With the vast production of data comes a great desire to understand and make use of its structure. One avenue has been to use generative models to provide insight. Topic models, in particular, have been quite popular to investigate documents and an increasing number of other domains (See [**?**] for a survey of this field.)

A relevant, prehistorical, viewpoint is the concept of 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 [**?**]. 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 [**?**]. Their 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 Hoffman []. 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 [**?**].

Both pLSI and LDA have had enormous impact, but remain challenging both in terms of the appropriateness of the model and in terms of algorithms fitting the model. The evaluation of efficacy departed from tradition in machine learning in that one evaluated the learned parameters in terms of predicted probability of the corpus ("perplexity") rather than on the performance of some task. Indeed, this perspective remains compelling. For example, in [**?**] the evaluation of new algorithms still proceeds very much along these internal measures. This has merits in that one goal of topic modelling is to find something "interesting", for which performance is difficult to define. Still, the form of the measures suggested in the literature can be very complex. See, for example, the definition of "free energy" in [**?**].

This complexity can make it very difficult to see when such models are appropriate or whether the algorithms are effectively learning the model. We repeat for emphasis that there are *two issues here*: the applicability of the model, and the effectiveness of the algorithm. Neither is easy to evaluate in engineering the methods for data analytics.

We seek to provide insight into the performance of algorithms and to some extent understand the application to data. In contrast to the tendencies in topic modelling [**?**], we wish to provide simple methods for evaluation. In this paper, we primarily study the effectiveness of algorithms for discerning LDA models *when we know* they apply; for data generated by the model. We derive some insights and discuss some real world datasets.

An effective model should lead to good performance on prediction. We thus

return to a simple prediction task as a primary figure of merit. Topic models, particularly LDA, generate documents word by word. A fair evaluation would be use a training set of documents and a subset of a document to predict an unseen word. This task, as we show, entails both learning the topics and the distribution of topics in a particular document. The "one-more" (or "k-more") word task is central to our study. This task also seems to have natural applications which we touch upon later.

A feature of this approach is that one can compare to methods not based on modelling. Thus, we can compare performance based on modelling topics to standard learning methods. In particular, we include nearest neighbor methods which have been found to be highly effective in the literature and in our experiments.

We found that for simulated data that the algorithms widely used for learning LDA appear to be reasonably effective, but traditional algorithms based on LSI are, in fact, *more effective.* We further engineered a new method combining LSI, k-means, and the inference part of LDA. This method is more robust than LSI and has the best performance of any method we tested.

We characterize the instances that we can learn, the instances where knowing the parameters give power (even when we fail to learn them), and finally where the model fails to give any predictive advantage at all even cheating. The resulting characterization is unsuprising: simply measure the independence or lack thereof of the selection of words in a particular document is key. On the other hand, taking advantage of such simple measures has been key to giving provable algorithms for numerous mixture problems ranging from learning mixtures of Gaussians [**?**], to hidden markov problems [**?**], to recent provable method for learning topic models [**?**] and even the LDA model [**?**]. Indeed, we must first recognize order before distinguishing it from chaos.

Finally, we test the methods on some real data sets. Here, we find that topic models do distinguish from chaos. Indeed, we find that the LDA algorithms do better than both LSI and our engineered methods which dominate on the simulated data. This is quite interesting, and gives gives testament to the notion that topic modellers have worked extensively with real data; model or not. Still, the most effective algorithms in our experiments by far for real datasets is the nearest neighbor algorithms.

We proceed in the sequel by describing the LDA model in section ??, We give some experiments for a range of parameters in section ??. We explore the border of where the algorithms and than even the topic parameterization fail to provide performance. We explore real datasets in section ??. We conclude in section ??.

0.1 Related Work

The most closely related work in theory are the aforementioned algorithms provided by Arora et.al. [] which learn pLSI under certain assumptions, and the

breakthrough by Anand...[] for LSI. There is a long literature in learning mixtures of distributions in statistics and more recently in theoretical computer science. See [**?**] for a recent breakthrough on learning mixtures of Gaussians and a discussion of this field.

To some extent, we seek to provide an infrastructure to evaluate topic modelling. We should point out that the machine learning community has been active at producing infrastructure for evaluation. See for example, CompML []. A particularly related java based infrastructure for topic modelling is []. We plan to test their implementations in the final version of this paper.

0.2 An handle

We wish to experiments from the previous section to explain the change in performance of the algorithms. There we defined the notion of typical number of words in the supprt 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.

Here we derive an easy to compute a quantity based on sig_words, sig_topics, the total number of topics, and the corpus size to predict when learning the models becomes difficult.

Again, in the previous section, we provided results for a totally cheating method that knew all the parameters of both topic and documents. We also included a "topic" cheater method that knew the topic distributions while not knowing anything about document topics (other than the words in the document.) Finally, the remainder of the methods were only given the actual documents.

We view the first method as optimal. What more could one do, then know all the parameters? Thus, we wish to understand when knowing just the topic matrix is sufficient and when our algorithm performs worse than the topic cheater. We compute a quantity that is reasonably predictive of both events. Moreover, the point at which the algorithm performs slowly against the cheater would perhaps be when the algorithm fails to learn the topics well. We examine this aspect later in this section.

The quantity is a heuristic accounting of when the distribution is distinguished from chaos. That is, the structure of the document generation should have the probability cooccurrence in documents of two words i and j differ significantly from what would be expected if they were independently chose: $p_i \times p_j$, if p_i and p_j are their probabilities in the corpus.

This can, of course, be calculated precisely from the topic and document distributions. But we give simple calculations that provide insight based on a rather small set of parameters.

0.3 Uniform Cases.

Let k be total number of topics. Let v be the vocabulary size.

Let t be the number of topics in a document (assuming uniform distribution over selected topics.)

Let w be the size of a topic (assuming uniform distribution over selected words.)

Let l be the number of words in the document.

We assume that topics are generated a priori.

Sparse Case: topics largely disjoint.

Here, we assume topics are disjoint.

The number of significant words is $s = wk$.

For a document, t topics are chosen, and each word is chosen uniformly from w.

For two words in the same topic. The chance that any two words are this pair is.

$$(\frac{t}{k})\left(\frac{1}{tw}\right)^2.$$

For words in different topics, we have

$$((\frac{t}{k})^2(\frac{1}{tw})^2 = \left(\frac{1}{kw}\right)^2.$$

The difference is a facotr of t/k. When this is large we should expect to be able to recover the topics easily.

Medium Dense Case.

We assume that topics overlap a great deal and any particular word pair is not likely to be in more than one topic together.

We add the parameter of expected number of topics per word. This is multiplicity or $m = kw/v$.

To simplify the calculations, we assume that the few pairs of words share more than one topic.

For two words in the same topic. The probability of the any pair being these two is

$$\left(\frac{t}{k}\right)(\frac{1}{tw})^2.$$

For words in different topics, we have

$$\left(\frac{mt}{k}\right)^2 \left(\frac{1}{tw}\right)^2 = \left(\frac{1}{v}\right)^2 .$$

Here the difference is a factor of $\frac{t}{k}(tw/v)^2 = \frac{t}{k}(m)^2$.