Retaining data from streams of social platforms with minimal regret

Nguyen Thanh Tam [1], Duong Chi Thang [1], Matthias Weidlich [2], Hongzhi Yin [3], Nguyen Quoc Viet Hung [4] *

[1] École Polytechnique Fédérale de Lausanne, [2] Humboldt-Universität zu Berlin,
[3] The University of Queensland, [4] Griffith University

Abstract

Today's social platforms, such as Twitter and Facebook, continuously generate massive volumes of data. The resulting data streams exceed any reasonable limit for permanent storage, especially since data is often redundant, overlapping, sparse, and generally of low value. This calls for means to retain solely a small fraction of the data in an online manner. In this paper, we propose techniques to effectively decide which data to retain, such that the induced loss of information, the regret of neglecting certain data, is minimized. These techniques enable not only efficient processing of massive streaming data, but are also adaptive and address the dynamic nature of social media. Experiments on large-scale real-world datasets illustrate the feasibility of our approach in terms of both, runtime and information quality.

1 Introduction

Current social platforms such as Twitter, Facebook, and Yelp, produce data streams with an unprecedented rate. For example, about half a billion tweets are generated every day [24]. To make sense of these streams, one can typically only retain a small fraction of the data for analysis, due to storage limits and the cognitive load induced by the sheer data volume.

Against this background, traditional methods for the analysis of social platforms perform data summarization, e.g., based on relevance detection or measures of information diversity [9; 35]. However, these approaches are inherently limited to a static setting: The data is crawled and stored, before the top-k most important data items are selected as a data summary. Even if feasible, such an approach incurs high storage cost and does not avoid the problem of retaining only a fraction of the data once its volume exceeds a storage limit.

In this paper, we consider the natural setting of social platforms, where data is dynamic and available as a stream. Then, retaining of data becomes more challenging compared to one-off summarization, as data selection has to be repeated every time new data arrives. Instead of considering the whole historical data, summarization now works on the retained data (i.e., a previous summary) and the new data. This further degrades the informativeness of the original data since the summary of the retained data already induces some loss of information. Specifically, the lack of historical data leads to a biased assessment of data importance: Data that was considered unimportant and thus discarded in the past may retrospectively turn out to be important.

To minimize the regret of discarding important data, two requirements have to be met. First, a compact data sketch needs to be maintained, in addition to the actual data summary, to capture the long-term history of data and enable a precise assessment of data utility over time. Yet, for data stemming from social platforms, this sketch needs to be adaptive to changes in the data stream. Second, a protocol needs to be specified to decide which data items to retain and to discard, such that the total regret in data utility is minimal. To cope with the data stream volume and velocity of social platforms, this protocol needs to be very efficient.

In this paper, we tackle these requirements and propose a novel statistical model, which does not only capture the traditional context of social data (importance of topics, user influence, information diffusion) [26; 35], but also embeds the dynamics of this context over time. For example, topics are not considered to be static. They may emerge or disappear over time and relate to recurring events. Striving for online processing of streaming data, we develop a scalable learning mechanism to quickly update the model with new data. We further show how the statistical model is used to define a utility function to assess the representativeness of a data summary. Minimizing the regret of discarding data then becomes the problem of minimizing the difference between the utility of the retained data and the utility of whole historical data. Finally, we present a progressive algorithm to select which data to retain, with guarantees on the induced regret factor. This algorithm scales linearly in time and space, solely in the size of the data summary (not the whole data stream).

Our contributions and the paper structure are summarized as follows. After outlining the retaining problem with minimal regret (§2), we present (i) a statistical model to sketch data properties; (ii) a utility function to assess data representativeness (§3); and (iii) a progressive algorithm to solve the retaining problem (§4). Using diverse datasets derived from Twitter, we demonstrate improvements of five orders of magnitude in efficiency and up to 42% in information quality of our approach over state-of-the-art baselines (§5). We then review related work (§6) and conclude the paper (§7).

*Corresponding author

2 Problem Statement

We model the stream of data stemming from a social platform by an infinite set of textual data items $E = \{e_1, e_2, \ldots\}$. The items are totally ordered based on their occurrence time, denoted by the subscript. By $E_t = \{e_1, \ldots, e_t\}$, we denote the set of items until time t. Acknowledging that not all items from E can be stored permanently, a representative subset of E of size k shall be retained. Here, the parameter k depends on the application context and typically reflects the storage limit. We further postulate a non-negative function $f : 2^E \to \mathbb{R}_{\geq 0}$ to quantify the utility of a set of items $S \subseteq E$, capturing how well S represents E according to some objective. Given that E is continuously extended with new data items, the *retaining problem with minimal regret* is to select k items, such that the regret ratio—the normalized difference between the utility of the retained items and the utility of the whole data stream—is minimal.

