Streaming Analysis of Discourse Participants

Benjamin Van Durme
Human Language Technology Center of Excellence
Johns Hopkins University

Abstract

Inferring attributes of discourse participants has been treated as a *batch*-processing task: data such as all tweets from a given author are gathered in bulk, processed, analyzed for a particular feature, then reported as a result of academic interest. Given the sources and scale of material used in these efforts, along with potential use cases of such analytic tools, discourse analysis should be reconsidered as a *streaming* challenge. We show that under certain common formulations, the batch-processing analytic framework can be decomposed into a sequential series of updates, using as an example the task of gender classification. Once in a streaming framework, and motivated by large data sets generated by social media services, we present novel results in approximate counting, showing its applicability to space efficient streaming classification.

1 Introduction

The rapid growth in social media has led to an equally rapid growth in the desire to mine it for useful information: the content of public discussions, such as found in tweets, or in posts to online forums, can support a variety of data mining tasks. Inferring the underlying properties of those that engage with these platforms, the *discourse participants*, has become an active topic of research: predicting individual attributes such as age, gender, and political preferences (Rao et al., 2010), or relationships *between* communicants, such as organizational dominance (Diehl et al., 2007). This research can benefit areas such as: (A) commercial applications, e.g., improved models for advertising placement, or detecting fraudulent or otherwise unhelpful product reviews (Jindal and Liu, 2008; Ott et al., 2011); and (B) in enhanced models of civic discourse, e.g., inexpensive, large-scale, *passive polling* of popular opinion (O'Connor et al., 2010).

Classification with streaming data has usually been taken in the computational linguistics community to mean individual decisions made on items that are presented over time. For example: assigning a label to each newly posted product review as to whether it contains positive or negative sentiment, or whether the latest tweet signals a novel topic that should be tagged for tracking (Petrovic et al., 2010).

Here we consider a distinct form of stream-based classification: we wish to assign, then *dynamically update*, labels on discourse participants based on their associated streaming communications. For instance, rather than classifying individual reviews as to their sentiment polarity, we might wish to classify the underlying author as to whether they are genuine or paid-advertising, and then update that decision as they continue to post new reviews. As the scale of social media continues to grow, we desire that our model be aggressively space efficient, which precludes a naive solution of storing the full communication history for all users.

In this paper we make two contributions: (1) we make explicit that a standard bag-of-words classification model for predicting latent author attributes can be simply decomposed into a series of streaming updates; then (2) show how the randomized algorithm, *Reservoir Counting* (Van Durme and Lall, 2011), can be extended to maintain approximate av-

erages, allowing for significant space savings in our classification model. Our running example task is gender prediction, based on spoken communication and microblogs/Twitter feeds.

2 Model

Assume that each discourse participant (e.g., speaker, author) a has an associated stream of communications (e.g., tweets, utterances, emails, etc.): $(c_i) = C$. Then let $C_t = (c_1, ..., c_t)$ represent the first t elements of C.

Assume access to a pretrained classifier Φ:[1]

$$\Phi(a) = \begin{cases} \texttt{1} & \text{if } w \cdot f(C) \geq 0, \\ \texttt{0} & \text{otherwise,} \end{cases}$$

which we initially take to be linear: author labels are determined by computing the sign of the dot product between a weight vector w, and feature vector $f(C)$, each of dimension d. Note that $f(C)$ is a feature vector over the entire set of communications from a given author.

For example, Φ might be trained to classify author gender:

$$\texttt{Gender}(a) = \begin{cases} \texttt{Male} & \text{if } w \cdot f(C) \geq 0, \\ \texttt{Female} & \text{otherwise.} \end{cases}$$

We now make explicit how under certain common restrictions on the feature space, the classification decision can be decomposed into a series of decision *updates* over the elements of C.

Define $\hat{f}(c_i)$ to be the vector containing the local, count-based feature values of communication c_j.[2] For convenience let us assume that $\hat{f}(c_i) \in \mathbb{N}^d$. Where $|v|_1 = \sum_i |v_i|$ is the L1-norm of vector v, let z_t be the *normalizing constant* at t:

$$z_t = \sum_{i=1}^{t} |\hat{f}(c_i)|_1$$

Now define $f_j(C)$, the j-th entry of $f(C)$, as:

$$f_j(C) = \frac{\sum_{i=1}^{n} \hat{f}_j(c_i)}{z_n}$$

Thus $f(C)$ represents the global relative frequency of each local, count-based feature. This allows us to rearrange terms:

$$\begin{aligned} w \cdot f(C) &= \sum_{j=1}^{d} w_j \, f_j(C) \\ &= \frac{1}{z_n} \sum_{j=1}^{d} w_j (\sum_{i=1}^{n} \hat{f}_j(c_i)) \\ &= \frac{1}{z_n} \sum_{i=1}^{n} (\sum_{j=1}^{d} w_j \, \hat{f}_j(c_i)) \end{aligned}$$

Let (s_t, z_t) be the current *state* of the classifier:

$$(s_t, z_t) \doteq (\sum_{i=1}^{t} \sum_{j=1}^{d} w_j \, \hat{f}_j(c_j), z_t)$$

which pairs the observed *rolling sum*, s_t with the feature stream length z_t.

