A Geometric Algorithm for LDA that's Predictive, Not Perplexing

Di Wang, Victor Huang, James Cook, Andrew Gambardella, Chenyu Zhao, Satish Rao *

ABSTRACT

The Latent Dirichlet Allocation (LDA) topic model has received enormous attention for extracting insight from documents and a wide variety of other data sources. A reasonable precondition for the utility of an LDA-based algorithm is that it perform well with data generated according to the LDA model.

We evaluate the model on a range of algorithms, both LDA-based and not. In addition, we also construct and evaluate a new algorithm, Projector, based on a geometric view, rather than an inference-based view of finding the optimal model. The algorithm is simple and much faster compared to the best inference-based methods available (e.g., the optimized Gibbs sampling based topic learning method in MALLET, Minmo et al.).

Our primary performance benchmark is the accuracy of predicting a new word in a document. This task is both natural for the LDA model and has widespread, practical applications. This notion of performance differs from statistical-based measures such as perplexity in that one can evaluate against non-parametric algorithms and it has a simple, intuitive interpretation.

The performance of Projector on generated data is better than the best inference-based methods in both predicting dropout words and learning the underlying topic models. Furthermore, the runtime of Projector is better than inference based algorithms.

Additionally, we identify features of the generated data to help characterize the performance of Projector and each other algorithm that we consider; in particular, the typical number of significant topics in a document, and the typical number of significant words in a topic. We give a simple analysis using these parameters to determine when performance should drop off and verify our characterization of predictability.

Finally, we test on various real-world document corpora on our word prediction task. In this context, our algorithm is competitive with the other LDA based algorithms while falling short of nearest-neighbor based methods.

*UC Berkeley

1. INTRODUCTION

The Latent Dirichlet Allocation topic model has been tremendously influential in the development of methods for analyzing documents and other types of data. Numerous algorithms for learning LDA, as well as variations LDA, have been proposed; see [4] for a recent survey. The bulk of these methods are based on inference techniques; they proceed by learning the parameters using methods such as Gibbs sampling, expectation-maximization, variational methods, and model modifications for faster computation. Evaluating these methods have typically proceeded along one of two lines. One trains on a set of documents and evaluates how well the resulting model predicts a test set, typically measured by perplexity. Another uses the model as a set of features for some classification task (e.g. document classification) followed by the application of some learning method (e.g. a support vector machine).

In this paper, we suggest an alternative to the inference approach in learning topic models as well as introduce a different evaluation method. The first yields an improved algorithm for learning LDA and the second allows for better understanding of topic learning efficacy.

In terms of inference algorithms, we view model parameters as geometric objects. For textual data, topic centers are simply points in the word space. This view is very traditional in that data is seen as generated from points in space under some noise model. This interpretation has long been associated with algorithms such as nearest neigbors for classification and k-means for clustering. As we show though, nearest neighbors and k-means are inferior to present LDA inference algorithms for LDA generated data. However, we show that a combination of k-means with dimension reduction and a scaling step produces an effective algorithm for this type of data. For real world data, our methods remain just as effective as LDA inference algorithms and more effective than k-means, but falls short of nearest neighbor techniques (as do all topic modelling approaches.) We refer to our algorithm as the Projector algorithm. We note here that recent work in [1] use linear algebraic methods, which can be viewed to some extent as geometric algorithms, to give an algorithm that provably learns the LDA parameters given a polynomial number of examples and polynomial time.

We measure our performance on the task of predicting a set of dropped out words in a document. Topic models, in particular the LDA topic model, generate documents on a word by word basis. Thus, a model's effectiveness for predicting a word in a document is both natural and easily interpreted. It also allows for the comparison of any prediction method, whether it explicitly learns the topic model or not. We also note that this task has been well studied in the field of recommender systems, where documents are people and words are products.

Before proceeding, we note that this paper proposes an algorithm, argues for more informative evaluations, and does experimental studies of existing algorithms. We feel these things go very much together. The proof for a methodology for algorithm development is, after all, an effective algorithm.

We proceed with related work in section 2, a description of the LDA model in section 3, the data generation process in section 4, a description of the algorithms we study in section 6, the results of our experiments in section 7, a bit of analysis in section 8, and conclude in 9.

2. RELATED WORK

An early version of topic modeling is seen in Latent Semantic Indexing (LSI) which views a document as a combination of a larger set of topics. Topics are in turn about certain words (represented by positive numbers) or not about certain words (represented by negative numbers) [13]. They provided a linear algebra algorithm that learned the structure of their model. They also argued that this model provides insight into the wide applicability of principal components analysis in data analysis. The mathematical tool at the core of their work (and PCA) was the singular value decomposition.

Shortly thereafter, Puzicha and Hofmann [7] provided a probabalistic framework, termed probabilisitc latent semantic indexing (PLSI). Topics are modelled as probability distributions over words (no negative numbers). Documents are then assumed to be generated as a mixture of these topics.

The large number of parameters in the PLSI model motivated Blei, Ng and Jordan to develop a generative model for these parameters, which resulted in the enormously influential Latent Dirichlet Allocation (LDA) topic model [6]. The LDA model has been widely applied in various domains such as information retrieval, collaborative filtering, computer vision, and bioinformatics.