Problem 1 (Retaining Problem with Minimal Regret)**.** *Given a data stream E_t until time t, a current set of retained items $S_t \subseteq E_t$, a window of new data items $W = \{e_{t+1}, \ldots, e_{t+|W|}\}$, the problem is to construct a new set of retained items S_{t+w}, such that:*

$$S_{t+w} = \underset{S \subseteq (S_t \cup W),\ |S|=k}{\arg\min} \frac{f(E_t \cup W) - f(S)}{f(E_t \cup W)}. \quad (1)$$

The problem setting is illustrated in Fig. 1. Upon the arrival of a window of new items, the set of retained items is updated. This is done by selecting items from the old set of retained items and from the window.

The figure further illustrates why the retaining problem with minimal regret cannot be addressed by applying traditional data summarization each time a window of new data items arrives. Traditional summarization would consider solely the current set of retained items and the items of the new window. Yet, the data stream history in terms of items discarded in the past would be neglected. Consequently, when constructing a new set of retained items, the utility of possible candidate sets cannot be assessed accurately.

As illustrated in Fig. 1, therefore, a concise sketch of historical data needs to be maintained. It captures essential properties of data items that have been discarded in the past, thereby enabling an accurate assessment of the regret ratio of a potential set of retained items. To realise such a sketch, §3 presents a statistical model and also shows how to assess the representativeness of an item set using an utility function. In §4, we then present a progressive algorithm to solve the retaining problem with minimal regret under this model.

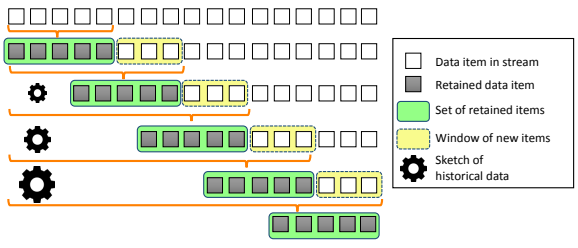


Figure 1: Illustration of the retaining problem ($k = 5$, $|W| = 3$).

3 Model and Approach

Below, we first propose a statistical model to sketch the historical data of a stream, before turning to the question of how to assess the utility of a set of retained data items. Finally, we discuss a simple strategy to solve the retaining problem.

3.1 A statistical model for social data

For textual data items that originate from social platforms, topics are a fundamental concept to understand the co-occurrence relations of words [6; 7]. We thus capture information on topics as a statistical means to select representative data items from important clusters of words. However, the dynamic nature of streaming data from social platforms prevents us from knowing a specific distribution of topics in some future state. Rather, we face the following phenomena:

- *Emergent topics*: In a streaming setting, new topics may emerge over time. Hence, topic modelling techniques such as LDA [5] or pLSI [18] that fix a pre-defined number of topics are not applicable. A small number of topics may lead to information loss, as different words might end up in the same cluster, whereas a large number of topics may imply sparse clusters, destroying data regularities.
- *Emergent vocabulary*: An evolving collection of topics implies that the vocabulary of words changes over time. Words may only be invented at a specific time [34] (e.g., 'brexit' during the events in the UK in June 2016). A fixed vocabulary as in traditional topic models [5; 14] does not capture these dynamics and tends to be inefficient due to unused and redundant words.
- *Recurring topics:* Traditional data summarization typically spans a short period of time [8; 35], due to data storage limits. In contrast, when processing data streams of social platforms, a long history of data items is considered, so that topics will recur over time [15] (e.g., the topic of 'football' shows seasonal patterns). Such effects influence the decision of which data items to retain.

Against this background, we propose a non-parametric [19] probabilistic model to sketch the properties of past data items. It features a potentially unbounded number of topics and words that are learned from textual data items over time. It also considers recurrent topics by means of temporal cluster variables, for which the time granularity can be customised.

Formally, a textual data item e is modelled as a multiset of words $\{v_1, \ldots, v_{|e|}\}$, where v_i is a word from a dynamic vocabulary V. A multiset of words further defines a semantic topic z. We model the dynamics of words per topic by means of a Dirichlet process to generate the word distribution ϕ_z and the vocabulary ρ_z of a topic z. Further, at time t, a topic distribution θ is used to generate the topics of a new data item e_t. To ensure that the temporal aspect of evolving topics is reflected in the generation, we use a hierarchical Dirichlet process to establish the link between data items in terms of topics. This yields a a concise and consistent set of topics rather than a sparse and unnecessarily large one. Finally, we model the recurring topics by means of temporal clusters, whose number is unbounded in general. To this end, a Chinese restaurant process [3] τ is used to non-parametrically generate a temporal cluster label c_t for each data item e_t. The model is summarised in Fig. 2.

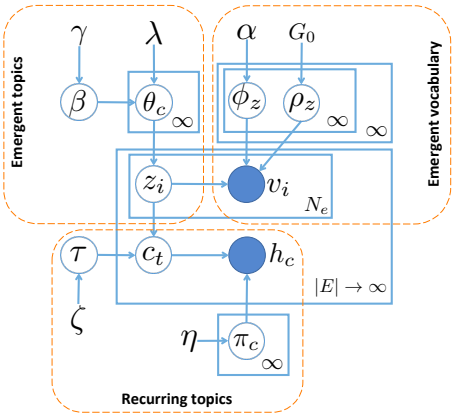


Figure 2: Model to sketch historical data. Shaded/blank circles are observed/latent variables, non-circles are model parameters.

