Alternatives for engineering and evalution of algorithms for LDA.

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

ABSTRACT

In this paper, we empirically study standard algorithms for topic modelling using Latent Dirichlet Allocation. We also develop geometry based algorithms for learning the Latent Dirichlet Allocation (LDA) which are as effective as inference based methods while being more efficient. In particular, the most effective algorithms use a novel combination of dimension reduction, projection, k-means, and a scaling procedure.

We argue that the evaluation of topic modelling should include a prediction task that is closely aligned with the model. LDA, in particular, generates the words in a document. Thus, predicting missing words is appropriate for the evaluation of the LDA model, both when applied to a corpus and for genarated data. Moreover, this task is useful for recommendation systems, keyword suggestion, tag prediction, etc. Finally, for practitioners this evaluation task may be more indicative of the usefulness of various methods than internal measures such as perplexity or than indirect methods where other prediction algorithms are applied to learned topic models.

Finally, to empirically study algorithms, one needs to understand properties of datasets. We thus provide an analysis that predicts whether a corpus can be effectively modelled. For generated datasets, the prediction is based on the diversity of topics in a typical document, and the concentration of words in a topic.

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 such models as well as variations of this model have been proposed; see [BleiCACM] for a recent survey. The bulk of these methods are based on inference techniques; they proceed by using learning the parameters using methods which include from Gibbs sampling, EM, variational methods, and modifying the model for easier computation. The evaluation of the methods have proceeded typically along two lines. One is to train on a set of documents and then to evaluate the how well the resulting model predicts a test set, typically using perplexity as a measure. Another is to use the model as a set of features for some classification task, say for example document classification, and apply some learning method, say for example, using a support vector machine.

In this paper, we suggest modifying both the inference approach to learning topic models and suggest a different evaluation method. The first yields an improved algorithm for learning LDA, and the second allows for better understanding of learning methods.

In terms of inference algorithms, we take the view that the model parameters are geometric objects. That is, topic centers are simply points in the word space when the data consist of documents. This view is quite a traditional view of data, e.g., that it is generated from points in space under some noise model. This view of data has long been associated with algorithms such as nearest neigbors or k-means for classification or summarization. Unfortunately, as we show, k-means and nearest neighbors algorithms are inferior to existing LDA inference algorithms for LDA generated data. We combine k-means, along with dimension reduction, and a scaling step to produce an effective algorithm for this type of data. For real world data, our methods remain as effective as LDA algorithms and more effective than k-means, but fall short of nearest neighbor techniques (as do all topic modelling approaches.) We call refer to our algorithm and its variants as Projector. We note here that recent work in [] use linear algebraic methods, which can be viewed to some extent as geometric algorithms, have been proved to be effective given a polynomial amount of data. XXX - ONLY RELATED WORK?

The context in which we do our study is new and we feel is an important contribution to empirical practice as it stands today. We formulate the task of predicting a set of dropped out words in a document. This is a natural problem as viewing related words for document similarity, or more generally as posing the recommendation problem when the "document" consists of product purchases. Moreover, topic models in general and the Latent Direchlet Allocation topic model, in particular, generates documents word by word. Thus, a model's effectiveness for predicting a word from the document is both fair and is easy to understand as a raw score and allows for the fcomparison to any prediction method whether it is generative or not.

Finally, we note that some datasets are more difficult than others for all the methods. Rather than characterize this difficulty using the arcane parameters of the LDA model, we suggest that using more natural measures is more useful. In particular, we give an analysis that suggests that the typical number topics in a document along with the typical number of words in a topic gives a good indication of a datasets' difficulty.

Before proceeding, we note that this paper may appear unusual; are we proposing an algorithm, analyzing dataset complexity, arguing for more informative evaluations, or doing an experimental study of existing algorithms? Indeed, these things go very much together. The proof for a methodology for algorithm development is, after all, an effective algorithm.

2. RELATED WORK

An early version of topic models could be seen in latent semantic indexing which views a document as being about a subset of a larger set of topics. Topics are in turn are about (positive numbers) or not about (negative numbers) certain words [15]. They used this view to provide an algorithm based on linear algebra that learned the structure of their model. They also argued that this model provided some insight into the wide applicability of principal components analysis in data analysis. The algorithmic tool at the core of their work (and principal components) was the singular value decomposition.

