Hash-Based Stream LDA: Topic Modeling in Social Streams

Anton Slutsky, Xiaohua Hu, and Yuan An

College of Computing and Informatics, Drexel University, USA
{as3463,xh29,yuan.an}@drexel.edu

Abstract. We study the problem of topic modeling in continuous social media streams and propose a new generative probabilistic model called Hash-Based Stream LDA (HS-LDA), which is a generalization of the popular LDA approach. The model differs from LDA in that it exposes facilities to include inter-document similarity in topic modeling. The corresponding inference algorithm outlined in the paper relies on efficient estimation of document similarity with Locality Sensitive Hashing to retain the knowledge of past social discourse in a scalable way. The historical knowledge of previous messages is used in inference to improve quality of topic discovery. Performance of the new algorithm was evaluated against classical LDA approach as well as the stream-oriented On-line LDA and SparseLDA using data sets collected from the Twitter microblog system and an IRC chat community. Experimental results showed that HS-LDA outperformed other techniques by more than 12% for the Twitter dataset and by 21% for the IRC data in terms of average perplexity.

1 Introduction

In this paper we are motivated by the problem of topic discovery in social media. We recognize that topic discovery systems for online social discourse need to address a set of challenges associated with the scale of modern social media outlets such as Twitter, chat systems and others. To be useful, these systems must operate continuously for extended periods of time, as social conversations do not stop, produce output in a timely fashion to remain relevant and ensure high quality of output.

Commonly used data mining techniques handle the problem of social stream topic discovery by applying batching heuristics to process the never-ending stream of messages. Since retaining all messages is not feasible in practice, current topic modeling approaches improve quality of topic discovery by retaining globally applicable statistics such as topic-word counters, but fail to take advantage of document-level information as no technique has existed so far to retain such information in a scalable and meaningful way.

Therefore, in this work we propose a new generative probabilistic model called Hash-based Stream LDA (HS-LDA), which is a generalization of the popular Latent Dirichlet Allocations (LDA) [1]. The model improves upon previous works by introducing a theoretical framework that makes it possible to retain the knowledge of

V.S. Tseng et al. (Eds.): PAKDD 2014, Part I, LNAI 8443, pp. 151–162, 2014.

historical stream messages in a scalable way and use this knowledge to improve the quality of topic discovery in social streams. Further, an efficient inference mechanism for the HS-LDA model is outlined, which makes use of the scalable hashing algorithm called Locality Sensitive Hashing (LSH) [2]. We show that the HS-LDA model and the associated inference algorithm are well suited for topic discovery in streams by comparing the predictive power of the topic models inferred by HS-LDA with that of topics learned by applying the classical LDA, On-line LDA [3] and SparseLDA [4] approaches to stream data. Evaluation was performed using data collected from the Twitter microblog site and an IRC chat system. Our experiments showed that HS-LDA outperformed other techniques by more than 12% for the Twitter dataset and by 21% for the IRC data in terms of average perplexity.

The paper is organized as follows. In section 2, current state of the art of topic modeling and stream mining is discussed. Section 3 introduces the HS-LDA model, outlines an efficient inference algorithm and discusses its application to stream data. In section 4, comparison of performance of our method to that of other modeling approaches in terms of perplexity is presented. Section 5 concludes the paper and outlines future work.

2 Related Works

The seminal work on Latent Dirichlet Allocation (LDA) [1] provides basis for numerous extensions and generalizations in the field topic modeling. LDA considers document collections as bag-of-words assemblies that are generated by stochastic processes. To generate a document, a random process first selects a topic from a distribution over topics and then generates a word by sampling the associated topic-word distribution. Both the topic and the word distributions are governed by hidden (or latent) parameters.

The LDA framework is designed to operate on a fixed set of documents and cannot be applied to stream data directly as converting an unbounded number of documents to a finite collection is not possible. To overcome this challenge, many approaches limit the training scope by aggregating messages based on attributes such as authorship or hash tag annotations and training models based on these aggregates [5], [6, 7].

An interesting recent work by Want et al. introduced an efficient topic modeling technique called TM-LDA for stream data. This approach is based on the notion that if document topic model is known at time t, at time $t + 1$ a new topic model can be predicted and an error can be computed by comparing the "old" and the "new" topic models. This error computation reduces the challenge of estimating topic models for new documents to a least-squares problem, which can be solved efficiently. Focusing on the popular Twitter micro-blog data, TM-LDA selects a set of individual authors and trains a separate model for each of the authors. To accomplish this, TM-LDA monitors Twitter for an extended period of time (a week's worth of data was collected in the original work) and then trains a model to be able to predict new messages.

