the standard Boolean BF that

[1] For extensions of the framework presented here to standalone smoothed Bloom filter language models, we refer the reader to a companion paper (Talbot and Osborne, 2007).

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 512–519,
Prague, Czech Republic, June 2007.

takes advantage of the Zipf-like distribution of corpus statistics to allow frequency information to be associated with n-grams in the filter in a space-efficient manner. We then propose a mechanism, *sub-sequence filtering*, for reducing the error rates of these models by using the fact that an n-gram's frequency is bound from above by the frequency of its least frequent sub-sequence.

We present machine translation experiments using these models to represent information regarding higher-order n-grams and additional larger monolingual corpora in combination with conventional smoothed trigram models. We also run experiments with these models in isolation to highlight the impact of different order n-grams on the translation process. Finally we provide some empirical analysis of the effectiveness of both the log frequency Bloom filter and sub-sequence filtering.

2 The Bloom filter

In this section, we give a brief overview of the Bloom filter (BF); refer to Broder and Mitzenmacher (2005) for a more in detailed presentation. A BF represents a set $\mathcal{S} = \{x_1, x_2, ..., x_n\}$ with n elements drawn from a universe $\mathcal{U}$ of size N. The structure is attractive when $N \gg n$. The only significant storage used by a BF consists of a bit array of size m. This is initially set to hold zeroes. To train the filter we hash each item in the set k times using distinct hash functions $h_1, h_2, ..., h_k$. Each function is assumed to be independent from each other and to map items in the universe to the range 1 to m uniformly at random. The k bits indexed by the hash values for each item are set to 1; the item is then discarded. Once a bit has been set to 1 it remains set for the lifetime of the filter. Distinct items may not be hashed to k distinct locations in the filter; we ignore collisons. Bits in the filter can, therefore, be *shared* by distinct items allowing significant space savings but introducing a non-zero probability of false positives at test time. There is no way of directly retrieving or ennumerating the items stored in a BF.

At test time we wish to discover whether a given item was a member of the original set. The filter is queried by hashing the test item using the same k hash functions. If all bits referenced by the k hash values are 1 then we *assume* that the item was a member; if any of them are 0 then we *know* it was not. True members are always correctly identified, but a false positive will occur if all k corresponding bits were set by other items during training and the item was not a member of the training set. This is known as a *one-sided error*.

The probability of a false postive, f, is clearly the probability that none of k randomly selected bits in the filter are still 0 after training. Letting p be the proportion of bits that are still zero after these n elements have been inserted, this gives,

$$f = (1 - p)^k.$$

As n items have been entered in the filter by hashing each k times, the probability that a bit is still zero is,

$$p^{'} = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-\frac{kn}{m}}$$

which is the expected value of p. Hence the false positive rate can be approximated as,

$$f = (1 - p)^k \approx (1 - p^{'})^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k.$$

By taking the derivative we find that the number of functions k^* that minimizes f is,

$$k^* = \ln 2 \cdot \frac{m}{n}.$$

which leads to the intuitive result that exactly half the bits in the filter will be set to 1 when the optimal number of hash functions is chosen.

The fundmental difference between a Bloom filter's space requirements and that of any lossless representation of a set is that the former does not depend on the size of the (exponential) universe N from which the set is drawn. A lossless representation scheme (for example, a hash map, trie etc.) must depend on N since it assigns a distinct representation to each possible set drawn from the universe.

3 Language modelling with Bloom filters

In our experiments we make use of both standard (i.e. Boolean) BFs containing n-gram types drawn from a training corpus and a novel BF scheme, the *log-frequency Bloom filter*, that allows frequency information to be associated efficiently with items stored in the filter.

