Larger Residuals, Less Work: Active Document Scheduling for Latent Dirichlet Allocation

Mirwaes Wahabzada* and Kristian Kersting*

Knowledge Discovery Department, Fraunhofer IAIS
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
firstname.lastname@iais.fraunhofer.de

Abstract. Recently, there have been considerable advances in fast inference for latent Dirichlet allocation (LDA). In particular, stochastic optimization of the variational Bayes (VB) objective function with a natural gradient step was proved to converge and able to process massive document collections. To reduce noise in the gradient estimation, it considers multiple documents chosen uniformly at random. While it is widely recognized that the scheduling of documents in stochastic optimization may have significant consequences, this issue remains largely unexplored. In this work, we address this issue. Specifically, we propose residual LDA, a novel, easy-to-implement, LDA approach that schedules documents in an informed way. Intuitively, in each iteration, residual LDA actively selects documents that exert a disproportionately large influence on the current residual to compute the next update. On several real-world datasets, including 3M articles from Wikipedia, we demonstrate that residual LDA can handily analyze massive document collections and find topic models as good or better than those found with batch VB and randomly scheduled VB, and significantly faster.

1 Introduction

Latent Dirichlet allocation (LDA) has recently become very popular due to its effectiveness at extracting low-dimensional representations from sparse high-dimensional data, with numerous applications in areas such as text analysis and computer vision [1]. Unfortunately, fitting a LDA topic model given a set of training documents requires approximate inference techniques that are computationally expensive. This makes it challenging to apply LDA to large-scale document collections that nowadays become increasingly common. Such datasets originate for example from online books at Google or image collections at Flickr. And as storage capacity continues to expand, today's "large" is certainly tomorrow's "medium" and next week's "small". For instance International Data Corporation[1] (IDC), a consultancy, has estimated that in August 2010 the amount of information available on the Internet did surpass the barrier of $1\text{ZB} = 10^{21}\text{B}$.

* Both authors contributed equally.
[1] Mar 2011, http://www.idc.com/getdoc.jsp?containerId=prUS22723811

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 475–490, 2011.

According to Eric Schmidt, CEO of Google, this amount currently grows at a rate of 2.5EB = $2.5 \cdot 10^{18}$B per day. Most of these data consist of user generated content such as videos, photos, blogs, or reviews. Apparently, processing such large-scale document collections opens up completely new and interesting applications for machine learning techniques in general and LDA in particular.

A promising approach to scaling LDA to large data sets are online variants, see e.g. [23,2,11,16] and references in there, that incrementally build topic models when a new document (or a set of documents) appears. For instance, Hoffman *et al.* [11] presented an online variational Bayes (VB) algorithm for LDA based on online stochastic optimization with a natural gradient step that can easily analyze massive document collections. In addition to providing a solution to the problem of growing document collections, online algorithms also open up different avenues for parallelization of inference from batch algorithms, providing ways to draw on the enhanced computing power of multiprocessor systems, and different tradeoffs in runtime and performance from other algorithms; another common approach to scaling LDA, see e.g. [18,15,22] and reference in there. Here, we explore another avenue opened up by online LDA algorithms, namely, to revisit batch LDA and ask the question whether we can improve it by viewing it as a quasi-online approach that processes documents respectively mini-batches one at a time? Somewhat surprisingly, there has been virtually no attempt to study the question of determining a good order for documents to be processed. While it is widely recognized that the scheduling of documents in stochastic optimization of LDA topic models may have significant consequences, this issue remains largely unexplored. Instead, practitioners schedule documents essentially uniformly at random, perhaps due to ease of implementation, and to the lack of clear guidelines on scheduling the documents. In this work, we address the question of how to schedule documents and show that convergence can be reached faster.

Specifically, triggered by recent results on randomized low-rank matrix factorization approaches [7,5,19,14] — they essentially randomly sample columns, i.e., documents according to a probability distribution that depends on the Euclidean norms of those documents — it was recently proposed to schedule documents that exert a disproportionately large influence on the topics of the corpus before less influential documents. Naively instantiating this idea results in a fix schedule: sort documents randomly biased towards those ones with higher norms. Then, we simply run online LDA following this fix schedule. In fact, this influence scheduled LDA (isLDA) can already result in considerable efficiency gains. However, as we will show in this paper, we can do considerably better. While isLDA is indeed a valid approach, it suffers from one drawback. Since it starts optimizing long before having seen the entire document collection even once, it tends to overfit on documents with many words; they are seen much earlier than documents with few words. Consequently, isLDA has to repair for this bias by several passes over the entire dataset. We therefore propose a novel schedule that overcomes this drawback. Intuitively, we schedule documents dynamically in each pass over the entire document collection according to a probability dis-

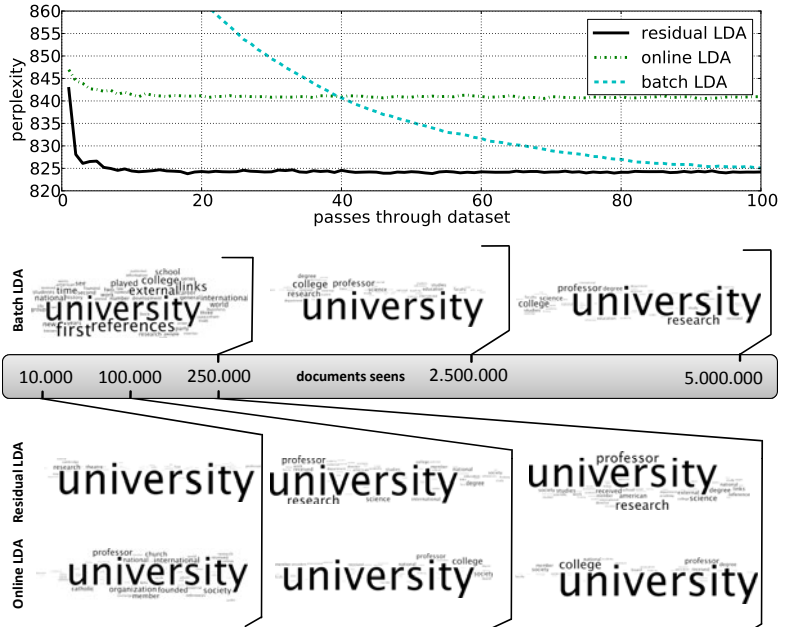


Fig. 1. Residual LDA can handily analyze massive document collections and finds topics better than those found with batch LDA and online LDA, and significantly faster: (Top) Held-out perplexity (the lower, the better) results on 3M Wikipedia articles as a function of full passes through the dataset. (Bottom) Evolution of the topic "university research" (bottom) after having seen 10K, 100K, and 250K (residual and online LDA) respectively 250K, 2.5m, 5m articles (batch LDA) on 250K Wikipedia. The 50 most likely words are shown as word cloud. The word size is proportional to the probability. Residual LDA is qualitatively more similar to batch LDA than online LDA is, university, professor and research prominently appear in both of them. This is also supported by smaller Hellinger distance. (Best viewed in color)

