A Practical Algorithm for Solving the Incoherence Problem of Topic Models In Industrial Applications

Amr Ahmed
Google Research
Mountain View, CA
amra@google.com

James Long
Google Research
Mountain View, CA
jamlong@google.com

Daniel Silva
Google Research
Mountain View, CA
dansilva@google.com

Yuan Wang
Google Research
Mountain View, CA
yuanwang@google.com

ABSTRACT

Topic models are often applied in industrial settings to discover user profiles from activity logs where documents correspond to users and words to complex objects such as web sites and installed apps. Standard topic models ignore the content-based similarity structure between these objects largely because of the inability of the Dirichlet prior to capture such side information of word-word correlation. Several approaches were proposed to replace the Dirichlet prior with more expressive alternatives. However, this added expressivity comes with a heavy premium: inference becomes intractable and sparsity is lost which renders these alternatives not suitable for industrial scale applications. In this paper we take a radically different approach to incorporating word-word correlation in topic models by applying this side information at the posterior level rather than at the prior level. We show that this choice preserves sparsity and results in a graph-based sampler for LDA whose computational complexity is asymptotically on bar with the state of the art Alias base sampler for LDA [6]. We illustrate the efficacy of our approach over real industrial datasets that span up to billion of users, tens of millions of words and thousands of topics. To the best of our knowledge, our approach provides the first practical and scalable solution to this important problem.

1. INTRODUCTION

Topic models [2] are an important statistical modeling tool in the arsenal of any data miner. Given a collection of documents, a topic model allows us to infer the hidden structure in this collection in the form of topics where each topic is a distribution over a given vocabulary. Each document can be then represented as a distribution over these topics which helps visualize and navigate the otherwise unstructured collection of documents.

While topic models were mostly developed for textual documents, they found widespread usage in industry where documents correspond to users and words correspond to their activities: for example clicked websites, issued queries and installed apps, to name a few. The inferred user distribution over topics can be used as a basis for personalization to help serve the users with relevant content. Unfortunately, industrial settings posit two main challenges for standard topic models: 1) words are no longer surface tokens but rather complex structures whose interdependency and relationships are mostly ignored and 2) the heavy tail nature of the vocabulary makes it hard to infer robust co-occurrence statistics from the data. Both of these problems result in discovering a large number of incoherent topics that need to be filtered manually which limits the applicability of topic models in large-scale industrial settings.

While the first problem can be addressed by representing the tokens as first citizen objects, for instance one can represent each website as a bag of words, this rather complicates posterior inference and forces the modeler to focus on objects that are not the main goal of her original analysis. Furthermore, in industrial settings, such as web companies, there exist many tools that measure similarities between tokens (such as website and app descriptions), using well-established IR techniques; ignoring these tools is a waste of modeling effort as it would help us combat the second problem of data sparsity.

Given a sparse graph of token-token similarities, one would like to bias the model to allocate similar tokens to the same topic. While simple to state, a robust and scalable solution to this problem is rather lacking from the literature. The wide majority of popular topic models uses a Dirichlet Prior over the topic-word distribution to force topics to be sparse. This design choice is mostly motivated by computational efficiency due to the conjugacy between the Dirichlet distribution and the multinomial distribution that enables the development of many efficient samplers [6, 5, 3]. However, the Dirichlet distribution can not model correlations between words and in fact the components of a sample from the Dirichlet distribution are almost independent. To address this problem, several alternative priors that can incorporate word-word [1] correlation were proposed to replace the

[1] we use token and word interchangeably in this paper to denote an element from the vocabulary

KDD'17, August 13-17, 2017, Halifax, NS, Canada.

ACM ISBN 978-1-4503-4887-4/17/08.
http://dx.doi.org/10.1145/3097983.3098200 .

said Dirichlet distribution [9, 10, 8], however, this came at a premium: the lack of conjugacy makes inference dense and inefficient compared to existing approaches that uses conjugacy to leverage the hidden sparsity pattern in the data in deriving better samplers such as [6, 5]. As a result, those more expressive priors are not applicable in industrial settings.

In this paper we provide, to the best of our knowledge, the first scalable and practical solution to the problem of incorporating word-word correlation into topic models. Our solution results in discovering significantly more coherent topics without incurring a radical increase in the computational complexity of the inference algorithm. To achieve this goal, we took a radically different approach that does not incorporate word-word correlation at the prior level but rather softly enforces it at the posterior level. To that regard our approach is related, in spirit, to Posterior regularization [12] but unlike Posterior regularization, we incorporate word-word correlations as a proposal distribution. Via a reparameterization of the prior knowledge, and a careful construction of the knowledge-enriched proposal distribution, we were able to derive a graph-based sampler whose computational complexity is on par with the state of the art Alias sampler [6].

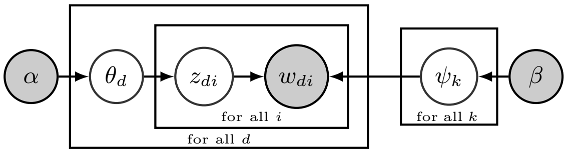
The rest of this paper is organized as follows: in Section 2 we review the basic topic model and related work, in Section 3, we detail our approach and provide an efficient graph-based sampler. Finally in Section 4, we illustrate the efficacy of our approach in several industry-scale datasets with favorable outcome.

2. BACKGROUND AND RELATED WORK

We give a brief introduction to topic models and the associated inference problems. Then we discuss related work to incorporate token-token side information in topics model and finally we give a brief introduction to the Shadow Dirichlet distribution that motivates our sampler in Section 3.