Generative process. The generative process for our model is defined as follows. For each item $e_t \in E$ at time t:

(1) Generate the lengths (number of words) of e_t from a Poisson distribution:
$N_e \sim Poiss(\epsilon)$

(2) Generate the topic of e_t from a categorical distribution:
$c_t \sim Cat(\tau)$

(3) Generate the temporal cluster label of e_t from a multinomial distribution:
$h_t|c_t, \pi \sim Mult(\pi_{c_t})$
where the value of h_t depends on the granularity according to which the topic distribution is captured for the data stream (e.g., h_t has a value of 31 to model topics per day of a month).

(4) For each of the $i = 1 \dots N_e$ word indices of e_t:
   a. Generate a topic from a multinomial distribution:
   $z_i|c_t, \theta \sim p(z|t) \triangleq Mult(\theta_{c_t})$
   b. Generate a word v_i from z_i via a multinomial distribution:
   $v_i|z_i, \phi, \rho \sim p(v|z) \triangleq Mult_{\rho_{z_i}}(\phi_{z_i})$
   where ρ_z is a vocabulary generated from an n-gram model and ϕ_z is the probability of selecting ρ_z for word v_i [34].

Model parameters. The sketch of historical data is designed by parametrising the model as follows. The sketch is denoted by $\Theta = (\alpha, G_0, \lambda, \gamma, \zeta, \eta, \epsilon)$, a vector of parameters for:

- The Dirichlet process [13] to generate ρ and ϕ:
  $\rho_{zk}, \phi_{zk} \sim DP(\alpha, G_0)$, for $t = 1, 2, \dots$ and $k = 1, 2, \dots$.
- The hierarchical Dirichlet process [31] to generate θ:
  $\beta \sim GEM(\gamma), \theta_c \sim DP(\lambda, \beta)$, for $c = 1, 2 \dots$
- The Chinese restaurant process [3] to generate τ:
  $\tau \sim CRP(\zeta)$
- The Dirichlet distribution [13] to generate π:
  $\pi_c \sim Dir(\eta)$, for $c = 1, 2, \dots$

This parametrisation of the model yields a compact, lightweight sketch of historical data. Since it contains solely single-variable parameters, it has constant-space requirements. In §4, we will show how to update the model parameters based on a data stream by means of an incremental inference mechanism.

3.2 Utility of retained data

To judge how well a set of retained data items stemming from social platforms represents a whole data stream, we argue that the following aspects shall be considered: First, *semantic* information such as topics and word frequencies has to be incorporated, see [5; 25; 35]. Second, the importance of data from social platforms is influenced by the *social context* (e.g., the authors of a textual statement) and its *freshness*, as the interestingness of social data degrades over time. Given the above sketch of historical data $\Theta^{(t)}$ at time t, these incorporate these aspects in the assessment of data utility as follows.

On semantic information. We first consider the probability $p(v, e)$ for observing a word w in a data item e at time t. It is defined based on the evolution of topics over time as

$$p(v, e) = \mathbb{E}_{p(z|\Theta^{(t)})} p(v|z)$$

where $p(v|z)$ represents the probability distribution of words given a topic, and $p(z|\Theta^{(t)})$ represents the probability distribution of topics at time t. The latter is derived from sketch Θ by the aforementioned generative distributions.

Based thereon, the probability of an item e being semantically important is defined as

$$p(e|\Theta^{(t)}) = \prod_{v \in e} p(v, e|\Theta^{(t)}) = \prod_{v \in V^{(t)}} p(v, e)^{n(v,e)}$$

where $V^{(t)}$ is the vocabulary at time t (maintained based on $\Theta^{(t)}$ in constant space [34]) and $n(v, e)$ denotes the frequency of a word $v \in V^{(t)}$ in item e.

On freshness and social context. To model that the interestingness of data degrades over time, we define a monotonic decreasing function $g(t)$. Specifically, the decay in interestingness of past data is described by an exponential form:

$$g(e) = \exp^{-\lambda(t - t(e))}$$

where λ is the decay rate and is set to 0.5 (maximal entropy principle), t is the current time and $t(e)$ is the time of e. Following [26; 35], we further associate each item e with a vector of social features $(h_1(e), \dots, h_m(e))$. Then, the aggregation of these features, denoted by $h(e)$, describes the social context of a data item.