tribution that depends on the Euclidean norms of the current residuals of the documents. As we will show, this dynamic schedule can actually be turned into an active schedule: in each iteration we only consider a small, informative subset of the documents sampled according to the residual distribution. This is sensible because, if we overfit on the current active set of documents, the residuals of the currently inactive documents will get larger. Moreover, it establishes a novel link between LDA and active learning, see [17] for an overview. Whereas our active schedule is unsupervised, most active learning approaches make use of label information. Even recent active clustering approaches, see e.g. [21], typically provide semi-supervision in terms of must-link and cannot-link constraints. Closest in spirit is probably early work on active data sampling [13]. Actually, as we will point out in the conclusions, there is also a close connection to search distributions for optimization [24].

We evaluated the resulting LDA, called *residual LDA*, both qualitatively and quantitatively using several benchmark datasets, including 3M articles from Wikipedia. The experimental results demonstrate that — as can also be seen in Fig. 1 — residual LDA can handily analyze massive document collections and find topic models as good or better than those found with randomly and informatively scheduled LDA, and significantly faster. In other words "larger residuals, less work", which also explains the title of the paper.

We proceed as follows. After reviewing LDA based on variational Bayes, we motivate residual LDA by drawing connections between LDA and low-rank matrix factorization. Based on this connection, we then introduce residual LDA and devise a active schedule that prefers documents with high influence on the gradient over less influence ones. Before concluding, we present our experimental evaluation.

2 Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) is a Bayesian probabilistic model of collections of text documents [1]. It assumes a fixed number of K underlying topics in a document collection D. Topics are assumed to be drawn from a Dirichlet distribution, $\beta_k \sim Dir(\eta)$, which is a convenient conjugate to the multinomial distribution of words appearing in documents. According to LDA, documents are generated by first drawing topic proportions according to $\theta_d \sim Dir(\alpha)$, where α is the parameter of the Dirichlet prior on the per-document topic distributions. Then for each word i a topic is chosen according to $z_{di} \sim Mult(\theta_d)$ and the observed word w_{di} is drawn from the selected topic, $w_{di} \sim Mult(\beta_{z_{di}})$. We assume symmetric priors for θ and β but asymmetric ones are possible, see e.g. [20]. The true posterior distribution can not be computed directly and is usually approximated using Markov chain monte carlo (MCMC) or variational inference. In this paper, we focus on variational Bayesian (VB) inference. Here, the true posterior is approximated using a simpler, fully factorized distribution q. Following Blei *et al.* and Hoffman *et al.* [1,11], we choose $q(z, \theta, \beta)$ of the form $q(z_{di} = k) = \phi_{dw_{di}k}$, $q(\theta_d) = Dir(\theta_d, \gamma_d)$, and $q(\beta_k) = Dir(\beta_k, \lambda_k)$. The posterior over the per-word topic assignments z is parameterized by ϕ, the posterior over the per-document topic weights θ is parameterized by γ, and the posterior over the topics β is parameterized by λ. These variational parameters are optimized to maximize the Evidence Lower BOund (ELBO) $\log p(w \mid \alpha, \eta) \geq \mathcal{L}(w, \phi, \gamma, \lambda) \triangleq$

$$\mathbb{E}_q[\log p(w, z, \theta, \beta \mid \alpha, \eta)] - \mathbb{E}_q[\log q(z, \theta, \beta)], \tag{1}$$

which is equivalent to minimizing the Kullback-Leiber divergence between $q(z, \theta, \beta)$ and the true posterior $p(z, \theta, \beta \mid w, \alpha, \eta)$. It can be shown that Eq. (1) factorizes to a summation over documents d:

$$\mathcal{L}(w, \phi, \gamma, \lambda) = \sum\nolimits_d \Big\{ \mathbb{E}_q[\log p(w_d \mid \theta_d, z_d, \beta)] + \mathbb{E}_q[\log p(z_d \mid \theta_d)] - \mathbb{E}_q[\log q(z_d)] + \mathbb{E}_q[\log p(\theta_d \mid \alpha)] - \mathbb{E}_q[\log q(\theta_d)] + (\mathbb{E}_q[\log p(\beta \mid \eta)] - \mathbb{E}_q[\log q(\beta)]) / \mathbf{D} \Big\}.$$

Algorithm 1. A single iteration of Variatonal Bayes for LDA (vbLDA). Setting $\rho_t = 1$ and $D = \tilde{D}$ corresponds to running batch LDA

Input: $\tilde{D}$, ρ_t
Output: $\boldsymbol{\lambda}$,$\boldsymbol{\gamma}_{\tilde{D}}$
foreach *document* d *in* $\tilde{D}$ **do**
  Initialize $\gamma_{dk} = 1$;
  `/* The const. is arbitrary */`
  repeat
    Set $\phi_{dwk} \propto \exp\{\mathbb{E}_q[\log \theta_{dk}] + \mathbb{E}_q[\log \beta_{kw}]\}$;
    Set $\gamma_{dk} = \alpha + \sum_w \phi_{swk} n_{dw}$;
  until $\frac{1}{K}\sum_k |change\ in\ \gamma_{dk}| < 0.00001$;
`/* Compute M step */`
Compute $\tilde{\lambda}_{kw} = \eta + \frac{\mathbf{D}}{|\tilde{D}|}\sum_{d\in\tilde{D}} n_{dw}\phi_{dwk}$;
Set $\boldsymbol{\lambda} = (1-\rho_t)\boldsymbol{\lambda} + \rho_t\tilde{\boldsymbol{\lambda}}$;