Algorithm 1 Training frequency BF

Input: $\mathcal{S}_{train}$, $\{h_1, ...h_k\}$ and $\mathcal{BF} = \emptyset$
Output: $\mathcal{BF}$
for all $x \in \mathcal{S}_{train}$ **do**
  $c(x) \leftarrow$ frequency of n-gram x in $\mathcal{S}_{train}$
  $qc(x) \leftarrow$ quantisation of $c(x)$ (Eq. 1)
  for $j = 1$ to $qc(x)$ **do**
    for $i = 1$ to k **do**
      $h_i(x) \leftarrow$ hash of event $\{x, j\}$ under h_i
      $\mathcal{BF}[h_i(x)] \leftarrow 1$
    end for
  end for
end for
return $\mathcal{BF}$

3.1 Log-frequency Bloom filter

The efficiency of our scheme for storing n-gram statistics within a BF relies on the Zipf-like distribution of n-gram frequencies in natural language corpora: most events occur an extremely small number of times, while a small number are very frequent.

We quantise raw frequencies, $c(x)$, using a logarithmic codebook as follows,

$$qc(x) = 1 + \lfloor \log_b c(x) \rfloor. \quad (1)$$

The precision of this codebook decays exponentially with the raw counts and the scale is determined by the base of the logarithm b; we examine the effect of this parameter in experiments below.

Given the quantised count $qc(x)$ for an n-gram x, the filter is trained by entering composite events consisting of the n-gram appended by an integer counter j that is incremented from 1 to $qc(x)$ into the filter. To retrieve the quantised count for an n-gram, it is first appended with a count of 1 and hashed under the k functions; if this tests positive, the count is incremented and the process repeated. The procedure terminates as soon as any of the k hash functions hits a 0 and the previous count is reported. The one-sided error of the BF and the training scheme ensure that the actual quantised count cannot be larger than this value. As the counts are quantised logarithmically, the counter will be incremented only a small number of times. The training and testing routines are given here as Algorithms 1 and 2 respectively.

Errors for the log-frequency BF scheme are one-sided: frequencies will never be underestimated.

Algorithm 2 Test frequency BF

Input: x, $MAXQCOUNT$, $\{h_1, ...h_k\}$ and $\mathcal{BF}$
Output: Upper bound on $qc(x) \in \mathcal{S}_{train}$
for $j = 1$ to $MAXQCOUNT$ **do**
  for $i = 1$ to k **do**
    $h_i(x) \leftarrow$ hash of event $\{x, j\}$ under h_i
    if $\mathcal{BF}[h_i(x)] = 0$ **then**
      return $j - 1$
    end if
  end for
end for

The probability of overestimating an item's frequency decays exponentially with the size of the overestimation error d (i.e. as f^d for $d > 0$) since each erroneous increment corresponds to a single false positive and d such independent events must occur together.

3.2 Sub-sequence filtering

The error analysis in Section 2 focused on the false positive rate of a BF; if we deploy a BF within an SMT decoder, however, the actual error rate will also depend on the *a priori* membership probability of items presented to it. The error rate Err is,

$$Err = Pr(x \notin S_{train}|Decoder)f.$$

This implies that, unlike a conventional lossless data structure, the model's accuracy depends on other components in system and how it is queried.

We take advantage of the monotonicity of the n-gram event space to place upper bounds on the frequency of an n-gram prior to testing for it in the filter and potentially truncate the outer loop in Algorithm 2 when we know that the test could only return postive in error.

Specifically, if we have stored lower-order n-grams in the filter, we can infer that an n-gram cannot present, if any of its sub-sequences test negative. Since our scheme for storing frequencies can never *underestimate* an item's frequency, this relation will generalise to frequencies: an n-gram's frequency cannot be greater than the frequency of its least frequent sub-sequence as reported by the filter,

$$c(w_1, ..., w_n) \leq \min \{c(w_1, ..., w_{n-1}), c(w_2, ..., w_n)\}.$$

We use this to reduce the effective error rate of BF-LMs that we use in the experiments below.

3.3 Bloom filter language model tests

A standard BF can implement a Boolean 'language model' test: have we seen some fragment of language before? This does not use any frequency information. The *Boolean BF-LM* is a standard BF containing all n-grams of a certain length in the training corpus, $\mathcal{S}_{train}$. It implements the following binary feature function in a log-linear decoder,

$$\phi_{\mathbf{bool}}(x) \geq \delta(x \in \mathcal{S}_{train})$$

Separate Boolean BF-LMs can be included for different order n and assigned distinct log-linear weights that are learned as part of a minimum error rate training procedure (see Section 4).

