On the Use of Linear Programming for Unsupervised Text Classification.

Mark Sandler
Department of Computer Science
Cornell University
Ithaca, NY, 14850
sandler@cs.cornell.edu

ABSTRACT

We present a new algorithm for large scale unsupervised text classification. Our method views each document as a sample of fixed size from a mixture model, and uses a novel L1-norm based theoretical approach due to Kleinberg and Sandler [14]. We show that our algorithm performs extremely well on data sets of 10^5 documents and more, and in particular out-performs Latent Semantic Indexing by a large margin. Furthermore, on some tests its prediction accuracy approaches that of *supervised* learning with training set of 5,000 or more documents. Unlike LSI, our algorithm produces a "well-behaved" projection in general, that in many cases does not require additional clustering algorithm to separate topics. We experiment with the arXiv- a collection of scientific abstracts and the 20 Newsgroup dataset - a small snapshot of 20 specific newsgroups.

1. INTRODUCTION

With growing availability of huge text collections, with documents on about every possible topic, the following natural question arises: is it possible to sort them into different categories without (or with minimal) human intervention? While a significant understanding of *supervised learning* - the setting where the algorithm has some pre-classified data as a part of the input, has achieved in the past years with a plethora of very fast and accurate methods[?]. However, the problem of unsupervised text classification, remained pretty much unsolved. To the best of our knowledge all existing clustering algorithms suffer from at least one of the following problems. Inefficiency - can not operate on collections larger than few hundred documents, and/or can not deliver reasonable classification accuracy.

With the present paper we believe that we make an important contribution toward practical and accurate algorithm for unsupervised text classification. We present a new technique for unsupervised text classification based on the joint work of the author with Kleinberg [14]. The approach that we use is based on the same idea that underlies LSI. Consider a term space, where each document can be represented as a single vector of term frequencies. Following the LSI approach, we find a good dimensional subspace and project each documents into it. The difference however, is that we build a different projection operator which is based on minimizing L_1 error and we show that it works much better than SVD projection. Furthermore as opposed to LSI, the basis of the topical space in many cases corresponds to underlying topics and no further clustering is needed.

KDD '05 Chicago, IL, USA

One important feature of our method is that it gives *provable guarantees of performance* within the model. In other words, under the assumption that the model adequately describes collection of documents and if particular conditions are satisfied we are *guaranteed* to recover almost accurate underlying term distributions with high probability. Such type of guarantees is not new: for example LSI does enjoy the same guarantees, and a lot of remarkable analysis have been done in the past years [1, 20, 19]. However the choice of model where L_2-norm of the error is minimized is not necessarily justified [10] and furthermore, recent work [11, 17, 6], suggest, that spectral analysis techniques might be not well suited for text classification and other problems where Zipf's law distributions are involved. Our approach is based on L_1 norm that seem to addresses this problem. It has also been observed by Ng [18] and Kanade [13] about preferability of L_1 over L_2-norm for particular learning problems. Our explanation is that L_2 norm puts too much weight on heavy components that in overall constitute only a small part of the system. For example in text classification problem, stop-words have much higher frequency of appearance than topic-specific terms, and yet in most cases they possess almost no information about topic structure.

The model we use for text collection is simply a generative mixture model, where each document is a sample generated from a mixture of overlapping distributions (topics). In contrast to latent Dirichlet allocation [4], we don't require our topic distributions to be of any specific shape, but our bounds depends on specific measure of independence which we define below.

Algorithm overview.

The algorithm starts by building an alternative 'mixture model', that would be equivalent to the underlying model. Term 'mixture model' is put in quotes because we relax definition to allow negative mixture coefficients (but resulting mixture must remain a probability distribution). Then it learns approximate coefficients using linear programming, and we use learned model to compute underlying term distribution. Found term distribution is useful on its own, as independent term-smoothing pre-processing step, and/or as a query enhancement mechanism that would automatically include synonymy search. However in this paper we will concentrate on the found mixture model, and each document mixture coefficients. Namely we use them for clustering documents into topics. Experimentally we have shown that when there are only "few" topics (e.g. 5 or less), classification according to the highest mixture coefficient is very effective. In this case our algorithm has closely approached the performance of supervised Support Vector Machines with *large* training dataset! To the best of our knowledge this is the first unsupervised algorithm which is capable of operating on large datasets, and to achieve such performance. All of our experiments were conducted on the collection of scientific abstracts - arXiv[1] and the 20 Newsgroup[2] dataset.