Shortly thereafter, a probabilistic framework was provided by Puzicha and Hofmann [10]. They termed their method probabilisitc latent semantic indexing (pLSI), and modelled topics as probability distributions over words (no negative numbers in this description). Documents were then presumed to be generated as a mixture of these topics. The large number of parameters in the PLSI model motivated Blei, Ng and Jordan to provide a generative model for these parameters as well; they provided the enormously influential Latent Dirichlet Allocation(LDA) topic model [9].

The most closely related work in theory are the aforementioned algorithms provided by Arora et.al. [5] which learn pLSI under certain assumptions, and a breakthrough by Anandkumar[4] which gives a provably converging algorithm for learning LDA with polynomial data requirements. There is a long literature in learning mixtures of distributions in statistics and more recently in theoretical computer science. See [11] for a recent breakthrough on learning mixtures of Gaussians and a discussion of this field.

There is ample work describing methods for optimizing LDA [7]. Again, recent a recent examples are described in [?], [14].

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 [7]. The hierarchical LDA model [8] 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 application.

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

This paper, in addition to providing some insight into the effectiveness of LDA algorithms, seeks to provide an infrastructure to evaluate effective topic modelling methods. We should point out that the machine learning community is made efforts in this regard. See for example, MLComp [12]. We also highlight the java based infrastructure that we used for topic modelling [13] which is part of a larger MALLET java package for machine learning.

3. LDA MODEL

LDA is introduced in [9] as a generative process. As a model it is widely applied in various domains sucha as information retrieval, collaborative filtering, vision, and bioinformatics. 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.

The basic idea is that there exist k underlying latent topics. Each document is a mixture of the latent topics, where the topic mixture is drawn from a dirichlet distribution. More precisely, there are n documents, m words, and k topics. The model has a $m \times k$ word-topic matrix A, where the i-th column A_i specifies a multinomial distribution on the m words for topic i. For a document w, we first choose $\vec{\theta}$, its distribution over topics, which can take values in the $(k-1)$-simplex, and has the following Dirichlet distribution

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

where $\vec{\alpha}$ is parameter of the model. 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, then the actual word $w_i \sim Multinomial(A_{z_i})$ is generated. Equivalently, in matrix form, there are the $m \times k$ word-topic matrix A, and $k \times n$ topic-document matrix W whose columns are drawn i.i.d from $Dirichlet(\vec{\alpha})$. The product $M = AW$ is the $m \times n$ term-document matrix where column M_i is document i's distribution on words. 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 columns 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 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.

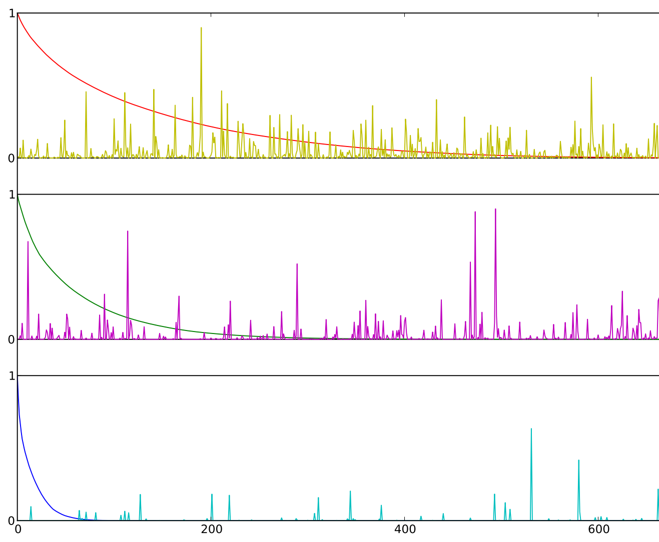


Figure 1: Plot of distributions on words for various β. $m = 1000$, each distribution is plotted along with its cdf after sorting the words by popularity. Refer to the y-axis on the right for the scaling of the distributions. In general, larger β values yield flatter distributions.