When using VB (as opposed to MCMC) documents can be summarized by their word counts, i.e., $\mathcal{L}(w,\phi,\gamma,\lambda) \triangleq \sum_{d=1}^{\mathbf{D}} \ell(n_d,\phi_d,\gamma_d,\lambda)$, where $\mathbf{D}$ denotes the number of documents, n_d the word count vector and $\ell(n_d,\phi_d,\gamma_d,\lambda)$ the contribution of document d to the ELBO. Now, $\mathcal{L}$ can be optimized using coordinate ascent over the variational parameters ϕ,γ,λ (see [1,11] for more details),

$$\phi_{dwk} \propto \exp\{\mathbb{E}_q[\log\theta_{dk}] + \mathbb{E}_q[\log\beta_{kw}]\},$$
$$\gamma_{dk} = \alpha + \sum\nolimits_w n_{dw}\phi_{dwk}, \text{ and } \lambda_{kw} = \eta + \sum\nolimits_d n_{dw}\phi_{dwk}, \tag{2}$$

iteratively optimizing each variational parameter to increase the objective. The conditional expectations under q of $\log\theta$ and $\log\beta$ are

$$\mathbb{E}_q[\log\theta_{dk}] = \Psi(\gamma_{dk}) - \Psi\left(\sum_{i=1}^{K}\gamma_{ki}\right) \text{ and } \mathbb{E}_q[\log\beta_{kw}] = \Psi(\lambda_{kw}) - \Psi\left(\sum_{i=1}^{W}\lambda_{di}\right),$$

where W is the size of the vocabulary, and Ψ denotes the digamma function. The updates in (2) are guaranteed to converge to a stationary point of the ELBO. By analogy to the EM algorithm, we can partition these updates into an *E-step* - iteratively updating γ and ϕ until convergence, holding λ fixed - and an *M-step* - updating λ given ϕ. In practice, this algorithm converges to a better solution if we reinitialize γ and ϕ before each *E-step*.

Based on VB, Hoffman *et al.* [11] have introduced an online variant that we here present for the batch case running over mini-batches (chunks of multiple observations) as summarized in Alg. 2. That is, we assume that the corpus of documents has been sorted according to some schedule, i.e. permutation $\pi(i)$ and chunked into l mini-batches $B_1, B_2, \ldots, B_l$ of size S with $B_i = d_{(i-1)\cdot S+1}, d_{(i-1)\cdot S+2}, \ldots, d_{i\cdot S}\}$ (w.l.o.g, we assume $n = l\cdot S$). That is, the ELBO $\mathcal{L}$ is set to maximize $\mathcal{L}(w,\phi,\gamma,\lambda) \triangleq \sum_{B_i}\sum_{d\in B_i}\ell(n_d,\phi_d(n_d,\lambda),\gamma_d(n_d,\lambda),\lambda)$. For each mini-batch, we perform an *E step* to find locally optimal values of γ_t

Algorithm 2. Online LDA (oLDA)

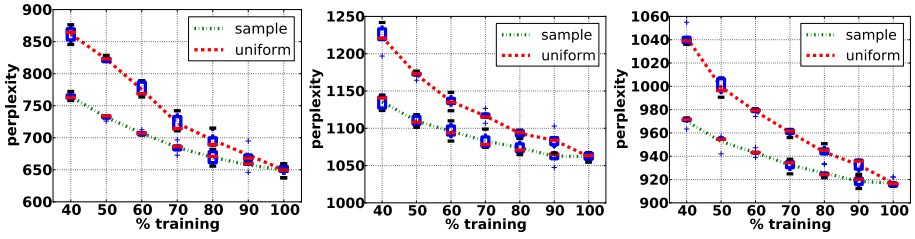
Input: D (documents), S (batchsize)
Define $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$ with $\kappa \in (0.5, 1]$;
Initialize $\boldsymbol{\lambda}$ randomly and set $t = 0$;
repeat
  Select S documents randomly forming the mini-batch $\tilde{D}$;
  Compute $(\boldsymbol{\lambda}, \boldsymbol{\gamma}_{\tilde{D}}) = \text{vbLDA}(\tilde{D}, \boldsymbol{\lambda}, \rho_t)$ using Alg. 1;
  Increment $t := t + 1$;
until *converged* ;

and ϕ_t, holding λ fixed. In the following *M step*, we compute $\tilde{\lambda}$, the setting of λ that would be optimal (given ϕ_t) if the entire corpus consisted only of repetitions of the batch B_i. λ is updated through a weighted average of its previous value and $\tilde{\lambda}$ computed in the current *M step*. The rate of change ρ_t is set to $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$. Here, $\tau_0 \geq 0$ slows down early iterations of the algorithm, and $\kappa \in (0.5, 1]$ controls the influence of old values of $\tilde{\lambda}$ and ensures convergence.

As Hoffman *et al.* [11] have shown this mini-batch VB-LDA corresponds to a stochastic natural gradient algorithm on the variational objective $\mathcal{L}$. Using mini-batches reduces the noise in the stochastic gradient estimation as we consider multiple observations per update, i.e., $\tilde{\lambda}_{kw} = \eta + \mathbf{D}/S \sum_{s \in B_i} n_{sw} \phi_{skw}$ where n_{sw} is the s-th document in the i-th mini-batch. The variational parameters ϕ_{ts} and γ_{ts} for this document are fit with a normal E step. Note that we recover batch VB when we set $\pi(i) = i$, the batch size to $S = \mathbf{D}$, and $\kappa = 0$. The benefits of this LDA approach are manifold, see [11]. It empirically converges faster than batch collapsed Gibbs sampling. It does not require a full pass through the entire corpus in order to compute an update. Instead, it makes an update per mini-batch and in turn can be quite fast when applying to large datasets. Furthermore, it converges as long as the expected number of times we see each document is the same for each document.