The idea of using authorship to improve topic modeling quality is not unique to TM-LDA. A recent work by Xu et al. modified the well-known Author-Topic [8] model for Twitter data [6]. Xu et al. extended the insight of the Author-Topic model by taking advantage of additional features available in Twitter such as links, tags, etc.

Another way to approach topic modeling in streams is to apply LDA machinery to snapshots or buffers of documents of fixed size. Online Variational Inference for

LDA [9] is one such technique. The algorithm assembles mini-batches of documents at periodic intervals and uses Expectation Maximization (EM) algorithm to infer distribution parameters by holding topic proportions fixed in the E-step and then re-computing topic proportions as if the entire corpus consisted of document mini-batches repeated numerous times. Topic parameters are then adjusted using the weighted average of previous values of each topic proportion.

Another approach termed On-line LDA [3] considers the data stream as a sequence of time-sliced batches of documents. The approach processes each time-slice batch using the classical LDA sampling techniques, with the variation being that the corresponding collapsed Gibbs sampler initialization is augmented with the inclusion of topic-word counters from histories of pervious time-slice batches. The histories are maintained using a fixed-length sliding window and the contribution of each history to the current slice initialization is predicated upon a set of weights associated with each element in the sliding window.

In another work, Yao et al in [4] considered topic discovery in streaming documents and proposed the SparseLDA model. Noticing that the efficiency of sampling-based inference depends on how fast the sampling distribution can be evaluated for each token, their work enhanced the inference procedure in a way as to allow parts of computations used in sampling to be pre-computed, thus improving performance. Further, the sampling procedure proposed by Yao et al. restricted training to a fixed collection of training documents and then, for each test document, sampled topics using counts from the training data and test document only, ignoring the rest of the stream.

The explosion of micro-blog popularity has attracted much attention from outside of the topic modeling community. One particularly interesting application is the field of first story detection. Conceptually, first story detection is concerned with locating emergent clusters of similar stream messages, which are said to be indicative of particularly interesting and currently relevant stories. First story detection approaches require the ability to discover clusters of similar documents in near real-time fashion, which is difficult to accomplished using classic clustering tools since the computational complexity of commonly used clustering algorithms (hierarchical, partitioning, etc.) is quite high. Therefore, recent works on first story detection have seized upon the concept of Locality Sensitive Hashing (LSH) [2], which is an approach for identifying a datum neighborhood in constant time [10]. In [10], Petrovic et al use a combination of LSH and inverse index searching to show that clusters of similar documents may be identified in constant time with exceptional accuracy and low variability.

3 Hash-Based Stream LDA

As noted in the preceding survey of related works, many approaches to topic modeling in streams have been developed in recent years. A number of these approaches [3, 9] attempted to enhance quality by preserving various aspects of topic inference calculations and predicating topic learning upon past knowledge. Unfortunately, none of these techniques were successful in retaining the knowledge of stream documents relying instead on storing global structures such as topic-word multinomials. Hurdles for retaining document knowledge are two-pronged – 1) the number of documents in streams is unbounded making storage of individual document information not feasible, and 2) since previous documents do not get replayed in streams, retaining records of their presence directly may be meaningless for topic modeling.

Therefore, this section introduces the new Hash-Based Stream LDA (HS-LDA) model, which provides a mechanism for retaining document knowledge for stream modeling in a scalable and meaningful way. HS-LDA is a generative probabilistic model that describes a process for generating a document collection. Like LDA, in HS-LDA each document is viewed as a mixture of underlying topics and each word is generated by drawing from a topic-word distribution. HS-LDA departs from LDA by imagining that, in addition to words, the generative process also emits certain auxiliary objects that are not directly observable in data. In order to refer to these objects in an intuitive way, we reach out to the physical world for a descriptive analogy. We borrow from particle physics nomenclature and recall that, in physics, a neutrino is a nearly massless, uncharged particle that is detectable only through its interactions with other matter [11]. Since the auxiliary objects postulated by the HS-LDA model are similarly ethereal, we introduce the notion of *HS-LDA neutrinos* (or *pseudo-neutrinos* for short), as the analogy with the real particle seems appropriate.

Following the analogy, as the physical neutrinos are said to be classifiable into a collection of categories [11], the HS-LDA *pseudo-neutrinos* are also thought to belong to a fixed set of possible types (or flavors). The physics analogy is abandoned at this point, however, as HS-LDA makes no further claims as to the properties or nature of each flavor. The generative process is graphically outlined in Figure 2.