2.1 Latent Dirichlet Allocation

In LDA [2] one assumes that documents are mixture distributions of language models associated with individual topics. That is, the documents are generated following the graphical model below:



For each document d draw a topic distribution θ_d from a Dirichlet distribution with concentration parameter α

$$\theta_d \sim \text{Dir}(\alpha). \tag{1}$$

For each topic t draw a word distribution from a Dirichlet distribution with concentration parameter β

$$\psi_t \sim \text{Dir}(\beta). \tag{2}$$

For each word $i \in \{1 \ldots n_d\}$ in document d draw a topic from the multinomial θ_d via

$$z_{di} \sim \text{Discrete}(\theta_d). \tag{3}$$

Draw a word from the multinomial $\psi_{z_{di}}$ via

$$w_{di} \sim \text{Discrete}(\psi_{z_{di}}). \tag{4}$$

A key property to derive an efficient sampler for LDA is the fact that the the Dirichlet distribution is a conjugate prior to multinomial distribution. This allows us to integrate out θ_d and ψ_k and express $p(w, z|\alpha, \beta, n_d)$ in closed-form [3]. This yields a Gibbs sampler to draw $p(z_{di}|\text{rest})$ efficiently. The conditional probability is given by

$$p(z_{di}|\text{rest}) \propto \frac{(n_{td}^{-di} + \alpha_t)(n_{tw}^{-di} + \beta_w)}{n_t^{-di} + \bar{\beta}}. \tag{5}$$

Where n_{td}, n_{tw} and n_t denote the number of occurrences of a given (topic,document) and (topic,word) pair, or a given topic respectively. We use the superscript $\cdot^{-di}$ to denote the same count without the contribution of (z_{di}, w_{di}). For instance, n_{tw}^{-di} is obtained when ignoring the (topic,word) combination at position (d, i). Finally, α_t is the prior of topic t and $\bar{\beta} := \sum_w \beta_w$ is the normalization constant.

A naive approach to sample from (5) costs $O(k)$ time since there are k nonzero terms in a sum that needs to be normalized. Two approaches were proposed to break this $O(k)$ time complexity: The sparse LDA sampler of [5] and the Alias Sampler of [6].

In SparseLDA [5], the authors proposed a very clever decomposition of (5) to exploit the sparsity of the sufficient statistics as:

$$p(z_{di}|\text{rest}) \propto \beta_w \frac{\alpha_t}{n_t^{-di} + \bar{\beta}} + n_{td}^{-di} \frac{\beta_w}{n_t^{-di} + \bar{\beta}} + n_{tw}^{-di} \frac{n_{td}^{-di} + \alpha_t}{n_t^{-di} + \bar{\beta}}$$

In this formulation only the first term is dense, and more specifically, whenever both n_{td} and n_{tw} are sparse, sampling from $p(z_{di}|\text{rest})$ can be accomplished efficiently in to $O(k_w + k_d)$ where, k_w is the number of non-zero topics per word w and k_d is the number of non-zero topics that appear in document d.

In Alias LDA, [6], the authors proposed a sampler to draw from $p(z_{di}|\text{rest})$ in amortized $O(k_d)$ time. They accomplish this via the following decomposition:

$$p(z_{di}|\text{rest}) \propto n_{td}^{-di} \frac{n_{tw}^{-di} + \beta_w}{n_t^{-di} + \bar{\beta}} + \frac{\alpha_t (n_{tw}^{-di} + \beta_w)}{n_t^{-di} + \bar{\beta}} \tag{6}$$

Here the first term is sparse in k_d and can be drawn from in $O(k_d)$ time. They use a Metropolis-Hastings-Walker sampling algorithm to draw in an amortized $O(1)$ time from the second term that corresponds to the language model $p(w|t)$ and changes slowly. This makes the overall complexity of the algorithm $O(k_d)$

2.2 Incorporating Token-Token Relationship

Several approaches were proposed to incorporate side information in the form token-token relationships into the generative process of the standard topic model. These approaches aim to bias the model towards grouping related words into the same topic. Most of these approaches modify the prior over the topic-word distribution ψ. For instance [9] uses a Dirichlet Forest prior encode the Must-Links and Cannot-Links relationship between words. Similarly, [7] uses side information to define the prior β over the topic-word distributions. Finally, [10] proposed a quadratic regularizer and a convolved Dirichlet regularizer. In contrast to the previous approaches, [8] proposed an LDA-MRF (Markov Random

Field) that uses side information to bias similar words to have similar topics inside each document. This was achieved by imposing an MRF prior over the topic indicators inside each document.

Notwithstanding these excellent developments, all of these methods lack an efficient inference algorithm similar to those described in Section 2.1. Since most of these methods break the conjugacy assumption, a collapsed Gibbs sampler, with its associated fast inference algorithms [6, 5, 1] can not be used, and instead a variational inference based techniques were proposed. The techniques materialize the topic-words matrix ψ which can be prohibitive in industrial settings with thousands of topics and hundred of millions of tokens. Furthermore, variational inference algorithms can not leverage sparsity in the model and as such the computational complexity of the inference algorithm is $O(k)$. Thus the usage of side information comes with a heavy computational premium which precluded the usage of these techniques in real industrial settings.

2.3 The Shadow Dirichlet Distribution

The problem with using the Dirichlet distribution as a prior over the topic-word distributions is its inability to deal with any correlation structure between its components. In other words, the components of a sample from the Dirichlet distribution, such as topic ψ_k, are almost independent as they just need to sum up to 1. Several alternatives were proposed to remedy this problem, and we chose the Shadow Dirichlet Distribution[11] as an example since it inspires the development of our graph-based sampler in Section 3.