The classifier decision after seeing everything up to and including communication c_t is thus a simple average:

$$\Phi_t(a) = \begin{cases} 1 & \text{if } \frac{s_t}{z_t} \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Finally we reach the observation that:

$$\begin{aligned} s_t &= s_{t-1} + w \cdot \hat{f}(c_t) \\ z_t &= z_{t-1} + |\hat{f}(c_t)|_1 \end{aligned}$$

which means that from an engineering standpoint we can process a stream of communication one element at a time, without the need to preserve the history explicitly. That is: for each author, for each attribute being analyzed, an online system only need maintain a state pair (s_t, z_t) by extracting and weighting features locally for each new communication. Beyond the computational savings of not needing to store communications nor explicit feature vectors in memory, there are potential privacy benefits as well: analytic systems need not have a lasting record of discourse, they can instead glean whatever signal is required locally in the stream, and then discard the actual communications.

Log-linear Rather than a strictly linear Φ, such as instantiated via perceptron or SVM with linear kernel, many prefer log-linear models as their classification framework:

$$\Phi(a) = \begin{cases} \texttt{1} & \text{if } \frac{1}{1+\exp(-w \cdot f(C))} \geq 0.5, \\ \texttt{0} & \text{otherwise.} \end{cases}$$

[1] While here we assume binary decision tasks, dynamic classification in a multiclass, or regression, setting is an interesting avenue of exploration, for which these definitions generalize.

[2] As seen later in Table 1, we have in mind features such as the frequency of the n-gram *my wife*.

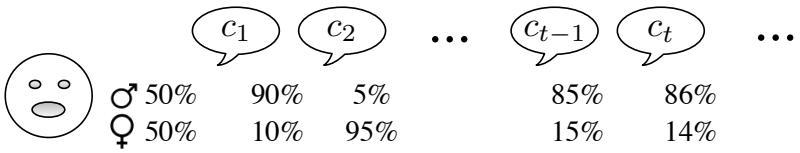

Figure 1: A streaming analytic model should update its decision with each new communication, becoming more stable in its prediction as evidence is acquired.

In either setting, the state of the classifier is sufficiently captured by the pair (s_t, z_t), under the restrictions on f.[3]

2.1 Validation

As an example of a model decomposed into a stream, we revist the task of gender classification based on speech transcripts, as explored by Boulis and Ostendorf (2005) and later Garera and Yarowsky (2009). In the original problem definition, one would collect all transcribed utterances from a given speaker in a corpus such as Fisher (Cieri et al., 2004) or Switchboard (Godfrey et al., 1992), known as a *side* of the conversation. Then by collapsing these utterances into a single document, one could classify it as to whether it was generated by a male or female. Here we define the task as: starting from scratch, report the classifier probability of the speaker being male, as each utterance is presented.

Intuitively we would expect that as more utterances are observed, the better our classification accuracy. Researchers such as Burger et al. (2011) have considered this point, but by comparing the classification accuracy based on the volume of batch data available per author (in that case, tweets): the more prolific the author had been, the better able they were to correctly classify their gender. We confirm here this can be reframed: as a speaker (author) continues to emit a stream of communication, a dynamic model tends to improve its online prediction.

Our collection based on Switchboard consisted of 520 unique speakers (240 female, 280 male), with a total of roughly 400k utterances. Similar to Boulis and Ostendorf, we extracted unigram and bigram counts as features, but without further

[3]Note that some non-linear kernels can be maintained online in a similar fashion. For instance, a polynomial kernel of degree p decomposes as: $(f(C_n) \cdot w)^p = (\frac{s_n}{z_n})^p$.

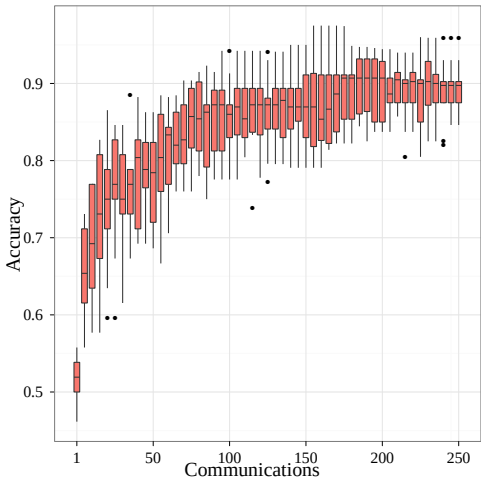

Figure 2: Accuracy on Switchboard gender classification, reported at every fifth utterance, using a dynamic log-linear model with 10-fold cross validation.

