Exponential Reservoir Sampling for Streaming Language Models

Miles Osborne[*]
School of Informatics
University of Edinburgh

Ashwin Lall
Mathematics and Computer Science
Denison University

Benjamin Van Durme
HLTCOE
Johns Hopkins University

Abstract

We show how rapidly changing textual streams such as Twitter can be modelled in fixed space. Our approach is based upon a randomised algorithm called *Exponential Reservoir Sampling*, unexplored by this community until now. Using language models over Twitter and Newswire as a testbed, our experimental results based on perplexity support the intuition that recently observed data generally outweighs that seen in the past, but that at times, the past can have valuable signals enabling better modelling of the present.

1 Introduction

Work by Talbot and Osborne (2007), Van Durme and Lall (2009) and Goyal et al. (2009) considered the problem of building very large language models via the use of randomized data structures known as *sketches*.[1] While efficient, these structures still scale linearly in the number of items stored, and do not handle deletions well: if processing an unbounded stream of text, with new words and phrases being regularly added to the model, then with a fixed amount of space, errors will only increase over time. This was pointed out by Levenberg and Osborne (2009), who investigated an alternate approach employing perfect-hashing to allow for deletions over time. Their deletion criterion was task-specific and based on how a machine translation system queried a language model.

Here we ask what the appropriate selection criterion is for streaming data based on a non-stationary process, when concerned with an intrinsic measure such as perplexity. Using Twitter and newswire, we pursue this via a sampling strategy: we construct models over sentences based on a sample of previously observed sentences, then measure perplexity of incoming sentences, all on a day by day, rolling basis. Three sampling approaches are considered: A fixed-width sliding window of most recent content, uniformly at random over the stream and a biased sample that prefers recent history over the past.

We show experimentally that a moving window is better than uniform sampling, and further that exponential (biased) sampling is best of all. For streaming data, recently encountered data is valuable, but there is also signal in the previous stream.

Our sampling methods are based on *reservoir sampling* (Vitter, 1985), a popularly known method in some areas of computer science, but which has seen little use within computational linguistics.[2] Standard reservoir sampling is a method for maintaining a uniform sample over a dynamic stream of elements, using constant space. Novel to this community, we consider a variant owing to Aggarwal (2006) which provides for an exponential bias towards recently observed elements. This *exponential reservoir sampling* has all of the guarantees of standard reservoir sampling, but as we show, is a better fit for streaming textual data. Our approach is fully general and can be applied to any streaming task where we need to model the present and can only use fixed space.

[*]Corresponding author: miles@inf.ed.ac.uk

[1]Sketches provide space efficiencies that are measured on the order of individual bits per item stored, but at the cost of being lossy: sketches trade off space for error, where the less space you use, the more likely you will get erroneous responses to queries.

[2]Exceptions include work by Van Durme and Lall (2011) and Van Durme (2012), aimed at different problems than that explored here.

2 Background

We address two problems: language changes over time, and the observation that space is a problem, even for compact sketches.

Statistical language models often assume either a local Markov property (when working with utterances, or sentences), or that content is generated fully i.i.d. (such as in document-level topic models). However, language shows observable priming effects, sometimes called *triggers*, where the occurrence of a given term decreases the surprisal of some other term later in the same discourse (Lau et al., 1993; Church and Gale, 1995; Beeferman et al., 1997; Church, 2000). Conventional cache and trigger models typically do not deal with new terms and can be seen as adjusting the parameters of a fixed model.

Accounting for previously unseen entries in a language model can be naively simple: as they appear in new training data, add them to the model! However in practice we are constrained by available *space*: how many unique phrases can we store, given the target application environment?

Our work is concerned with modeling language that might change over time, in accordance with current trending discourse topics, but under a strict space constraint. With a fixed amount of memory available, we cannot allow our list of unique words or phrases to grow over time, even while new topics give rise to novel names of people, places, and terms of interest. Thus we need an approach that keeps the size of the model constant, but that is geared to what is being discussed now, as compared to some time in the past.