We also mention that accurate clustering is *provable* in the case of 2-topic classification (again, within the model): the basis we build is so closely related to the underlying mixture model, that it is sufficient to do classification. For multi-topic problem, our empirical evaluation suggest that it might be true. However larger datasets and/or additional analysis are needed to confirm or reject this statement in its full generality.

For problems with more than 6 topics, we evaluate performance of our algorithm as an intermediate dimensionality reduction step. We show that accuracy loss due our method is much smaller then the one of LSI, which is the only method known to us which would allow to do *fully*[3] unsupervised large scale dimensionality reduction.

The rest of the paper is organized as follows. In the next section we describe standard generative mixture model and introduce all necessary definitions and notation. In the section 3 we describe the algorithm along with intuition behind it. The forth section contains experimental results, and finally we conclude the paper with open problems and further directions.

Remark 1. Linear Programming has been used for clustering before. Particularly, correlation clustering introduced by Bansal, Blum and Chawla [3], uses linear and semidefinite programs to produce approximations for a specific graph clustering problem [23, 5]. However their application is very different from ours in a sense that programs were obtained as relaxations of integer program. Whereas for our method, as we show below, it is a natural (and optimal) setting.

2. GENERATIVE TEXT MODEL

We use *Multiple Cause Mixture Model*[14, 4, 20, 21, 22, 15, 9, 10] to describe the process of generating corpus. Each document is modeled as a sample of fixed size from a mixture of possibly overlapping distributions over set of all possible terms. The coefficients for the mixture are different for every document, but the underlying distributions are the same for the whole collection.

More formally, there is a collection of documents $\mathfrak{C}$ of the size $m = |\mathfrak{C}|$. All words used in the collection form a dictionary $\mathfrak{D}$ of the size $n = |\mathfrak{D}|$. Finally there are k topics and each topic c induces a probability distribution W_c over $\mathfrak{D}$. Each document d is simply a sample of a fixed size from a distribution D_d, defined by the mixture with coefficients $P_{d1}, \dots P_{dk}$. E.g. $D_d = WP_d$. Vector P_d of hidden mixing coefficients is called *relevance vector*, and D_d is called *term distribution* vector. Naturally we require that $\|P_d\|_1 = \|D_d\|_1 = 1$. Normalized vector $\tilde{D}_d$ of term frequencies as they occur in d is called a signature vector, and is the only information available about the document[4].

[1]See *http://www.arxiv.org*,

[2]See *http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html*,

[3]There are methods which perform dimensionality reduction after seeing some labelled data, for example [2]

Few more words on notation: We consistently use Latin letters c, w and d to denote arbitrary topics (clusters), topics(words) and documents respectively. We will also use these letters to denote matrix indices and unless specifically stated otherwise, they will "type-check" with the semantic meaning of the index. For example W_{cw} would denote the probability, topic c puts on the word w.

For a particular document d, we will use $\mathbf{d}$ and $\tilde{\mathbf{d}}$ to denote its term distribution and signature vector respectively and $\mathbf{p}$ for its relevance vector. E.g. $\mathbf{p}$ corresponds to a column P_d, and $\mathbf{d}$ is equal to $W\mathbf{p}$.

By slightly abusing notation we will be using $\tilde{x}$, to denote both random variable with distribution x, and a particular observation drawn from x.

3. ALGORITHM AND ANALYSIS

In this section we present our algorithm. While being very similar to the algorithm presented in [14], it is different in a number of key points. Therefore for the sake of completeness we describe our algorithm in full and present some basic intuition behind it. For a more formal exposure we refer to the original work. At the end of the section we present additional results about binary classification problem.