The Shadow Dirichlet distribution, $ShadowDir(\beta, S)$, is parametrized by two parameters: the generating mean, β, and a correlation structure S. S is a sparse stochastic matrix of size $V \times V$ where each row sums to 1 and V is the number of words (i.e. components in the sampled multinomial). A sample is drawn from the $ShadowDir(\beta, S)$ as follows:

$$\hat{\psi} \sim Dir(\beta) \text{ and } \psi := \hat{\psi} S^T. \tag{7}$$

The ShadowDir distribution can encode various constraints over the simplex that subsumes the regularizers enforced in [10]. Unfortunately, The *ShadowDir* distribution is not conjugate to the multinomial distribution, however, inspecting the structure of (7) gives us a key insight towards our sampler: the model enforces *apriori* that components of ψ are correlated via a *convex* combination according to the sparsity structure in S. We use this insight in the next section, however, not to define a prior that leads to an intractable posterior, but rather to define a correlated posterior that leads to a tractable proposal distribution.

3. GRAPH SAMPLER FOR TOPIC MODELS

In this section we detail our solution to utilizing token-token side information to improve the coherence of topic models. Unlike all previous approaches, we keep the generative process of LDA intact, thus enjoying the nice conjugate properties between the Dirichlet and Multinomial distributions. Instead, we use the word-word correlation structure to influence the posterior distribution over correlated words to be similar. In that regard our approach is related to posterior regularization [12], however, unlike Posterior regularization, we utilize side information to define a proposal distribution that biases the model towards the desired effect. We will show in this section, that this results in a very efficient sampler with minimal overhead over the samplers presented in Section 2.1. We first specify how we represent side information, then revisit the Alias sampler for notational consistency and finally present our graph sampler.

3.1 Representation of Side Information

We represent side information as a sparse graph G where two words are connected if they are semantically similar. For instance, when modeling user interests from search click history, words correspond to urls, and two urls are connected in G if they are semantically similar based on their content – an information that is otherwise unavailable for the unsupervised LDA model. We also assume that G is a stochastic graph, that is the edge weights are probabilities (i.e. $G_{uv} \in [0, 1]$) and each node defines a probability distribution over its neighbors (i.e. $\sum_{v \in \mathcal{N}(w)} G_w v = 1$), where we use $\mathcal{N}(w)$ to denote neighbors of w in G, and $\mathcal{N}_w$ to denote the average degree of nodes in G. Using G, we define a similarity matrix S as follows :

$$S = (1 - \lambda) \times I + \lambda \times G \tag{8}$$

where, I is the $V \times V$ identity matrix and V is the number of words. $\lambda \in [0, 1]$ and we refer to λ as the *smoothness factor* since we use λ to control the influence of the words in $\mathcal{N}(i)$ on i. By construction, S is a stochastic matrix as each row of S sums to 1. Also by construction, the non-zero entires in row $S_{w.}$ are given by $\mathcal{N}(w) \cup \{w\}$.

3.2 Alias Sampler Revisited

Recall from Section 2.1 that the main operation in a Collapsed Gibbs sampler is to sample a topic for each word using:

$$p(z_{di}|\text{rest}) \propto \frac{(n_{td}^{-di} + \alpha_t)(n_{tw}^{-di} + \beta_w)}{n_t^{-di} + \bar{\beta}}. \tag{9}$$

To break the naive $O(k)$ complexity barrier, the alias sampler postulates the following proposal distribution for each instance (d, w):

$$q(t, w, d) := \frac{P_{dw}}{P_{dw} + Q_w} p_{dw}(t) + \frac{Q_w}{P_{dw} + Q_w} q_w(t) \tag{10}$$

where

$$Q_w := \sum_t \alpha_t \frac{n_{tw} + \beta_w}{n_t + \bar{\beta}} \text{ and } q_w(t) := \frac{\alpha_t}{Q_w} \frac{n_{tw} + \beta_w}{n_t + \bar{\beta}} \tag{11}$$

and

$$P_{dw} := \sum_t n_{td}^{-di} \frac{n_{tw}^{-di} + \beta_w}{n_t^{-di} + \bar{\beta}} \text{ and } p_{dw}(t) := \frac{n_{td}^{-di}}{P_{dw}} \frac{n_{tw}^{-di} + \beta_w}{n_t^{-di} + \bar{\beta}} \tag{12}$$

The proposal in (10) is computed for every (d, w) pairs and comprises two components: a document dependent part p_{dw} and a word-dependent (document-independent) part, q_w that can be shared across documents. Each of these components is computed using the existing sufficient statistics in the sampler. A sample can be generated from p_{dw} in $O(k_d)$

whereas a sample can be sampled from q_w in an amortized $O(1)$ using Alias-Walker algorithm [4]. This can be done by freezing the sufficient statistics in q_w, computing an Alias table over them in $O(k_w)$ and then generating $O(k_w)$ samples in $O(1)$ per sample. Once this supply is exhausted, we regenerate the table and compute new samples. This makes the total amortized cost of generating a sample from q_w to be $O(1)$ and as such making the cost of generating a sample from $q(t, d, w)$ to be $O(k_d)$. Once a sample is generated, we use the standard MH acceptance probability to accept or reject this sample.