3 Reservoir Sampling

3.1 Uniform Reservoir Sampling

The reservoir sampling algorithm (Vitter, 1985) is the classic method of sampling without replacement from a stream in a single pass when the length of the stream is of indeterminate or unbounded length. Say that the size of the desired sample is k. The algorithm proceeds by retaining the first k items of the stream and then sampling each subsequent element with probability $f(k, n) = k/n$, where n is the length of the stream so far. (See Algorithm 1.) It is easy to show via induction that, at any time, all the items in the stream so far have equal probability of appearing in the reservoir.

The algorithm processes the stream in a single pass—that is, once it has processed an item in the stream, it does not revisit that item unless it is stored in the reservoir. Given this restriction, the incredible feature of this algorithm is that it is able to guarantee that the samples in the reservoir are a uniformly random sample with no unintended biases even as the stream evolves. This makes it an excellent candidate for situations when the stream is continuously being updated and it is computationally infeasible to store the entire stream or to make more than a single pass over it. Moreover, it is an extremely efficient algorithm as it requires $O(1)$ time (independent of the reservoir size and stream length) for each item in the stream.

Algorithm 1 Reservoir Sampling Algorithm

Parameters:
k: maximum size of reservoir

1: Initialize an empty reservoir (any container data type).
2: $n := 1$
3: **for** each item in the stream **do**
4: **if** $n < k$ **then**
5: insert current item into the reservoir
6: **else**
7: with probability $f(n, k)$, eject an element of the reservoir chosen uniformly at random and insert current item into the reservoir
8: $n := n + 1$

3.2 Non-uniform Reservoir Sampling

Here we will consider generalizations of the reservoir sampling algorithm in which the sample items in the reservoir are more biased towards the present. Put another way, we will continuously decay the probability that an older item will appear in the reservoir. Models produced using such biases put more modelling stress on the present than models produced using data that is selected uniformly from the stream. The goal here is to continuously update the reservoir sample in such a way that the decay of older items is done consistently while still maintaining the benefits of reservoir sampling, including the single pass and memory/time constraints.

The time-decay scheme we will study in this paper is *exponential bias* towards newer items in the stream. More precisely, we wish for items that

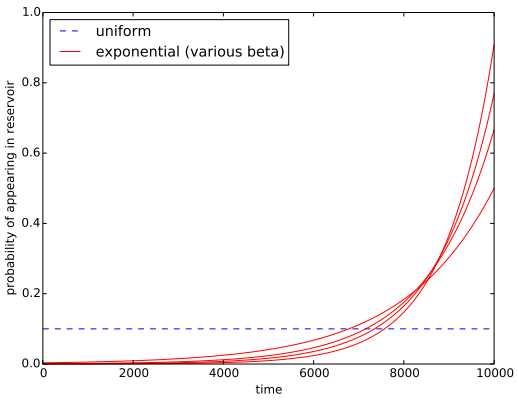

Figure 1: Different biases for sampling a stream

have age a in the stream to appear with probability

$$g(a) = c \cdot \exp(-a/\beta),$$

where a is the age of the item, β is a scale parameter indicating how rapidly older items should be deemphasized, and c is a normalization constant. To give a sense of what these time-decay probabilities look like, some exponential distributions are plotted (along with the uniform distribution) in Figure 1.

Aggarwal (2006) studied this problem and showed that by altering the sampling probability ($f(n, k)$ in Algorithm 1) in the reservoir sampling algorithm, it is possible to achieve different age-related biases in the sample. In particular, he showed that by setting the sampling probability to the constant function $f(n, k) = k/\beta$, it is possible to approximately achieve exponential bias in the sample with scale parameter β (Aggarwal, 2006). Aggarwal's analysis relies on the parameter β being very large. In the next section we will make the analysis more precise by omitting any such assumption.

3.3 Analysis

In this section we will derive an expression for the bias introduced by an arbitrary sampling function f in Algorithm 1. We will then use this expression to derive the precise sampling function needed to achieve exponential decay.[3] Careful selection of f allows us to achieve anything from zero decay (i.e., uniform sampling of the entire stream) to exponential decay. Once again, note that since we are only changing the sampling function, the one-pass, memory- and time-efficient properties of reservoir sampling are still being preserved.