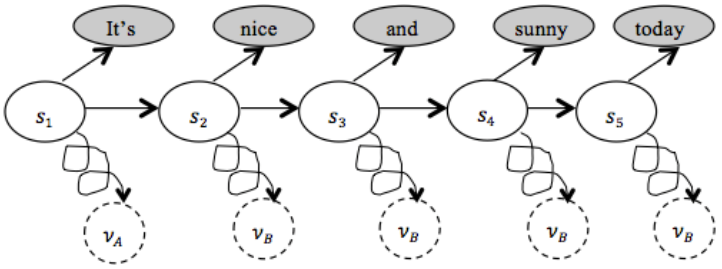

Fig. 1. Visualization of the HS-LDA generative process. Ovals $s_1,\ldots,s_5$ represent process states, shaded ovals represent word generation and dashed circles represent emissions of neutrinos ν of types A and B. Dashed circles surrounding neutrinos labels aim to emphasize the notion that neutrinos are assumed to be present but difficult to detect.

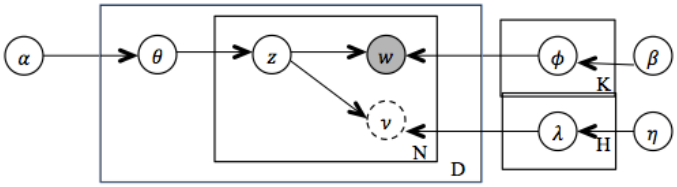

Fig. 2. Graphical model representation of HS-LDA. N is the number of words in a document, D is the number of documents, K is the number of topics and H is the number of pseudo-neutrino types. α, η and β are Dirichlet prior vectors that are assumed to be symmetrical in this paper. θ represents the vector multinomial over topics, ϕ is the multinomial over words, z is the topic draw, w stands for a word realization and ν is the emitted pseudo-neutrino. The clear circles represent hidden entities, shaded circles represent directly observable entities and the dashed circles stand for indirectly detectable ones.

In Figure 3, the generative process is outlined. There, words are generated in a way common to many LDA-type models by drawing from a distribution over words. Unlike other approaches, however, a pseudo-neutrino is also emitted by a draw from a multinomial distribution parameterized by a vector of topic-specific neutrino type proportions.

1. For each topic $k \in \{1, \ldots, K\}$:
2. Generate $\phi_k = \{\phi_{k,1}, \ldots, \phi_{k,V}\}^T \sim Dir(\cdot|\beta)$
3. Generate $\lambda_k = \{\lambda_{k,1}, \ldots, \lambda_{k,H}\}^T \sim Dir(\cdot|\eta)$
4. For each document d:
5. Generate $\theta^{(d)} \sim Dir(\cdot|\alpha)$
6. For each $i \in \{1, \ldots, N_d\}$
7. Generate $z_i \in \{1, \ldots, K\} \sim Mult(\cdot|\theta^{(d)})$
8. Generate $w_i \in \{1, \ldots, V\} \sim Mult(\cdot|\phi_{z_i})$
9. Generate $v_i \in \{1, \ldots, H\} \sim Mult(\cdot|\lambda_{z_i})$

Fig. 3. Generative process for HS-LDA: ϕ_k is a vector consisting of parameters for the multinomial distribution over words corresponding to *k*th topic, λ_k is a vector consisting of parameters for the multinomial distribution over neutrino types corresponding to *k*th topic, α is the Dirichlet document topic prior vector, β word prior vector, η is the neutrino type prior vector and N_d is the number or words in document d and K is the number of topics

It is important to note that if a user were to restrict the set of possible neutrino types to just a single type (say {"root"}), HS-LDA would become equivalent to LDA as all draws of type label assignments would be the same making the generative branch from z to v redundant. Therefore, HS-LDA is a generalization of Latent Dirichlet Allocations [1], which is important to note since the general nature of HS-LDA suggests that its insight can be applied to other models that extend LDA, of which there are many. Later sections will take advantage of this fact and show the experimental results of application of HS-LDA to other successful models.

3.1 Gibbs Sampling with HS-LDA

The generative probabilistic HS-LDA model describes the process of document collection creation. The hidden model parameters θ, ϕ and λ may be estimated using a Monte Carlo procedure, which is relatively easy to implement, does not require a lot of memory and produces output that is competitive with that of other more complicated and slower algorithms [3, 12]. The rest of the section describes the derivation of an efficient sampling algorithm used to infer models parameters with HS-LDA.