3.3 Graph Sampler

Now we are ready to define our graph-based proposal distribution that incorporates the similarity matrix defined in Section 3.1. In a nutshell, the basic idea is to define the graph based proposals as a convex combination of their non-graph-based counterparts where the convex combination is defined using the non-zero elements of the stochastic similarity matrix S. Formally we have:

$$q(t, w, d)^S := \frac{P^S_{dw}}{P^S_{dw} + Q^S_w} p^S_{dw}(t) + \frac{Q^S_w}{P^S_{dw} + Q^S_w} q^S_w(t) \qquad (13)$$

where

$$q^S_w(t) := \sum_{v \in \mathcal{N}(w) \cup \{w\}} S_{wv} q_v(t) = \frac{\alpha_t}{Q^S_w} \frac{\sum_{v \in \mathcal{N}(w) \cup \{w\}} S_{wv} n_{tv} + \beta_w}{n_t + \bar{\beta}} \qquad (14)$$

and

$$\begin{aligned} p^S_{dw}(t) &:= \sum_{v \in \mathcal{N}(w) \cup \{w\}} S_{wv} p_{dv}(t) \\ &= \frac{n^{-di}_{td}}{P_{dw}} \frac{\sum_{v \in \mathcal{N}(w) \cup \{w\}} S_{wv} n^{-di}_{tv} + \beta_w}{n^{-di}_t + \bar{\beta}} \qquad (15) \end{aligned}$$

Finally Q^S_w and P^S_{dw} are the normalization constants for both q^S_w and p^S_{dw} respectively. Since S is a stochastic matrix, It is easy to show that:

$$Q^S_w = \sum_{v \in \mathcal{N}(w) \cup \{w\}} S_{wv} Q_v \qquad (16)$$

It is very instructive to notice how S appears in the graph based proposals. In fact, the graph based proposals can be simply obtained by replacing the word topic counts statistics n_{tw} in (11-12) by their graph-based convex counterparts:

$$n^S_{tw} := \sum_{v \in \mathcal{N}(w) \cup \{w\}} S_{wv} n_{tv}. \qquad (17)$$

As such, it is easy to see that naively sampling from $q(t, w, d)^S$ using the same alias based algorithm can be done in an amortized $O(\mathcal{N}_w k_d)$ time, however, we will show below how to improve over this bound. Furthermore, according to (8), if $\lambda = 0$ then $S_{ij} = 0$ if $i \neq j$ and as such the the graph based proposals are reduced to the standard alias proposals. As we increase λ, the contribution of neighboring words in the graph increases.

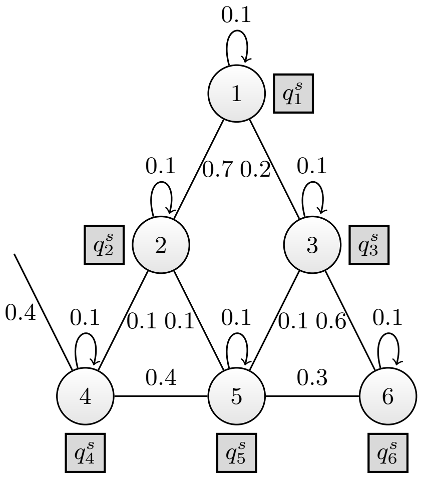


Figure 1: Illustrating the efficient sampler for $q^S_w(t)$. The graph pictorially depicts entries in the similarity matrix S. Each node is endowed with its own samples generated from the graph-oblivious proposal q_w. To sample a topic from $q^S_w(t)$, we first sample a neighbor and then chose a sample from that neighbor's alias samples. Both of these steps can be accomplished in $O(1)$ amortized cost.

Algorithm 1 Graph Sampling

```
1 [Graph Initialization] $\forall w$ compute an alias table for
  the distribution $S_{w.}$.
2 [Word Initialization] $\forall w$ compute an alias table for the
  distribution $q_w$ and generate $O(k_w + \mathcal{N}_w)$ samples from
  it.
3 [Normalization Initialization] $\forall w$ compute $Q^S_w$ using
  (16) in $O(\mathcal{N}_w)$.
4 [Maintenance]: generate a sample from $q^S_w$.
    4.1 generate a sample node $v$ from the alias table of
        $S_{w.}$.
    4.2 pull a sample from those cached for $q_v$, if no samples
        are left then repeat steps 2, 3 only for word
        $v$.
```

3.3.1 *Sampling from $q^S_w(t)$:*

As discussed earlier in (17, 14) the cost to compute the alias table for q^S_w is $O(\mathcal{N}_w k_w)$ and as such one needs to draw $O(\mathcal{N}_w k_w)$ samples from this alias table to achieve an amortized cost of $O(1)$. However this increases the risk of sampling from a stale distribution. Inspecting (14) one notices that since S_w is a distribution, we can sample from $q^S_w(t)$ in a two step process: first, sample a node $v \in \mathcal{N}(w) \cup \{w\}$ according to S_w in $O(1)$ and then generate a sample from q_v in an amortized cost of $O(1)$. In this case, one needs only to generate only $O(\mathcal{N}_w + k_w)$ to achieve an amortized cost of $O(1)$. The full algorithm is defined below and the overall idea is illustrated in Figure 1. Note that we need to generate $O(k_w + \mathcal{N}_w)$ from q_w since we need also to compute Q^S_w which is essential to sample from (13). The main steps are summarized in Algorithm 1.

3.3.2 *Sampling from $p^S_{dw}(t)$:*

Sampling from this distribution is dominated by the cost to compute the normalization constant $P^S_{dw}(t)$. From (15), it is clear that this costs $O(k_d \mathcal{N}_w)$. This cost is dominated by the need to compute the smoothed word topic counts in (17) for every topic that appear in the document. However, we can leverage the fact that weights in S naturally follow a power-law distribution and use a sample with replacement technique to approximate the smoothed word-topic sum as follows. For every pair occurrence of (d, w) we *sample with replacement* a fixed set of nodes from $\mathcal{N}(w) \cup \{w\}$ according to $S_{w.}$ and assign each of these samples (that might include repeated occurrences of the same neighbor) a uniform weight to approximate the computation of (17). This fixed-size neighborhood is regenerated dynamically for every occurrence of (d, w) and is different every time we encounter the same pair in subsequence iterations. In addition, these samples can be generated in $O(1)$ since as we described in Section 3.3.1 we compute and store an alias table table for every row of S. This allows us to sample from $p^S_{dw}(t)$ in $O(k_d)$ and in the experimental section, we will see that a size between $[1, 5]$ is sufficient to achieve decent performance.