3 isLDA ≈ Random Matrix Factorization

It is known that LDA and its precursor probabilistic latent semantic analysis (pLSA) [12] are closely related. In particular, one can show that pLSA is tantamount to LDA with a uniform prior [10]. Given this connection, one can also establish a relation between LDA and certain settings of low-rank matrix factorization. Specifically, Gaussier and Goutte [8] and Ding *et al.* [4] have noted that pLSA correspond to specific instances of the problem of non-negative matrix factorization. pLSA can thus be reduced to a low-rank matrix factorization problem. Consequently, LDA topic models can be determined using algorithmic approaches that differ from the conventional idea of expectation maximization, MCMC, and VB. In particular, randomized matrix factorization approaches become applicable to LDA as well. For instance, Frieze *et al.* [7] introduced a sampling approach, in which the rows of a matrix are picked with probabilities proportional to their squared lengths. This and similar randomized matrix



(a) **WebKB** (b) **N20** (c) **Wiki 25K**

Fig. 2. More influence, lower perplexity; lowest perplexity when using the full training set: Held-out perplexity results (the lower, the better) for different proportions of the original dataset. We selected $40\%, 50\%, \ldots, 100\%$ documents and ran LDA for $K = 100$. Results are averaged over five reruns. (Best viewed in color)

factorization approaches have been proved to approximately minimize the reconstruction error in terms of the Frobenius norm and be successful in several tasks and applications, see also e.g [5,19,14].

Given the link between low-rank matrix factorization and LDA, it is natural to ask "Can we improve LDA by adapting techniques developed for matrix factorization?" Recently, we and colleagues have shown that this is indeed the case. A common randomized matrix factorization approach, see e.g. [5,19], is to approximate a given matrix A by S rescaled rows/columns sampled from A. To do so, we compute an importance score for each row, and sample rows using that score as an importance sampling probability distribution. A common score for matrix factorization is

$$p(i) = \sum_j n_{ij}^2 / \sum_{i,j} n_{ij}^2 \ , \tag{3}$$

and the rescaling factor is $1/\sqrt{p(i) \cdot S}$. Thus, this importance score depends on the whole corpus and intuitively captures the "influence" of a given document on the LDA topic model. By preferentially choosing documents that exert a disproportionately large influence on the topic model, we expect to capture the important part of a given corpus at hand. Fig. 2 illustrates that this is actually the case. Using the probability (3) as importance score yields significantly lower perplexities than selecting documents uniformly at random when selecting only $40\%, 50\%, \ldots, 100\%$ documents of the original corpus. In other words, the way we schedule documents (and in turn the way we build mini-batches) can make a crucial difference on how well and long mini-batch LDA takes to converge.

Based on this insight, we can devise a naive "fix" schedule for LDA that is easy-to-implement. Essentially, we apply the importance sampling procedure to LDA. However, whereas the randomized matrix factorization approaches sample a subset of documents with replacement, we keep all documents exactly once. Consequently, we compute a schedule by sampling all documents without replacement biased towards those ones with higher norms. In other words, the documents with higher importance score will have higher chance to be processed

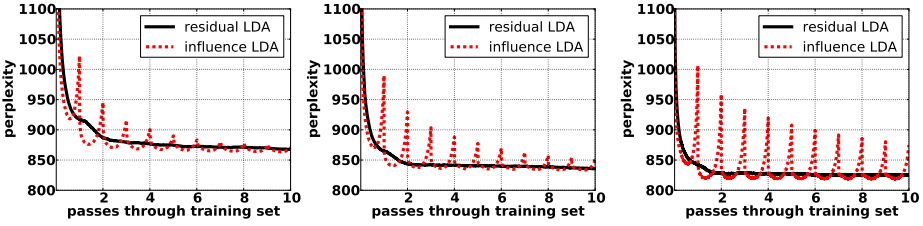


Fig. 3. Residual LDA avoids overfitting on large documents. Held-out perplexity results (for $K = 100$) as a function of passes through training set on Wikipedia (from left to right 50K, 250K, and 3M documents). The dashed line denotes the performance of isLDA, the solid lines that of residual LDA. (Best viewed in color)

earlier. We and colleaguescalled the resulting LDA — compute $p(i)$ as in Eq. (3), rescale and shuffle the documents according to π, build mini-batches as described earlier, and run Alg. 1 — *influence scheduled* LDA (isLDA).

However, while isLDA is indeed a valid approach and provably converges, it suffers from one major drawback. Since isLDA starts optimizing long before having seen the entire document collection even once, it tends to overfit on documents with many words. They are seen much earlier than the documents with few words only. Consequently, isLDA has to repair for this bias by several passes over the entire dataset. This is illustrated in Fig. 3. As one can see, the held-out perplexity drops after each full pass over the entire document collection. During one pass over the entire document collection, however, influence LDA overfits on the large documents. One is tempted to simply use the first half of the data set only since the minimal held-out perplexity per pass is achieved at this point. This, however, is not sensible. Recall the results presented in Fig. 2. They illustrate that omitting least important documents actually decreases the overall performance. Only when the entire document collection is potentially considered for training, the best overall performance is achieved.

In other words, isLDA is simply too static. The fix schedule does not allow to change the order of documents processed during learning, although the impact of a document on the LDA topic model is indeed constantly changing. A dynamic schedule would be more sensible. How do we realize this? Intuitively, we schedule documents in each pass over the entire document collection according to a probability distribution that depends on the Euclidean norms of the residuals of those documents. This is not only sensible but actually can also be turned into an active schedule: in each iteration we only consider a small, informative subset of the documents sampled according to the residual distribution. Indeed, we may overfit on the current active documents. However, their importance score will drop, and the residuals of the currently inactive documents will increase. That is, their importance score increases and so does the probability of selecting an inactive document and making it active. This is the basic idea underlying residual LDA that we will now introduce.

4 Residual Latent Dirichlet Allocation

Intuitively, we want to select documents which are most informative with respect to the variational bound. So, what is the impact of a single document on the current variational bound? How do we compute this per document influence?

Recall that VB uses coordinate ascent to maximize the objective function. That is, each variational parameter $\boldsymbol{\gamma}$, $\boldsymbol{\phi}$, respectively $\boldsymbol{\lambda}$ is varied while all others are held constant. Since we are only interested in the influence of a single document d on the variational bound, however, it is sufficient to consider $\boldsymbol{\phi}$ only. That is, we view the "per document d influence" as the change of the variational parameter $\boldsymbol{\phi}$ for each word w in d $\tilde{\nabla}_d^t := \sum_k \sum_w n_{dw} \mid \phi_{dwk}^t - \phi_{dwk}^{t-1} \mid$. Here, ϕ_{wk}^t is the optimized parameter at iteration t representing the probability that w is generated by topic k, and n_{dw} represents the number of times w appears in d.