In the following analysis, we fix n to be the size of the stream at some fixed time and k to be the size of the reservoir. We assume that the ith element of the stream is sampled with probability $f(i, k)$, for $i \leq n$. We can then derive the probability that an element of age a will still be in the reservoir as

$$g(a) = f(n-a, k) \prod_{t=n-a+1}^{n} \left(1 - \frac{f(t, k)}{k}\right),$$

since it would have been sampled with probability $f(n-a, k)$ and had independent chances of being replaced at times $t = n-a+1, \ldots, n$ with probability $f(t, k)/k$. For instance, when $f(x, k) = \frac{k}{x}$, the above formula simplifies down to $g(a) = \frac{k}{n}$ (i.e., the uniform sampling case).

For the exponential case, we fix the sampling rate to some constant $f(n, k) = p_k$, and we wish to determine what value to use for p_k to achieve a given exponential decay rate $g(a) = ce^{-a/\beta}$, where c is the normalization constant (to make g a probability distribution) and β is the scale parameter of the exponential distribution. Substituting $f(n, k) = p_k$ in the above formula and equating with the decay rate, we get that $p_k(1 - p_k/k)^a \equiv ce^{-a/\beta}$, which must hold true for all possible values of a. After some algebra, we get that when $f(x, k) = p_k = k(1 - e^{-1/\beta})$, the probability that an item with age a is included in the reservoir is given by the exponential decay rate $g(a) = p_k e^{-a/\beta}$. Note that, for very large values of β, this probability is approximately equal to $p_k \approx k/\beta$ (by using the approximation $e^{-x} \approx 1 - x$, when $|x|$ is close to zero), as given by Aggarwal, but our formula gives the precise sampling probability and works even for smaller values of β.

4 Experiments

Our experiments use two streams of data to illustrate exponential sampling: Twitter and a more conventional newswire stream. The Twitter data is interesting as it is very multilingual, bursty (for example, it talks about memes, breaking news, gossip etc) and written by literally millions of different people. The newswire stream is a lot more well behaved and serves as a control.

4.1 Data, Models and Evaluation

We used one month of chronologically ordered Twitter data and divided it into 31 equal sized

[3] Specifying an arbitrary decay function remains an open problem.

Stream	Interval	Total (toks)	Test (toks)
Twitter	Dec 2013	3282M	105M
Giga	1994 – 2010	635.5M	12M

Table 1: Stream statistics

blocks (roughly corresponding with days). We also used the AFP portion of the Giga Word corpus as another source of data that evolves at a slower pace. This data was divided into 50 equal sized blocks. Table 1 gives statistics about the data. As can be seen, the Twitter data is vastly larger than newswire and arrives at a much faster rate.

We considered the following models. Each one (apart from the exact model) was trained using the same amount of data:

- **Static**. This model was trained using data from the start of the duration and never varied. It is a baseline.
- **Exact**. This model was trained using *all* available data from the start of the stream and acts as an upper bound on performance.
- **Moving Window**. This model used all data in a fixed-sized window immediately before the given test point.
- **Uniform**. Here, we use uniform reservoir sampling to select the data.
- **Exponential**. Lastly, we use exponential reservoir sampling to select the data. This model is parameterised, indicating how strongly biased towards the present the sample will be. The β parameter is a multiplier over the reservoir length. For example, a β value of 1.1 with a sample size of 10 means the value is 11. In general, β always needs to be bigger than the reservoir size.

We sample over whole sentences (or Tweets) and not ngrams.[4] Using ngrams instead would give us a finer-grained control over results, but would come at the expense of greatly complicating the analysis. This is because we would need to reason about not just a set of items but a multiset of items. Note that because the samples are large[5], variations across samples will be small.

[4] A consequence is that we do not guarantee that each sample uses exactly the same number of grams. This can be tackled by randomly removing sampled sentences.

[5] Each day consists of approximately four million Tweets and we evaluate on a whole day.