The most closely related works in theory are the aforementioned algorithms provided by Arora et.al. [2] which learn PLSI under certain assumptions, and a breakthrough by Anandkumar [1] which gives a provably convergent algorithm for learning LDA with polynomial data requirements. There is a long literature in learning mixture models in statistics and more recently in theoretical computer science. See [9] for a recent breakthrough on learning mixtures of Gaussians and a discussion of this field.

There is ample work describing methods for optimizing LDA [4]. Recent examples are described in [11], [12].

There is also significant work on extending LDA and topic models to better model data as well as to augment them with other types of information. These are discussed in [4]. The hierarchical LDA model [5] is perhaps the most relevant. This model is based on the chinese restuarant process where new customers arrive and chose to join a table or create a new one. This process can be used to create a hierarchical structure of documents and topics. The authors argue that this is a better model for real data and we tend to agree though our preliminary experiments do not bear this out. Still, we began by understanding the simpler model which continues to have wide applications.

Other examples of variations of the LDA model include adding the notion of a manifold to the model [8], adding partially labeled data [14], and sparse LDA models [19].

This paper, in addition to providing some insight into the effectiveness of LDA algorithms, seeks to provide an infrastructure to evaluate effective topic modeling methods. We should point out that the machine learning community has made efforts in this regard. See for example MLComp [10]. We also mention the Java-based infrastructure that we used for topic modelling [11] which is part of the larger MALLET machine learning package.

3. LDA MODEL

LDA was introduced in [6] as a generative process. In this work we will adopt the language of text collections, and denote entities as 'word', 'document', and 'corpus', since they give intuition in discussing topic models.

We begin with n documents, m words, and k latent topics. Each document is a mixture of the latent topics θ, where the mixture is drawn from a Dirichlet distribution with parameter $\vec{\alpha}$.

$$p(\vec{\theta}|\vec{\alpha}) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\Pi_{i=1}^{k}\Gamma(\alpha_i)} \theta_i^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1}$$

This results in an $n \times k$ document-topic matrix W, where row i gives the parameters of the multinomial distribution over topics for document i. Similarly, we have a $k \times m$ word-topic matrix A, where the j-th row gives the parameters of the multinomial distribution over words for topic j. The columns are drawn from a Dirichlet distribution with parameter $\vec{\beta}$.

For a document w, we first choose $\vec{\theta}$, which can take values in the $(k-1)$-simplex. The number of words in document w is sampled from Poisson(l). For each word w_i, a topic $z_i \sim$ Multinomial($\vec{\theta}$) is chosen, followed by the actual word $w_i \sim$ Multinomial(A_{z_i}). The product $M = WA$ is the $n \times m$ document-word matrix where row M_i is document i's multinomial distribution over on words.

Equivalently, document i is generated by sampling words i.i.d from Multinomial(M_i). We are interested in the case where A is of full rank, since if the row of A are not independent, intuitively it means there exists some document which is covered completely by a set of topics I, but at the same time also completely covered by another set of topics J which is disjoint from I. In our experiments, the randomly generated A matrices are almost always of full rank.

4. DATA

Since our focus is on how effectively the algorithms learn the model, we use synthetic datasets generated from the LDA model for a range of parameters. Our data generator takes in parameters

- n, m, k, number of documents, words, and topics respectively
- α, the Dirichlet parameter for generating documents' distributions over topics as in the LDA model. In our

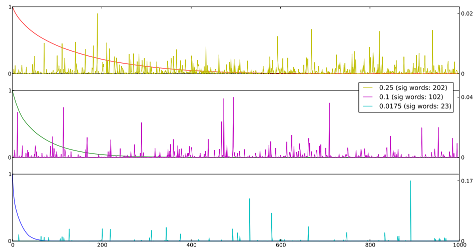


Figure 1: Plot of distributions on words for various β. $m = 1000$, each distribution is plotted along with its cdf after sorting the words by popularity. Refer to the y-axis on the right for the scaling of the distributions. In general, larger β values yield flatter distributions.

experiments we work with symmetric Dirichlet distributions, where $\alpha_i = \ldots = \alpha_k = \alpha$

- β, we generate the columns of word-topic matrix A from a m dimensional Dirichlet distribution with parameter $\vec{\beta}$. Again we work with symmetric Dirichlet where $\beta_1 = \ldots = \beta_m = \beta$.
- l, the Poisson parameter controlling the expected number of words in a document.

Intuitively the Dirichlet parameter α is a crude measure of the sparsity of the sampled distribution over topics. When $\alpha = 1$, all points in the $k - 1$ simplex have the same probability density. When $\alpha < 1$, distributions that are more concentrated on a few topics are prefered by the Dirichlet. The same applies to β and the topic's distribution on words. See figure 1 for typical word distributions sampled from the Dirichlet distribution with various β's as parameter.
To help understand the dataset, we compute the values sig_topic and sig_word. For a document with distribution $\vec{\theta}$ over topics, sig_topic is the smallest t such that the union of the t heaviest topics in $\vec{\theta}$ has an aggregate probability of at least 0.8. Intuitively, sig_topic is the number of significant topics of a document. Analogously, for a topic's distribution over words, sig_word is the smallest number of most popular words with an aggregate probability of at least 0.8.Instead of using α and β, we use the average sig_topic and average sig_word to characterize our datasets.
We also evaluate on a topic distribution which obeys a power law. Instead of using the same α parameter for all the topics, we vary it so that there are topics of varying popularity.
We also evaluate the methods on some standard real world datasets. We used the Classic-3 datasets (Cran,Med,Cisi) [15], a corpus (AP) from the Associated Press [16] pruning stop words and infrequent words, and a bag of words dataset (NIPS) from UC Irvine[18], MovieLens [17].