3.3.3 *Summary*

To recap our graph-based sampler. We have shown in the previous two subsection that one can sample from the graph based proposal defined in (13) in an amortized cost of $O(k_d)$ which matches the same amortized cost of sampling from the non-graph-based alias sampler defined in [6]. Once a sample t' is generated, we compute the acceptance ratio $\pi = \min(1, \frac{p(z_{dt'}|\text{rest})q^S(d,t,w)}{p(z_{dt}|\text{rest})q^S(d,t',w)})$, and finally accept the new topic assignment t' with probability $\propto \pi$.

3.4 Distributed Implementation

Since we run our experiments on industry-scale data, we use the same parameter server architecture in [14] to implement the proposed sampler. In this architecture, the global sufficient statistics (n_t, n_{tw}) are stored on a set of servers. Documents are partitioned among clients and each client maintains a partial view of the global state that is sufficient to perform sampling over its assigned documents. For the Graph-sampler discussed in this paper, the partial view at each client contains the sufficient statistics corresponding to words that appear in documents assigned to this client, in addition to their closures in the word-word similarity graph S. The later set of words are necessary in order to be able to compute the graph-based proposal distributions locally. A synchronization thread runs on each client in parallel with the sampling thread(s) to synchronize the local partial view with the global state at the servers.

4. EXPERIMENTS

In this section we illustrate the efficacy of our approach over various datasets both qualitatively by inspecting the coherence of the learnt topics and quantitatively using held out Log Likelihood (LL). In the rest of this section we refer to our approach as **LDA+graph Sampler** or **Graph Sampler** for short. We compare our approach to the basic LDA model trained using the Alias Sampler in [6]. Since none of the related work discussed in Section 2.2 is applicable at this scale, we also compare our approach on a small dataset to the recently proposed LDA-MRF [8] topic model. Unless otherwise stated, all experiments were ran on a cluster of 100 clients and 20 servers over 2000 topics. We set the parameters of the graph sampler as follows: smoothness factor $\lambda = 0.01$ (Section 3.1) and dynamic neighborhood (i.e. number of samples with replacement) of size 5 (see Section 3.3.2).

4.1 Datasets Overview

We considered four different datasets show characteristics are detailed in Table 1.

- **Wikipedia**: set of Wikipedia pages. Each Wikipedia page is considered as an LDA document and each tokenized word in that page body is considered as a word in that document. As side information we used both Word2Vec [13] embedding and a similarity measure based on Google's Knowledge Graph [16].
- **App Installs**: set of mobile applications installed by users. Each user is regarded as an LDA document and the set of apps installed by that user is considered as the words in that document. For privacy reasons we only subsampled some of the users and some of the apps. As side information we used a graph based on app tags and textual descriptions. Note that since installed data is binary (either an app is installed or not), the frequency of each word in a given document will always be at most 1, which is a challenging situation for LDA .
- **Search Click History**: sub-set of user URL search click history. Each user is regarded as an LDA document and the list of URLs is considered as the words in that document. As side information, we derive the similarity graph based on the URL content using standard IR techniques. For privacy reasons we only subsampled some of the users and some of the top urls.
- **Query History**: We subsampled a sub-set of search queries from a web search engine. Each user query is assigned into a pre-computed query cluster where the clustering is based on the language model of each query . Each user is regarded as an LDA document and each whole query cluster id is considered as a word. As side information, we used similarities between language models of the query clusters.

4.2 Qualitative Results

In Tables 2,3,4,5,6,7,8 we show some qualitative results from each of the four analyzed datasets. We see that for most cases, the incoherent words within the topics disappear when the LDA training includes side information. For all figures, we show the top words discovered by different samplers. Some observations are in order. For the Wikipedia dataset, as shown in Tables 2 and 3 both forms of side-information results in different form of topics and qualitatively, while both sources of side information results in more coherent topics compared to the baseline, none of them dominated the other source, however, quantitively, the model with KG similarity results in overall better held-out LL. Perhaps the biggest improvement was observed over the query history dataset in Tables 6 and 7 and in the app install datasets show in Table 8. This happens largely due to the heavy tail nature of these datasets and the binary nature of the app-install dataset. As evident from Table 6, the "gardening" topic was largely garbled beforehand and became coherent

Dataset	Vocabulary size	Number of Documents	Average similarity Degree.
Wikipedia KG	50,000	220,000	50
Wikipedia Word2Vec	50,000	220,000	50
App Installs	100,000	200,000,000	100
Search Click History	5,000,000	1,000,000,000	75
Query History	10,000,000	400,000,000	50

Table 1: General statistics for the document datasets and the side information similarity graphs. Voc: vocabulary size; Train Docs: number of training documents; Test Docs: number of held out test documents; Avg Degree: average node degree of side info similarity graph. . For all user data, we only subsampled a set of users /items for privacy reasons.

Table 2: Wikipedia topic about World Class Sports

	Graph Sampler	Graph Sampler
LDA	(KG)	(W2V)
World	world	championships
at	men's	won
championships	women's	championship
born	olympics	women's
won	olympic	sport
team	competition	olympic
olympics	results	international
april	tournament	event
medal	sports	final

Table 3: Wikipedia topic about Architecture

	Graph Sampler	Graph Sampler
LDA	(KG)	(W2V)
building	building	building
buildings	buildings	buildings
holders	design	built
timatic	england	construction
were	houses	located
england	tower	site
information	architect	city
ground	side	designed
birmingham	hospital	park

The left most topics in each table are generated from regular LDA model, the non coherent words are in red colors, the middle topics in each table are generated with graph sampler utilizing side information from either the Knowledge Graph or W2V embedding.

