More Influence Means Less Work: Fast Latent Dirichlet Allocation by Influence Scheduling

Mirwaes Wahabzada, Kristian Kersting, Anja Pilz, Christian Bauckhage
Fraunhofer IAIS, Sankt Augustin, Germany
{firstname.lastname}@iais.fraunhofer.de

ABSTRACT

There have recently been considerable advances in fast inference for (online) latent Dirichlet allocation (LDA). While it is widely recognized that the scheduling of documents in stochastic optimization and in turn in LDA may have significant consequences, this issue remains largely unexplored. Instead, practitioners schedule documents essentially uniformly at random, due perhaps to ease of implementation, and to the lack of clear guidelines on scheduling the documents. In this work, we address this issue and propose to schedule documents for an update that exert a disproportionately large influence on the topics of the corpus before less influential ones. More precisely, we justify to sample documents randomly biased towards those ones with higher norms to form mini-batches. On several real-world datasets, including 3M articles from Wikipedia and 8M from PubMed, we demonstrate that the resulting influence scheduled LDA can handily analyze massive document collections and find topic models as good or better than those found with online LDA, often at a fraction of time.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Algorithms

Keywords

Latent Dirichlet Allocation, Stochastic Gradient

1. INTRODUCTION

LDA has recently become 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.

A promising approach to scaling LDA to large datasets are online variants, see e.g. [5] and references in there, that incrementally build topic models when a new document (or a set of documents) appears. Specifically, Hoffman *et al.* [5] 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. Here, we explore another avenue opened up by online LDA (oLDA) algorithms, namely, to view LDA as a search process and to ask the question whether we can improve it by scheduling documents respectively mini-batches for processing. 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, due perhaps 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.

2. ONLINE LDA

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. 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, $w_{di} \sim Mult(\beta_{z_{di}})$.

In this paper, we focus on variational Bayesian (VB) inference. Here, the true posterior is approximated using a simpler, fully factorized distribution q. Following [1, 5], 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 variational parameters ϕ, γ, and λ are optimized to maximize the Evidence

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

Lower BOund (ELBO) $\log p(w \mid \alpha, \eta) \geq \mathcal{L}(w, \phi, \gamma, \lambda) \triangleq \mathbb{E}_q[\log p(w, z, \theta, \beta \mid \alpha, \eta)] - \mathbb{E}_q[\log q(z, \theta, \beta)]$, which is equivalent to minimizing the Kullback - Leibler divergence between $q(z, \theta, \beta)$ and the true posterior $p(z, \theta, \beta \mid w, \alpha, \eta)$.

Based on VB, Hoffman *et al.* [5] have introduced an online variant that we here present for the batch case running over mini-batches (chunks of multiple observations). That is, we assume that the corpus of documents has been sorted according to some schedule, i.e. permutation π and chunked into l mini-batches $B_1, B_2, \ldots, B_l$ of size S. That is, the ELBO $\mathcal{L}$ is set to maximize $\mathcal{L}(w, \phi, \gamma, \lambda) \triangleq \sum_{B_i} \sum_{d \in B_i} \ell(n_d, \phi_d(n_d, \lambda), \gamma_d(n_d, \lambda), \lambda)$, where n_d is the word count vector and $\ell(n_d, \phi_d(n_d, \lambda), \gamma_d(n_d, \lambda), \lambda)$ denotes the contribution of document d to the ELBO. As Hoffman *et al.* [5] 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: $\tilde{\lambda}_{kw} = \eta + D/S \sum_{s \in B_i} n_{sw} \phi_{skw}$ where n_{sw} is the s-th document in the i-th mini-batch and D denote the number of documents.

3. INFLUENCE SCHEDULED (O)LDA

It is known that LDA and its precursor probabilistic latent semantic analysis (pLSA) are closely related. In particular, one can show that pLSA is tantamount to LDA with a uniform prior [4]. Moreover, it is well known that pLSA is deeply connected to instances of the problem of non-negative matrix factorization [2]. Putting both results together, there is a relation between LDA and certain settings of low-rank matrix factorization. Thus, it is natural to ask: "*Can we improve LDA by adapting techniques developed for matrix factorization?*" Here, we show that this is actually the case. Specifically, we utilize randomized matrix factorization approaches, see e.g. [3, 6]. That is, we 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 is $p(i) = \sum_j n_{ij}^2 / \sum_{i,j} n_{ij}^2$, and the rescaling factor is $1/\sqrt{p(i) \cdot S}$. 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.