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 have used some standard real world datasets. We used the Classic-3 datasets (Cran,Med,Cisi) [3], a corpus (AP) from the Associated Press [1] pruning stop words and infrequent words, and a bag of words dataset (NIPS) from UC Irvine[2], MovieLens [**?**].

5. EXPERIMENTS

5.1 Prediction task

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 $p_t = 0.9, H = 30\%, s = 3$ 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 algortihms 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 method produces a word distribution for teach test document and outputs the most frequently unseen words in this distribution.

- **Baseline:** uses the training set word distribution.
- **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 predicts the s unseen words that are most frequent in its k closest training documents. Notice baseline is just KNN with k equal to the number of training documents.
- **LDA:** Davide Blei's implementation [6] based of variational EM uses an "estimation" phase to train a topic model on the training data. Then use the "inference" routine to infer a topic distribution for a test document which than uses the topic definitions to produce a word distribution for the document.
- K-means: Uses k-means with cosine similaritly to produce a topic matrix, and uses LDA inference to produce a word distibution for each test document.
- **LDA(MALLET)** Uses the implementation of LDA in Mallet based on Gibbs sampling [**?**]
- **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.
- **LSI**. Computes the best rank k subspace approximation of the document-word matrix. Then projects test document into the subspace to find a word distribution.
- **Projector**. We have two versions of Projector which is described below which produces a topic word matrix. Then LDA inference is used on each test document to produce a word distribution.

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. Our algorithm is based on geometric intuition of the documents as points in the high dimensional word space. 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.
$center = \frac{1}{n}\sum_{i=1}^{n}\hat{M}_i$
$\hat{M}_i = \hat{M}_i - center \qquad \forall i = 1,\ldots,n$

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 $C_1, \ldots, C_k$ be the centers of the k clusters (center in the sense as in euclidean distance).

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

Whitening Make all C_i distribution over words: $C_i = C_i + center$, 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. Notice after the *Shift* step, we want to find the best $(k-1)$-dimensional subspace since the columns from the topic-document matrix are from the $(k-1)$-simplex.
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, and then 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. EXPERIMENT 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. We experimented with k from 3 to 30, and in each case, a set of α and β to cover a wide range of *sig_word* and *sig_topic*. See figure 3 for the prediction results of the algorithms on a representative set of datasets.
Also see table 1 for prediction results on real datasets. There we see that LDA is no longer dominated by LSI and Projector. Still, nearest neighbor's performance dominates. In the appendix, in table 2 we show the most popular words in a sampling of topic vectors recovered by Projector in the AP dataset.
Typically, we saw the topic matching to correlate with prediction performance. See figure 5 to see this result.

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]

[1]We note that it is possible to set up topics so that the additional correlations one recieves in one topic are

	Baseline	GibbsLda	LSI	PLSI	knn-25	lda	projector
AP	0.14	0.21	0.24	0.15	0.25	0.2	0.21
Cisi	0.09	0.12	0.09	0.1	0.14	0.12	0.1
Cran	0.17	0.21	0.19	0.17	0.25	0.23	0.21
Kos	0.21	0.38	0.26	0.19	0.38	0.38	0.3
Med	0.08	0.11	0.16	0.08	0.17	0.11	0.16
NIPS	0.64	0.77	0.47	0.64	0.75	0.73	0.57

Table 1: Experiment results on real datasets. We pick the result of the best among a few parameters for each algorithm.

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 4, we plot this ratio against the performance of the projector algorithm and see that things degrade as this value increases.

8.2 Dependence on document size and length.

When the ratio is close to 1, the topic cooccurrence should be viewed as an additive advantage over the "background" probability of two words co-occurring, e.g., $1/v^2$ where v is the effective vocabulary size as defined before.

That is, for a pair of words not in the same topic the probability of co-occurrence is $1/v^2$ and for a pair of words in the same topic the probability of cooccurrence is $1/v^2(1-(t/k)) + (t/k)(1/tw)^2$. The latter summand being the contribution from when the common topic is chosen, the former being an approximation to the background.

Indeed with enough data the distribution will be distinguishable from a random one whenever $tw < v$. The question then is "How much data does one need?"

