Smoothed Bloom filter language models: Tera-Scale LMs on the Cheap

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 fall significantly below lossless information-theoretic lower bounds but it produces false positives with some quantifiable probability. Here we present a general framework for deriving smoothed language model probabilities from BFs.

We investigate how a BF containing n-gram statistics can be used as a *direct replacement* for a conventional n-gram model. Recent work has demonstrated that corpus statistics can be stored efficiently within a BF, here we consider how smoothed language model probabilities can be derived efficiently from this randomised representation. Our proposal takes advantage of the one-sided error guarantees of the BF and simple inequalities that hold between related n-gram statistics in order to further reduce the BF storage requirements and the error rate of the derived probabilities. We use these models as replacements for a conventional language model in machine translation experiments.

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 (Chiang, 2005). Although it is well-known that higher-order language models and models trained on additional monolingual corpora can significantly improve translation performance, deploying such language models is not trivial. Increasing the order of an n-gram model can result in an exponential increase in the number of parameters; for the English Gigaword corpus, for instance, there are 300 million distinct trigrams and over 1.2 billion distinct five-grams. Since a language model is potentially 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 from our corpus, we create an implicit randomised representation of these statistics. This allows us to drastically reduce the space requirements of our models. In this paper, we build on recent work (Talbot and Osborne, 2007) that demonstrated how the *Bloom filter* (Bloom (1970); BF), a space-efficient randomised data structure for representing sets, could be used to store corpus statistics efficiently. Here, we propose a framework for deriving smoothed n-gram models from such structures and show via machine translation experiments that these *smoothed Bloom filter language models* may be used as direct replacements for standard n-gram models in SMT.

The space requirements of a Bloom filter are quite spectacular, falling significantly below information-theoretic error-free lower bounds. 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

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 468–476, Prague, June 2007.

never occur: the error is said to be *one-sided*. Our framework makes use of the *log-frequency Bloom filter* presented in (Talbot and Osborne, 2007), and described briefly below, to compute smoothed conditional n-gram probabilities on the fly. It takes advantage of the one-sided error guarantees of the Bloom filter and certain inequalities that hold between related n-gram statistics drawn from the same corpus to reduce both the error rate and the computation required in deriving these probabilities.

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.

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

Recent work (Talbot and Osborne, 2007) presented a scheme for associating static frequency information with a set of n-grams in a BF efficiently.[1]

3.1 Log-frequency Bloom filter

The efficiency of the scheme for storing n-gram statistics within a BF presented in Talbot and Osborne (2007) relies on the Zipf-like distribution of n-gram frequencies: most events occur an extremely small number of times, while a small number are very frequent. We assume that raw counts are quantised and employ a logarithmic codebook that maps counts, $c(x)$, to quantised counts, $qc(x)$, as follows,

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

[1]Note that as described the Bloom filter is *not* an associative data structure and provides only a Boolean function characterising the set that has been stored in it.

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 $\mathcal{BF}$

Algorithm 2 Test frequency BF

Input: x, $MAXQCOUNT$, $\{h_1, ...h_k\}$ and $\mathcal{BF}$
Output: Upper bound on $c(x) \in \mathcal{S}_{train}$
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 $E[c(x)|qc(x) = j - 1]$ (Eq. 2)
    end if
  end for
end for

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 on our language models 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 an n-gram's frequency, the n-gram is first appended with a counter set to 1 and hashed under the k functions; if this tests positive, the counter is incremented and the process repeated. The procedure terminates as soon as any of the k hash functions hits a 0 and the previous value of the counter 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 is usually incremented only a small number of times.

We can then approximate the original frequency of the n-gram by taking its expected value given the quantised count retrieved,

$$E[c(x)|qc(x) = j] = \frac{b^{j-1} + b^j - 1}{2}. \tag{2}$$

These training and testing routines are repeated here as Algorithms 1 and 2 respectively.

As noted in Talbot and Osborne (2007), errors for this log-frequency BF scheme are one-sided: frequencies will never be underestimated. 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.

The efficiency of the log-frequency BF scheme can be understood from an entropy encoding perspective under the distribution over frequencies of n-gram types: the most common frequency (the singleton count) is assigned the shortest code (length k) while rarer frequencies (those for more common n-grams) are assigned increasingly longer codes ($k \times qc(x)$).