Table 4: Search Click History topic about food recipes

LDA	Graph Sampler
allrecipes.com	allrecipes.com
foodnetwork.com	foodnetwork.com
alohaorderonline.com	food.com
food.com	tasteofhome.com
bettycrocker.com	marthastewart.com
realsimple.com	realsimple.com
tasteofhome.com	bettycrocker.com
kobobooks.com	delish.com
accuweather.com	myrecipes.com
thekitchn.com	cookinglight.com

Table 5: Search Click History topic about sports

LDA	Graph Sampler
nfl.com	nfl.com
espn.com	espn.com
fanduel.com	fantasypros.com
ufrgs.br	rotoworld.com
cbssports.com	foxsports.com
foxsports.com	bleacherreport.com
nba.com	cbssports.com
nflshop.com	nbcsports.com
rotoworld.com	dallascowboys.com
ifsp.edu.br	fanduel.com

The left topics in each table are generated from regular LDA model, the red urls are incoherent compared to the rest words in the same topic; the right topics of each table are generated with side info, which have must less incoherent words.

Table 6: Query History topic about gardening

LDA	Graph Sampler
companion planting	companion planting
canker sore	growing watermelon
age of wushu	growing tomatoes
mre	onion plant
words with friends app	growing strawberries
waves lyrics	powdery mildew
clarkson university	heirloom seeds
car rental with debit card	how to plant a garden
mountain view high school	blueberry bush
words that end in z	non gmo seeds

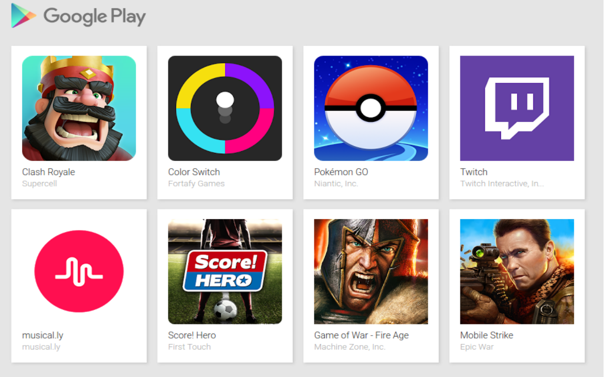
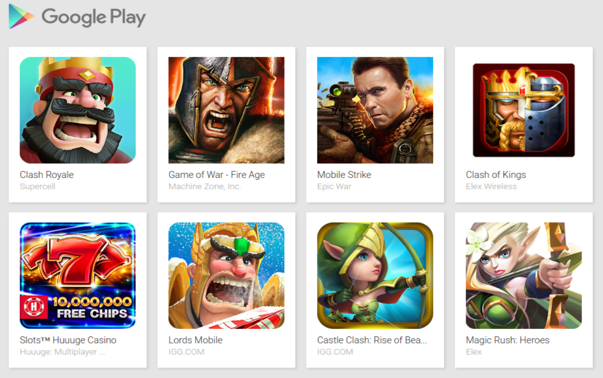
Table 7: Query History topic about Python

LDA	Graph Sampler
python list	python list
numpy array	numpy array
pyplot	python try except
python csv	python sort
python random	python random
python try except	python try except
sloppy joes	python string
machine learning	python subprocess
deep learning	python if else
naive bayes	pycharm

Topics on the left side of each table are topics generated from regular LDA model, the non coherent words are in red . Topics on the right side of each table are generated with side info. Note that while the Python topic was already decent in the baseline, it still contains words not semantically related to Python.

with side information, while the Topic about Python in Table 7 was decent beforehand but contains various spurious queries about Machine learning largely because of the large usage of Python to code ML models, however, with side-

Table 8: App install topic about action game

LDA	Graph Sampler
 Google Play Clash Royale Supercell Color Switch Fortafy Games Pokémon GO Niantic, Inc. Twitch Twitch Interactive, In... musical.ly musical.ly Score! Hero First Touch Game of War - Fire Age Machine Zone, Inc. Mobile Strike Epic War	 Google Play Clash Royale Supercell Game of War - Fire Age Machine Zone, Inc. Mobile Strike Epic War Clash of Kings Elex Wireless Slots™ Huuuge Casino Huuuge: Multiplayer ... Lords Mobile IGG.COM Castle Clash: Rise of Bea... IGG.COM Magic Rush: Heroes Elex

The left side is an action game topic generated from the regular LDA model, the app themes are mixed and and are not very coherent. The right side is a similar topic generated with sideinfo which gives more coherent results.

	Baseline	LDA-MRF (KG)	LDA+GraphSampler (KG)	LDA-MRF (W2V)	LDA+GraphSampler(W2V)
Log-Likelihood	-2327	-2275	-2256	-2314	-2307

Table 9: Comparison of held-out data log-likelihood for regular *LDA*, *LDA-MRF*, and *LDA+GraphSampler* for Wikipedia dataset.