The *log-frequency BF-LM* implements a multinomial feature function in the decoder that returns the value associated with an n-gram by Algorithm 2.

$$\phi_{\mathbf{logfreq}}(x) \geq qc(x) \in \mathcal{S}_{train}$$

Sub-sequence filtering can be performed by using the minimum value returned by lower-order models as an upper-bound on the higher-order models.

By boosting the score of hypotheses containing n-grams observed in the training corpus while remaining agnostic for unseen n-grams (with the exception of errors), these feature functions have more in common with maximum entropy models than conventionally smoothed n-gram models.

4 Experiments

We conducted a range of experiments to explore the effectiveness and the error-space trade-off of Bloom filters for language modelling in SMT. The space-efficiency of these models also allows us to investigate the impact of using much larger corpora and higher-order n-grams on translation quality. While our main experiments use the Bloom filter models *in conjunction* with a conventional smoothed trigram model, we also present experiments with these models in isolation to highlight the impact of different order n-grams on the translation process. Finally, we present some empirical analysis of both the log-frequency Bloom filter and the sub-sequence filtering technique which may be of independent interest.

Model	EP-KN-3	EP-KN-4	AFP-KN-3
Memory	64M	99M	1.3G
gzip size	21M	31M	481M
1-gms	62K	62K	871K
2-gms	1.3M	1.3M	16M
3-gms	1.1M	1.0M	31M
4-gms	N/A	1.1M	N/A

Table 1: Baseline and Comparison Models

4.1 Experimental set-up

All of our experiments use publically available resources. We use the French-English section of the *Europarl* (EP) corpus for parallel data and language modelling (Koehn, 2003) and the *English Gigaword Corpus* (LDC2003T05; GW) for additional language modelling.

Decoding is carried-out using the Moses decoder (Koehn and Hoang, 2007). We hold out 500 test sentences and 250 development sentences from the parallel text for evaluation purposes. The feature functions in our models are optimised using minimum error rate training and evaluation is performed using the BLEU score.

4.2 Baseline and comparison models

Our baseline LM and other comparison models are conventional n-gram models smoothed using modified Kneser-Ney and built using the SRILM Toolkit (Stolcke, 2002); as is standard practice these models drop entries for n-grams of size 3 and above when the corresponding discounted count is less than 1. The baseline language model, EP-KN-3, is a trigram model trained on the English portion of the parallel corpus. For additional comparisons we also trained a smoothed 4-gram model on this Europarl data (EP-KN-4) and a trigram model on the Agence France Press section of the Gigaword Corpus (AFP-KN-3).

Table 1 shows the amount of memory these models take up on disk and compressed using the *gzip* utility in parentheses as well as the number of distinct n-grams of each order. We give the gzip compressed size as an optimistic lower bound on the size of any lossless representation of each model.[2]

[2]Note, in particular, that gzip compressed files do *not* support direct random access as required by our application.

Corpus	Europarl	Gigaword
1-gms	61K	281K
2-gms	1.3M	5.4M
3-gms	4.7M	275M
4-gms	9.0M	599M
5-gms	10.3M	842M
6-gms	10.7M	957M

Table 2: Number of distinct n-grams

4.3 Bloom filter-based models

To create Bloom filter LMs we gathered n-gram counts from both the Europarl (EP) and the whole of the Gigaword Corpus (GW). Table 2 shows the numbers of distinct n-grams in these corpora. Note that we use no pruning for these models and that the numbers of distinct n-grams is of the same order as that of the recently released Google Ngrams dataset (LDC2006T13). In our experiments we create a range of models referred to by the corpus used (EP or GW), the order of the n-gram(s) entered into the filter (1 to 10), whether the model is Boolean (Bool-BF) or provides frequency information (Freq-BF), whether or not sub-sequence filtering was used (FTR) and whether it was used in conjunction with the baseline trigram (+EP-KN-3).