TFIDF reweighting. Ngrams were required to occur at least 10 times in the training set, recomputed for each split of 10-fold cross validation. Weights were computed under a log-linear model using `LibLinear` (Fan et al., 2008), with 5% of training held out for tuning an L2 regularizing term. Feature extraction and dynamic aspects were handled through additions to the `Jerboa` package (Van Durme, 2012). Similar to previous work, we found intuitive features such as *my husband* to be weighted heavily (see Table 1), along with certain non-lexical vocalizations such as transcribed laughter.

Table 1: Top ten features by gender.

Male	a, wife, is, my wife, right, of, the, uh, actually, [vocalized-noise]
Female	have, and, [laughter], my husband, really, husband, children, are, would

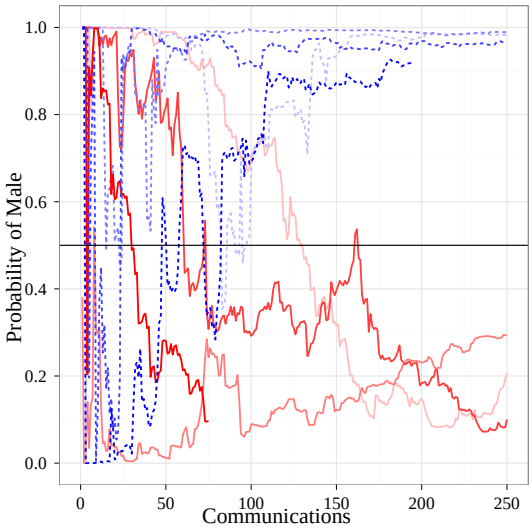

Figure 3: Streaming analysis of eight randomly sampled speakers, four per gender (red-solid: female, blue-dashed: male). Being a log-linear model, the decision boundary is marked at $y = 0.5$.

As seen in Figure 2, accuracy indeed improves as more content is emitted. Figure 3 highlights the streaming perspective: individual speakers can be viewed as distinct trajectories through $[0, 1]$, based on features of their utterances.

3 Randomized Model

Now situated within a streaming context we exact space savings through approximation, extending the approach of Van Durme and Lall (2011), there concerned with online Locality Sensitive Hashing, here initially concerned with taking averages.

When calculating the average over a sequence of values, $X_n = (x_1, ..., x_n)$, we divide the sum of the sequence, $\text{sum}(X_n) = \sum_{i=1}^{n} x_i$, by its length, $\text{length}(X_n) = |X_n|$:

$$\text{avg}(X_n) = \frac{\text{sum}(X_n)}{\text{length}(X_n)}$$

Our goal in this section is to maintain a space efficient approximation of $\text{avg}(X_t)$, as t increases, by using a bit-saving approximation of both the sum, and the length of the sequence.

We begin by reviewing the method of Reservoir Counting, then extend it to a new notion we refer to as Reservoir Averaging. This will allow in the subsequent section to map our analytic model to a form explicitly amenable to keeping an online average.

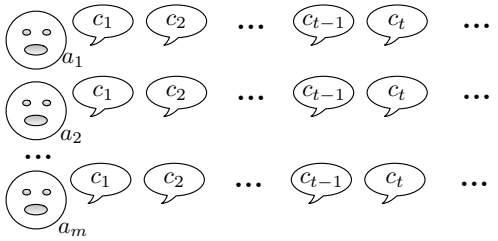

Figure 4: Social media platforms such as Facebook or Twitter deal with a very large number of individuals, each with a variety of implicit attributes (such as gender). This motivates a desire for online space efficiency.

3.1 Reservoir Counting

Reservoir Counting plays on the folklore algorithm of reservoir sampling, first described in the literature by Vitter (1985). As applied to a stream of arbitrary elements, reservoir sampling maintains a list (reservoir) of length k, where the contents of the reservoir represents a uniform random sample over all elements $1...t$ observed thus far in the stream.

When the stream is a sequence of positive and negative integers, reservoir counting implicitly views each value as being unrolled into a sequence made up of either 1 or -1. For instance, the sequence: (3, **-2**, 1) would be viewed as:

$$(1, 1, 1, \textbf{-1}, \textbf{-1}, 1)$$

Since there are only two distinct values in this stream, the contents of the reservoir can be characterized by knowing the fixed value k, and then s: how many elements in the reservoir are 1.[4] This led to Van Durme and Lall defining a method, `ReservoirUpdate`, here abbreviated to `ResUp`, that allows for maintaining an approximate sum, defined as $t(\frac{2s}{k} - 1)$, through updating these two parameters t and s with each newly observed element. Expected accuracy of the approximation varies with the size of the sample, k. Reservoir Counting exploits the fact that the reservoir need only be considered implicitly, where s represented as a b-bit unsigned integer can be used to characterize a reservoir of size $k = 2^b - 1$. This allowed those authors to show a 50% space reduction in the task of online

[4] As the number of -1 values is simply: $k - s$.