5. EXPERIMENTS

5.1 Prediction task: the plusone framework.

For a corpus of documents, we randomly divide the documents into the training set and the testing set, where each document is put into the training set with probability p_t independently. For each document in the testing set, we hold out a certain percentage H of the distinct words in the document uniformly at random. The training set is given to the algorithms. After training, each algorithm gets the testing documents, and for each document predicts s terms not in the observed part of the testing document. We use the precison of the s predicted terms as the score of an algorithm on a specific testing document. The score of an algorithm on the corpus is its average score over all testing documents. In our experiments, we use $p_t = 0.5, H = 30\%, s = 3$ as our parameters.
This prediction task is widely applicable in practice, especially in online settings with a large amount of user-generated data. A familiar example is the collaborative filtering system by which Netflix leverages the preferences of a large user population to recommend new films to a customer after observing his or her viewing history. For our purpose, the prediction task provides a more straightforward algorithmic measure than statistical measures such as perplexity and likelihood, which are commonly adopted in machine learning, but not applicable to algotihms that don't reconstruct a statistical model.

5.2 Recovery task

For the algorithms that reconstruct a topic matrix $\hat{A}$ in the process, we also measure how close $\hat{A}$ and A are. For each learned topic $\hat{A}_i$ and real topic A_j, we compute $cos(\hat{A}_i, A_j)$, then find a maximum matching between the learned topics and real topics. We evaluate the average cosine similarity between the matched real and learned topics. We also carry out the above computation using total variation distance between distributions, and get same qualitative results between algorithms.

6. ALGORITHMS

6.1 Previous Methods.

We adapt the methods to generate a word distribution for each test document, and output the s most likely unseen words according to this distribution.

- **Baseline:** The word distribution for every test document is the distribution over words in the training set.
- **KNN:** KNN finds the k most similar training documents to a testing document, where similarity is defined as the cosine between the two documents as vectors in the word space. For a testing document, KNN uses the distribution over words in its k closest training documents to predict. Notice baseline is just KNN with k equal to the number of training documents.
- **LDA:** It is NP-hard to find the maximum likelihood fit to the LDA model, so in practice, the prevailing approach to learn LDA model is local search. We adopted the algorithm implemented by Blei [3] using variational EM (see [6]).

  The LDA algorithm has two procedures,*"estimation"* and *"inference"*. *"estimation"* takes a collection of documents as input, and outputs estimated model parameters for the corpus, in particular the estimated term-topic matrix. *"inference'* takes a collection of documents as well as a LDA model, and outputs an inferred topic mixture for each document. For our prediction task, we use the estimated term-topic matrix of the training set and the topic mixture for each testing document to estimate the most likely unseen words.

- **LDA(MALLET)** We also experiment with another implementation of LDA in Mallet [11] using fast Gibbs sampling [19]. The implementation in Mallet follows a similar *"estimation"* and *"inference"* workflow.

- **LDAT, LDAC** For generated LDA datasets, we have two "cheating" algorithms as benchmarks. LDAT knows the real term-topic matrix A of the model and uses the *"inference"* routine to find a word distribution for each document. LDAC knows the real term-document matrix M for each testing document, and uses the real word distribution for a test document. LDAC is the best we can do given sampling noise.

- **K-means:** We partition the training documents using k-means, with $1 - \cos(w, w')$ as the distance between documents w and w'. We use the k cluster centers as the term-topic matrix, and uses the *"inference"* procedure of BLei's LDA implementation to produce a word distibution for each test document.

- **LSI**. We compute the best rank-k subspace approximation of the document-word matrix for the training set. For each test document, we project it onto the subspace, and get a word distribution by shifting and normalizing its projection.

6.2 Projector

Projector is our new algorithm that builds upon LSI, and reconstructs a term-topic probability matrix $\hat{A}$. The motivation is that SVD is computationally more efficient than the LDA algorithm, and has a clear geometric interpretation, but doesn't recover the topics as distributions of words. We aim to start from the subspace computed by SVD, and use some straightforward operation to construct the topics. The algorithm is as follows

Input $\hat{M}$: observed distributions of training documents, k: number of topics, δ: algorithm parameter

Shift Shift the training documents to be centered at the origin.
$center = \frac{1}{n}\sum_{i=1}^{n}\hat{M}_i$
$\hat{M}_i = \hat{M}_i - center \qquad \forall i = 1, \dots, n$

SVD Compute the U, the best rank $(k-1)$-dimension approximation to the column space of $\hat{M}$
Project all $\hat{M}_i$'s to the subspace U, denote V_i as the projections.

Clustering Use k-means to cluster the V_i's into k clusters, where in the k-means algorithm the distance between two points x, y is defined as $1 - cos(x, y)$.
Let $C_1, \dots, C_k$ be the centers of the k clusters (center in the sense as in euclidean distance).

Scale Scale $C_1, \dots, C_k$ by the smallest common scalar so that δn of $V_1, \dots, V_n$ are contained in the hull with $C_1, \dots, C_k$ as vertices.