We define the density, Δ to be the ratio total number of co-occurrences in the documents to the total number of word, word pairs. For our model, this is roughly $\frac{n\binom{l}{2}}{\binom{v}{2}}$.

We consider a corpus with additive advantage $\epsilon = (t/k)((1/tw)^2 - (1/v)^2)$.

The total signal on a topic of size w is $\epsilon\Delta w^2$. The noise (standard deviation) of this process is roughly $\sqrt{\Delta w^2}$. As there are $\binom{v}{w}$ subsets the ratio of signal to noise needs to exceed $\sqrt{\log\binom{v}{w}} \approx \sqrt{w\log v}$ to properly distinguish a topic from one that may arise "on accident" (assuming a Gaussian distribution on the deviation). Thus, we would roughly need

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.

$$\frac{\epsilon\Delta w^2}{\sqrt{\Delta w^2}} \geq \sqrt{w\log v}.$$

That is, the density Δ should be larger than roughly, $\frac{\log v}{\epsilon^2 w}$.

Recalling that Δ is $(nl^2)/v^2$ where n is the number of documents and l is the number of words in a document. So, as the number of words in each document and number of documents increase so does the distinguishability. Moreover, longer documents are more advantageous than more documents.

In figure 5, we see that algorithms become increasing effective with either an increase in document size or number as suggested by these calculations. Also, we see from the first few points that doubling the document size was far more effecting than incresing the number of documents.

Finally, we note this is statistically distinguishable but to date not necessarily algorithmically distinguishable. For example, principal components analysis (e.g., LSI) has greater difficulty distinguishing distributions when there are many small topics. In particular, LSI finds "clusters" real or imagined that have the greatest excess of common pairs. Thus the algorithm must contend with the noise on a large fake topic, i.e., the noise is roughly $\sqrt{\Delta m^2}$ versus just the noise on a small topic $\sqrt{\Delta w^2}$. Thus, one needs Δ to be a factor of $(m/w)^2$ greater for LSI to learn the topics than would be required information theoretically.

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 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. This prediction task itself may be a reasonable candidate for practitioners to use to evaluate algorithms that are trying to find interesting topics.
We will provide our framework either in MALLET [13] or separately for this purpose.
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 working on this aspect currently.

10. REFERENCES

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

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

[3] Med, cran, cisi. http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets/.

[4] 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.

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

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

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

[8] 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.

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

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

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

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

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

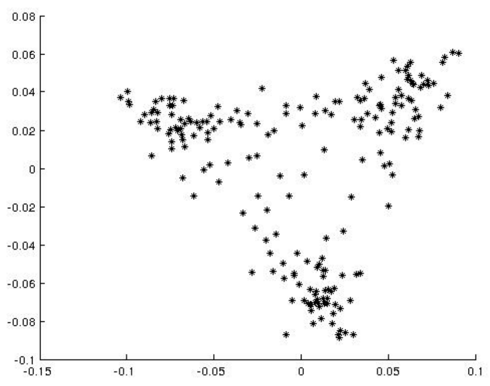
[14] Indraneel Mukherjee and David M. Blei. Relative performance guarantees for approximate inference in latent dirichlet allocation. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *NIPS*, pages 1129–1136. Curran Associates, Inc., 2008.

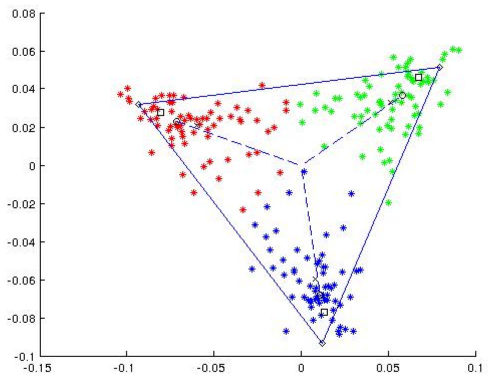
[15] Christos H Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent Semantic Indexing : A Probabilistic Analysis. In *PODS*, 1997.