Locality Sensitive Hashing, at similar levels of accuracy, by replacing explicit 32-bit counting variables with approximate counters of smaller size. See (Van Durme and Lall, 2011) for further details.

3.2 Reservoir Averaging

For a given integer x, let $m = |x|$ be the magnitude of x, and $\sigma = \text{sign}(x)$. For a given sequence, let m^* be the largest such value of m.

Modifying the earlier implicit construction, consider the sequence (3, **-2**, 1), with $m^* = 3$, mapped to the sequence:

(1, 1, 1, 1, 1, 1, **-1**, **-1**, **-1**, **-1**, **-1**, **1**, 1, 1, 1, 1, -1, -1)

where each value x is replaced with $m^* + m$ elements of σ, and $m^* - m$ elements of $-\sigma$. This views x as a sequence of length $2m^*$, made up of 1s and -1s, where each x in the discrete range $[-m^*, m^*]$ has a unique number of 1s.

Now recognize that the average over the original sequence, here $\frac{3-2+1}{3} = \frac{2}{3}$, is proportional to the average over the implicit sequence, $\frac{1+1+...-1-1}{18} = \frac{4}{18} = \frac{2}{3}(\frac{1}{m^*})$.

Generally for a sequence $(x_1, ..., x_n)$, with m^* as defined, the average times $\frac{1}{m^*}$ is equal to:

$$\begin{aligned}\frac{\sum_{i=1}^{n} x_i}{n}(\frac{1}{m^*}) &= \frac{1}{n2m^*}\sum_{i=1}^{n}(\sum_{l=1}^{m^*+m_i}\sigma_i + \sum_{l=1}^{m^*-m_i}-\sigma_i) \\ &= \frac{\sum_{i=1}^{n} m_i\sigma}{nm^*}\end{aligned}$$

where $n2m^*$ is the total number of 1s and -1s observed in the implicit stream, up to and including the mapping of element x_n. If applying Reservoir Counting, s would then record the sampled number of 1s, as per norm, where t maintained as the implicit stream length can also be viewed as storing $t = n2m^*$. At any point in the stream, the average over the original value sequence can then be approximated as: (1) the approximate sum of the implicit stream; divided by (2) the implicit stream length; times (3) m^* to cancel the $\frac{1}{m^*}$ term:

$$(t(\frac{2s}{k} - 1))_1(\frac{1}{t})_2(m^*)_3 = (\frac{2s}{k} - 1)m^*$$

Granularity As defined this scheme operates on streams of integers. We extend the definition to work with a stream of fixed precision floating point variables. Let g be a positive integer that we refer to as the *granularity*. Modify the mapping of value x from a sequence of length $2m^*$, to a sequence of length g, comprised of $\frac{m^*+m}{2m^*}g$ instances of σ, and $(1-\frac{m^*-m}{2m^*})g$ instances of -σ. As seen in line 4 of Algorithm 1, a random coin flip determines placement of the remainder.

For example, the value 1.3, with $m^* = 3$, and $g = 10$, would now be represented as a sequence of $\frac{3+1.3}{6}g = 7.16 \in (7, 8)$ instances of 1, followed by however many instances of -1 that lead to a sequence of length g, after probabilistic rounding. The possible sequences are thus:

(1, 1, 1, 1, 1, 1, 1, -1, -1, -1)
(1, 1, 1, 1, 1, 1, 1, 1, -1, -1)

with the former more likely.

At this point we have described the framework captured by Algorithm 1, where Van Durme and Lall (2011) defined `ResUp`.

Algorithm 1 UPDATEAVERAGE($n, k, m, m^*, \sigma, g, s$)

Parameters:
n : size of stream
k : size of reservoir, also maximum value of s
m : magnitude of update
m^* : maximum magnitude of all updates
σ : sign of update
g : granularity
s : current value of reservoir

1: **if** $m = 0$ or $\sigma = 0$ **then**
2: Return without doing anything
3: $v := \frac{m+m^*}{2m^*}g$
4: $v := \lceil v \rceil$ with probability $v - \lfloor v \rfloor$, $\lfloor v \rfloor$ otherwise
5: $s' := \texttt{ResUp}(ng, k, v, \sigma, s)$
6: $s' := \texttt{ResUp}((ng + v), k, g - v, -\sigma, s')$
7: Return s'

Log-scale Counting For additional space savings we might approximate the length parameter t with a small bit representation, using the approximate counting scheme of Morris (1978). The method enables counting in log-scale by probabilistically incrementing a counter, where it becomes less and less likely to update the counter after each increment. This scheme is popularly known and used in a variety of contexts, recently in the community by Talbot (2009) and Van Durme and Lall (2009)