Whitening Make all C_i distribution over words: Map all the C_i's to the original word space, and then set $C_i = C_i + center$, truncate the negative entries in C_i to be 0, normalize C_i so the sum of entries is 1.
Return $\hat{A}_i = C_i$ be the recovered topics.

We illustrate in figure 2 how our algorithm works using the a visualization on two datasets with $k = 3$ topics.
The design of our algorithm is based on geometric intuitions of the documents as points in the high dimensional word space. With LDA as the underlying generative model, if we consider each document as its true underlying distribution over words without the sampling noise, they reside in the $(k-1)$-simplex subspace with the k topics as vertices. Our *Shift* and *SVD* steps aim to approximatley find this $(k-1)$-dimensional subspace.
After projecting all documents onto the subspace, the projected points lie approximately inside the convex hull of the k topics. The origin also lies inside the convex hull, and the k topics are along the pointy directions. We heuristically find the pointy directions using k-means with distances defined based on cosine similarity.
Documents are convex combinations of the topics. The more interior points have more balanced weights on a relatively large number of topics, while points further away from the origin have large portion from a few topics, and the vertices are pure topics. The *Scale* step exploits this intuition by starting from the cluster centers, which gives the pointy directions, and pushing the centers away from the origin to find a convex hull approximately contains all the documents. As long as there is a slightly stronger presence of one topic in a cluster center, the scaling will amplify this topic. Finally we do some post processing to get valid distributions over words using the corners.
We use estimated $\hat{A}$ and the inference procedure of the LDA algorithm to predict words for testing documents. We use the inference procedure of LDA since LDAT also uses it, so we can attribute the performance difference between LDAT and Projector to the quality of $\hat{A}$ compared to the real topics.
We also experimented with a version of projector which does not use LSI as a first step; it just proceeds with k-means, then we project the documents into the subspace that contains the means and scale as above. The results were quite similar to the results above so we do not include them here.

7. EXPERIMENTAL RESULTS

We generated datasets from the LDA model for a range of parameters, and tested the algorithms discussed in previous section on the prediction task. For the LDA algorithm and our Projector algorithm, we also have results for the recovery task.
In table 1, we give results for the various algorithms on the uniform α and β datasets. Recall the prediction task is to predict $s = 3$ unseen words for each documents, and the numbers shown in the table are the average precisions. Here, we see that Projector is the best of all non-cheating algorithms. In table 2, we give results of the recovery task on these datasets for Mallet LDA, k-means, and Projector. Recall the score for the recovery task is the average weight of the max weight matching between learned topics and real topics, with weights being the cosine similarities. Typically, we saw that topic cosine similarity correlates with prediction performance.

sig_topics	sig_words	Baseline	LSI-15	kmeans-15	knn-25	LDA-15	LDAC	LDAT	MalletLDA-15	Projector-15
5.09	19.4	0.21	0.5	0.23	0.46	0.28	0.57	0.54	0.52	**0.54**
3.03	199	0.06	**0.18**	0.06	0.15	0.06	0.23	0.23	0.15	**0.18**
5.01	53.6	0.11	0.3	0.11	0.27	0.12	0.36	0.34	0.31	**0.32**
1.19	19.73	0.04	0.81	0.81	0.81	0.73	0.84	0.84	0.79	**0.82**
7.42	55	0.17	0.21	0.17	0.19	0.17	0.3	0.26	**0.25**	0.23
1.19	206.13	0.06	**0.32**	0.29	0.31	0.27	0.32	0.32	0.29	**0.32**
2.46	53.2	0.09	0.53	0.48	0.51	0.49	0.56	0.56	0.52	**0.54**
7.36	20.6	0.22	0.35	0.22	0.3	0.22	0.48	0.41	**0.41**	0.4
3.06	21.53	0.15	0.63	0.56	0.59	0.6	0.69	0.66	0.65	**0.67**
3.07	120.4	0.07	0.25	0.1	0.24	0.08	0.3	0.28	0.23	**0.26**
2.48	119.53	0.06	0.29	0.18	0.29	0.12	0.32	0.32	0.28	**0.3**
2.51	204.33	0.06	0.19	0.08	0.15	0.07	0.24	0.23	0.17	**0.22**
3.06	53.53	0.11	0.48	0.33	0.46	0.39	0.52	0.5	0.48	**0.5**
5.07	205.67	0.07	**0.09**	0.06	0.06	0.07	0.14	0.13	0.07	**0.09**
1.2	51.6	0.04	0.68	0.64	**0.69**	0.6	0.71	0.7	0.68	**0.69**
5.05	120.87	0.08	0.18	0.08	0.14	0.08	0.23	0.21	0.16	**0.2**
7.37	116.53	**0.1**	0.09	**0.1**	0.09	**0.1**	0.17	0.15	**0.1**	**0.1**
2.5	19.4	0.12	0.68	0.64	0.66	0.61	0.75	0.73	0.68	**0.72**
7.39	208.13	**0.08**	0.05	**0.08**	0.05	**0.08**	0.11	0.1	0.07	0.06
1.18	118.07	0.06	**0.48**	**0.48**	**0.48**	0.42	0.49	0.49	0.47	**0.48**

Table 1: Results of the prediction task. Bold indicates champion non-cheating method for each row.