We start by framing the problem of topic discovery in terms of collections of D documents containing K topics expressed over W words and H pseudo-neutrino types. The task of learning topic models is to discover the makeup of θ, ϕ and λ, which can be estimated by evaluating the probability of a topic having observed both a word and a pseudo-neutrino. The posterior distribution is formally stated as $P(\boldsymbol{z}|\boldsymbol{w}, \boldsymbol{v}) = \frac{P(\boldsymbol{w},\boldsymbol{z},\boldsymbol{v})}{\sum_{\boldsymbol{z}} P(\boldsymbol{w},\boldsymbol{z},\boldsymbol{v})}$

The joint distribution $P(\boldsymbol{w}, \boldsymbol{\nu}, \boldsymbol{z})$ can be computed by considering that Dirichlet priors α, β and η in the HS-LDA model are conjugate to θ, ϕ and λ respectively. Since $P(\boldsymbol{w}, \boldsymbol{\nu}, \boldsymbol{z}) = P(\boldsymbol{w}|\boldsymbol{\nu}, \boldsymbol{z})P(\boldsymbol{\nu}|\boldsymbol{z})P(\boldsymbol{z})$ by the chain rule and since $\boldsymbol{w}$ and $\boldsymbol{\nu}$ are conditionally independent in our model (see Figure 2), $P(\boldsymbol{w}|\boldsymbol{\nu}, \boldsymbol{z}) = P(\boldsymbol{w}|\boldsymbol{z})$ which simplifies the joint distribution as

$$P(\boldsymbol{w}, \boldsymbol{\nu}, \boldsymbol{z}) = P(\boldsymbol{w}|\boldsymbol{z})P(\boldsymbol{\nu}|\boldsymbol{z})P(\boldsymbol{z})$$

Observing that ϕ, λ and θ only appear in first, second and third terms respectively, each term may be evaluated separately. Integrating out ϕ, λ and θ in each term gives

$$P(\boldsymbol{w}|\boldsymbol{z}) = \left(\frac{\Gamma(W\beta)}{\Gamma(\beta)^W}\right)^K \prod_{j=1}^K \left(\frac{\prod_w \Gamma\left(n_j^{(w)}+\beta\right)}{\Gamma\left(n_j^{(\cdot)}+W\beta\right)}\right) \tag{1a}$$

$$P(\boldsymbol{\nu}|\boldsymbol{z}) = \left(\frac{\Gamma(H\eta)}{\Gamma(\eta)^H}\right)^K \prod_{j=1}^K \left(\frac{\prod_\nu \Gamma\left(n_j^{(\nu)}+\eta\right)}{\Gamma\left(n_j^{(\cdot)}+H\eta\right)}\right) \tag{1b}$$

$$P(\boldsymbol{z}) = \left(\frac{\Gamma(K\alpha)}{\Gamma(a)^K}\right)^D \prod_{d=1}^D \left(\frac{\prod_j \Gamma\left(n_j^{(d)}+\alpha\right)}{\Gamma\left(n_\cdot^{(d)}+K\alpha\right)}\right) \tag{1c}$$

where $n_j^{(w)}$ is the number of times word w has been assigned to topic j, $n_j^{(d)}$ is the number of time a word from document d has been assigned to topic j, $n_j^{(\nu)}$ is the number of times a neutrino of type ν has been assigned to topic j, $n_j^{(\cdot)}$ and $n_\cdot^{(d)}$ are the total numbers of assignments in topic j and document d respectively. $\Gamma(\cdot)$ is the standard gamma function.

Since computing the exact distributions in Equations 1a-c is intractable [1, 12], we follow the pattern in other topic modeling approaches [3, 4, 6-8, 13] and estimate θ, ϕ and λ by relying on the Gibbs sampling procedure. The Gibbs procedure operates by iteratively sampling all variables from their distributions conditioned on their current values and data and updating variables for each new state. The full conditional distribution $P(z_i = j|\boldsymbol{z}_{-i}, \boldsymbol{w}, \boldsymbol{\nu})$ that is necessary for the Gibbs sampling algorithm is obtained by probabilistic argument [12] as well as by observing that first terms in each of the Equations 1a-c are constant and values of denominators and numerators of second terms are proportional to the arguments of their gamma functions. Therefore, the sampling equation is as follows:

$$P(z_i = j|\boldsymbol{z}_{-i}, \boldsymbol{w}, \boldsymbol{\nu}) \propto \frac{n_{-i,j}^{(w_i)}+\beta}{n_{-i,j}^{(\cdot)}+W\beta} \frac{n_{-i,j}^{(d)}+\alpha}{n_{-i,\cdot}^{(d)}+K\alpha} \frac{n_{-i,j}^{(\nu_i)}+\eta}{n_{-i,j}^{(\cdot)}+H\eta} \tag{2}$$

where, $n_{-i,j}^{(\nu_i)}$ is the count of times neutrino ν_i has been assigned to topic j excluding current assignment and $n_{-i,j}^{(\cdot)}$ is the total number of topics j assignments in any document excluding current assignment. Reader may notice that denominators in the first and third product terms in Equation 2 have identical counters. That is because, in the HS-LDA model, the number of words is always exactly the same as the number of neutrino emissions by process construction.