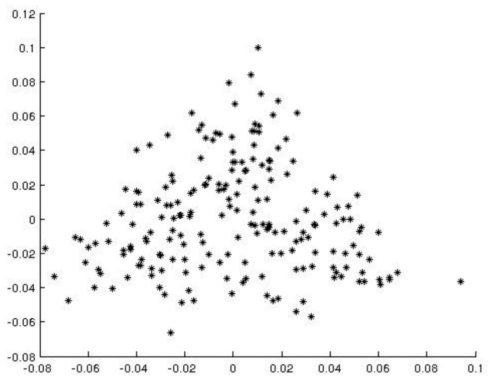
11. APPENDIX

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

Table 2: The top 20 words (ranked by probability) for 5 sparse topics recovered by Projector on the AP dataset. Topic names derived from top words.


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


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


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

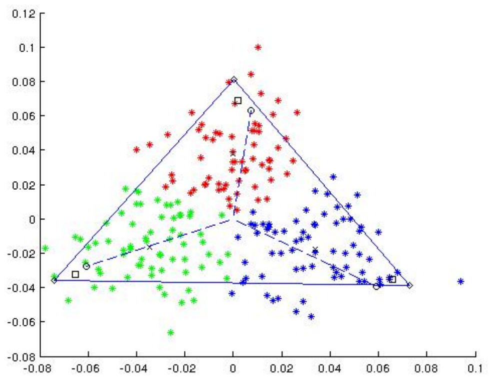
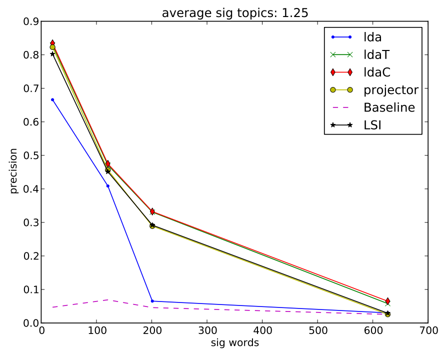
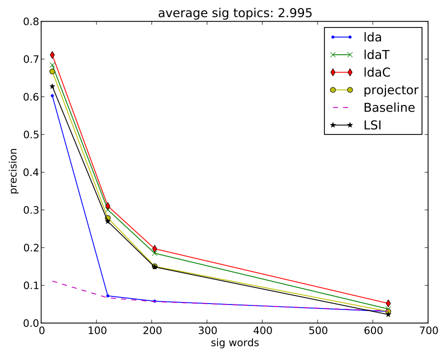
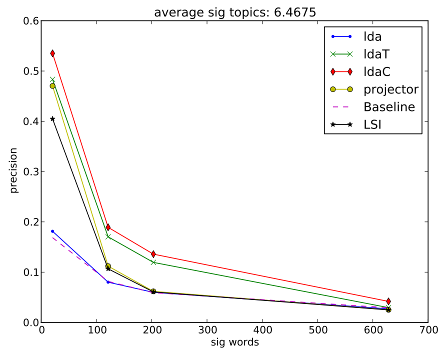

(d) Algorithm illustration with $\delta = 0.8$

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 $\Box$ are the recovered $\hat{A}_i$'s. All points in the plot are after shifting and projected on the SVD subspace.







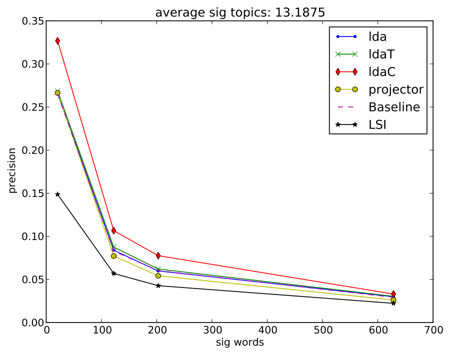


Figure 3: Results of various algorithms on 16 generated datasets, with $k = 20, n = 1000, m = 1000, l = 75$. Each subfigure has a fixed *sig_topic*, **the plots are result of the prediction task versus** *sig_word*