This importance measure essentially needs $\mathcal{O}(U \cdot K \cdot \mathbf{D})$ storage effort with $U = \max_d |w_d|$. We have to store the values $\boldsymbol{\phi}$ of all unique words in document d times the number of topics K. This is essentially intractable for large document collections. To restrict the overall complexity, we therefore summarize $\boldsymbol{\phi}$ for each topic in a document. More formally, using the partial derivative of the Eq. (1) w.r.t. γ_{dk} it is shown that it yields a maximum at $\gamma_{dk} = \alpha + \sum_w n_{dw}\phi_{dwk}$ (see e.g. [1]). Additionally applying the triangle inequality, we arrive at

$$\begin{aligned}\tilde{\nabla}_d^t &= \sum_k \sum_w \mid n_{dw}\phi_{dwk}^t - n_{dw}\phi_{dwk}^{t-1} \mid \geq \sum_k \mid \sum_w n_{dw}\phi_{dwk}^t - \sum_w n_{dw}\phi_{dwk}^{t-1} \mid \\ &= \sum_k \mid \alpha + \sum_w n_{dw}\phi_{dwk}^t - (\alpha + \sum_w n_{dw}\phi_{dwk}^{t-1}) \mid = \sum_k \mid \gamma_{dk}^t - \gamma_{dk}^{t-1} \mid =: \nabla_d^t \,. \quad (4)\end{aligned}$$

This measure is intuitive for finding high impact documents, since the documents that have the highest ∇_d^t are the ones we are most uncertain about. Unfortunately, it does not consider the expected reduction or growth in uncertainty. Therefore, we instead search for highly informative documents by maximizing the information gained about ∇_d^t. More formally, the influence $\xi(d)$ of a document d is $\xi(d) = \max\{c\ , \nabla_d^t - \nabla_d^{t-1}\}$. Here, $c > 0 \in \mathbb{R}$ is some small constant that prevents the influence to become too small or even zero. Now, following Eq. 3, the importance score $\mathbf{p}(d)$ of the document d is: $\mathbf{p}(d) = \xi(d)^2 / \sum_i \xi(i)^2$ Proceeding as for LDA — run Alg. 1 computing $p(i)$ using this new influence score, rescaling, and shuffling the documents according to π in each pass over the entire document collection — essentially yields *residual LDA*.

However, we can do better. The dynamic schedule can actually be turned into an active schedule: in each iteration we only consider a small, informative subset of the documents sampled according to the residual distribution. This is sensible because, if we overfit on the current active set of documents, the residuals of the currently inactive documents will get larger. In turn, the probability that they will be selected and become active in the next iteration increases. However, how should we initialize respectively update the values $\boldsymbol{\xi}$ for previously unseen documents? Assuming that the expected variational parameters are uniform for

Algorithm 3. Residual LDA

Input: D (documents), S (batchsize)
1 Define $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$ with $\kappa \in (0.5, 1]$;
2 Initialize $\boldsymbol{\lambda}$ randomly, set $t := 0$ and $\sigma = 0$;
3 **foreach** $d \in D$ **do**
4 $\quad \xi^2(d) := (\sum_w n_{dw})^2$;
5 $\quad \sigma := \sigma + \xi^2(d)$;
6 Set $c = \min\{\xi(d)\}$;
7 **repeat**
`/* Update document importance scores */`
8 **foreach** $d \in D$ **do**
9 $\quad \mathbf{p}(d) := \sigma^{-1} \cdot \xi^2(d)$;
10 Sample S active documents $\tilde{D}$ according to $\mathbf{p}$;
`/* Run one iteration of variational inference (Alg. 1) */`
11 $(\boldsymbol{\lambda}, \boldsymbol{\gamma}_{\tilde{D}}) := \text{vbLDA}(\tilde{D}, \boldsymbol{\lambda}, \rho_t)$;
`/* Update document influence for the active documents in` $\tilde{D}$ `*/`
12 **foreach** $\tilde{d} \in \tilde{D}$ **do**
13 $\quad \nabla^t_{\tilde{d}} := \sum_k \mid \gamma^t_{\tilde{d}k} - \gamma^{t-1}_{\tilde{d}k} \mid$;
14 $\quad \sigma := \sigma - \xi^2(\tilde{d})$;
15 $\quad \xi^2(\tilde{d}) := (\max(c, \nabla^t_{\tilde{d}} - \nabla^{t-1}_{\tilde{d}}))^2$;
16 $\quad \sigma := \sigma + \xi^2(\tilde{d})$;
17 $\quad$ Increment $t := t + 1$;
18 **until** *converged* ;

documents not seen yet, i.e., $\phi_{dwk} = K^{-1}$, Eq. (4) simplifies as follows: $\xi(d)_{init} =$

$$\sum_k \mid \gamma_{dk} - 0 \mid -0 = \sum_k \gamma_{dk} = \sum_k (\alpha + \sum_w n_{dw}\phi_{dwk}) = K\alpha + \sum_w n_{dw} \; .$$

Thus, $\xi(d)_{init}$ can essentially be set to the number of words in d, i.e. $\sum_w n_{dw}$.

The overall approach is summarized in Alg. 3. We start off by initializing the influence scores of all documents (lines 3-5). Then, in each iteration, we compute $\mathbf{p}(d)$ (lines 8-9), sample S documents according to $\mathbf{p}(d)$ (line 10), run one iteration of variational inference to the active mini-batch (line 11), update the influence scores of the active documents (line 12-16), and iterate until convergence. For efficiency reasons, we can switch to online LDA in later iterations, i.e., we avoid computing any schedule. We can do so because the gradients will eventually get very similar. When this happens, residual LDA essentially mimics online LDA. To decide when to switch, we can simply check whether the entropy of importance scores drops below some threshold ϵ. If $\epsilon \geq \mid \sum_d \mathbf{p}(d) \log \mathbf{p}(d) - \log \frac{1}{\mathbf{D}} \mid$, we switch to online LDA.