4.4 Machine translation experiments

Our first set of experiments examines the relationship between memory allocated to the BF and BLEU score. We present results using the Boolean BF-LM in isolation and then both the Boolean and log-frequency BF-LMS to add 4-grams to our baseline 3-gram model.Our second set of experiments adds 3-grams and 5-grams from the Gigaword Corpus to our baseline. Here we constrast the Boolean BF-LM with the log-frequency BF-LM with different quantisation bases (2 = fine-grained and 5 = coarse-grained). We then evaluate the sub-sequence filtering approach to reducing the actual error rate of these models by adding both 3 and 4-grams from the Gigaword Corpus to the baseline. Since the BF-LMs easily allow us to deploy very high-order n-gram models, we use them to evaluate the impact of different order n-grams on the translation process presenting results using the Boolean and log-frequency BF-LM *in isolation* for n-grams of order 1 to 10.

Model	EP-KN-3	EP-KN-4	AFP-KN-3
BLEU	28.51	29.24	29.17
Memory	64M	99M	1.3G
gzip size	21M	31M	481M

Table 3: Baseline and Comparison Models

4.5 Analysis of BF extensions

We analyse our log-frequency BF scheme in terms of the additional memory it requires and the error rate compared to a non-redundant scheme. The non-redundant scheme involves entering *just* the exact quantised count for each n-gram and then searching over the range of possible counts at test time starting with the count with maximum *a priori* probability (i.e. 1) and incrementing until a count is found or the whole codebook has been searched (here the size is 16).

We also analyse the sub-sequence filtering scheme directly by creating a BF with only 3-grams and a BF containing both 2-grams and 3-grams and comparing their *actual* error rates when presented with 3-grams that are all known to be negatives.

5 Results

5.1 Machine translation experiments

Table 3 shows the results of the baseline (EP-KN-3) and other conventional n-gram models trained on larger corpora (AFP-KN-3) and using higher-order dependencies (EP-KN-4). The larger models improve somewhat on the baseline performance.

Figure 1 shows the relationship between space allocated to the BF models and BLEU score (left) and false positive rate (right) respectively. These experiments do *not* include the baseline model. We can see a clear correlation between memory / false positive rate and translation performance.

Adding 4-grams in the form of a Boolean BF or a log-frequency BF (see Figure 2) improves on the 3-gram baseline with little additional memory (around 4MBs) while performing on a par with or above the Europarl 4-gram model with around 10MBs; this suggests that a lossy representation of the unpruned set of 4-grams contains more useful information than a lossless representation of the pruned set.[3]

[3] An unpruned modified Kneser-Ney 4-gram model on the Eurpoparl data scores slightly higher - 29.69 - while taking up 489MB (132MB gzipped).

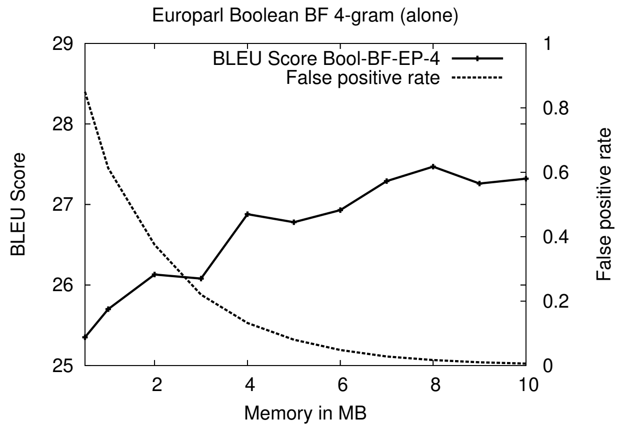

Figure 1: Space/Error vs. BLEU Score.

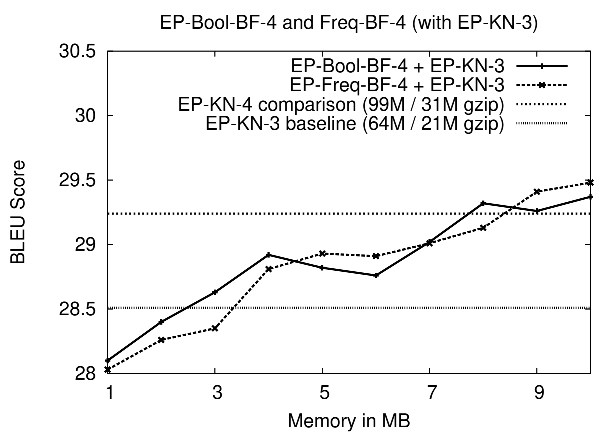

Figure 2: Adding 4-grams with Bloom filters.