Day	Uniform	β value			
	∞	1.1	1.3	1.5	2.0
5	619.4	619.4	619.4	619.4	619.4
6	601.0	**601.0**	603.8	606.6	611.1
7	603.0	**599.4**	602.7	605.6	612.1
8	614.6	**607.7**	611.9	614.3	621.6
9	623.3	**611.5**	615.0	620.0	628.1
10	656.2	**643.1**	647.2	650.1	658.0
12	646.6	**628.9**	633.0	636.5	644.6
15	647.7	**628.7**	630.4	634.5	641.6
20	636.7	**605.3**	608.4	610.8	618.4
25	631.5	**601.9**	603.3	604.4	610.0

Table 2: Perplexities for different β values over Twitter (sample size = five days). Lower is better.

We test the model on unseen data from all of the next day (or block). Afterwards, we advance to the next day (block) and repeat, potentially incorporating the previously seen test data into the current training data. Evaluation is in terms of perplexity (which is standard for language modelling).

We used *KenLM* for building models and evaluating them (Heafield, 2011). Each model was an unpruned trigram, with Kneser-Ney smoothing. Increasing the language model order would not change the results. Here the focus is upon which data is used in a model (that is, which data is added and which data is removed) and not upon making it compact or making retraining efficient.

4.2 Varying the β Parameter

Table 2 shows the effect of varying the β parameter (using Twitter). The higher the β value, the more uniform the sampling. As can be seen, performance improves when sampling becomes more biased. Not shown here, but for Twitter, even smaller β values produce better results and for newswire, results degrade. These differences are small and do not affect any conclusions made here. In practise, this value would be set using a development set and to simplify the rest of the paper, all other experiments use the same β value (1.1).

4.3 Varying the Amount of Data

Does the amount of data used in a model affect results? Table 3 shows the results for Twitter when varying the amount of data in the sample and using exponential sampling ($\beta = 1.1$). In parentheses for each result, we show the corresponding moving window results. As expected, using more data improves results. We see that for each sample size, exponential sampling outperforms our moving window. In the limit, all sampling methods would produce the same results.

Day	Sample Size (Days)		
	1	2	3
5	652.5 (661.2)	629.1 (635.8)	624.8 (625.9)
6	635.4 (651.6)	611.6 (620.8)	604.0 (608.7)
7	636.0 (647.3)	611.0 (625.2)	603.7 (612.5)
8	654.8 (672.7)	625.6 (641.6)	614.6 (626.9)
9	653.9 (662.8)	628.3 (643.0)	618.8 (632.2)
10	679.1 (687.8)	654.3 (666.8)	646.6 (659.7)
12	671.1 (681.9)	645.8 (658.6)	633.8 (647.5)
15	677.7 (697.9)	647.4 (668.0)	636.4 (652.6)
20	648.1 (664.6)	621.4 (637.9)	612.2 (627.6)
25	657.5 (687.5)	625.3 (664.4)	613.4 (641.8)

Table 3: Perplexities for different sample sizes over Twitter. Lower is better.

4.4 Alternative Sampling Strategies

Table 4 compares the two baselines against the two forms of reservoir sampling. For Twitter, we see a clear recency effect. The static baseline gets worse and worse as it recedes from the current test point. Uniform sampling does better, but it in turn is beaten by the Moving Window Model. However, this in turn is beaten by our exponential reservoir sampling.

Day	Static	Moving	Uniform	Exp	Exact
5	619.4	619.4	619.4	619.4	619.4
6	664.8	**599.7**	601.8	601.0	597.6
7	684.4	602.8	603.0	**599.3**	595.6
8	710.1	612.0	614.6	**607.7**	603.5
9	727.0	617.9	623.3	**613.0**	608.7
10	775.6	651.2	656.2	**642.0**	640.5
12	776.7	639.0	646.6	**628.7**	627.5
15	777.1	638.3	647.7	**626.7**	627.3
20	800.9	619.1	636.7	**604.9**	607.3
25	801.4	621.7	631.5	**601.5**	597.6

Table 4: Perplexities for differently selected samples over Twitter (sample size = five days, $\beta = 1.1$). Results in **bold** are the best sampling results. Lower is better.