5 Experimental Evaluation

Our intention here is to investigate the following questions: **(Q1)** Can residual LDA be faster than batch LDA, online LDA and isLDA? **(Q2)** If so, does residual

LDA find solutions that are as good as batch LDA? **(Q3)** Does residual LDA scale better than online LDA and isLDA in terms of number of documents resp. the size of the vocabulary?

To this aim, we implemented residual LDA, batch LDA and isLDA in Python based on Hoffman *et al.*'s [11] Python code[2] for online LDA. We evaluated the performance of the methods on several datasets where **D** denotes the number of documents, **W** the number of unique words, and **N** the number of terms. The **WebKB** dataset consists of webpages of various universities with four different categories (student, course, faculty, project). We used the dataset provided by Ana Cardoso-Cachopo[3] with $\mathbf{D} = 3869$ and $\mathbf{N} = 217671$. We chose a vocabulary of $\mathbf{W} = 3000$ unique words consisting of the terms with the highest TFIDF (term frequency inverse document frequency). The Reuters dataset **R8**, a classic corpus for text classification algorithms, contains newswire articles. We used the version provided by Ana Cardoso-Cachopo[3] with $\mathbf{D} = 5378$ and $\mathbf{N} = 234650$. As for **WebKB**, we chose a vocabulary of $\mathbf{W} = 3000$ unique words consisting of the terms with the highest TFIDF. Finally, we also used the 20-newsgroups dataset **20N** as in [9][4] with $\mathbf{D} = 18576$, $\mathbf{N} = 1847456$, and $\mathbf{W} = 10000$. From all corpora, we removed documents with less than six words.

For evaluation we computed perplexity on the held-out test sets to measure a model's ability to generalize to unseen data. Perplexity is a common criterion of clustering quality that does not require a priori categorizations and thus often used in the context of topic modelling. For a corpus of test documents $D^{test} = \{\boldsymbol{w}_d\}$, perplexity is the reciprocal geometric mean of the likelihood of this corpus given the model:

$$\mathit{perplexity}\left(D^{test}\right) = \exp\left\{-\frac{\sum_d \log p(\boldsymbol{w}_d)}{\sum_d N_d}\right\} .$$

Additionally, we evaluated all approaches in a classification setting for **WebKB**, **R8**, and **N20** . Specifically, we used the 7 first-level classes for **N20**, and all classes for **WebKB** and **R8**. Then, we used a multi-class linear support vector machine[5] to predict the class labels merely using the topic distributions of the documents as estimated by the learned LDA models. We report on the average accuracy achieved in a 5-fold cross-validation.

For the scaling experiment (**Q3**), we crawled our own **Wikipedia** (english) corpus of $\mathbf{D} = 2914700$ randomly selected documents with $\mathbf{N} = 292941239$ and a fixed vocabulary of $\mathbf{W} = 7686$ words. We processed the crawled Wikipedia articles as done in [11] removing alls words from their vocabulary that did not appear in our articles. Additionally, we used a subset of **Grolier** encyclopedia articles, provided by Sam Roweis[6] with $\mathbf{D} = 30991$ documents, $\mathbf{N} = 3484393$ and a vocabulary of $\mathbf{W} = 15276$ most common words, and the **NYTimes** news

[2] `http://www.cs.princeton.edu/~mdhoffma`
[3] `http://web.ist.utl.pt/~acardoso/datasets/`
[4] `http://www.kyb.tuebingen.mpg.de/bs/people/pgehler/rap/`
[5] We used PyML `http://pyml.sourceforge.net/` with the default settings.
[6] `http://www.cs.nyu.edu/~roweis/data.html`

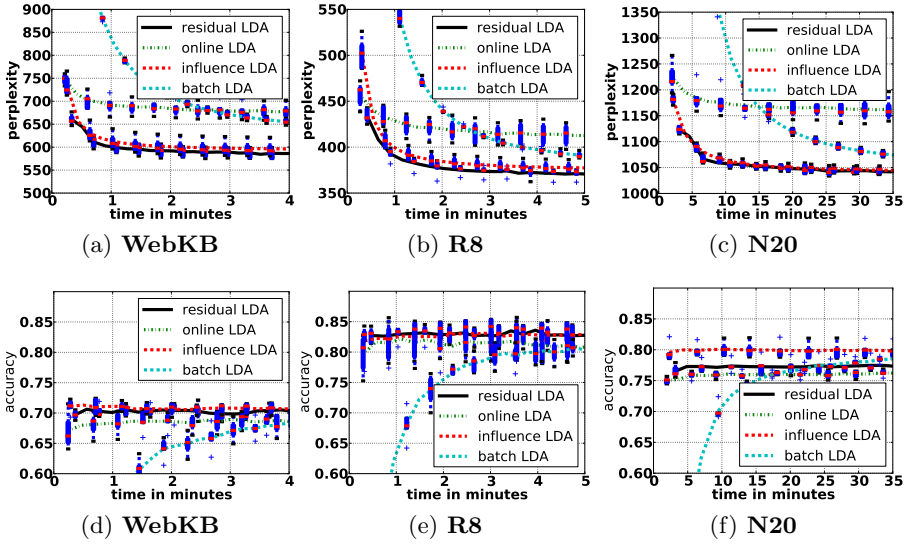


Fig. 4. Small and medium scale experiments: Residual LDA outperforms batch and online LDA and is comparable to influence LDA. (a-c) Held-out perplexity (the lower, the better) as a function of CPU time (for K = 100). (d-e) Classification accuracy (the higher the better) as a function of CPU time taken to learn the low dimensional representation using different methods. (Best viewed in color)

articles provided by [6] with $\mathbf{D} = 300000$ documents with $\mathbf{N} = 99542125$ and a vocabulary of $\mathbf{W} = 102660$ words. For **NYTimes**, after stoppword removal the vocabulary was reduced by keeping just words which appeared more than ten times, and additionally we excluded all multi token phrases. Again, we removed all articles with less than six words.

For all perplexity experiments we used 1000 documents as held-out test set for **Wikipedia** and **NYTimes**, and 500 documents for the remaining datasets. For all experiments, we set κ close to 0.5 as suggested in [11] and $\tau_0 = 4$ (determined by cross-validation on the training set of a subset of 25K webpages from **Wikipedia** and **N20** but used for all experiments). To set the batch size we used the following heuristic: $B = \frac{\mathbf{D} \cdot \| D \|_2}{\sum_i \| d_i \|_2}$, where $\| d_i \|_2$ resp. $\| D \|_2$ denotes the Frobenius norm of document i resp. the whole corpus. Intuitively, it measures how many documents are required to capture the important part of a corpus. We used fixed symmetric hyperparameters $\alpha = 0.01$ and $\eta = 0.01$ in all our experiments.