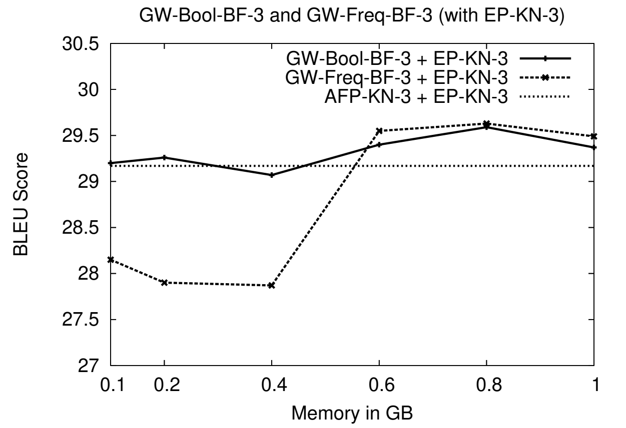

Figure 3: Adding GW 3-grams with Bloom filters.

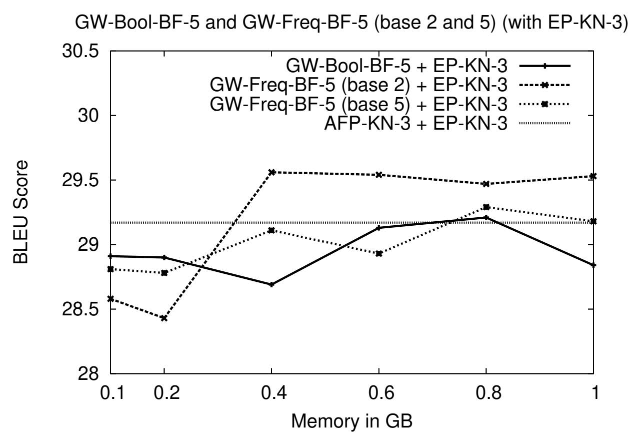

Figure 4: Comparison of different quantisation rates.

As the false positive rate exceeds 0.20 the performance is severly degraded. Adding 3-grams drawn from the whole of the Gigaword corpus rather than simply the Agence France Press section results in slightly improved performance with signficantly less memory than the AFP-KN-3 model (see Figure 3).

Figure 4 shows the results of adding 5-grams drawn from the Gigaword corpus to the baseline. It also contrasts the Boolean BF and the log-frequency BF suggesting in this case that the log-frequency BF can provide useful information when the quantisation base is relatively fine-grained (base 2). The Boolean BF and the base 5 (coarse-grained quantisation) log-frequency BF perform approximately the same. The base 2 quantisation performs worse for smaller amounts of memory, possibly due to the larger set of events it is required to store.

Figure 5 shows sub-sequence filtering resulting in a small increase in performance when false positive rates are high (i.e. less memory is allocated). We believe this to be the result of an increased *a priori* membership probability for n-grams presented to the filter under the sub-sequence filtering scheme.

Figure 6 shows that for this task the most useful n-gram sizes are between 3 and 6.

5.2 Analysis of BF extensions

Figure 8 compares the memory requirements of the log-frequencey BF (base 2) and the Boolean

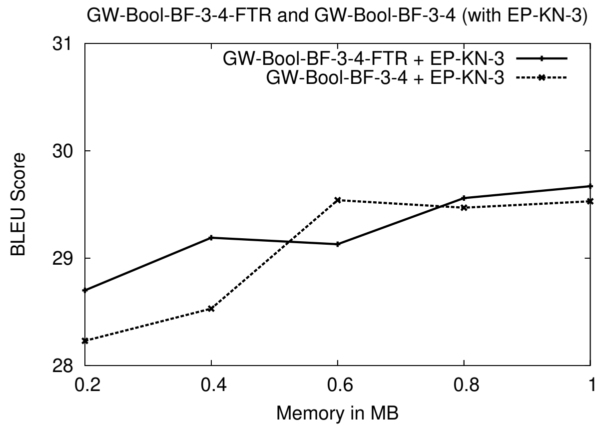

Figure 5: Effect of sub-sequence filtering.