3.2 Smoothed BF language models

A standard n-gram language model assigns conditional probabilities to target words given a certain context. In practice, most standard n-gram language models employ some form of interpolation whereby probabilities conditioned on the most specific context consisting usually of the $n - 1$ preceding tokens are combined with more robust estimates based on less specific conditioning events. To compute smoothed language model probabilities, we generally require access to the frequencies of n-grams of length 1 to n in our training corpus. Depending on the smoothing scheme, we may also need auxiliary statistics regarding the number of distinct suffixes for each n-gram (e.g., Witten-Bell and Kneser-Ney smoothing) and the number of distinct prefixes or contexts in which they appear (e.g., Kneser-Ney). We can use a single BF to store these statistics but need to distinguish each type of event (e.g., raw counts, suffix counts, etc.). Here we use a distinct set of k hash functions for each such category.

Our motivation for storing the corpus statistics directly rather than precomputed probabilities is twofold: (i) the efficiency of the scheme described above for storing frequency information together with items in a BF relies on the frequencies having a Zipf-like distribution; while this is definitely true for corpus statistics, it may well not hold for probabilities estimated from them; (ii) as will become apparent below, by using the corpus statistics directly, we will be able to make additional savings in terms of both space and error rate by using simple inequalities that hold for related information drawn consistently from the same corpus; it is not clear whether such bounds can be established for probabilities computed from these statistics.

3.2.1 Proxy items

There is a potential risk of redundancy if we represent related statistics using the log-frequency BF scheme presented in Talbot and Osborne (2007). In particular, we do not need to store information explicitly that is necessarily implied by the presence of another item in the training set, if that item can be identified efficiently at query time when needed. We use the term *proxy item* to refer to items whose presence in the filter implies the existence of another item *and* that can be efficiently queried given the implied item. In using a BF to store corpus statistics for language modelling, for example, we can use the event corresponding to an n-gram and the counter set to 1 as a proxy item for a distinct prefix, suffix or context count of 1 for the same n-gram since (ignoring sentence boundaries) it must have been preceded and followed by at least one distinct type, i.e.,

$$qc(w_1, ..., w_n) \geq 1 \in BF \Rightarrow s(w_1, ..., w_n) \geq 1,$$

where $s(\cdot)$ is the number of the distinct types following this n-gram in the training corpus. We show below that such lower bounds allow us to significantly reduce the memory requirements for a BF language model.

3.2.2 Monotonicity of n-gram event space

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.

Assuming that statistics are entered consistently from the same corpus, we can take advantage of the monotonicity of the n-gram event space to place upper bounds on the frequencies of events to be retrieved from the filter prior to querying it, thereby reducing the *a priori* probability of a negative and consequently the error rate.

Specifically, since the log-frequency BF scheme will never underestimate an item's frequency, we can apply the following inequality recursively and bound the frequency of an n-gram by that of its least frequent subsequence,

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

We use this to reduce the error rate of an interpolated BF language model described below.

3.3 Witten-Bell smoothed BF LM

As an example application of our framework, we now describe a scheme for creating and querying a log-frequency BF to estimate n-gram language model probabilities using Witten-Bell smoothing (Bell et al., 1990). Other smoothing schemes, notably Kneser-Ney, could be described within this framework using additional proxy relations for infix and prefix counts.

In Witten-Bell smoothing, an n-gram's probability is discounted by a factor proportional to the number of times that the $n-1$-gram preceding the current word was observed preceding a novel type in the training corpus. It is defined recursively as,

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

where λ_x is defined via,

$$1 - \lambda_x = \frac{c(x)}{s(x) + c(x)},$$

and $P_{ml}(\cdot)$ is the maximum likelihood estimator calculated from relative frequencies.

The statistics required to compute the Witten-Bell estimator for the conditional probability of an n-gram consist of the counts of all n-grams of length

1 to n as well as the counts of the number of distinct types following all n-grams of length 1 to $n-1$. In practice we use the $c(w_1,...,w_i) = 1$ event as a proxy for $s(w_1,...,w_i) = 1$ and thereby need not store singleton suffix counts in the filter.