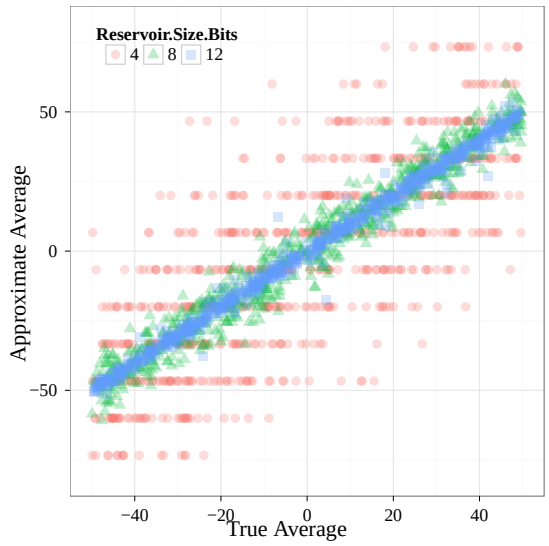

Figure 5: Results on averaging randomly generated sequences, with $m^* = 100, g = 100$, and using an 8 bit Morris-style counter of base 2. Larger reservoir sizes lead to better approximation, at higher cost in bits.

to provide a streaming extension to the Bloom-filter based count-storage mechanism of Talbot and Osborne (2007a) and Talbot and Osborne (2007b). See (Flajolet, 1985) for a detailed analysis of Morris-style counting.

3.3 Experiment

We show through experimentation on synthetic data that this approach gives reasonable levels of accuracy at space efficient sizes of the length and sum parameter. Random sequences of 1,000 values were generated by: (1) fix a value for m^*; (2) draw a polarity bias term μ uniformly from the range [0,1]; then (3) for each value, x: (a) σ was positive with probability μ; (b) m was drawn from [0, m^*]. Figure 5 shows results for varying reservoir sizes (using 4, 8 or 12 bits) when $g = 100$, $m^* = 100$, and the length parameter was represented with an 8 bit Morris-style counter of base 2.

3.4 Justification

Before we close this section, one might ask why this extension is needed in the first place. As Reservoir Counting already allows for keeping an online sum, and pairs it with a length parameter, then this would presumably be what is needed to get the average we are focussed on. Unfortunately that is not the case: the parameter recording the current stream length, here called t, tracks the length of the implicit stream of 1s and -1s, it does not track the length of the original stream of values that gave rise to the mapped version. As an example, consider again the sequence: (3, -2, 1), as compared to: (2,1,-1,-1,1). Both have the same sum, and would therefore be viewed the same under the pre-existing Reservoir Counting algorithm, giving rise to implicit streams of the same length. But critically the sequences have different averages: $\frac{2}{3} \neq \frac{2}{5}$, which we cannot detect based on the original counting algorithm.

Finally, we restate the constraint: for the sequence to averaged, one must know m^* ahead of time.

4 Application to Classification

Going back to our streaming analysis model, we have a situation that can be viewed as a sequence of values, such that we do know m^*. First reinterpret the fraction $\frac{s_t}{z_t}$ equivalently as the normalized sum of a stream of elements sampled from w:

$$\frac{s_t}{z_t} = \frac{1}{z_t} \sum_{i=1}^{t} \sum_{j=1}^{d} \sum_{l=1}^{\hat{f}_j(c_i)} w_j$$

The value m^* is then: $m^* = \max_j |w_j|$, over a sequence of length z_t. Rather than updating s_t and z_t through basic addition, we can now use a smaller bit-wise representation for each variable, and update via Reservoir Averaging.

4.1 Problems in Practice

Reconsidering the earlier classification experiment, we found this approximation method led to terrible results: while our experiments on synthetic data worked well, those sequences were sampled somewhat uniformly over the range of possible values. As seen in Figure 6, sequences arising from observed feature weights in a practical setting may not be so broadly distributed. In brief: the more the maximum possible update, m^*, can be viewed as an outlier, then the more the resulting implicit encoding of g elements per observed weight becomes dominated by "filler". As few observed elements will in that case require the provided range, then the implicit representation will be a mostly balanced set of

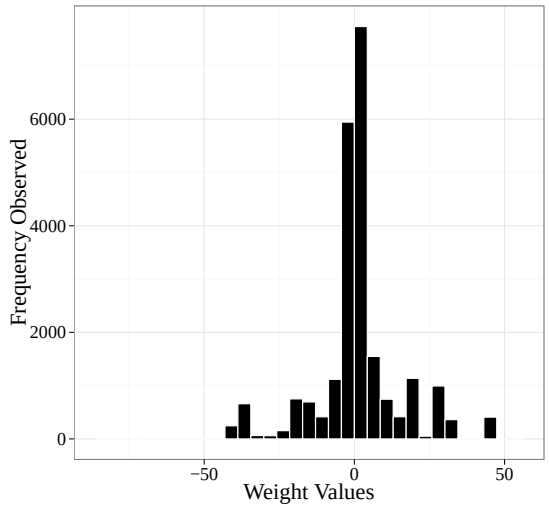

Figure 6: Frequency of individual feature weights observed over a full set of communications by a single example speaker. Most observed features have relatively small magnitude weight. The mean value is 1.3, with $\frac{1}{1+e^{-1.3}} = 0.79 > 0.5$, which properly classifies the speaker as MALE.