Our key idea now is to apply the importance sampling procedure to LDA. However, whereas the randomized matrix factorization approaches sample a subset of documents with replacement, we want to keep all documents exactly once. Consequently, we schedule documents by sampling all documents with replacement biased towards those ones with higher norms. In other words, the documents with higher entry values will have higher chance to be processed earlier.

We compute a schedule, i.e., a permutation π of the document collection $\{d_1, d_2, \ldots, d_n\}$ by sweeping through the list of documents in order of increasing indices until all documents have been selected. To decide whether the current document d_i should be placed at position $\pi(idx)$ of the schedule π, we draw a random number in $[0, 1]$ and check whether it is larger than the influence score $p(i)$ of d_i. If that is not the case, we place d_i at position $\pi(idx)$ of the schedule π. Alternatively, we just sort the documents according to their norm, from large to small documents. Setting π to a random permutation would recover oLDA.

Putting everything together results in *influenced* scheduled LDA (isLDA): Compute $p(i)$, rescale and shuffle the documents according to π, and build mini-batches as described earlier — and then run online variational Bayes inference using these mini-batches. Because we only change the schedule in which documents are considered, we do not change the expected number of times a document is seen. In turn, the analysis of [5] carries over to isLDA: isLDA converges to a stationary point of the objective $\mathcal{L}(w, \phi, \gamma, \lambda)$.

As we will show now, influence scheduling can be seen as a stochastic search problem [7]. More formally, assume that we have a distribution $p(\mathbf{z}|\theta)$ over search directions, i.e., gradients parameterized by θ. In each iteration, we generate m samples of search directions $\mathbf{z}_1, \ldots, \mathbf{z}_m$ and use some fitness function to evaluate them — such as the contribution ℓ of document d to the ELBO — to evaluate and in turn to adjust the parameters θ of the search distribution.

Let $J(\theta)$ be the expected fitness under search distribution $p(\mathbf{z}|\theta)$, namely $J(\theta) = \int f(\mathbf{z}) p(\mathbf{z}|\theta) d\mathbf{z}$. We now want to adjust θ such that the expected fitness $J(\theta + \delta\theta)$ is increased. The most straightforward way to do this is to set $\delta\theta = \nabla_\theta J(\theta) = \int p(\mathbf{z}|\theta) \cdot (f(\mathbf{z}) \nabla_\theta \ln p(\mathbf{z}|\theta) d\mathbf{z})$. Using Monte Carlo, we can approximate the last term as $\nabla_\theta{}^s J(\theta) = (1/m) \sum_{i=1}^m f(\mathbf{z}_i) \nabla_\theta \ln p(\mathbf{z}_i|\theta)$. Apriori, i.e., when have not seen any document, it is natural to assume the fitness $f(\mathbf{z}_i)$ is identical for all search directions, for the ease of simplification say $f(\mathbf{z}_i) = 1$ for $i = 1, \ldots, m$. The last equation simplifies to $\nabla_\theta{}^s J(\theta) = (1/m) \sum_{i=1}^m \nabla_\theta \ln p(\mathbf{z}_i|\theta)$. So, what is $\nabla_\theta \ln p(\mathbf{z}_i|\theta)$? With a rough approximation, one can assume all the search directions are independent and identically distributed. The central limit theorem then gives $\mathbf{z} \sim \mathcal{N}(\mathbf{x}, (C/m))$ where $\mathbf{x}$ represents the mean and C is the covariance matrix of the search directions. Again, if we have not seen any document it is sensible to assume $\tilde{C}$ is the identity matrix. Now, it is easy to show that $\nabla_\mathbf{x} \ln p(\mathbf{z}|\theta) = C^{-1}(\mathbf{z} - \mathbf{x}) = \mathbf{z} - \mathbf{x}$.

What do we gain by this? It allows us to see isLDA as an offline, greedy search learning approach and in turn that it has close links to well-known active learning approaches. Specifically, we have a set of i.i.d. documents simultaneously available. Then, isLDA queries documents so as to attempt to improve the ELBO as much as possible without computing the ELBO. Actually, isLDA takes a myopic approach that greedily chooses the next query based on this criterion. To see this, assume that the expected variational parameters are uniform for documents not seen, i.e., $\phi_{dwk} = K^{-1}$. Now, the expected influence of an unseen document d to $\tilde{\lambda}$ is $\xi_d = \sum_k \sum_w n_{dw} \phi_{dwk} = \sum_k \sum_w n_{dw} K^{-1} = \sum_w n_{dw}$, i.e. the number of words in the document. This tells us that the document with the largest norm will have the greatest expected impact on the search distribution. The sampling implements a simple exploration and exploitation strategy. Because the norm of a document is not changing, we can stick to the corresponding schedule after having seen each document once and continue running batch LDA following the computed schedule. This essentially proves:

THEOREM 3.1. *isLDA is a myopic stochastic search policy assuming all documents are i.i.d. and uncorrelated.*

Moreover, we can generalise isLDA to the online case. Intu-

Algorithm 1 Influence Scheduled Online LDA (isoLDA)

Define $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$ with $\kappa \in (0.5, 1]$
Initialize $\boldsymbol{\lambda}$ randomly and set $t = 0$
repeat
  Randomly select a batch B of documents
  Compute a schedule π for B
  Sort the documents in B according to schedule π
  Build mini-batches $B_1, B_2, \ldots, B_l$ according to π
  for each batch B_i in turn **do**
    for each document s in B_i **do**
      Initialize $\gamma_{sk} = 1$. */*The const. is arbitrary*/*
      repeat
        Set $\phi_{swk} \propto \exp\{\mathbb{E}_q[\log \theta_{sk}] + \mathbb{E}_q[\log \beta_{kw}]\}$
        Set $\gamma_{sk} = \alpha + \sum_w \phi_{swk} n_{sw}$
      until $\frac{1}{K}\sum_k |\text{change in } \gamma_{sk}| < 0.00001$
    Compute $\tilde{\lambda}_{kw} = \eta + \frac{D}{S}\sum_{s \in B_i} n_{sw}\phi_{swk}$
    Set $\boldsymbol{\lambda} = (1 - \rho_t)\boldsymbol{\lambda} + \rho_t \tilde{\lambda}$
    Increment $t = t + 1$
until converged

itively, we draw a batch of documents at random, apply one iteration of isLDA on this batch, and iterate. That is, we draw again randomly a batch, apply one iteration of isLDA, and so on. This *influence scheduled oLDA* (isoLDA) is summarized in Alg. 1.

The benefits of isoLDA are manifold. In contrast to isLDA, it does not require a full pass through the entire corpus in order to schedule documents for an update. Instead, it realizes a dynamic scheduling by scheduling each random batch again and again. In turn, it can be quite fast and flexible when applying to massive datasets. Finally, it has a constant memory consumption and naturally applies to growing datasets. In a nutshell, it combines the benefits of isLDA and oLDA: faster convergence and better quality. And, since we sample batches uniformly at random, the analysis of [5] again carries over:

THEOREM 3.2. *isoLDA converges to a stationary point of the objective* $\mathcal{L}(w, \phi, \gamma, \lambda)$.

4. EXPERIMENTAL EVALUATION

Our intention here is to investigate the following questions: **Q1** Can isLDA be faster than batch LDA and oLDA on small and medium scale datasets? **Q2** If so, does isLDA find solutions that are as good as the ones of batch LDA and oLDA for small and medium scale datasets? **Q3** Does isLDA scale also well to large datasets? **Q4** If not, does isoLDA scale better than isLDA and oLDA?

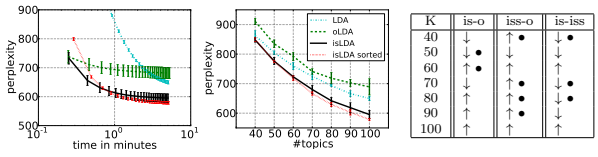
To this aim, we implemented batch LDA and all isLDA variants in Python based on Hoffman *et al.*'s [5] Python code[1] for oLDA and evaluated their performances on several datasets. Specifically, we considered the following datasets where **D** denotes the number of documents, **W** the number of unique words, and **N** the total number of words.

The **WebKB**[2] dataset consists of webpages of various universities with four different categories (student, course, faculty, project) with **D** = 3,869 and **N** = 217,671. We chose a vocabulary of **W** =3,000 unique words consisting of the terms with the highest TFIDF. Finally, we also used the 20-newsgroups[3] dataset (**20N**) with **D** = 18,576, **N** =1,847,456, and **W** = 10,000. From all corpora, we removed documents with less than six words. For the scaling experiment (**Q3**,