The Gibbs sampling algorithm can be implemented in an on-line fashion by first initializing topic assignments to a random state and then using Equation 2 to assign words to topics. The algorithm operates by reconsidering data for a number of iterations during which new states of topic assignments are found using Equation 2. The algorithm is fast as the only information necessary to estimate the new state is the word, topic and neutrino counters, which can be cached and updated efficiently [12].

3.2 Pseudo-Neutrino Detection

The sampling algorithm outlined in the previous section estimates parameter values by relying on two detectable quantities – words and pseudo-neutrino emissions. To detect pseudo-neutrinos, which cannot be observed directly in text, we assumed a Gaussian distribution of pseudo-neutrinos in documents, as this distribution was common to many phenomena [14]. With this assumption, we could refer to all pseudo-neutrinos in a given document in a meaningful way by identifying the most common (or mean) neutrino type. That is, for $H \in \mathbb{Z}^+$ possible pseudo-neutrino types, we assumed that there existed a mean pseudo-neutrino type $1 \leq c_\mu^d \leq H$ for each document d. With that, a rough approximation vector of pseudo-neutrino assignments $h_d = \{h_{d,1}, \dots, h_{d,H}\}$ could be constructed for each document d of size N_d such that $h_{d,i} = \begin{cases} N_d, if\ i = c_\mu^d \\ 0, otherwise \end{cases}$.

Constructing the vector h_d as described in the previous paragraph suggested that a meaningful approximation of document pseudo-neutrinos could be found by identifying a representative (mean) neutrino type for each document. To locate the representative flavor, we noticed that pseudo-neutrino types essentially constituted a kind of vocabulary akin to that of words. With that, considering topics from conceptual point of view, intuitively, documents on the same topic would be close to one another in terms of similarity of their content regardless of the vocabulary used to express the content (e.g. for any language, documents about the 'World Cup' sporting event would contain text related to the even in that language). With that, since the number of pseudo-neutrino types was known, clustering documents into H clusters based on word similarity would approximate document-level (mean) neutrino types as cluster indices could be used as the neutrino type identifiers.

To implement this intuition in practice, we searched for a clustering strategy that would perform in a scalable way while at the same time ensuring that similar documents were likely to share a cluster. We realized that by restricting $H = 2^n$ for some positive integer n, it would be possible to make use of Locality Sensitive Hashing (LSH)[2].

LSH relies on existence of a set of hash functions $\mathcal{H}$ (referred to as a *function family*) for some *d*-dimensional coordinate space $\mathbb{R}^d$ where each hash function can be efficiently implemented with the help of Random Projections (RP) [15]. To use LSH, we start by defining a function space $f: \mathbb{R}^d \to \{0,1\}$ and constructing a function family $\mathcal{H} = \{f_1, \dots, f_{\log_2 H} | f_i \in f\}$. Each function $f_i \in \mathcal{H}$ is associated with a random projection vector $p_i^{\text{random}} \in \mathbb{R}^d$ with components that are selected at random from a Gaussian distribution $\mathcal{N}(0,1)$ [16]. Each random projection is used to compute a dot-product between it and any point $p \in \mathbb{R}^d$ allowing the mapping function to be constructed in the following way:

$$f_i(p) = \begin{cases} 1\ if\ p \cdot p_i^{\text{random}} \geq 0 \\ 0\ if\ p \cdot p_i^{\text{random}} < 0 \end{cases} \text{ [2]}$$

Then, for any $p \in \mathbb{R}^d$, LSH hash value is constructed by invoking each of the functions in $\mathcal{H}$ on p and concatenating output bits as a bit string. Treating the bit

string as a binary number, a mapping function assigns p to a number between one and H as follows:

$$map(p) = ||_{i=1}^{|\mathcal{H}|} f_i(p)$$

Since the bit string generated by the above procedure is of finite size, the space of possible values is bound by $2^{|\mathcal{H}|}$. Recalling that $H = 2^n$ and $|\mathcal{H}| = \log_2 H = n$, function map can be used to map each point in $\mathbb{R}^d$ to a positive integer bound by H.