Distinct suffix counts of 2 and above are stored by subtracting this proxy count and converting to the log quantisation scheme described above, i.e.,

$$qs(x) = 1 + \lfloor \log_b(s(x) - 1) \rfloor$$

In testing for a suffix count, we first query the item $c(w_1,...,w_n) = 1$ as a proxy for $s(w_1,...,w_n) = 1$ and, if found, query the filter for incrementally larger suffix counts, taking the reconstructed suffix count of an n-gram with a non-zero n-gram count to be the expected value, i.e.,

$$E[s(x)|qs(x) = j \cap j > 0] = 1 + \frac{(b^{j-1} + b^j - 1)}{2}$$

Having created a BF containing these events, the algorithm we use to compute the interpolated WB estimate makes use of the inequalities described above to reduce the *a priori* probability of querying for a negative. In particular, we bound the count of each numerator in the maximum likelihood term by the count of the corresponding denominator and the count of distinct suffixes of an n-gram by its respective token frequency.

Unlike more traditional LM formulations that back-off from the highest-order to lower-order models, our algorithm works up from the lowest-order model. Since the conditioning context increases in specificity at each level, each statistic is bound from above by its corresponding value at the previous less specific level. The bounds are applied by passing them as the parameter $MAXQCOUNT$ to the frequency test routine shown as Algorithm 2. We analyze the effect of applying such bounds on the performance of the model within an SMT decoder in the experiments below. Working upwards from the lower-order models also allows us to truncate the computation before the highest level if the denominator in the maximum likelihood term is found with a zero count at any stage (no higher-order terms can be non-zero given this).

4 Experiments

We conducted a range of experiments to explore the error-space trade-off of using a BF-based model as a replacement for a conventional n-gram model within an SMT system and to assess the benefits of specific features of our framework for deriving language model probabilities from a BF.

4.1 Experimental set-up

All of our experiments use publically available resources. Our main experiments use the French-English section of the *Europarl* (EP) corpus for parallel data and language modelling (Koehn, 2003). Decoding is carried-out using the Moses decoder (Koehn and Hoang, 2007). We hold out 1,000 test sentences and 500 development sentences from the parallel text for evaluation purposes. The parameters for the feature functions used in this log-linear decoder are optimised using minimum error rate (MER) training on our development set unless otherwise stated. All evaluation is in terms of the BLEU score on our test set (Papineni et al., 2002).

Our baseline language models were created using the SRILM toolkit (Stolcke, 2002). We built 3, 4 and 5-gram models from the Europarl corpus using interpolated Witten-Bell smoothing (WB); no n-grams are dropped from these models or any of the BF-LMs. The number of distinct n-gram types in these baseline models as well as their sizes on disk and as compressed by *gzip* are given in Table 1; the *gzip* figures are given as an approximate (and optimistic) lower bound on lossless representations of these models.[2]

The BF-LM models used in these experiments were all created from the same corpora following the scheme outlined above for storing n-gram statistics. Proxy relations were used to reduce the number of items that must be stored in the BF; in addition, unless specified otherwise, we take advantage of the bounds described above that hold between related statistics to avoid presenting known negatives to the filter. The base of the logarithm used in quantization is specified on all figures.

The SRILM and BF-based models are both queried via the same interface in the Moses decoder.

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

n	Types	Mem.	Gzip'd	BLEU
3	5.9M	174Mb	51Mb	28.54
4	14.1M	477Mb	129Mb	28.99
5	24.2M	924Mb	238Mb	29.07

Table 1: WB-smoothed SRILM baseline models.

We assign a small cache to the BF-LM models (between 1 and 2MBs depending on the order of the model) to store recently retrieved statistics and derived probabilities. Translation takes between 2 to 5 times longer using the BF-LMs as compared to the corresponding SRILM models.

4.2 Machine translation experiments

Our first set of experiments examines the relationship between memory allocated to the BF-LM and translation performance for a 3-gram and a 5-gram WB smoothed BF-LM. In these experiments we use the log-linear weights of the baseline model to avoid variation in translation performance due to differences in the solutions found by MER training: this allows us to focus solely on the quality of each BF-LM's approximation of the baseline. These experiments consider various settings of the base for the logarithm used during quantisation (b in Eq. (1)).

We also analyse these results in terms of the relationships between BLEU score and the underlying error rate of the BF-LM and the number of bits assigned per n-gram in the baseline model.

MER optimised BLEU scores on the test set are then given for a range of BF-LMs.

4.3 Mean squared error experiments

Our second set of experiments focuses on the accuracy with which the BF-LM can reproduce the baseline model's distribution. Unfortunately, perplexity or related information-theoretic quantities are not applicable in this case since the BF-LM is not guaranteed to produce a properly normalised distribution. Instead we evaluate the mean squared error (MSE) between the log-probabilites assigned by the baseline model and by BF-LMs to n-grams in the English portion of our development set; we also consider the relation between MSE and the BLEU score from the experiments above.