sig_topics	sig_words	LDA-15	MalletLDA-15	Projector-15	kmeans-cosine	LDA-cosine	Mallet-cosine	Projector-cosine
5.09	19.4	0.28	0.52	**0.54**	0.14	0.62	**0.98**	**0.98**
3.03	199	0.06	0.15	**0.18**	0.45	0.48	0.67	**0.88**
5.01	53.6	0.12	0.31	**0.32**	0.23	0.41	0.94	**0.95**
1.19	19.73	0.73	0.79	**0.82**	0.15	0.93	0.99	**1.0**
7.42	55	0.17	**0.25**	0.23	0.23	0.37	**0.76**	0.73
1.19	206.13	0.27	0.29	**0.32**	0.45	0.81	0.83	**0.94**
2.46	53.2	0.49	0.52	**0.54**	0.22	0.84	0.97	**0.98**
7.36	20.6	0.22	**0.41**	0.4	0.14	0.39	**0.98**	0.94
3.06	21.53	0.6	0.65	**0.67**	0.15	0.91	**0.99**	**0.99**
3.07	120.4	0.08	0.23	**0.26**	0.34	0.49	0.87	**0.94**
2.48	119.53	0.12	0.28	**0.3**	0.34	0.59	0.88	**0.95**
2.51	204.33	0.07	0.17	**0.22**	0.45	0.49	0.75	**0.9**
3.06	53.53	0.39	0.48	**0.5**	0.22	0.75	0.96	**0.97**
5.07	205.67	0.07	0.07	**0.09**	0.45	0.46	0.42	**0.69**
1.2	51.6	0.6	0.68	**0.69**	0.23	0.84	0.98	**0.99**
5.05	120.87	0.08	0.16	**0.2**	0.34	0.41	0.78	**0.86**
7.37	116.53	**0.1**	**0.1**	**0.1**	0.33	0.39	0.43	**0.58**
2.5	19.4	0.61	0.68	**0.72**	0.14	0.86	**0.99**	**0.99**
7.39	208.13	**0.08**	0.07	0.06	**0.46**	**0.46**	0.23	0.41
1.18	118.07	0.42	0.47	**0.48**	0.33	0.84	0.93	**0.97**

Table 2: Bold indicates champion results and we use cosine similiarity where matching to model topics uses maximum weight matching.

In figure 3, we examine results for datasets that are more difficult for the various algorithms (see section 8). In particular, the product of *sig_words* and *sig_topics* ranges from $\frac{1}{2}$ to 2 times the vocabulary size.
For α chosen so that the topic distribution obeys a power law, we present results in table 3. Here, we see that projector again performs robustly well. We note that the Mallet implementation optimizes over varying values of α.
Finally, table 4 contains results for the word prediction task on the standard real world datasets we discussed earlier.
There we see that LDA is no longer dominated by LSI and Projector. Still, nearest neighbor's performance dominates. We note that nearest neighbor prediction requires access to all the original data where topic model based algorithms substantially compress the original data.
In table 7 we show the most popular words in a sampling of topic vectors recovered by Projector in the AP dataset.

7.1 Runtime

Figure 4, compares the runtimes of Projector with Mallet LDA and suggests that Mallet's runtime increases linearly with respect to average document length and the number of documents. Projector appears sublinear in each, which suggests that the our timings are influenced by fixed startup costs. Not shown, is Mallet's better scaling with respect to vocabulary size as Projector uses dense matrices. Projector's dependence remains linear in this case.

8. ANALYSIS

In the previous section, we defined the notion of typical number of words in the support of a topic, or sig_words, and the notion of a typical number of topics in a document. Clearly, if both get very large one gets to a trivial topic model where each document is generated by choosing words independently from a single probability distribution.
On the other hand if sig_words and sig_topics are small each document should have relatively small support compared with the corpus. Thus, in such cases we should easily distinguish the data from the trivial topic model.
We will proceed by showing that the probability of cooccurrence in documents of two words i and j differs significantly from in the trivial topic model. This can be represented as a matrix which refer to as the co-occurence matrix.
The expected co-occurence matrix can, of course, be calculated precisely from the topic and document distributions. But we give simple, even trivial, calculations that provide insight based on a simplified topic model.

8.1 A Uniform Topic Model.

We proceed by calculating the difference in co-occurence matrices of a corpus generated by a nontrivial topic model from the trivial topic model.
Let k be the total number of topics. Let m be the vocabulary size. Let t be the number of topics in a document and assume that each word is chosen uniformly from these t topics. Each topic is a uniform distribution over w words. Let l be the number of words in the document.
We examine word cooccurrences for w_i and w_j: that is, the probability that two words generated in a document are w_i and w_j. Note that in a document of length l, there are $\binom{l}{2}$ possible cooccurences between w_i and w_j.
We compute the probability of a word cooccurrence for data from a topic distribution versus the trivial model with the working vocabulary: the union of significant words in all topics which roughly has size $v = \min(m, kw)$.
For two words from the same topic, a document contains their common topic with probabiltiy t/k, and the probability that both words are chosen is $(1/tw)^2$. Thus the probability of cooccurence from being in the same topic is $\frac{t}{k}(1/tw)^2$. We assume that the background probability that the two words cooccur in the other case (or in the case that there are no topics) is $1/v^2$. [1]
When the background co-occurrence is much smaller than the topic cooccurrence the topic model should be easy to learn. For example, when $\frac{t}{k}1/(tw)^2 >> 1/v^2$ or when $1 >> (\frac{kw}{v})^2\frac{t}{k}$, then we should have good performance. In figure 5, we plot this ratio against the performance of the projector algorithm and see that things degrade as this value increases. We note that technically even under the weaker condition that $tw < v$, an algorithm could information theoretically determine the topics with enough data but the benefit over the baseline algorithm will be small.