[1] `http://www.cs.princeton.edu/~mdhoffma`

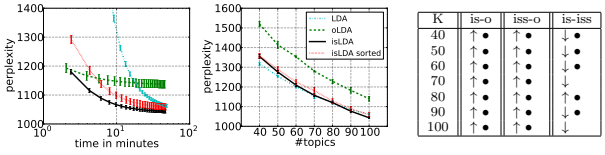
[2] `http://web.ist.utl.pt/~acardoso/datasets/`

[3] `http://www.kyb.tuebingen.mpg.de/bs/people/pgehler/rap/`



K	is-o	iss-o	is-iss
40	↓	↑ •	↓ •
50	↓ •	↓	↓
60	↑ •	↑	↑
70	↓	↑ •	↓ •
80	↑	↑ •	↓ •
90	↑	↑ •	↓
100	↑	↑	↑

(a) **WebKB**: Time - #Topics - Classification



K	is-o	iss-o	is-iss
40	↑ •	↑ •	↓ •
50	↑ •	↑ •	↓ •
60	↑ •	↑ •	↓ •
70	↑ •	↑ •	↓
80	↑ •	↑ •	↑ •
90	↑ •	↑ •	↓ •
100	↑ •	↑ •	↓

(b) **20N**: Time - #Topics - Classification

Figure 1: (Left and middle col.) Perplexity vs. CPU time (sec.; $K = 100$; in log scale) resp. the number of topics. (Right col.) Influence scheduling yields competitive classification performance. Here, is = isLDA, iss = isLDA (sorted), and o = oLDA. A ↑ (↓) denotes that the first (second) algorithm achieved a higher mean. A • denotes significant differences (paired t-test, $p = 0.05$). (Best viewed in color)

Q4), we crawled our own **Wiki**pedia (english) corpus of **D** = 3,295,656 documents with **N** ≈ 294,000,000 and a fixed vocabulary of **W** =7,686 words. We preprocessed the crawled Wikipedia articles as done in [5], using the provided positive-list vocabulary to remove terms not appearing in our articles. Additionally we used PubMed (**PM**) abstracts from the UCI repository with **D** = 8,200,000 documents with **N** ≈ 737,000,000 and a vocabulary of **W** =141,043 words. And finally, to compare the influence of different parameter settings, we use the NY Times (**NYT**) news articles from UCI with **D** = 300,000 documents with **N** ≈ 100,000,000 and a vocabulary of **W** = 102,660 words. For both (after stoppword removal), the vocabulary was reduced by keeping just words which appeared more than ten times, and additionally for **NYT** we excluded all multi token phrases.

For **WebKB** and **20N**, we held out 500 randomly selected documents for evaluation purposes; 1000 documents for the **Wiki**, **PM** and **NYT** datasets. On the test sets, we computed perplexity (the lower, the better) to measure the model's ability to generalise to unseen data [1]. Additionally, we evaluated all approaches in a classification setting for **WebKB** and **20N**. Specifically, we used the 7 first-level classes for **20N** and all classes for **WebKB**. Then, we used a multi-class linear support vector machine[4] 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 all experiments, we set κ close to 0.5 as suggested in [5] and $\tau_0 = 4$ for the small and medium corpora (determined by cross-validation on the training set of **20N** but used for all experiments). To set the mini-batch size for isLDA on the small datasets we used the following heuristic: $S = \frac{\mathbf{D} \cdot \|D\|_2}{\sum_i \|d_i\|_2}$, 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

[4] We used PyML `http://pyml.sourceforge.net/` with the default parameter settings.

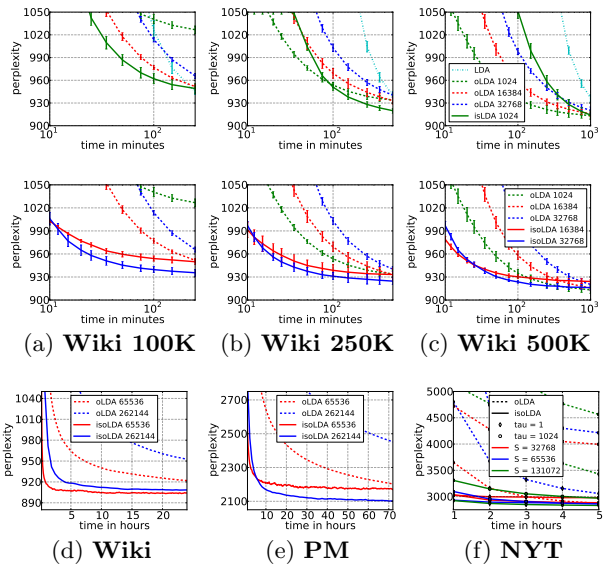


Figure 2: Perplexity vs. CPU time on Wiki subsets (min.; $K = 100$; upper two rows) resp. for (d) Wiki and (e) PM (h.; $K = 100$). (f) isoLDA is more stable for different parameter settings than oLDA: Perplexity on NYT after 5 hours. The numbers behind (is)(o)LDA indicate the batch sizes used. (Best viewed in color)

the important part of a corpus. We always stopped when each document was seen 25 times. We used fixed symmetric hyperparameters $\alpha = 0.01$ and $\eta = 0.01$.