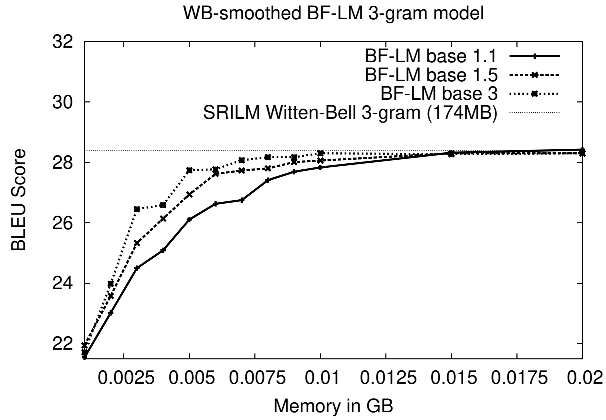

Figure 1: WB-smoothed 3-gram model (Europarl).

4.4 Analysis of BF-LM framework

Our third set of experiments evaluates the impact of the use of upper bounds between related statistics on translation performance. Here the standard model that makes use of these bounds to reduce the *a priori* negative probability is compared to a model that queries the filter in a memoryless fashion.[3]

We then present details of the memory savings obtained by the use of proxy relations for the models used here.

5 Results

5.1 Machine translation experiments

Figures 1 and 2 show the relationship between translation performance as measured by BLEU and the memory assigned to the BF respectively for WB-smoothed 3-gram and 5-gram BF-LMs. There is a clear degradation in translation performance as the memory assigned to the filter is reduced. Models using a higher quantisation base approach their optimal performance faster; this is because these more coarse-grained quantisation schemes store fewer items in the filter and therefore have lower underlying false positive rates for a given amount of memory.

Figure 3 presents these results in terms of the relationship between translation performance and the false positive rate of the underlying BF. We can see that for a given false positive rate, the more coarse-grained quantisation schemes (e.g., base 3) perform

[3]In both cases we apply 'sanity check' bounds to ensure that none of the ratios in the WB formula (Eq. 3) are greater than 1.

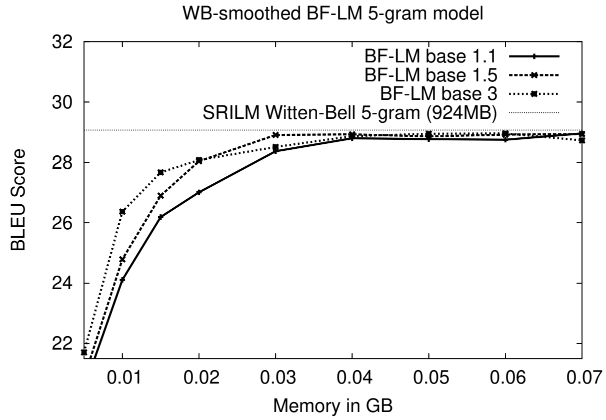

Figure 2: WB-smoothed 5-gram model (Europarl).

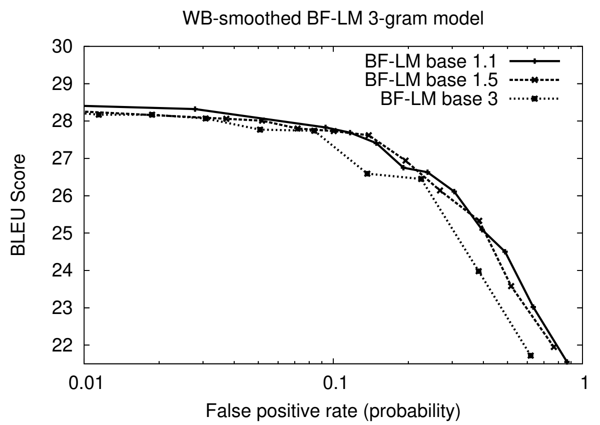

Figure 3: False positive rate vs. BLEU .