8.2 Dependence on document length and number.

The basic unit for determining the co-occurence values is the number of word pairs in the corpus. The number of word pairs grows quadratically with number of words in a document and linearly with number of document. This would indicate that the performance should improve more as document length grows as compared with document number. We note that the pairs inside the document are not independent but there is an analysis that shows a nonlinear benefit.

8.3 An external measure.

For real world data, there is no access, of course, to the parameters used in the discussion above. We instead use the chi squared measure on the cooccurence matrix, W. The analysis above is really just an anlysis of the cooccurence matrix. The Chi Squared measure is defined as

$$\sum_{i,j} \frac{(W_{i,j} - E_{i,j})^2}{E_{i,j}},$$

where $E_{i,j}$ is the expected number of cooccurrences if documents are generated from a trivial topic model with a word distribution equal to the marginal distribution of the data.
We remove all words for the corpus that occur infrequently and compute the Chi-squared measure on the corresponding co-occurrence matrix. In figure 6, we see that this measure gives us reasonable insight on the generated data into when we get good performance on the prediction task.

9. CONCLUSION

In this paper, we made progress toward understanding the performance of algorithms on data generated by the LDA model. Using our prediction task, we were able to find an improved algorithm for this task as well as for learning the topics in a generated topic model. This prediction task itself

[1]We note that it is possible to set up topics so that the additional correlations one receives in one topic are exactly cancelled out by other topics. This setup corresponds to coding up a parity problem in the set of topics, but it seems unlikely to arise in any reasonable topic model. Still, the "rough" calculations here fail. Indeed, we emphasize that the arguments are heuristic.

sig_topics	sig_words	Baseline	lda-15	ldaC	ldaT	malletlda-15	projector-15	lda-cosine	mallet-cosine	projector-cosine
1.4	98.73	0.12	0.5	0.53	0.52	0.43	**0.51**	0.95	0.8	0.98
3.59	203	0.06	0.06	0.18	0.15	0.12	**0.13**	0.46	0.65	0.87
2.29	52.93	0.13	0.49	0.55	0.54	0.47	**0.5**	0.91	0.88	0.92
3.61	54.8	0.19	0.19	0.44	0.45	0.35	**0.43**	0.46	0.79	0.98
1.69	100.8	0.08	0.4	0.45	0.45	0.38	**0.44**	0.91	0.87	0.97
2.29	202	0.06	0.07	0.23	0.22	0.19	**0.21**	0.5	0.73	0.87
1.38	204.67	0.06	0.15	0.34	0.33	0.27	**0.3**	0.58	0.78	0.95
3.58	99.53	0.16	0.16	0.35	0.35	0.27	**0.33**	0.44	0.81	0.9
1.66	54.2	0.1	0.58	0.61	0.6	0.47	**0.58**	0.92	0.89	0.99
2.27	97.53	0.11	0.32	0.37	0.38	0.32	**0.36**	0.88	0.85	0.97
1.72	205.2	0.06	0.08	0.25	0.23	0.19	**0.23**	0.52	0.7	0.94
1.4	52.87	0.12	0.61	0.69	0.68	0.6	**0.64**	0.85	0.91	0.92

Table 3: Pareto distribution for topics. Bold indicates best non-cheating result and cosine similarities are included.

	Baseline	LSI-15	knn-15	lda-15	malletlda-15	projector-15
AP	0.21	0.3	0.28	0.26	0.26	0.24
Cacm	0.07	0.07	0.12	0.1	0.08	0.1
Cisi	0.13	0.13	0.16	0.15	0.15	0.15
Cran	0.18	0.27	0.29	0.25	0.25	0.24
Med	0.08	0.13	0.13	0.12	0.13	0.13
Nips	0.66	0.69	0.77	0.75	0.75	0.68

Table 4: Experiment results on real datasets. We pick the result of the best among a few parameters for each algorithm. We note that increasing the number of topics to 30 does not change the results.

may be a reasonable candidate for practitioners to use to evaluate algorithms that are trying to find interesting topics. We provided some rules of thumb for when algorithms can make use of topic structure in generated data. It is important to see if these rules of thumb extend to real world datasets. We are continuing on this aspect currently.
We note that we tested Hierarchical LDA on our task of predicting more words. Admittedly, we had to modify the algorithm for this task by sampling many hLDA hidden states to estimate the word distributions for the test documents. We found it to be substantially inferior to Gibbs LDA on real datasets.
Our framework is currently available on github and we are working to make it user friendly.

10. REFERENCES

[1] Animashree Anandkumar, Dean P. Foster, Daniel Hsu, Sham M. Kakade, and Yi-Kai Liu. Two svds suffice: Spectral decompositions for probabilistic topic modeling and latent dirichlet allocation. *CoRR*, abs/1204.6703, 2012.

[2] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning Topic Models âĂŤ Going beyond SVD. 2012.

[3] David M. Blei. lda-c, 2003.

[4] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, 2012.