Further, since it is proven in [17] (proof omitted here) that $\mathrm{P}(f_i(p) = f_i(q)) = 1 - \frac{\angle(p,q)}{\pi}$ holds for any function $f_i \in \mathcal{H}$ and all points $p, q \in \mathbb{R}^d$, the probability of LSH hash collision for two vectors increases with the decrease to the angle between them. Then, since the value of cosine of two vectors is directly related to the size of the angle

$$\mathrm{P}(f_i(p) = f_i(q)) \propto \cos(\angle(p,q))$$

where $\angle$ is the angle between the two vectors in radians[1].

Therefore, since LSH hashing allowed for fast clustering of vectors in a way that preserved document similarity, LSH was used to approximate the mean pseudo-neutrino type by treating LSH hash value as the type identifier. To make use of LSH hashing in topic modeling, we restricted the size of the set H to be a power of two and rewrote the sampling equation (Equation 2) in terms of LSH hash family F of size $\log_2 H$ as:

$$P(z_i = j|\boldsymbol{z}_{-i}, \boldsymbol{w}, \boldsymbol{x}) \propto \frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{(\cdot)} + W\beta} \frac{n_{-i,j}^{(d)} + \alpha}{n_{-i,\cdot}^{(d)} + K\alpha} \frac{n_{-i,j}^{(h_d^F)} + \eta}{n_{-i,j}^{(\cdot)} + H\eta} \tag{3}$$

where h_d^F is the hash value of document d, $n_{-i,j}^{(h_d^F)}$ is the number of words from documents with hash value h_d^F assigned to topic j excluding current assignment and $n_{-i,j}^{(\cdot)}$ is the total number of words in any document assigned to topic j excluding current assignment. The sampling algorithm, then, proceeds as outlined in section 3.1 using Equation 3 to assign words to topics.

4 Evaluation

In order to validate the utility of our model, the approach was tested on two distinct data sets. Our first data set consisted of 1,000,000 English language messages collected from Twitter micro-blog site using its public sampling API over a period of one week. The second data set was comprised of 300,000 English language chatroom messages collected by connecting to the public *irc.freenode.net* public chat server and monitoring chat rooms with more than 150 chatters for the same one week period. Filtering of non-English texts was accomplished with the help of the open source *language-detection*[2] library.

The language models produced by our approach were compared to those learned by On-line LDA [3] and SparseLDA [4] as these models were designed to operate efficiently on stream data. In addition, to provide a common baseline, topic models

[1] Unusual angle operator used to avoid confusion with topic modeling notation.

[2] `https://code.google.com/p/language-detection/`

learned by HS-LDA were compared to those discovered by the classic LDA [1] algorithm. We did not evaluate our approach against TM-LDA as it required partitioning by author as well as a significant and static training sample to be collected prior to producing any output at all. These constraining requirements made TM-LDA unfit for continuous topic modeling application, which was the motivation of this work.

To compare language models, evaluation was performed using the perplexity measure over held-out subset of data $\overline{W} = \{\overline{w}_1, \dots, \overline{w}_n\}$ given language model M and the training data calculating perplexity using the following formula:

$$perp_M(\overline{W}) = \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\frac{1}{|\overline{w}_i|}\sum_{j=1}^{|\overline{w}_i|}\log\left(p_m(\overline{w}_{ij})\right)\right)$$

where $n = |\overline{W}|$, $\overline{w} \in \overline{W}$, $\overline{w}$ is the jth term in the ith string in the held-out collection and $p_M(w \in \overline{w})$ is the probability of term w as per the learned language model M. Further, to account for possible overfitting, our evaluation was validated using the 5-fold cross-validation.

4.1 Parameter Selection

As pointed out in earlier works [10, 18] Locality Sensitive Hashing is highly sensitive to choices of the hash family size. This choice governs the scatter within each hash bucket as chance of collision decreases with the increase of hash family size. Therefore, hash family size selection was approached from the point of view of estimating a reasonable number of buckets for the number of messages expected.

Considering the Twitter micro-blog service as being one of the most vibrant and popular social forums today, we experimented with the numbers of English language messages that could be downloaded over a given period. Recalling the industry-oriented motivation for this work and selecting one working week as the target period (timeframe common to the industry environment) the number of messages that could be gathered from Twitter's sampling service was empirically estimated to number in some millions. Realizing that if the number of hash family function was chosen to be high (ex.: $2^{20} = 1{,}048{,}576$) the algorithm could potentially map every message into an individual bucket, negating the entire insight of HS-LDA. With that, the reasonable number of hash functions for our experiments was chosen to be 17 ($2^{17} = 131{,}072$) as this number would allow for variability within each cluster while at the same time providing reasonable specificity.