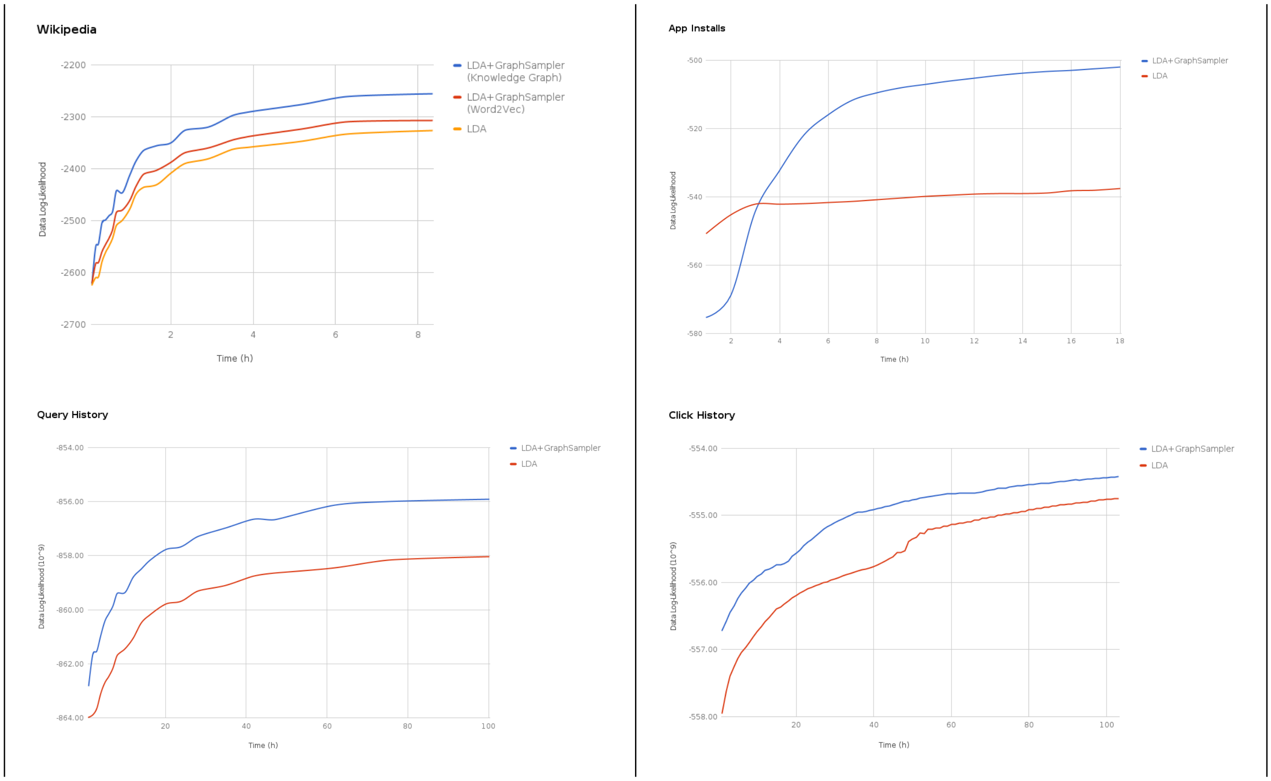


Table 10: Side-by-side comparison of baseline *LDA* and *LDA+GraphSampler* for all four datasets.

information, these ML related queries were factored out and this improves the readability of the topic. In fact it is better to factor these ML queries out since not every python coder is interested in ML. Moreover, as clear from Table 8 the app topic about Action games is much more clearer with side-information. Overall, the addition of side-information enables the model to trade lexical similarity with behavioral similarity and prevents the model from capturing spurious behavioral correlations in the data.

4.3 Quantitative Results

	Search Click History			App Installs		
# Top Words (M)	LDA	LDA+GraphSampler	Improvement	LDA	LDA+GraphSampler	Improvement
5	-59.97	-54.72	8.75%	-30.808	-28.601	7.16%
10	-283.49	-252.79	10.83%	-161.81	-151.56	6.33%
20	-1236.45	-1094.82	11.45%	-799.41	-743.93	6.94%
30	-2876.27	-2532.55	11.95%	-1995.4	-1853.6	7.11%

Table 11: Topic coherence improvement observed by using *LDA+GraphSampler*. Table shows results for different values of top words (M) across Search Click History and App Installs datasets.

Dataset	Baseline	SF = 0.001	SF = 0.01	SF = 0.1	SF = 0.2
App Installs	-91.37	-91.51	-90.20	-87.65	-85.34

Table 12: Final data log-likelihood ($\times 10^9$) for baseline (*LDA*) and for *LDA+GraphSampler* under different smoothing factors. SF: Smoothing factor.

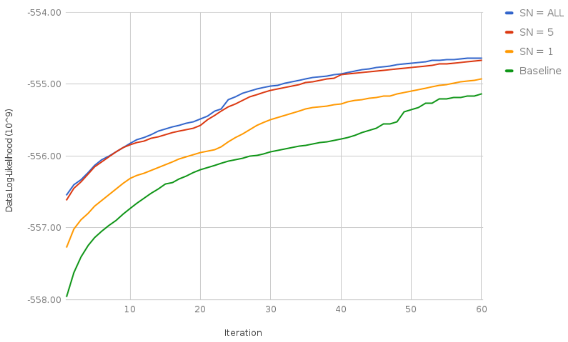


Figure 2: Comparison of baseline *LDA* vs. *LDA+GraphSampler* with different values of number of neighbors sampled (with replacement) for the Search Click History dataset. SN = Sampled Neighbors.

Table 10 contrasts the held-out LL over the four datasets between LDA and LDA+Graph Sampler. As evident form the Figure our Graph sampler algorithm that utilizes side information both A) improves held-our LL and b) converges at the same speed or even faster than the baseline LDA standard Alias sampler. We also notice that improvement is more noticeable as the dataset is more sparse (app-install and query history). Furthermore, we noticed that for the wikipedia dataset that using KG-based similarity gives better results than using similarity driven from another form non-linearity (i.e. w2v) that might be more relevant in the context it was derived from but not as side information for other models.