[5] David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *NIPS*. MIT Press, 2003.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[7] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[8] Seungil Huh and Stephen E. Fienberg. Discriminative topic modeling based on manifold learning. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 653–662, New York, NY, USA, 2010. ACM.

[9] Adam Tauman Kalai, Ankur Moitra, and Gregory Valiant. Disentangling gaussians. *Commun. ACM*, 55(2):113–120, 2012.

[10] Percy Lang. Mlcomp. http://mlcomp.org, 2010.

[11] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[12] Indraneel Mukherjee and David M. Blei. Relative performance guarantees for approximate inference in latent dirichlet allocation. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *NIPS*, pages 1129–1136. Curran Associates, Inc., 2008.

[13] Christos H Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent Semantic Indexing : A Probabilistic Analysis. In *PODS*, 1997.

[14] Daniel Ramage, Christopher D. Manning, and Susan Dumais. Partially labeled topic models for interpretable text mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 457–465, New York, NY, USA, 2011. ACM.

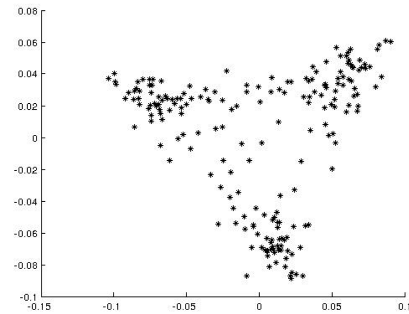
[15] Med, cran, cisi.

[16] Ap. http://www.cs.princeton.edu/ blei/lda-c/index.html.

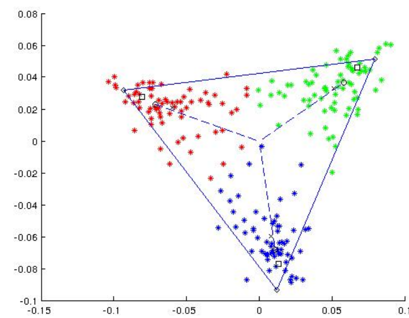
[17] Movielens. http://www.grouplens.org/node/73.

[18] Kos, nips. http://archive.ics.uci.edu/ml/datasets/Bag+of+Words.

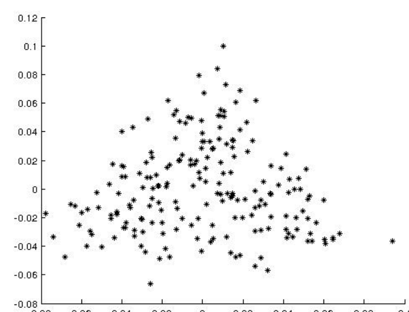
[19] Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 937–946, New York, NY, USA, 2009. ACM.



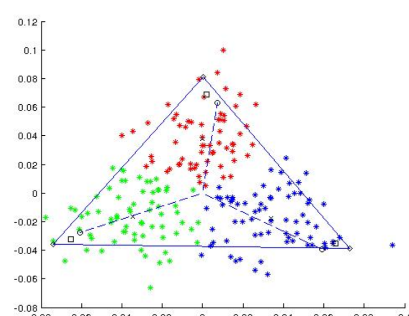
(a) $\alpha = 0.1, \beta = 0.25, k = 3$



(b) Algorithm illustration with $\delta = 0.8$



(c) $\alpha = 0.8, \beta = 0.25, k = 3$



(d) Algorithm illustration with $\delta = 0.8$

Figure 2: Illustration of Projector. The left figures are the V_i's after the SVD step. In the right figures, the black 'o's at the ends of dotted lines are the real topic, black $\times$ are the C_i's before scaling, black $\diamond$ are C_i's after scaling, and black $\square$ are the recovered $\hat{A}_i$'s. All points in the plot are after shifting and projected on the SVD subspace.