A utility measure. In social data, topics with highly frequent words may dominate other topics. To avoid such vocabulary bias, we define the utility of a set of items S as the log-likelihood over its items based on the information entropy. This measure of utility incorporates the above notions of semantic information, freshness, and social context:

$$f(S) = \sum_{e \in S} \sum_{v \in V} n(v, e) p(v, e) \log \frac{1}{p(v, e)} g(e) h(e) \quad (2)$$

Using this formulation, an algorithm to select data items will prefer sets with high entropy, i.e., sets with items that cover diverse topics and preserve the evolving topic distribution. As proven in the appendix, the above measure is monotonic (selecting more items increases utility) and submodular (marginal gains by selecting more items start to diminish due to saturation of the utility objective).

Proposition 1. *$f(.)$ is a monotonic function.*

Proposition 2. *$f(.)$ is a submodular function.*

3.3 A simple retaining algorithm

A straight-forward approach to solve Problem 1 under the above model applies traditional data summarization [35] on the retained items and the content of a new window. Then, the new set of retained items is selected as:

$$\max_{S \subseteq S_t \cup W, |S|=k} f(S) \tag{3}$$

which is equivalent to Eq. 1 since the value of $f(E_{t+w})$ is constant in terms of selecting any S. However, this problem is known to be NP-complete [35; 27; 21]

Due to the computational complexity, greedy approximation algorithms (inspired by the knapsack problem) are commonly employed. They start with the empty set $S^{(0)} = \emptyset$, and at each iteration i over the current data, choose an item $e \in S_t \cup W$ maximizing the utility, i.e.,

$$S^{(i)} = S^{(i-1)} \cup \arg\max_{e \in S_t \cup W} f(S^{(i-1)} \cup e) - f(S^{(i-1)}) \tag{4}$$

For a monotonic and submodular function (as our utility function defined above), this greedy algorithm yields a $(1 - 1/e) \approx 0.63$ approximation [27]. However, this algorithm has an update time complexity of $\mathcal{O}(k(k + |W|))$, which is undesirable for streaming applications. Also, the update process needs to be repeated every time new data arrives.

4 A Progressive Retaining Algorithm

Given the above results on hardness of the retaining problem and the time complexity of a simple greedy algorithm, we now present a progressive algorithm. It is tailored to the stream processing setting and shows linear time and space complexity. The algorithm comprises of (i) an incremental inference mechanism to update the sketch of historical data; and (ii) a mechanism to select a new set of retained items.

4.1 Updating the data sketch

To update the parameters Θ of our model, we realise an online learning mechanism. When data arrives, we compute the observed variables and propagate back the information to the model parameters. Once the parameters have been updated, the conditional and marginal probabilities for the utility function are computed following the generative process.

Many online learning techniques have been developed based on Markov chain Monte Carlo sampling (e.g., incremental Gibbs sampling [6]). However, these techniques either reduce model complexity (loosing the guarantee to converge for the complete model) or have a space complexity that is linear in the number of items to analyse [16]. To overcome these issues, we rely on stochastic variational inference [17]. Here, the idea is to minimize the evidence lower bound (ELBO) of the expected difference between the observed distribution and the latent distribution, defined as:

$$L(Z) = \mathbb{E}_{q(Z)}[ln\frac{p(Z,E)}{q(Z)}]$$

where Z is the set of all latent variables in the model (except model parameters and observed variables); and $q(Z)$ is an approximate distribution of $p(Z|E)$, which can be factorised over the distributions of model parameters in the generative process. Then, we apply stochastic optimisation to the ELBO function over a data stream, which basically updates the new parameter values from the previous ones following the direction of the ELBO gradient of new data with a fixed step size. The more data is received, the more the model parameters will converge to minimize the ELBO function.

Instead of single-item update, our approach considers multiple observations per update to reduce noise [17], which also aligns with window-based processing to avoid order distortion in data streaming settings. Receiving data as a series of windows W_b, $b = 1, 2, \ldots$, of items, we proceed as follows:

1. We update the local parameters of the variational distribution of the word-topic variable z and the topic-cluster variable c. This requires us to maintain additional M_z local parameters for possible values of z, and M_c local parameters for possible values of c. Here, M_c, M_z are 'prior beliefs' on the maximum number of topics and recurring topic clusters. Yet, their effects are marginal, as the updates are dominated by the observed information, so that they can be safely set to large constant values (e.g., 1000).
2. We compute the natural gradients using previous parameter values $\Theta^{(t-1)}$ and the above local parameters of the ELBO function decomposed over each item $e \in E_b$ [16]. Formally, we obtain $\nabla_{\Theta^{(t-1)}} L_e$ as a vector of gradients for each parameter in $\Theta^{(t-1)}$.
3. The new values for model parameters are computed from their previous value:

$$\Theta^{(t)} = \Theta^{(t-1)} + w_b \frac{1}{|W_b|} \sum_{e \in W_b} \nabla_{\Theta^{(t-1)}} L_e \tag{5}$$

where w_b is the learning rate to control the learning quality and convergence of the inference. w_b is often modelled as a power function of b with a forgetting rate r [17]. Setting $r \in (0.5, 1]$ guarantees convergence [17], while larger values often lead to higher learning quality and faster convergence (but not monotonically).