1 and -1 values. These mostly balanced encodings make it difficult to maintain an adequate approximation of the true average, when reliant on a small, implicit uniform sample. Here we leave further analysis aside, focusing instead on a modified solution for the classification model under consideration.

4.2 Rewriting History

Practically we would like to restrict our range to the dense region of weight updates, while at the same time not throwing away or truncating larger weights that appear outside a reduced window. We do this by fitting a replacement to m^*: $m' \leq m^*$, based on the classifier's training data, such that too-large elements will be *accommodated* into the stream by implicitly assuming that the portion of a value that falls outside the restricted window is "spread out" over the previously observed values. That is, we modify the contents of the implicit reservoir by *rewriting history*: pretending that earlier elements were larger than they were, but still within the reduced window. As long as we don't see too many values that are overly large, then there will be room to accommodate the overflow without any theoretical damage to the implicit stream: all count mass may still be accounted for. If a moderately high number of overly large elements are observed, then we expect in practice for this to have a negligible impact on downstream performance. If an exceptional number of elements are overly large, then the training data was not representative of the test set.

The newly introduced parameter m' is used in MODIFIEDUPDATEAVERAGE (MUA), which relies on REWRITEHISTORY. Note that MUA uses the same value of n when calling REWRITEHISTORY as it does in the subsequent line calling UPDATEAVERAGE: we modify the state of the reservoir without incrementing the stream length, taking the current overflow and pretending we saw it earlier, spread out across previous elements. This happens by first estimating the number of 1 values seen thus far in the stream: $\frac{s}{k}n$, then adding in twice the overflow value, which represents removing o instances of $-\sigma$ from the stream, and then adding o instances of σ. We probabilistically round the resultant fraction to achieve a modified version of s, which is returned.

Algorithm 2 MUA$(n, k, m, m', \sigma, g, s)$

```
if m < m' then
    Return UPDATEAVERAGE(n, k, m, m', σ, g, s)
s' := REWRITEHISTORY(n, k, m, m', σ, g, s)
Return UPDATEAVERAGE(n, k, m', m', σ, g, s')
```

Algorithm 3 REWRITEHISTORY$(n, k, m, m', \sigma, g, s)$

Parameters:
o : overflow to be accommodated

```
o := ((m − m')/(2m')) g
if σ > 0 then
    if s = k then
        Return s
    p := min(1.0, s/k + 2o/n)
else
    if s = 0 then
        Return s
    p := max(0.0, s/k − 2o/n)
Return ⌈pk⌉ with prob. pk − ⌊pk⌋, ⌊pk⌋ otherwise
```

4.3 Experiment

Figure 7 compares the results seen in Figure 2 to a version of the experiment when using approximation. Parameters were: $g = 100$; $k = 255$; and a Morris-style counter for stream length using 8 bits

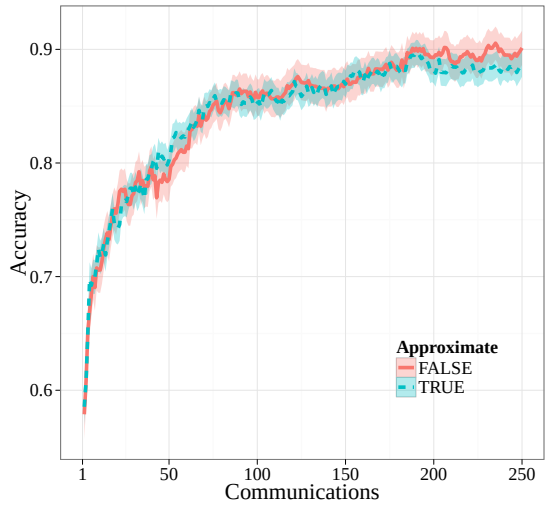

Figure 7: Comparison between using explicit counting and approximation on the Switchboard dataset, with bands reflecting 95% confidence.

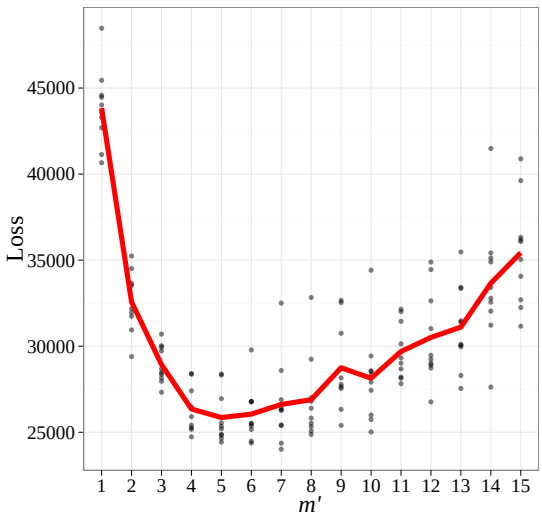

Figure 8: Summed 0/1 loss over all utterances by each speaker in the Switchboard training set, across 10 splits. A value of $m' = 5$ was on average that which minimized the number of mistakes made.

and a base of 1.3. The value m' was fit independently for each split of 10-fold cross validation, by finding through simple line search that which minimized the number of prediction errors on the original training data (see Figure 8). This result shows our ability to replace 2 variables of 32 bits (sum and length) with 2 approximation variables of 8 bits (reservoir status s, and stream length n), leading to a 75% reduction in the cost of maintaining online classifier state, with no significant cost in accuracy.