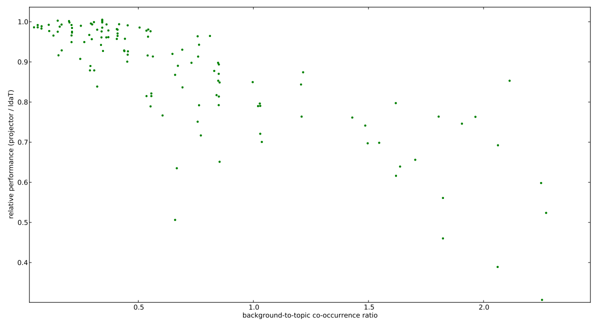
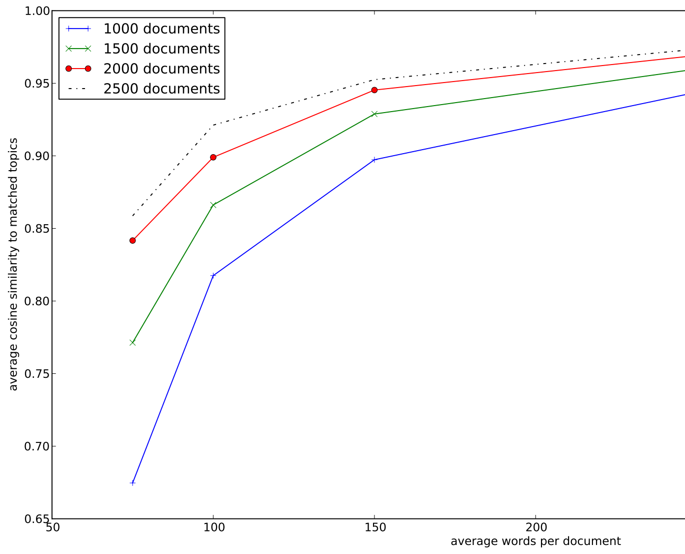
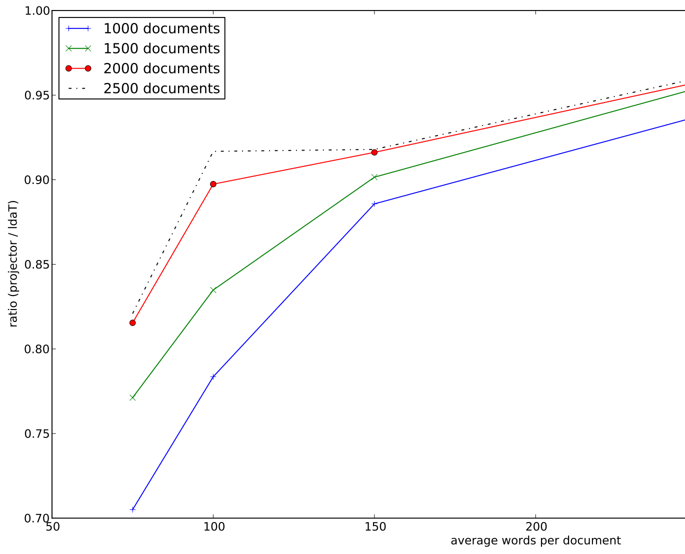


Figure 4: The performance of projector relative to LDAT falls off as the ratio of the background to topic cooccurrences increases.



(a)



(b)

Figure 5: Figure 5(a) above shows that as documents grow and document lengths grow, the projector algorithm learns the topics better. Figure 5(b) a corresponding increase in prediction performance comparing the LdaT algorithm that has perfect knowledge of the topics. The different lines correspond to different numbers of documents, and the *x*-axis corresponds to the number of words in the document.

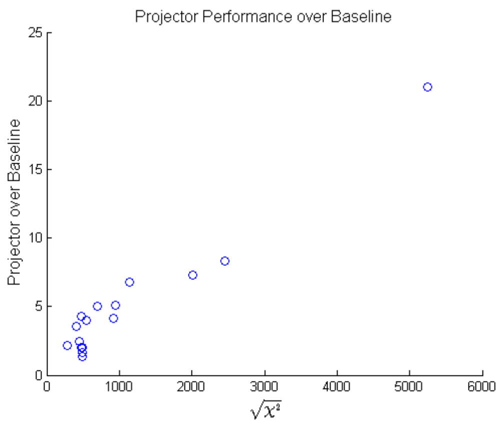


Figure 6: The x-axis is the log of the Chi Squared measure on the co-occurrence matrix for the words that occur with more than average frequency. Performance relative to baseline increases with increasing Chi squared value.