Note that windows of large size reduce the number of updates, but may lead to a poor estimation of model parameters. To further improve scalability, updating of model parameters can be parallelised by exploiting the conditional independence property. When the global variables (i.e., the most outer parameters in the model) are given, the updates to local variables (i.e., inner parameters) become independent and can thus be computed concurrently. Also, the computation of semantic information, decay in interestingness, and social features per data item, see §3.2, is independent once the model parameters are updated and thus can be parallelised.

4.2 Retaining data items

We now turn to the selection of data items to retain. In essence, at each step, a new set of items is selected as the one with the highest utility among all candidate sets. The candidate sets are created by swapping at most one new item with a retained item, if the utility increases. However, the arrival of a new item may change the model parameters, influences the decay in interestingness of old items, and may introduce new social features. Hence, the utility of the retained set of items has to be updated, before the swapping procedure is started.

Retaining algorithm. Our progressive retaining algorithm is formalised in Alg. 1. We illustrate the algorithm with a window size of $|W| = 10$ (line 6). The algorithm starts with the empty set $S_0 = \emptyset$. As long as no more than k elements $e_1, \ldots, e_t$ have arrived, all of them are kept, i.e., $S_t = S_{t-1} \cup \{e_t\}$, for $t \leq k$. For each new item e_t, where $t > k$, we update the utility value of $f(S_{t-1})$ and compute the semantic importance $p(e_t)$, the decay in interestingness $g(e_t)$ and the social features $h(e_t)$. Then, we check whether swapping this item and an item in S_{t-1} will increase the utility value. If so, the one that maximizes the utility is selected for swapping.

Theorem 1. *Alg. 1 does a single pass over data stream, uses $\mathcal{O}(k)$ memory, and has $\mathcal{O}(k)$ update time per item.*

The proof of Theorem 1 can be found in the appendix. Correctness of Alg. 1 is established as follows: First, the decay in interestingness is a monotonic decreasing function. Thus, an optimal selection remains optimal after the utility has been updated. Also, even if a new item does not increase utility in terms of entropy and social features, the algorithm still swaps it with an old item to preserve the freshness of the retained set of items. Second, adding a new item increases the entropy of the topic distribution. Thus, while the algorithm favours new items, it still preserves topics by ensuring an even distribution of the selected items across all topics (old and new). Moreover, an optimal selection remains optimal as the entropy increases for all old items.

Incremental utility computation. The above complexity result assumes that the computation of utility (line 14), when swapping data items, is done in constant time. This indeed holds true, since utility can be computed incrementally, i.e., $f(S_i)$ is derived from $f(S_{i-1})$ in constant time as follows:

$$f(S_{i-1} \setminus \{e\} \cup \{e'\}) = f(S_{i-1}) - \sum_{v \in V} n(v,e)p(v,e) \log \frac{1}{p(v,e)}$$

$$g(e)h(e) + \sum_{v \in V} n(v,e')p(v,e') \log \frac{1}{p(v,e')} g(e')h(e').$$

Here, values $p(.,.)$, $n(.,.)$, $g(.)$, $h(.)$ have been computed already in the previous steps of the algorithm (lines 9 to 13).

Algorithm 1: A Progressive Retaining Algorithm

input : An infinite sequence E of data items
output: A selected set S_t of size k of data items at any time t

1 $S_0 = \emptyset$;
2 **for** $t = 1$ *to* k **do** $S_t = S_{t-1} \cup \{e_t\}$;
3 $W = \emptyset$; ▷ Window
4 **for** $t = k + 1$ *to* $|E|$ **do**
5 $W = W \cup \{e_t\}$;
6 **if** $|W| < 10$ *and* $t < n$ **then**
7 $S_t = S_{t-1}$;
8 continue;
`// Incremental learning of model parameters`
9 Compute $\Theta^{(t)}$ from $\Theta^{(t-1)}$ and W;
10 Update $p(v,e)$ and $n(v,e)$ by new parameter $\Theta^{(t)}$, $\forall v \in V, e \in S_{t-1}$;
11 Update $g(e)$ $\forall e \in S_{t-1}$;
12 Update $f(S_{t-1})$;
`// Prepare computation of utility of new items`
13 Compute $p(v,e')$, $n(v,e')$, $g(e')$, $h(e')$ for all $e' \in W$ and $v \in V$;
`// Find swapping pair`
14 $e^*, e_b = \arg\max_{e \in S_{t-1}, e' \in W} f(S_{t-1} \setminus \{e\} \cup \{e'\})$;
15 **if** $f(S_{t-1} \setminus \{e^*\} \cup \{e_b\}) \geq f(S_{t-1})$ **then**
16 $S_t = S_{t-1} \setminus \{e^*\} \cup \{e_b\}$;
17 **else** $S_t = S_{t-1}$;
18 $W = \emptyset$; ▷ Reset for new window

5 Empirical Evaluation

Below, we first elaborate on the experimental setup, before we analyse our method's efficiency and effectiveness.