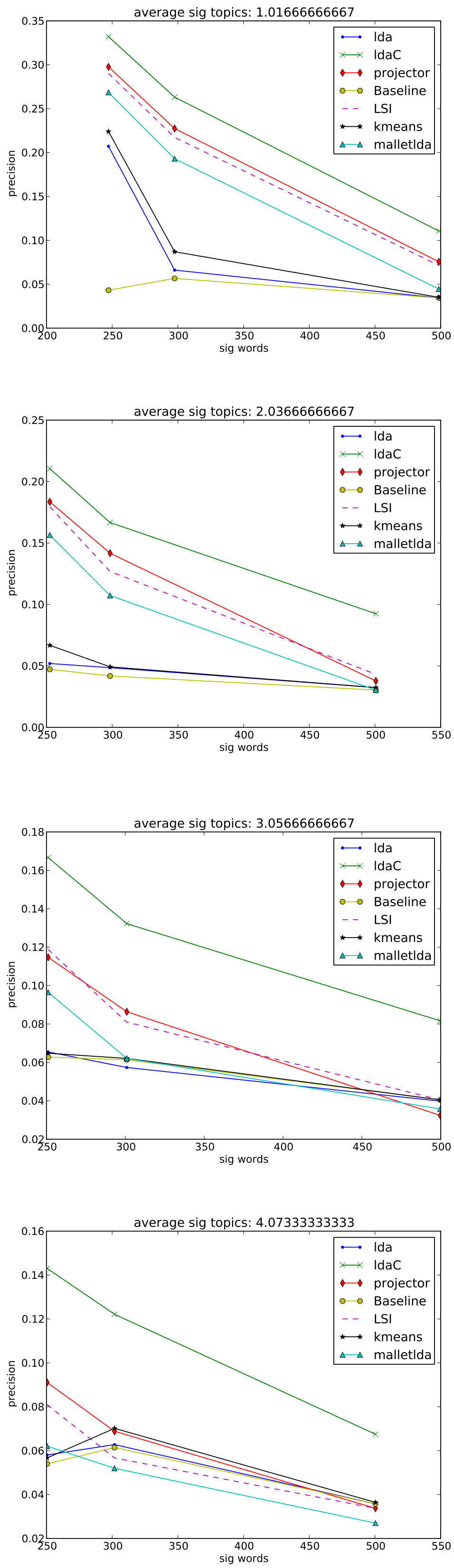


Figure 3: Results of various algorithms on 16 generated datasets, with $k = 20, n = 1000, m = 1000, l = 75$. Each subfigure has a fixed *sig_topic*, the plots are result of the prediction task versus *sig_word*

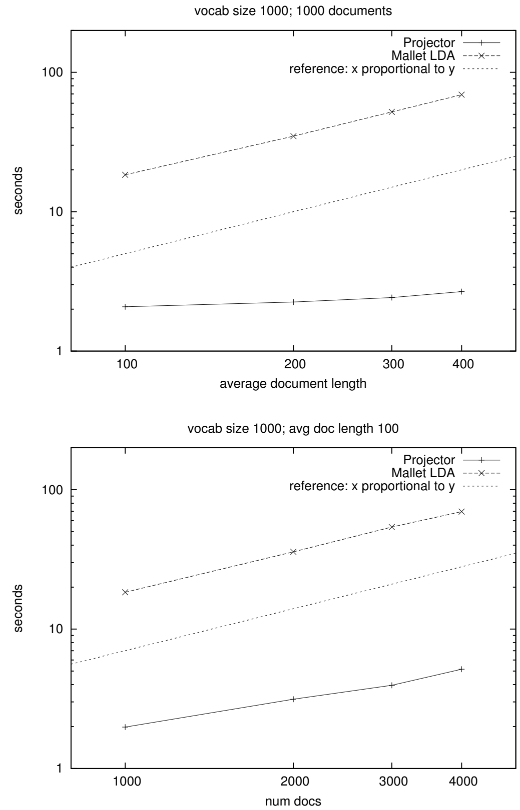


Figure 4: The runtimes of Mallet's LDA implementation and projector a plotted against the number of documents and the average document lengths in the figures above.

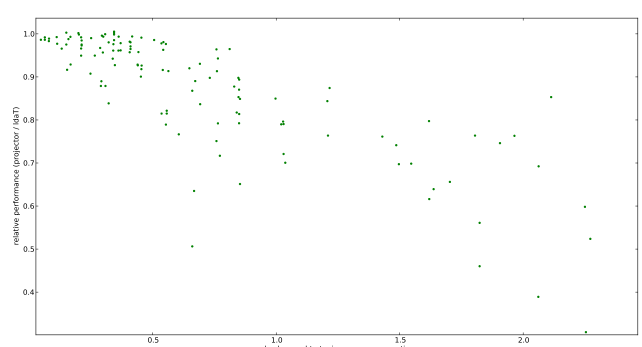


Figure 5: The performance of projector relative to LDAT falls off as the ratio of the background to topic cooccurrences increases.

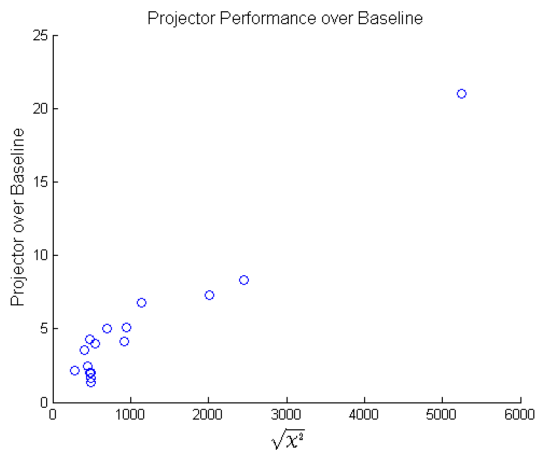


Figure 6: The x-axis is the log of the Chi Squared measure on the co-occurrence matrix for the words that occur with more than average frequency. Performance relative to baseline increases with increasing Chi squared value.

Civil Rights	Economics	Politics	Stock	Legal
years	percent	states	stock	court
west	rate	government	exchange	wednesday
black	year	united	points	federal
american	prices	union	closed	judge
war	reported	war	tuesday	trial
lived	increase	minister	average	convicted
jr	compared	countries	tokyo	state
chicago	rose	military	nikkei	death
story	report	meeting	close	attorney
social	higher	soviet	share	police
martin	highest	leader	financial	prison
progress	time	world	shares	district
series	index	president	index	charges
blacks	march	administration	volume	accused
side	mortgages	political	timesstock	man
neighborhood	billion	forces	times	charged
dream	figures	nations	million	filed
king	inflation	foreign	percent	case
influence	retail	prime	prices	years
remains	august	capital	investors	ordered

Figure 7: The top 20 words (ranked by probability) for 5 sparse topics recovered by Projector on the AP dataset. Topic names derived from top words.