Q1, Q2: Small and Medium Datasets: The perplexity results for the small and medium datasets are summarized in Fig. 4 (a-c). In each experiment we computed perplexity after each time **D** documents were seen. Here, residual LDA finds sollutions in the range of influence scheduled LDA, and batch LDA's solution but with much less computation. Compared to online LDA on those

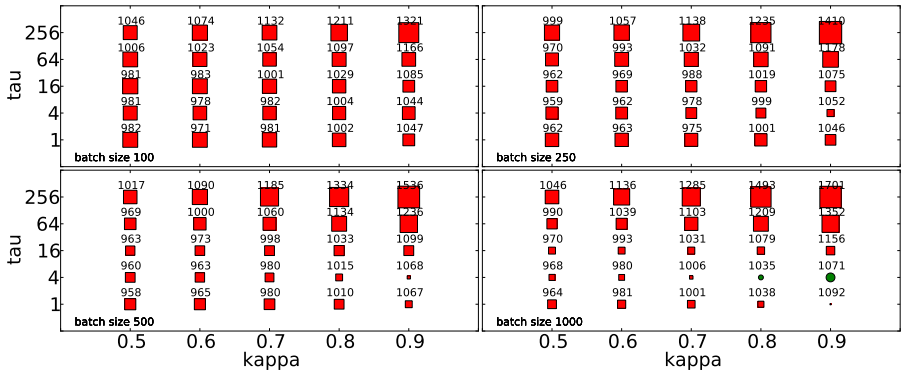
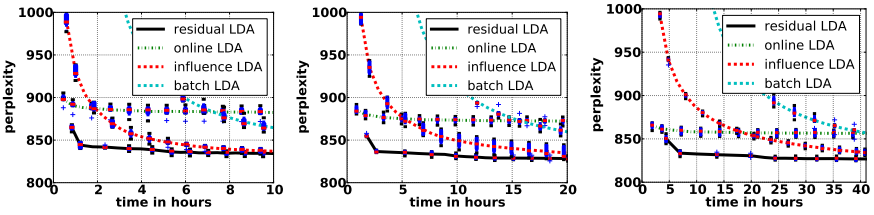


Fig. 5. Stability analysis: Hinton diagrams of held-out perplexity for different settings of the parameters $\tau_0 \in \{1, 4, 16, 64, 25\}$, $\kappa \in \{0.5, 0.6, \ldots, 0.9\}$ and $B \in \{100, 250, 500, 1000\}$ for a subset of 25K Wiki articles on a validation set of 1000 articles. Residual and online LDA were ran for 30 minutes. The achieved perplexity results were averaged over 5 reruns. The size of markers denote the difference in perplexity. Red squares indicate that residual LDA achieved lower perplexity, green circles that online LDA was the winner. The numbers are the absolute perplexities of the winners. Residual LDA is the winner in 98 of 100 parameter settings. (Best viewed in color)

datasets, residual LDA produces lower perplexity, i.e. it achieves significantly better performance. Here, we compared also residual LDA against online LDA on a subset of 25K Wikipedia articles for different parameter settings, as shown in Fig. 5 (we used 100 different parameter settings, for a total number of 500 runs). As one can see, the performance of residual LDA is more stable, and in 98% of cases better (red squares) than online LDA's perplexity (green circles).

Another point is, there is some question as to the meaningfulness of perplexity as a metric for comparing different topic models [3]. The accuracies of the classification experiments as summarized in Fig. 4 (d-e) provide a different metric. The results clearly show that residual LDA and isLDA yield predictive accuracies in the range of the batch model's with much less computation. They even outperform online LDA.

Q3: Scaling Experiments on Wikipedia: The results on the small datasets suggest that residual LDA and influence scheduled LDA are faster than batch LDA. To further investigate how they scale, we ran experiments on several Wikipedia corpora ranging from 250K to 1M documents. The results are summarized in Fig. 6. As one can see, isLDA indeed takes longer to converge than residual LDA. In contrast, residual LDA can handily analyze massive datasets and it can find topic models as good or better than those found with batch LDA or online LDA, and it converges much faster than isLDA. It is essentially always converged after the kink, which happens after about $2 \cdot \mathbf{D}$ documents were seen (again, here we measured the perplexity after each time $\mathbf{D}$ documents were seen). Also on the 3M Wikipedia dataset, see Fig. 1 (top), it produced better



(a) **Wiki 250K** (b) **Wiki 500K** (c) **Wiki 1M**

Fig. 6. Large scale experiments: Residual LDA outperforms all other methods. Held-out perplexity (the lower, the better) as a function of CPU time (for K = 100) for subsets of Wikipedia with different numbers **D** of articles. Results are averaged over five reruns. Residual LDA is essentially converged after the kink in its curve, which happens after about $2 \cdot \mathbf{D}$ documents were seen. (Best viewed in color)

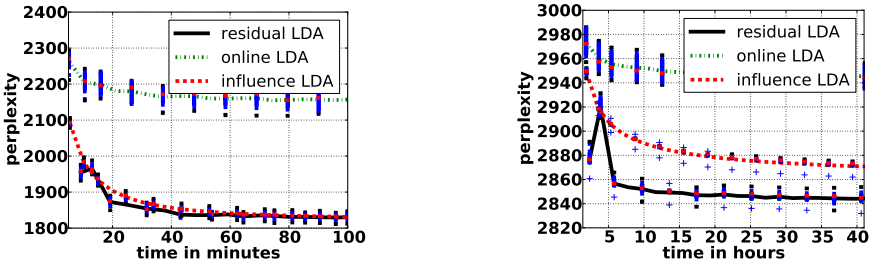
topics (in terms of perplexity) as those found by online LDA and batch LDA, and significantly faster. The topic evolution is shown in Fig. 1 (bottom) and illustrates that residual LDA's topic "university research" is qualitatively as good as the one found by batch LDA, but with a fraction of documents seen. Online LDA only starts to get similar to batch LDA. This is also supported by a smaller Hellinger distance to the batch LDA topic: residual LDA 0.12 vs. online LDA 0.53. Finally, to test the scaling behaviour with respect to both the vocabulary and dataset size, we ran experiments on **Grolier** and **NYTimes**. The results are summarized in Fig. 7. As one can see, residual LDA can handily analyse large corpora with large vocabulary size. For the smaller **Grolier**, residual LDA outperforms online LDA and is comparable to influence LDA. For the larger **NYTimes**, it outperforms online and influence LDA.