4.2 Experimental Setup and Results

Having thus chosen the hash family size, HS-LDA was evaluated against LDA, Online LDA and Sparse LDA using the two test datasets. For all models, the number of topics was chosen to be 100 and experimented with various hyperparameter settings. Results reported here were for hyperparameter values of $\alpha = 0.05$, $\beta = 0.05$ and $\eta = 1$ as these values produced best results for all models.

Figure 4 shows perplexity results for the two test datasets. In order to provide a readable graphic, the Simple Moving Average (SMA) smoothing technique was applied to raw results, setting the moving average window set to 10,000.

While cross-model comparison shows that HS-LDA approach outperformed other models in terms of perplexity, performance of LDA-type models could be sensitive to parameter choices [19]. Since some parameter choices could be more beneficial to performances of some frameworks and less so for others and since all models used for evaluation were derivative of the classic LDA model, we applied the insight of the HS-LDA approach to each test algorithm and conducted a pairwise comparison in terms of perplexity, thus controlling for model parameter sensitivity. Figures 5-6 show pairwise comparisons for each test model with the same approach augmented with HS-LDA (LDA/HS-LDA pairwise comparison is not reported as it can be found in Figure 4).

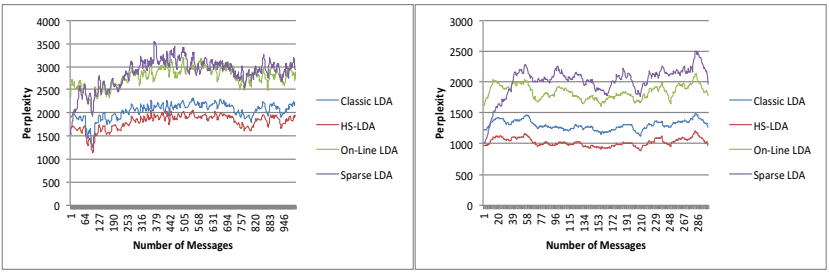

Fig. 4. Smoothed perplexity results for Twitter (left) and IRC (right) dataset

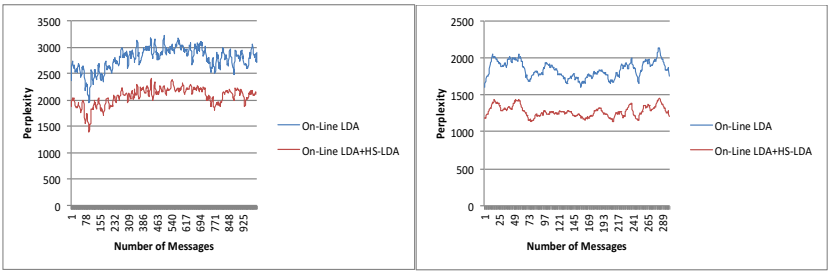

Fig. 5. Pairwise comparison of On-Line LDA and On-Line LDA augmented with HS-LDA for Twitter (left) and IRC (right) test sets

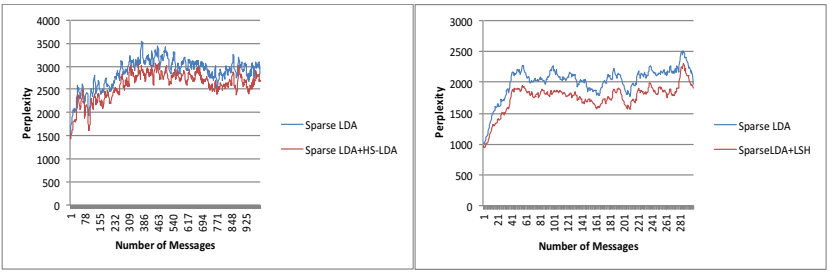

Fig. 6. Pairwise comparison of Sparse LDA and Sparse LDA augmented with HS-LDA for Twitter (left) and IRC (right) test sets

To summarize results in numerical way, average perplexities are reported for all tested models in Table 1. The purpose of this report is to identify the model with the highest predictive prowess as well as to quantify amount of improvement in terms of percentages.