5.1 Experimental Setup

Datasets. We extracted datasets using the Twitter Streaming API [35]. Over a year, we considered five different domains (climate change, vaccination, processed food, genetically modified organism, general public) and randomly selected 1 million English tweets per domain. Furthermore, a total of five important social features (e.g., user influence, retweet score, and affective language) had been extracted for each data item using existing frameworks [35].

Baselines. We compare our approach with several baselines:

- *Traditional summarization:* a state-of-the-art summarization technique [35] for social data.
- *Greedy*: the simple greedy algorithm (see §3.3) to select the items to retain.
- *Offline learning*: an iterative algorithm to compute model parameters using deterministic variational inference [22; 4], which requires a full pass of the data in each iteration.
- *Static:* our retaining algorithm tailored to the traditional, static setting of data summarization: *offline learning* to compute the sketch of historical data and the *greedy* algorithm to select the items to retain.

Environment. All results have been obtained on an Intel i7 3.8GHz system (4 cores, 16GB RAM). Following [17; 16], we vary the forget rate in $(0.5, 1]$, choose a stable window size = 10 and report average values.

5.2 Efficiency

We evaluate the update time of our approach, when new data arrives. To assess the average time per window needed to update the model parameters, we compare our online learning algorithm with its offline version. The latter considers the whole data received so far when computing the parameters upon the arrival of a new window. Fig. 3 illustrates the results averaged over all datasets, reporting the average update time until $100K$, $500K$, and $1M$ data items are received. Here, the update time of our progressive approach remains constant and small (< 0.01s), whereas the baseline yields a high and increasing runtime (up to 10^3s).

Focusing on how to select the items to retain, we compare the efficiency of our progressive algorithm (*fast*) with the *greedy* algorithm. We realise the online setting as above and vary the size of the set of retained items ($k = 0.2\%$ to 1%). Fig. 4 depicts the update time of the algorithms, averaged over all datasets. Our progressive algorithm outperforms the *greedy* one. It also scales better to large data summaries.

5.3 Effectiveness

We compare the quality of model parameters, in terms of utility, obtained with our online learning algorithm and its offline

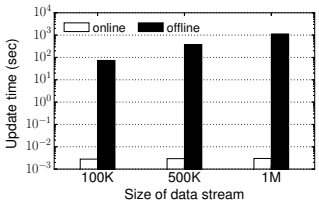


Figure 3: Model update

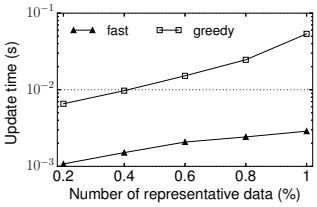
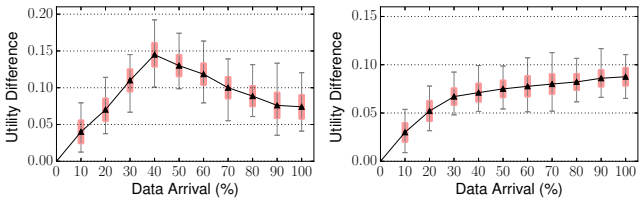


Figure 4: Retaining data

version. To mitigate the randomness of the Twitter streaming API, we select 100K items $\tilde{E}$ from the original datasets and construct a set of retained items S as the $k = 1\%$ oldest items in $\tilde{E}$. We stream $\tilde{E}$ and learn model parameters online ($\Theta_{\tilde{E}}$). Offline learning considers all data received so far, the result being $\Theta'_{\tilde{E}}$. We then assess the relative difference in utility of S, computed with either method, i.e., $\frac{|f_{\Theta}(S)-f_{\Theta'}(S)|}{f_{\Theta'}(S)}$.



(a) Updating model parameters **(b)** Retaining data items

Figure 5: Effectiveness relative to amount of processed data

The difference in utility relative to the amount of processed data is shown in Fig. 5a (averaged over 100 runs of different $\tilde{E}$ and the five datasets). Due to the stochastic property of online learning (data is needed to converge in the model parameters), the utility increases initially. Also, the difference between the online and offline learning results is less than 15% in general, underlining the usefulness of our approach.

We further compare the utility of retained items selected by the greedy and our progressive algorithm. Similar to the above setting, we stream all data and use online learning to update the model parameters. Both algorithms start from the set of $k = 1\%$ old items and update it upon the arrival of a new window. We then measure the relative difference of the obtained sets of retained items at each step. The results in Fig. 5b (averaged over 100 runs and the five datasets) show that the utility difference increases with the arrival of data. This is because the progressive algorithm accumulates some loss of information. However, the difference is small ($\leq 15\%$) and converges with more data. As such, data quality is not compromised too much.