Q1, Q2: Small and Medium Corpora The perplexity results are summarized in Fig. 1. As one can see, isLDA approaches find solutions in the range of the batch algorithm's solution with much less computation. For small corpora, the higher bias of isLDA (sorted) results in significantly better performance compared to isLDA and oLDA. With increasing sizes of the corpora, however, isLDA catches up. Taking all experiments together, either influence scheduling outperforms oLDA on all document selections. We note, however, that batch LDA's perplexity can drop below the ones of the mini-batch approaches. The accuracies of the classification experiments as summarized in Fig. 1 clearly show that influence scheduling yield competitive predictive accuracies. In 8 cases, isLDA produced significantly higher classification accuracies than oLDA. Only in one case oLDA produced a significantly higher classification accuracy.

Q3, Q4: Scaling Experiments The results so far suggest that isLDA finds solutions in the range of (o)LDA but much faster. To investigate this further for large datasets, we ran experiments on several subsets of the Wikipedia corpus ranging from 100K to 500K documents using the optimal parameters as determined by Hoffman *et al.* [5]. For isoLDA, we used a mini-batch size of 1024. The results are summarized in Fig. 2. As one can see, isLDA does not scale well. It actually slows down for larger document collections. In contrast, isoLDA scales well. To further investigate if isoLDA scales better than oLDA we conducted experiments on two massive datasets with approx. 3.3M (**Wiki**) respectively 8.2M (**PM**) documents. Here, we used $|B|/64$ to set the mini-batch size for isoLDA given that oLDA used a batch size $|B|$. For **Wiki** we set $\tau_0 = 1$ based on the proposed settings in [5] for large batch sizes. For **PM** we set $\tau_0 = 1024$ to account for its large vocabulary size. We considered large batch sizes of $|B| = 65536$ and $|B| = 262144$ as we can expect to find models of higher quality for them. The results are summarized in Fig. 2. On **Wiki**, isoLDA converged after about only 4 hours to a model with a lower perplexity than the model oLDA found after 24 hours. On **PM**, this difference in performances is even more pronounced. Moreover, the learning curves are much more stable compared to oLDA. That isoLDA is more stable w.r.t. to different parameter settings is also confirmed by our final experiment on the **NYT** corpus. Fig. 2 shows learning curves for several different parameter settings. Clearly, the variance is much lower for isoLDA than for oLDA.

Putting all experimental results together, we can clearly answer questions **Q1**, **Q2** and **Q4** affirmatively.

Conclusion

Triggered by the recent success stories of oLDA 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 forms mini-batches of highly influential documents first and processes them before less influential ones, called isLDA. Then, we turned isLDA into a novel, easy-to-implement oLDA approach, called isoLDA, that scales well to massive and growing datasets by applying influence scheduling to randomly formed batches. Based on the results of the present paper, [8] have recently developed the first active LDA.

Acknowledgements: MW and KK were supported by the Fraunhofer ATTRACT fellowship STREAM. AP was funded by the German Federal Ministry of Economy and Technology (BMWi) under the THESEUS project.

References

[1] D.M. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[2] C. Ding, T. Li, and W. Peng. NMF and PLSI: Equivalence and a Hybrid Algorithm. In *Proc. SIGIR*, 2006.

[3] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM*, 51(6):1025–1041, 2004.

[4] M. Girolami and A. Kaban. On an Equivalence between PLSI and LDA. In *Proc. SIGIR*, 2003.

[5] M. Hoffman, D.M. Blei, and F. Bach. Online learning for latent dirichlet allocation. In *Proc. NIPS*, 2010.

[6] M.W. Mahoney and P. Drineas. 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.

[7] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Stochastic search using the natural gradient. In *Proceedings of ICML*, page 146, 2009.

[8] M. Wahabzada and K. Kersting. Larger residuals, less work: Active document scheduling for latent dirichlet allocation. In *Proceedings of ECML PKDD*, 2011.