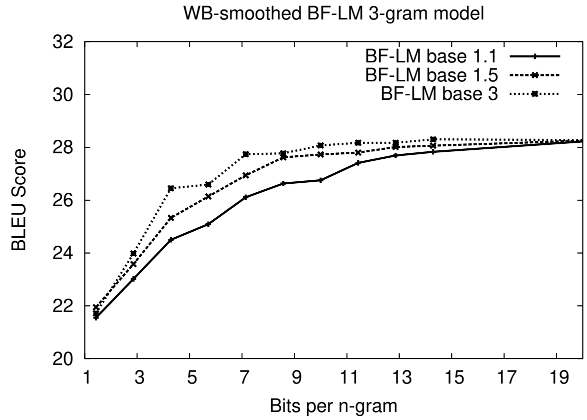

Figure 4: Bits per n-gram vs. BLEU.

n	Memory	Bits / n-gram	base	BLEU
3	10MB	14 bits	1.5	28.33
3	10MB	14 bits	2.0	28.47
4	20MB	12 bits	1.5	28.63
4	20MB	12 bits	2.0	28.63
5	40MB	14 bits	1.5	28.53
5	40MB	14 bits	2.0	28.72
5	50MB	17 bits	1.5	29.31
5	50MB	17 bits	2.0	28.67

Table 2: MERT optimised WB-smoothed BF-LMS.

worse than the more fine-grained schemes.[4]

Figure 4 presents the relationship in terms of the number of bits per n-gram in the baseline model. This suggests that between 10 and 15 bits is sufficient for the BF-LM to approximate the baseline model. This is a reduction of a factor of between 16 and 24 on the plain model and of between 4 and 7 on *gzip* compressed model.

The results of a selection of BF-LM models with decoder weights optimised using MER training are given in Table 2; these show that the models perform consistently close to the baseline models that they approximate.

5.2 Mean squared error experiments

Figure 5 shows the relationship between memory assigned to the BF-LMs and the mean squared error (MSE) of log-probabilities that these models assign to the development set compared to those assigned by the baseline model. This shows clearly that the more fine-grained quantisation scheme (e.g. base 1.1) can reach a lower MSE but also that the more coarse-grained schemes (e.g., base 3) approach their minimum error faster.

Figure 6 shows the relationship between MSE between the BF-LM and the baseline model and BLEU. The MSE appears to be a good predictor of BLEU score across all quantisation schemes. This suggests that it may be a useful tool for optimising BF-LM parameters without the need to run the decoder assuming a target (lossless) LM can be built and queried for a small test set on disk. An MSE of below 0.05 appears necessary to achieve translation performance matching the baseline model here.

5.3 Analysis of BF-LM framework

We refer to (Talbot and Osborne, 2007) for empirical results establishing the performance of the log-frequency BF-LM: overestimation errors occur with

[4]Note that in this case the base 3 scheme will use approximately two-thirds the amount of memory required by the base 1.5 scheme.

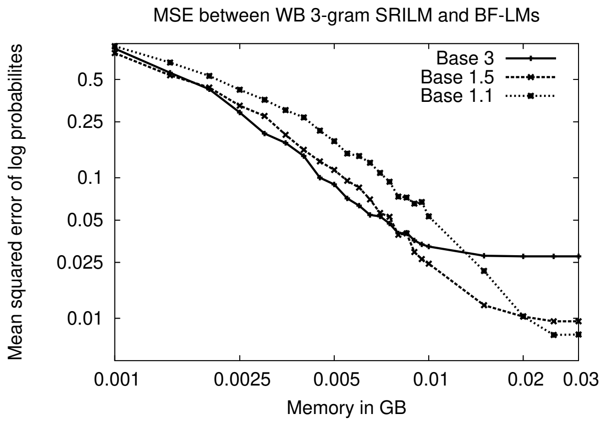

Figure 5: MSE between SRILM and BF-LMs

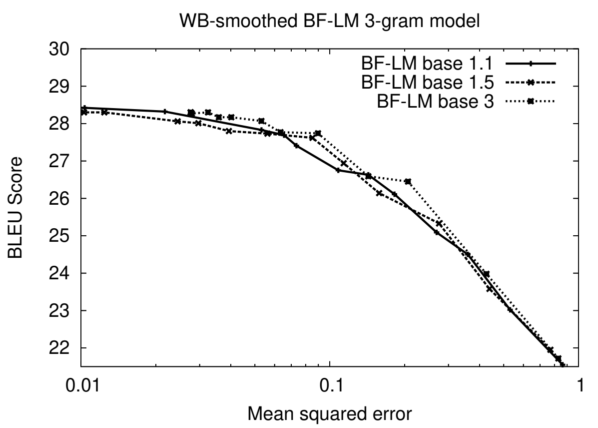

Figure 6: MSE vs. BLEU for WB 3-gram BF-LMs

a probability that decays exponentially in the size of the overestimation error.