5 Real World Stream: Twitter

5.1 Setup

Based on the tweet IDs from the data used by Burger et al. (2011), we recovered 2,958,107 of their roughly 4 million original tweets.[5] These tweets were then matched against the gender labels established in that prior work. As reported by Burger et al., the dominant language in the collection is English (66.7% reported), followed by Portuguese (14.4%) then Spanish (6.0%), with a large variety of other languages with small numbers of examples.

Content was lowercased, then processed by regular expression to collapse the following into respective single symbols: emoticons; URLs; usernames (@mentions); and hashtags. Any content deemed to be a retweet (following the characters *RT*) was removed. Text was then tokenized according to a modified version of the Penn TreeBank tokenization standard[6] that was less English-centric.

5.2 Experiment

A log-linear classifier was built using all those authors in the training set[7] with at least 10 tweets. Similar to the previous experiment, unigrams and bigrams features were used exclusively, with the parameter m' fit on the training data.

As seen in Figure 9, results were as in Switchboard: accuracy improves as more data streams in per author, and our approximate model sacrifices perhaps a point of accuracy in return for a 75% reduction in memory requirements per author.

Table 2 gives the top features per gender. We see similarities to Switchboard in terms such as *my*

[5] Standard practice in Twitter data exchanges is to share only the unique tweet identifications and then requery the content from Twitter, thus allowing, e.g., the individual authors the ability to delete previous posts and have that reflected in future data collects. While respectful of author privacy, it does pose a challenge for scientific reproducibility.

[6] Such as codified in `http://www.cis.upenn.edu/~treebank/tokenizer.sed`

[7] The same training, development and test set partitions were used as by Burger et al. (2011), minus those tweets we were unable to retrieve (as previously discussed).

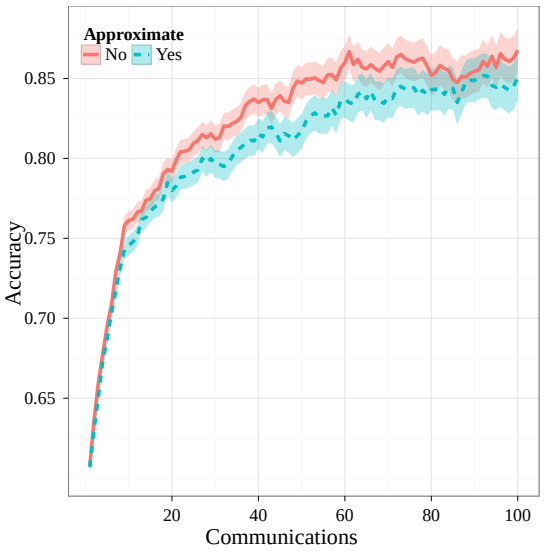

Figure 9: Comparison between using explicit counting and approximation, on the Twitter dataset, with bands reflecting 95% confidence.

wife, along with terms suggesting a more youthful population. Foreign terms are recognized by their parenthetical translation and 1st- or 2nd-person + Male/Female gender marking. For example, the Portuguese *obrigado* can be taken to be literally saying: *I'm obliged (thank you), and I'm male*.

6 Related Work

Streaming algorithms have been developed within the applied communities of networking, and (very large) databases, as well as being a popular topic in the theoretical computer science literature. A summary of the streaming algorithms community is beyond the scope of this work: interested readers are directed to Muthukrishnan (2005) as a starting point.

Table 2: Top thirty-five features by gender in Twitter.

Male	obrigado (*thank you* [1M]), wife, my wife, bro, cansado (*tired* [1M]), gay, mate, dude, [@username] why, buddy, windows, album, dope, beer, [@username] yo, sir, ps3, comics, galera (*folks/people*), amigo (*friend* [2M]), man !, fuckin, omg omg, cheers, ai n't
Female	obrigada (*thank you* [1F]), hubby, husband, cute, my husband, ?, cansada (*tired* [1F]), hair, dress, soooo, lovely, etsy, boyfriend, jonas, loved, book, sooo, girl, sé (*I*), lindo (*cute*), shopping, amiga (*friend* [2F]), yummy, ppl, cupcakes

Within computational linguistics interest in streaming approaches is a more recent development; we provide here examples of representative work, beyond those described in previous sections. Levenberg and Osborne (2009) gave a streaming variant of the earlier perfect hashing language model of Talbot and Brants (2008), which operated in batch-mode. Using a similar decomposition to that here, Van Durme and Lall (2010) showed that Locality Sensitive Hash (LSH) signatures (Indyk and Motwani, 1998; Charikar, 2002) built using count-based feature vectors can be maintained online, as compared to their earlier uses in the community (Ravichandran et al., 2005; Bhagat and Ravichandran, 2008). Finally, Goyal et al. (2009) applied the frequent items[8] algorithm of Manku and Motwani (2002) to language modeling.