4.5 GigaWord

Twitter is a fast moving, rapidly changing multilingual stream and it is not surprising that our exponential reservoir sampling proves beneficial. Is it still useful for a more conventional stream that is drawn from a much smaller population of reporters? We repeated our experiments, using the same rolling training and testing evaluation as before, but this time using newswire for data.

Table 5 shows the perplexities when using the Gigaword stream. We see the same general trends, albeit with less of a difference between exponential sampling and our moving window. Perplexity values are all lower than for Twitter.

Block	Static	Moving	Uniform	Exp
11	416.5	**381.1**	382.0	382.0
15	436.7	353.3	357.5	**352.8**
20	461.8	347.0	354.4	**344.6**
25	315.6	214.9	222.2	**211.3**
30	319.1	200.5	213.5	**199.5**
40	462.5	304.4	313.2	**292.9**

Table 5: Perplexities for differently selected samples over Gigaword (sample size = 10 blocks, $\beta = 1.1$). Lower is better.

4.6 Why does this work for Twitter?

Although the perplexity results demonstrate that exponential sampling is on average beneficial, it is useful to analyse the results in more detail. For a large stream size (25 days), we built models using uniform, exponential ($\beta = 1.1$) and our moving window sampling methods. Each approach used the same amount of data. For the same test set (four million Tweets), we computed per-Tweet log likelihoods and looked at the difference between the model that best explained each tweet and the second best model (ie the margin). This gives us an indication of how much a given model better explains a given Tweet. Analysing the results, we found that most gains came from short grams and very few came from entire Tweets being reposted (or retweeted). This suggests that the Twitter results follow previously reported observations on how language can be bursty and not from Twitter-specific properties.

5 Conclusion

We have introduced exponential reservoir sampling as an elegant way to model a stream of unbounded size, yet using fixed space. It naturally allows one to take account of recency effects present in many natural streams. We expect that our language model could improve other Social Media tasks, for example lexical normalisation (Han and Baldwin, 2011) or even event detection (Lin et al., 2011). The approach is fully general and not just limited to language modelling. Future work should look at other distributions for sampling and consider tasks such as machine translation over Social Media.

Acknowledgments This work was carried out when MO was on sabbatical at the HLTCOE and CLSP.

References

Charu C Aggarwal. 2006. On biased reservoir sampling in the presence of stream evolution. In *Proceedings of the 32nd international conference on Very large data bases*, pages 607–618. VLDB Endowment.

Doug Beeferman, Adam Berger, and John Lafferty. 1997. A model of lexical attractions and repulsion. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 373–380. Association for Computational Linguistics.

K. Church and W. A. Gale. 1995. Poisson mixtures. *Natural Language Engineering*, 1:163–190.

Kenneth W Church. 2000. Empirical estimates of adaptation: the chance of two noriegas is closer to p/2 than p 2. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 180–186. Association for Computational Linguistics.

Amit Goyal, Hal Daumé III, and Suresh Venkatasubramanian. 2009. Streaming for large scale NLP: Language Modeling. In *Proceedings of NAACL*.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 368–378, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Raymond Lau, Ronald Rosenfeld, and SaIim Roukos. 1993. Trigger-based language models: A maximum entropy approach. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 45–48. IEEE.

Abby Levenberg and Miles Osborne. 2009. Stream-based randomised language models for smt. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 756–764. Association for Computational Linguistics.

Jimmy Lin, Rion Snow, and William Morgan. 2011. Smoothing techniques for adaptive online language models: topic tracking in tweet streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 422–429. ACM.

David Talbot and Miles Osborne. 2007. Randomised language modelling for statistical machine translation. In *Proceedings of ACL*.

Benjamin Van Durme and Ashwin Lall. 2009. Probabilistic Counting with Randomized Storage. In *Proceedings of IJCAI*.

Benjamin Van Durme and Ashwin Lall. 2011. Efficient online locality sensitive hashing via reservoir counting. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 18–23. Association for Computational Linguistics.

Benjamin Van Durme. 2012. Streaming analysis of discourse participants. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 48–58. Association for Computational Linguistics.

Jeffrey S. Vitter. 1985. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11:37–57, March.