Thus, putting all experimental results together, we can clearly answer questions **Q1**-**Q3** affirmatively.

6 Conclusions

Triggered by the recent success stories of online LDA approach for the problem of inferring topics in growing document collections, we revisited batch LDA. We turned batch LDA into a quasi-online LDA approach that actively forms mini-batches of highly influential documents, called residual LDA. Specifically, residual LDA actively selects a subset of documents that exert a disproportionately large influence on the current residual to compute the next update. On several real-world datasets, including 3M articles from Wikipedia, we demonstrated that residual LDA can handily analyze massive document collections and find topic models as good or better than those found with batch VB and randomly scheduled VB, and significantly faster. In other words, *large residual* means *less work*.

Indeed, much remains to be done. One interesting avenue for future work is to employ influence schedules in parallel LDA approaches. Another is to discard



(a) **Grolier** (b) **NYTimes**

Fig. 7. Residual LDA can handily analyse large corpora with large vocabulary size. Held-out perplexity (the lower, the better) as a function of CPU time (for K = 100) for **Grolier** and **NYTimes** datasets. Results are averaged over five reruns. For the smaller dataset residual LDA outperforms online LDA and is comparable to influence LDA. For the larger dataset, residual LDA outperforms both. (Best viewed in color)

documents during learning (based on inference). Most interesting, however, is to develop strong theoretical guarantees for residual LDA and to transfer them to the general stochastic optimization case. That this is possible shows a connection to the recent work of Yi *et al.* [24]. We assume a fitness function and want to maximize the expected fitness under the search distribution. Following Yi *et al.*, we assume that gradient entries are Gaussian distributed and compute the covariance over all gradients. Assuming a uniform fitness over gradients and computing the gradient with the steepest ascent of the expected fitness, Yi *et al.*'s results show that the gradient of the likelihood of the search direction is the most influential one. This measure, however, depends on the norm of the corresponding gradient, which is exactly what residual LDA employs.

Acknowledgements. The authors thank the anonymous reviewers for their comments and their colleagues Christian Bauckhage and Anja Pilz for valuable discussions. They also thank Anja Pilz for crawling the Wikipedia dataset and Matt Hoffman for making the online LDA Python code publically available. MW and KK were supported by the Fraunhofer ATTRACT fellowship STREAM.

References

1. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
2. Canini, K., Shi, L., Griffiths, T.: Online inference of topics with latent dirichlet allocation. In: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, AISTATS 2009 (2009)
3. Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., Blei, D.: Reading tea leaves: How humans interpret topic models. In: Proceeding of NIPS (2009)
4. Ding, C., Li, T., Peng, W.: NMF and PLSI: Equivalence and a Hybrid Algorithm. In: Proc. SIGIR (2006)

5. Drineas, P., Kannan, R., Mahoney, M.: Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. SIAM Journal of Computing 36, 184–206 (2006)
6. Frank, A., Asuncion, A.: UCI machine learning repository (2010), http://archive.ics.uci.edu/ml
7. Frieze, A., Kannan, R., Vempala, S.: Fast monte-carlo algorithms for finding lowrank approximations. Journal of the ACM 51(6), 1025–1041 (2004)
8. Gaussier, E., Goutte, C.: Relations between PLSA and NMF and Implications. In: Proc. SIGIR (2005)
9. Gehler, P., Holub, A., Welling, M.: The rate adapting poisson model for information retrieval and object recognition. In: Proceedings of ICML, pp. 337–344 (2006)
10. Girolami, M., Kaban, A.: On an Equivalence between PLSI and LDA. In: Proc. SIGIR (2003)
11. Hoffman, M., Blei, D., Bach, F.: Online learning for latent dirichlet allocation. In: Proceedings of Neural Information Processing Systems (NIPS 2010) (2010)
12. Hofmann, T.: Probabilistic latent semantic indexing. Research and Development in Information Retrieval, pp. 50–57 (1999)
13. Hofmann, T., Buhmann, J.: Active data clustering. In: Proceedings of NIPS (1997)
14. Mahoney, M., Drineas, P.: Cur matrix decompositions for improved data analysis. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 106(3), 697–703 (2009)
15. Newman, D., Asuncion, A., Smyth, P., Welling, M.: Distributed algorithms for topic models. Journal of Machine Learning Research 10, 1801–1828 (2009)
16. Sato, I., Kurihara, K., Nakagawa, H.: Deterministic single-pass algorithm for lda. In: Proceedings of Neural Information Processing Systems, NIPS 2010 (2010)
17. Settles, B.: Active learning literature survey. Tech. Rep. 1648, University of Wisconsin-Madison (2010)
18. Smola, A., Narayanamurthy, S.: An architecture for parallel topic models. PVLDB 3(1), 703–710 (2010)
19. Sun, J., Xie, Y., Zhang, H., Faloutsos, C.: Less is more: Sparse graph mining with compact matrix decomposition. Statistical Analysis and Data Mining 1(1), 6–22 (2008)
20. Wallach, H., Mimno, D., McCallum, A.: Rethinking lda: Why priors matter. In: Advances in Neural Information Processing Systems, vol. 22, pp. 1973–1981 (2009)
21. Wang, X., Davidson, I.: Active spectral clustering. In: Proceedings of the IEEE International Conference on Data Mining (ICDM 2010) (2010)
22. Yan, F., Xu, N., Qi, Y.: Parallel inference for latent dirichlet allocation on graphics processing units. In: Proceedings of NIPS (2009)
23. Yao, L., Mimno, D., McCallum, A.: Efficient methods for topic model inference on streaming document collections. In: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 937–946 (2009)
24. Yi, S., Wierstra, D., Schaul, T., Schmidhuber, J.: Stochastic search using the natural gradient. In: Proceedings of ICML, p. 146 (2009)