Finally, we compare the overall utility ratio (utility of output over utility of whole data) of our retaining algorithm (*dynamic*) and two baselines: the *static* version of our algorithm and traditional summarization (*sum*). The utility ratios obtained after processing all data of the five datasets are shown in Table 1, for different sizes of the set of retained items ($k = 0.1\%$ and $k = 1\%$). While the *static* approach outperforms traditional summarization (*sum*), we need to acknowledge that both, *static* and *sum*, are inapplicable for streaming data. However, the results of the *dynamic* technique are relatively close to those of the *static* approach, highlighting its usefulness for online processing. Also, its decrease in utility for a smaller number of retained items (k) is less drastic compared to traditional summarization.

Table 1: Overall utility ratio

k		climate	vacc	food	gmo	public
1%	static	0.84	0.83	0.84	0.81	0.86
	dynamic	0.70	0.74	0.73	0.74	0.76
	sum	0.72	0.70	0.69	0.71	0.71
.1%	static	0.74	0.76	0.73	0.72	0.75
	dynamic	0.69	0.67	0.68	0.65	0.68
	sum	0.51	0.53	0.48	0.51	0.52

6 Related work

Social data analysis is often based on topic modelling [5], feature extraction [26; 32; 28], and temporal-aware information processing [9]. Methods for topic modelling, e.g., Latent Dirichlet Allocation [5], hierarchical Dirichlet processes [14], or word modelling [34], are not applicable for a streaming setting, since they require multiple passes over the data. Streaming versions of these techniques [16; 6], in turn, ignore the dynamics of social data. Our model follows a non-parametric approach, where the number of topics and vocabulary words is learned from the data rather than specified in advance. Moreover, our model incorporates social features [26; 12] when assessing data utility. Also, methods to query a streams of social data [23; 33] are not applicable for data summarization, since the query is not known in advance.

Traditional data summarization works on offline data [35; 30; 29; 20] and, even if temporal aspects are considered [9; 1; 10], on the whole data. Existing streaming algorithms for data summarization [2] also rely on access to the complete data, as they sample the data for an estimation of the utility. A relaxed version of the retaining problem has been addressed in [11], which finds subsets of items with maximal utility using a sliding window. Yet, different from our problem formulation, this method targets solely the recent data bounded by a fixed window size, discarding all old items. Whereas, our approach retains old items as long as they are valuable. Also, unlike [11], our approach summarizes the entire history of a data stream. Finally, the algorithm in [11] assumes apriori knowledge of an upper bound of utility. While this assumption may be reasonable for some types of data streams, it it unrealistic for dynamic data produced by social platforms.

7 Conclusions

This paper proposed a technique to retain a representative set of items from a stream of social data. That is, we acknowledge the online nature of data produced by social platforms, which prevents us from storing the complete data stream. This led the retaining problem with minimal regret, where a protocol decides which data to retain, such that the loss of utility is minimized. To address this problem, we proposed a light-weight, adaptive sketch of historical data and a progressive algorithms for the selection of data items. Experiments on large-scale real-world data showed that our approach is efficient (five orders of magnitude faster than the baseline) and effective (less than 15% reduction in the utility ratio).

References

[1] N. Alsaedi, P. Burnap, and O. F. Rana. Automatic summarization of real world events using twitter. In *ICWSM*, pages 511–514, 2016.

[2] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: Massive data summarization on the fly. In *KDD*, pages 671–680, 2014.

[3] D. M. Blei, T. L. Griffiths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *JACM*, pages 7:1–7:30, 2010.

[4] D. M. Blei, M. I. Jordan, et al. Variational inference for dirichlet process mixtures. *Bayesian analysis*, pages 121–143, 2006.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, pages 993–1022, 2003.

[6] K. R. Canini, L. Shi, and T. L. Griffiths. Online inference of topics with latent dirichlet allocation. In *AIStats*, pages 65–72, 2009.

[7] Y. Cha, B. Bi, C.-C. Hsieh, and J. Cho. Incorporating popularity in topic models for social network analysis. In *SIGIR*, pages 223–232, 2013.

[8] D. Chakrabarti and K. Punera. Event summarization using tweets. *ICWSM*, pages 66–73, 2011.

[9] Y. Chang, J. Tang, D. Yin, M. Yamada, and Y. Liu. Timeline summarization from social media with life cycle models. In *IJCAI*, pages 3698–3704, 2016.

[10] F. C. T. Chua and S. Asur. Automatic summarization of events from social media. In *ICWSM*, pages 81–90, 2013.

[11] A. Epasto, S. Lattanzi, S. Vassilvitskii, and M. Zadimoghaddam. Submodular optimization over sliding windows. *arXiv preprint arXiv:1610.09984*, 2016.

[12] P. Ernst, C. Meng, A. Siu, and G. Weikum. Knowlife: a knowledge graph for health and life sciences. In *ICDE*, pages 1254–1257, 2014.

[13] T. S. Ferguson. A bayesian analysis of some nonparametric problems. *AOS*, pages 209–230, 1973.

[14] D. Griffiths and M. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *NIPS*, pages 17–24, 2004.

[15] S. Herrera-Damas. Recurring topics in the social media policies of mainstream media. *AJMS*, pages 155–173, 2014.