Table 1. Average perplexity results for Twitter and IRC datasets

Model	**Average Perplexity (Twitter)**	**Average Perplexity (IRC)**
LDA	2044.42	1300.92
On-Line LDA	2773.99	1835.74
Sparse LDA	2860.27	1998.53
HS-LDA	**1803.67**	**1023.12**

In Table 1, HS-LDA outperformed other models by at least approximately 12% for the Twitter dataset and 21% for the IRC chatroom data. Significantly better predictive power of resulting topic models learned from the chatroom discourse may be explained by noting that chatrooms are often oriented towards particular themes, thus introducing loose structuring to social discourse. Such structuring does not exist in Twitter where the discourse is entirely unstructured, making the job of theme discovery more difficult.

5 Conclusions and Future Work

To improve the quality of topic models learned from social media streams, we introduced the new HS-LDA model for topic modeling, which was a generalization of the well-known LDA topic discovery technique. We experimented on large data sets collected from popular social media services and showed that our model outperformed other state-of-the-art stream topic modeling techniques in all cases. Further, we enhanced other topic modeling approaches with the insight of HS-LDA and showed that applying core notions of HS-LDA to other techniques improves their performance in terms of predictive power of resulting topic models.

While our results showed improvement in all cases where HS-LDA insight was used, combining HS-LDA with other models aimed at preserving global context did not immediately result in substantial performance gains. It seems, however, that such a combination has merit and we will continue this investigation in the future work.

Further, while this work was instrumental in moving towards the goal of constructing an industry-grade stream topic monitoring system, one of the major hurdles for constructing such a system with HS-LDA was the necessity to specify the number of topics. In our future work, we plan to investigate topic modeling approaches based on the popular Chinese Restaurant Process paradigm and will attempt to apply the insight of HS-LDA to dynamically discovered topic allocations.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022 (2003)
2. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 604–613. ACM (1998)
3. AlSumait, L., Barbará, D., Domeniconi, C.: On-line LDA: Adaptive Topic Models for Mining Text Streams with Applications to Topic Detection and Tracking. In: ICDM, pp. 3–12. IEEE Computer Society (2008)
4. Yao, L., Mimno, D., McCallum, A.: Efficient methods for topic model inference on streaming document collections. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 937–946. ACM (2009)
5. Hong, L., Davison, B.D.: Empirical study of topic modeling in Twitter. In: Proceedings of the First Workshop on Social Media Analytics, pp. 80–88. ACM (2010)
6. Xu, Z., Lu, R., Xiang, L., Yang, Q.: Discovering User Interest on Twitter with a Modified Author-Topic Model. In: 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), pp. 422–429 (2011)
7. Wang, Y., Agichtein, E., Benzi, M.: TM-LDA: efficient online modeling of latent topic transitions in social media. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 123–131. ACM (2012)
8. Rosen-Zvi, M., Griffiths, T., Steyvers, M., Smyth, P.: The author-topic model for authors and documents. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, pp. 487–494. AUAI Press (2004)
9. Hoffman, M.D., Blei, D.M., Bach, F.: Online learning for latent dirichlet allocation. In: NIPS (2010)
10. Petrović, S., Osborne, M., Lavrenko, V.: Streaming first story detection with application to Twitter. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 181–189. Association for Computational Linguistics (2010)
11. Wang, K.C.: A Suggestion on the Detection of the Neutrino. Phys. Rev. 61, 97 (1942)
12. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proceedings of the National Academy of Sciences 101, 5228–5235 (2004)
13. Kim, H., Sun, Y., Hockenmaier, J., Han, J.: ETM: Entity Topic Models for Mining Documents Associated with Entities. In: ICDM 2012, pp. 349–358 (2012)
14. Patel, J.K., Read, C.B.: Handbook of the normal distribution. Marcel Dekker Inc. (1996)
15. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 245–250. ACM (2001)
16. Slaney, M., Casey, M.: Locality-Sensitive Hashing for Finding Nearest Neighbors [Lecture Notes]. IEEE Signal Processing Magazine 25, 128–131 (2008)
17. Ravichandran, D., Pantel, P., Hovy, E.: Randomized algorithms and NLP: using locality sensitive hash function for high speed noun clustering. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 622–629. Association for Computational Linguistics (2005)
18. Ture, F., Elsayed, T., Lin, J.: No free lunch: brute force vs. locality-sensitive hashing for cross-lingual pairwise similarity. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 943–952. ACM (2011)
19. Panichella, A., Dit, B., Oliveto, R., Di Penta, M., Poshyvanyk, D., De Lucia, A.: How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In: Proceedings of the 2013 International Conference on Software Engineering, pp. 522–531. IEEE Press (2013)