For further background in predicting author attributes such as gender, see (Garera and Yarowsky, 2009) for an overview of previous work and (non-streaming) methodology.

7 Conclusions and Future Work

We have taken the predominately batch-oriented process of analyzing communication data and shown it to be fertile territory for research in large-scale streaming algorithms. Using the example task of automatic gender detection, on both spoken transcripts and microblogs, we showed that classification can be thought of as a continuously running process, becoming more robust as further communications become available. Once positioned within a streaming framework, we presented a novel approximation technique for compressing the streaming memory requirements of the classifier (per author) by 75%.

There are a number of avenues to explore based on this framework. For instance, while here we assumed a static, pre-built classifier which was then applied to streaming data, future work may consider the interplay with online learning, based on methods such as by Crammer et al. (2006). In the appli-

[8] See the survey by Cormode and Hadjieleftheriou (2009) for approaches to the *frequent items* problem, otherwise known as finding *heavy hitters*.

cations arena, one might take the savings provided here to run multiple models in parallel, either for more robust predictions (perhaps "triangulating" on language ID and/or domain over the stream), or predicting additional properties, such as age, nationality, political orientation, and so forth. Finally, we assumed here strictly count-based features; streaming log-counting methods, tailored Bloom-filters for binary feature storage, and other related topics are assuredly applicable, and should give rise to many interesting new results.

Acknowledgments I thank the reviewers and my colleagues at Johns Hopkins University for helpful feedback, in particular Matt Post, Mark Dredze, Glen Coppersmith and David Yarowsky. Thanks to David Yarowsky and Theresa Wilson for their assistance in collecting data.

References

Rahul Bhagat and Deepak Ravichandran. 2008. Large Scale Acquisition of Paraphrases for Learning Surface Patterns. In *Proceedings of ACL*.

Constantinos Boulis and Mari Ostendorf. 2005. A quantitative analysis of lexical differences between genders in telephone conversations. In *Proceedings of ACL*.

John D. Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Proceedings of EMNLP*.

Moses Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of STOC*.

Christopher Cieri, David Miller, and Kevin Walker. 2004. The fisher corpus: a resource for the next generations of speech-to-text. In *Proceedings of LREC*.

Graham Cormode and Marios Hadjieleftheriou. 2009. Finding the frequent items in streams of data. *Communications of the ACM*, 52(10):97–105.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Christopher P. Diehl, Galileo Namata, and Lise Getoor. 2007. Relationship identification for social network discovery. In *Proceedings of AAAI*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsief, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, (9).

Philippe Flajolet. 1985. Approximate counting: a detailed analysis. *BIT*, 25(1):113–134.

Nikesh Garera and David Yarowsky. 2009. Modeling latent biographic attributes in conversational genres. In *Proceedings of ACL*.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of ICASSP*.

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

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of STOC*.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the International Conference on Web Search and Wed Data Mining*, pages 219–230.

Abby Levenberg and Miles Osborne. 2009. Stream-based randomised language models for smt. In *Proceedings of EMNLP*.

Gurmeet Singh Manku and Rajeev Motwani. 2002. Approximate frequency counts over data streams. In *Proceedings of the 28th international conference on Very Large Data Bases (VLDB)*.

Robert Morris. 1978. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842.

S. Muthu Muthukrishnan. 2005. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2).

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of ICWSM*.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of ACL*.

Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *Proceedings of NAACL*.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in twitter. In *Proceedings of the 2nd International Workshop on Search and Mining User-generated Contents (SMUC)*.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering. In *Proceedings of ACL*.

David Talbot and Thorsten Brants. 2008. Randomized language models via perfect hash functions. In *Proceedings of ACL*.

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

David Talbot and Miles Osborne. 2007b. Smoothed Bloom filter language models: Tera-Scale LMs on the Cheap. In *Proceedings of EMNLP*.

David Talbot. 2009. Succinct approximate counting of skewed data. In *Proceedings of IJCAI*.

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

Benjamin Van Durme and Ashwin Lall. 2010. Online Generation of Locality Sensitive Hash Signatures. In *Proceedings of ACL*.

Benjamin Van Durme and Ashwin Lall. 2011. Efficient Online Locality Sensitive Hashing via Reservoir Counting. In *Proceedings of ACL*.

Benjamin Van Durme. 2012. Jerboa: A toolkit for randomized and streaming algorithms. Technical Report 7, Human Language Technology Center of Excellence, Johns Hopkins University.

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