[16] M. Hoffman, F. R. Bach, and D. M. Blei. Online learning for latent dirichlet allocation. In *NIPS*, pages 856–864, 2010.

[17] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *JMLR*, pages 1303–1347, 2013.

[18] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57. ACM, 1999.

[19] M. Hollander, D. A. Wolfe, and E. Chicken. *Nonparametric statistical methods*. John Wiley & Sons, 2013.

[20] N. Q. V. Hung, H. Jeung, and K. Aberer. An evaluation of model-based approaches to sensor data compression. *TKDE*, pages 2434–2447, 2013.

[21] N. Q. V. Hung, D. C. Thang, M. Weidlich, and K. Aberer. Minimizing efforts in validating crowd answers. In *SIGMOD*, pages 999–1014, 2015.

[22] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, pages 183–233, 1999.

[23] C. Kedzie, F. Diaz, and K. McKeown. Real-time web scale event summarization using sequential decision making. In *IJCAI*, pages 3754–3760, 2016.

[24] M. Mathioudakis and N. Koudas. Twittermonitor: trend detection over the twitter stream. In *SIGMOD*, pages 1155–1158, 2010.

[25] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley. Is the sample good enough? comparing data from twitter's streaming api with twitter's firehose. In *ICWSM*, pages 400–408, 2013.

[26] S. Mukherjee, G. Weikum, and C. Danescu-Niculescu-Mizil. People on drugs: credibility of user statements in health communities. In *KDD*, pages 65–74, 2014.

[27] G. L. Nemhauser and L. A. Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. *NHMS*, pages 279–301, 1981.

[28] Q. V. H. Nguyen, C. T. Duong, T. T. Nguyen, M. Weidlich, K. Aberer, H. Yin, and X. Zhou. Argument discovery via crowdsourcing. *JVLDB*, pages 1–25, 2017.

[29] T. T. Nguyen, Q. V. H. Nguyen, M. Weidlich, and K. Aberer. Result selection and summarization for web table search. In *ICDE*, pages 231–242, 2015.

[30] J. Nichols, J. Mahmud, and C. Drews. Summarizing sporting events using twitter. In *IUI*, pages 189–198, 2012.

[31] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical dirichlet processes. *JASA*, pages 1566–1581, 2006.

[32] Y. Wang and A. Pal. Detecting emotions in social media: A constrained optimization approach. In *IJCAI*, pages 996–1002, 2015.

[33] D. Yang, B. Li, and P. Cudré-Mauroux. Poisketch: Semantic place labeling over user activity streams. In *IJCAI*, pages 2697–2703, 2016.

[34] K. Zhai and J. L. Boyd-Graber. Online latent dirichlet allocation with infinite vocabulary. *ICML*, pages 561–569, 2013.

[35] H. Zhuang, R. Rahman, X. Hu, T. Guo, P. Hui, and K. Aberer. Data summarization with social contexts. In *CIKM*, pages 397–406, 2016.

Appendix

Proof of Proposition 1. Let E be a sequence of items, $S \subset E$ is a selection, and $e \in E \setminus S$ is from a set of non-selected tweets. Then it holds that: $f(S \cup \{e\}) \geq f(S)$. Indeed, we denote all the words that occurs in e but not in the selection S as the set of words $e \setminus V_S$. Then we have $f(S \cup \{e\}) - f(S) = \sum_{v \in e \setminus V_S} n(v, e) p(v, e) \log \frac{1}{p(v,e)}$, which is non-negative.

Proof of Proposition 2. Let E be a sequence of items, S a selection, and $e, e' \in E \setminus S$ from a set of non-selected tweets. Then it holds that: $f(S \cup \{e\}) - f(S) \geq f(S \cup \{e, e'\}) - f(S \cup \{e'\})$. Similar to the proof of monotonicity, we expand the inequality to: $\sum_{v \in e \setminus V_S} n(v, e) p(v, e) \log \frac{1}{p(v,e)} \geq \sum_{v \in e \setminus V_{S \cup \{e'\}}} n(v, e) p(v, e) \log \frac{1}{p(v,e)}$, which is equivalent to $\sum_{v \in e'} n(v, e) p(v, e) \log \frac{1}{p(v,e)} \geq 0$. The equality happens if and only if $e \cap e' = \emptyset$.

Proof of Theorem 1. The proof is straightforward from the algorithm. The loops in line 2 and line 4 pass over the data stream only once. We need to maintain a frequency matrix and a probability matrix for $n(v, e)$ and $p(v, e)$ for all $v \in V$ and $e \in S_t$ where $|S_t| = k$. Other maintenance of $g(e)$ and $p(e)$ take only $\mathcal{O}(k)$ memory. Considering the number of model parameters as constant yields the required memory as $\mathcal{O}(k)$ and the update time per each loop in line 14 as $\mathcal{O}(k|W|)$. Since $|W|$ is often small ($|W| \ll k$) and can be considered as constant, the update time complexity becomes $\mathcal{O}(k)$.