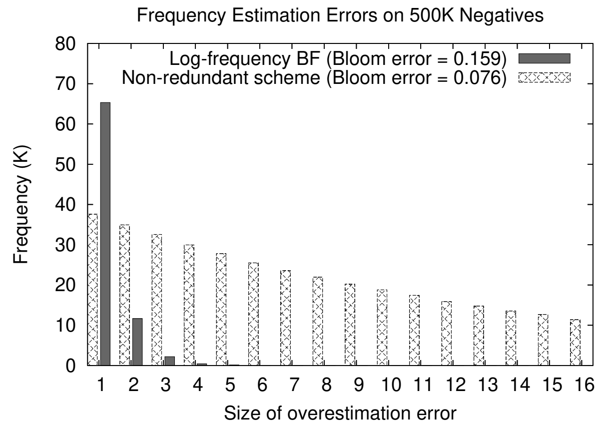

Figure 7: Frequency estimation errors.

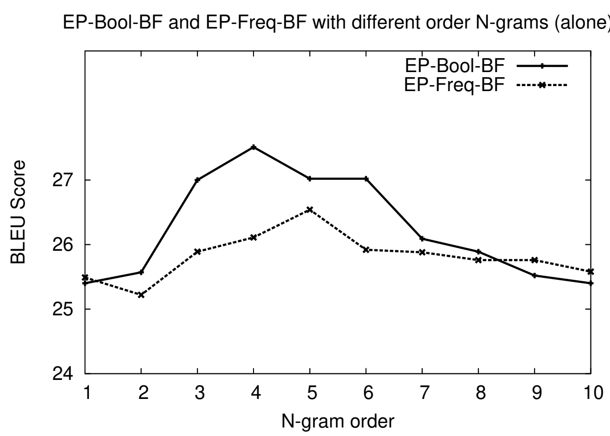

Figure 6: Impact of n-grams of different sizes.

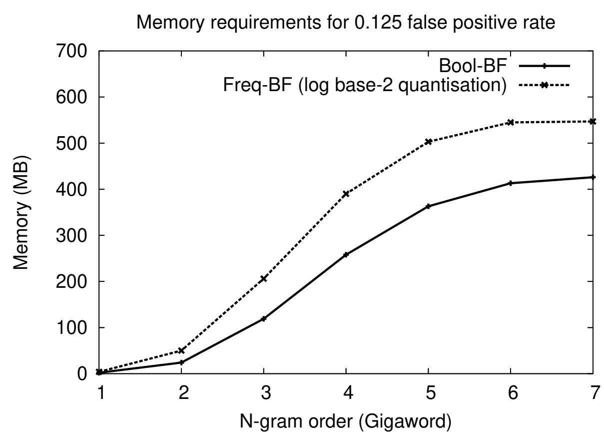

Figure 8: Comparison of memory requirements.

BF for various order n-gram sets from the Gigaword Corpus with the same underlying false positive rate (0.125). The additional space required by our scheme for storing frequency information is less than a factor of 2 compared to the standard BF.

Figure 7 shows the number and size of frequency estimation errors made by our log-frequency BF scheme and a non-redundant scheme that stores only the exact quantised count. We presented 500K negatives to the filter and recorded the frequency of overestimation errors of each size. As shown in Section 3.1, the probability of overestimating an item's frequency under the log-frequency BF scheme decays exponentially in the size of this overestimation error. Although the non-redundant scheme requires fewer items be stored in the filter and, therefore, has a lower underlying false positive rate (0.076 versus 0.159), in practice it incurs a *much* higher error rate (0.717) with many large errors.