As we mentioned earlier, none of the baselines discussed in Section 2.2 scales to massive datasets, thus we compare our graph sampler with the LDA-MRF baseline on the smaller wikipedia dataset using K = 100 since LDA-MRF uses a Variational Inference based inference algorithm that scale as O(K). The results are shown in Table 9. We see that, while *LDA+MRF* outperforms the regular *LDA* baseline, *LDA+SideInfo* yields slightly better results than *LDA+MRF*. Which means that our approach while scalable doesn't compromise quality when compared to models that use alternative priors.

Finally, we also compute the semantic coherence improvement generated by our graph sampler. As proposed in [15], topic coherence is defined as:

$$C(t; V^{(t)}) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} log \frac{D(v_m^{(t)}, v_l^{(t)}) + 1}{D(v_l^{(t)})} \tag{18}$$

Where $V^{(t)} = (v_1^{(t)}, \ldots, v_M^{(t)})$ are the top M words in topic t, $D(w)$ is the word frequency in a document, and $D(w, w')$ is the co-document frequency across the dataset.

We observed a significant improvement in topic coherence for different values of M, as shown in Table 11. As expected, the topic coherence improvement was better for higher values of M since the incoherent words tend to become more frequent as we move towards the lower-scored words in a topic. For $M = 30$ we observed a topic coherence improvement of 11.95% for Search Click History and 7.11% for App Installs.

4.4 Ablation Study

In order to understand the effects of the different parameters involved in our *Graph Sampler*, we ran a sequence of experiments varying the values of the number of sampled neighbors (Section 3.3.2) and the smoothing factor (λ, section 3.1).

4.4.1 Sampled Neighbors (SN)

This parameter corresponds to the number of neighbors we sample with replacement for a given word during the document proposal computation. As detailed in Section 3.3.2, we use those sampled sub-set of $\mathcal{N}(w) \cup \{w\}$ to compute the effective smoothed word probability instead of $\mathcal{N}(w) \cup \{w\}$ itself.

We observe in Figure 2 that, as expected, the *per-iteration* log-likelihood curve improves as we increase the number of neighbors sampled since we are incorporating more information from the similarity graph. However, the running time of each iteration grows linearly with the number of sampled neighbors. Even using a relatively low value of SN (such as 5 in our Figure) was enough to yield great improvements over the baseline. Furthermore, using 5 samples we were able to obtain a model that is similar in quality to the model obtained with $SN = ALL$ with practically little change in runtime over the baseline LDA model. The key idea here is that we use different 5 fresh samples each time we touch a given word, hence we can provide a good approximation with little extra computation.

4.4.2 Smoothing Factor (SF)

This parameter λ corresponds to the probability allocated to neighbouring words (See section 3.1).

We can see in Table 12 that *LDA+GraphSampler* will outperform *LDA* in a very large range of values for the *App Installs* dataset (due to the high cost of these experiments, we have not run similar tests for the other 3 datasets). We observe that $SF = 0.001$ (under-smoothing) will yield results similar to baseline. From $SF = 0.001$ to $SF = 0.5$, we observed a monotonically increasing behavior of final data log-likelihood, reaching final values that are significantly better than the baseline. In general this value can be tuned using cross validation if need be.

5. CONCLUSION

In this paper we proposed a Graph Sampler for topic models that utilizes side information in the form of a word-word similarity graph. We showed that our approach is computationally efficient, and asymptotically scales as the efficient Alias Sampler for LDA [6]. Our approach preserves sparsity in topic-word representation during inference and scales gracefully with both number of documents, number of topics and vocabulary size. We demonstrated the efficacy of our approach over real big-data applications and showed qualitatively and quantitatively the added value of our technique. To the best of our knowledge, we believe that our approach is the first practical and scalable solution to the problem of incorporating word-word side information in topic models.

6. REFERENCES

[1] D. Blei, T. Griffiths, and M. Jordan. The nested chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):1–30, 2010.

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

[3] T. Griffiths and M. Steyvers. Finding scientific topics. *PNAS*, 101:5228–5235, 2004.

[4] A. J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM TOMS*, 3(3):253–256, 1977.

[5] L. Yao, D. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In *KDD'09*, 2009.

[6] Li, Aaron Q. and Ahmed, Amr and Ravi, Sujith and Smola, Alexander J. Reducing the Sampling Complexity of Topic Models. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM*, 2014.

[7] Petterson, J., Buntine, W., Narayanamurthy, S.M., Caetano, T.S. and Smola, A.J. Word features for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, 2010.

[8] Xie, Pengtao, Diyi Yang, and Eric P. Xing. Incorporating Word Correlation Knowledge into Topic Modeling. In *HLT-NAACL*, 2015.

[9] Andrzejewski, David, Xiaojin Zhu, and Mark Craven. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning. ACM*, 2009.

[10] Newman, David, Edwin V. Bonilla, and Wray Buntine. Improving topic coherence with regularized topic models. In *Advances in neural information processing systems*, 2011.

[11] Bela Frigyik and Gupta, Maya and Yihua Chen. Shadow Dirichlet for Restricted Probability Modeling. In *NIPS'10*, 2010.

[12] Ganchev, Kuzman, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. In *Journal of Machine Learning Research 11.Jul (2010): 2001-2049*, 2010.

[13] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (pp. 3111-3119)*, 2013.

[14] Li, M., Andersen, D.G., Park, J.W., Smola, A.J., Ahmed, A., Josifovski, V., Long, J., Shekita, E.J. and Su, B.Y. Scaling Distributed Machine Learning with the Parameter Server. In *OSDI. Vol. 14*, 2014.

[15] Mimno, D., Wallach, H. M., Talley, E., Leenders, M., and McCallum, A. Optimizing semantic coherence in topic models. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (pp. 262-272), 2011.

[16] https://www.google.com/intl/bn/insidesearch/features/search/knowledge.html