Figure 7 shows the effect of applying upper bounds to reduce the *a priori* probability of presenting a negative event to the filter in our interpolation algorithm for computing WB-smoothed probabilities. The application of upper bounds improves translation performance particularly when the amount of memory assigned to the filter is limited. Since both filters have the same underlying false positive rate (they are identical), we can conclude that this improvement in performance is due to a reduction in the number of negatives that are presented to the filter and hence errors.

Table 3 shows the amount of memory saved by the use of proxy items to avoid storing singleton suffix counts for the Witten-Bell smoothing scheme. The savings are given as ratios over the amount of memory needed to store the statistics without proxy items. These models have the same underlying false

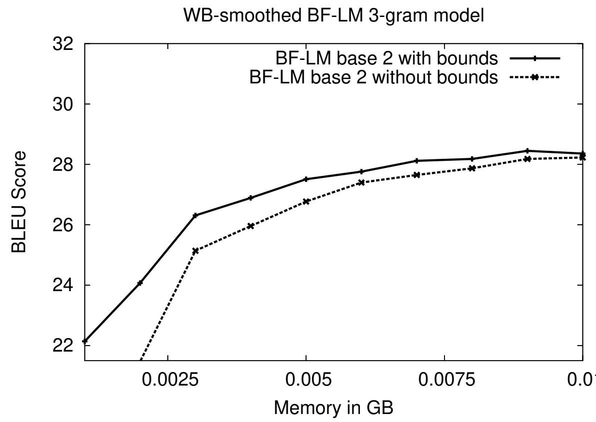

Figure 7: Effect of upper bounds on BLEU

n-gram order	Proxy space saving
3	0.885
4	0.783
5	0.708

Table 3: Space savings via proxy items .

positive rate (0.05) and quantisation base (2). Similar savings may be anticipated when applying this framework to infix and prefix counts for Kneser-Ney smoothing.

6 Related Work

Previous work aimed at reducing the size of n-gram language models has focused primarily on quantisation schemes (Whitaker and Raj, 2001) and pruning (Stolcke, 1998). The impact of the former seems limited given that storage for the n-gram types themselves will generally be far greater than that needed for the actual probabilities of the model. Pruning on the other hand could be used in conjunction with the framework proposed here. This holds also for compression schemes based on clustering such as (Goodman and Gao, 2000). Our approach, however, avoids the significant computational costs involved in the creation of such models.

Other schemes for dealing with large language models include per-sentence filtering of the model or its distribution over a cluster. The former requires time-consuming adaptation of the model for each sentence in the test set while the latter incurs significant overheads for remote calls during decoding. Our framework could, however, be used to complement either of these approaches.

7 Conclusions and Future Work

We have proposed a framework for computing smoothed language model probabilities efficiently from a randomised representation of corpus statistics provided by a Bloom filter. We have demonstrated that models derived within this framework can be used as direct replacements for equivalent conventional language models with significant reductions in memory requirements. Our empirical analysis has also demonstrated that by taking advantage of the one-sided error guarantees of the BF and simple inequalities that hold between related n-gram statistics we are able to further reduce the BF storage requirements and the effective error rate of the derived probabilities.

We are currently implementing Kneser-Ney smoothing within the proposed framework. We hope the present work will, together with Talbot and Osborne (2007), 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.

Acknowledgements

References

T.C. Bell, J.G. Cleary, and I.H. Witten. 1990. *Text Compression*. Prentice Hall, Englewood Cliffs, NJ.

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, June. Association for Computational Linguistics.

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, draft. Available at:http://people.csail.mit.edu/ koehn/publications/europarl.ps.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL-2002: 40th Annual meeting of the Association for Computational Linguistics*.

Andreas Stolcke. 1998. Entropy-based pruning of back-off language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*.

D. Talbot and M. Osborne. 2007. Randomised language modelling for statistical machine translation. In *45th Annual Meeting of the Association of Computational Linguists (To appear)*.

E. Whitaker and B. Raj. 2001. Quantization-based language model compression (tr-2001-41). Technical report, Mitsubishi Electronic Research Laboratories.