Figure 9 shows the impact of sub-sequence filtering on the actual error rate. Although, the false positive rate for the BF containing 2-grams, in addition, to 3-grams (filtered) is higher than the false positive rate of the unfiltered BF containing only 3-grams, the actual error rate of the former is lower for models with less memory. By testing for 2-grams prior to querying for the 3-grams, we can avoid performing some queries that may otherwise have incurred errors using the fact that a 3-gram cannot be present if one of its constituent 2-grams is absent.

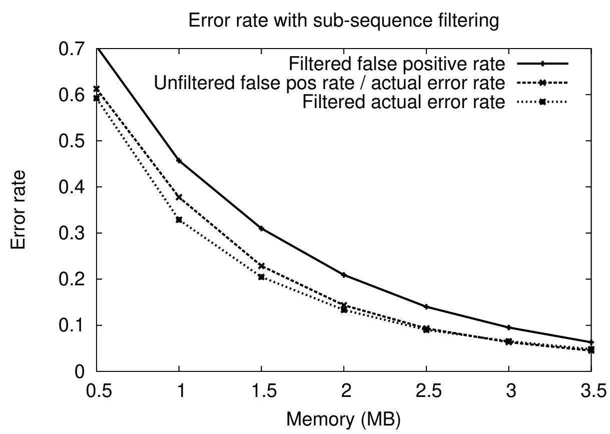

Figure 9: Error rate with sub-sequence filtering.

6 Related Work

We are not the first people to consider building very large scale LMs: Kumar et al. used a four-gram LM for re-ranking (Kumar et al., 2005) and in unpublished work, Google used substantially larger n-grams in their SMT system. Deploying such LMs requires either a cluster of machines (and the overheads of remote procedure calls), per-sentence filtering (which again, is slow) and/or the use of some other lossy compression (Goodman and Gao, 2000). Our approach can complement all these techniques.

Bloom filters have been widely used in database applications for reducing communications overheads and were recently applied to encode word frequencies in information retrieval (Linari and Weikum, 2006) using a method that resembles the non-redundant scheme described above. Extensions of the BF to associate frequencies with items in the set have been proposed e.g., (Cormode and Muthukrishn, 2005); while these schemes are more general than ours, they incur greater space overheads for the distributions that we consider here.

7 Conclusions

We have shown that Bloom Filters can form the basis for space-efficient language modelling in SMT. Extending the standard BF structure to encode corpus frequency information and developing a strategy for reducing the error rates of these models by sub-sequence filtering, our models enable higher-order n-grams and larger monolingual corpora to be used more easily for language modelling in SMT. In a companion paper (Talbot and Osborne, 2007) we have proposed a framework for deriving conventional smoothed n-gram models from the log-frequency BF scheme allowing us to do away entirely with the standard n-gram model in an SMT system. We hope the present work will help establish the Bloom filter as a practical alternative to conventional associative data structures used in computational linguistics. The framework presented here shows that with some consideration for its workings, the randomised nature of the Bloom filter need not be a significant impediment to is use in applications.

References

B. Bloom. 1970. Space/time tradeoffs in hash coding with allowable errors. *CACM*, 13:422–426.

A. Broder and M. Mitzenmacher. 2005. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan.

G. Cormode and S. Muthukrishn. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75.

J. Goodman and J. Gao. 2000. Language model size reduction by pruning and clustering. In *ICSLP'00*, Beijing, China.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proc. of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP/Co-NLL).*

P. Koehn. 2003. Europarl: A multilingual corpus for evaluation of machine translation philipp koehn, draft. Available at:http://people.csail.mit.edu/ koehn/publications/europarl.ps.

S. Kumar, Y. Deng, and W. Byrne. 2005. Johns Hopkins University - Cambridge University Chinese-English and Arabic-English 2005 NIST MT Evaluation Systems. In *Proceedings of 2005 NIST MT Workshop*, June.

Alessandro Linari and Gerhard Weikum. 2006. Efficient peer-to-peer semantic overlay networks based on statistical language models. In *Proceedings of the International Workshop on IR in Peer-to-Peer Networks*, pages 9–16, Arlington.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proc. of the Intl. Conf. on Spoken Lang. Processing, 2002*.

David Talbot and Miles Osborne. 2007. Smoothed Bloom filter language models: Tera-scale LMs on the cheap. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP/Co-NLL)*, June.