The algorithm runs in two stages. First it finds some "suitable" mixture model for a given collection $\mathfrak{C}$. This defines our topical subspace. We do this via analyzing a matrix of word co-occurrences. Second, for each document it recovers mixture coefficients in the built mixture model. Here we use Linear Programming to build optimal projection operator.

In order to maintain clarity, these steps are presented in the reverse order, as the second part can be viewed as independent learning technique, while the first relies on the ideas used in the second step.

3.1 Analyzing Mixture Model

Suppose we know mixture model, how one would find mixture coefficients? A traditional approach would be to use EM-based methods [16], which in general might converge to a suboptimal solution. Out method based on generalized pseudoinverses [14], and it provably recovers coefficients (within a small error) with high probability.

We begin with a simple definition.

DEFINITION 1 (GENERAL PSEUDOINVERSE[14]). *For arbitrary rectangular $n \times k$ $(n \geq k)$ matrix W, matrix W^* is called a generalized pseudoinverse of W if $W^*W = I$.*

If $n > k$ and if W^* exists, then it is not unique. For example one possible generalized pseudoinverse matrix is obtained from singular value decomposition[7]. However for our needs we will use a pseudoinverse which minimizes its maximal element.

[4]We are ignoring the order and word correlations here, but that's the property of "bag of words" formulation in general.

The reason why we need pseudoinverse and why we need this property is as follows. Consider an arbitrary document d of length s with term distribution and relevance vectors $\mathbf{d}$ and $\mathbf{p}$ respectively. Obviously $\mathbf{d} = W\mathbf{p}$, and thus

$$W^*\mathbf{d} = W^*W\mathbf{p} = \mathbf{p}, \tag{1}$$

This implies if we knew $\mathbf{d}$ we could find $\mathbf{p}$ and thus solve the classification problem.

However instead of $\mathbf{d}$ we have a signature vector $\tilde{\mathbf{d}}$, and while $\mathbf{E}\ \tilde{\mathbf{d}}\ = \mathbf{d}$, the vector $\tilde{\mathbf{d}}$ is sparse and thus is a bad approximation for $\mathbf{d}$. However in the (1), we only use elements of $\mathbf{d}$ as a weighted sum and therefore one can apply tail inequalities to bound an error of the sum of random variable This fact is formulated in the following lemma.

LEMMA 3.1 ([14]). *Let $\tilde{\mathbf{d}}$ be a document signature with at least s words in it, and let V be an arbitrary $k \times n$ matrix, with maximal element bounded by B, then if $s \geq \frac{B^2 k}{\varepsilon^2 \delta}$,*

$$\Pr[\|V\tilde{\mathbf{d}} - V\mathbf{d}\|_\infty \geq \varepsilon] \leq \delta.$$

PROOF. Note that $\tilde{\mathbf{d}}$ can be represented as a sum $\sum_{i=1}^{s} \tilde{\mathbf{d}}(i)$, of independent vectors, where ith term represents i-th word selection. The rest is a simple corollary of Chebyshev inequality and the union bound $\square$

This lemma implies that if a pseudoinverse with bounded maximal element exists, then it immediately gives us an algorithm to find a document relevance vector: multiply pseudoinverse W^* by signature vector $\tilde{\mathbf{d}}$ and this gives a good approximation to relevance vector. Obviously the smaller we can get a maximal element of W^* the better error bounds we obtain.

Now we again use the result of [14] which states that there is always a pseudoinverse such that its maximal element is bounded by $\frac{1}{\Gamma}$ where Γ is a quantity defined as

$$\Gamma(W) = \min_{\mathbf{y}\neq 0} \frac{\|W\mathbf{y}\|_1}{\|\mathbf{y}\|_1}$$

and called *independence coefficient*. Note that if $\Gamma = 0$, then W has linearly dependent columns and no generalized pseudoinverse exists. Conversely if $\Gamma = 1$, then distributions in the mixture are disjoint, and corresponding pseudoinverse is simply a "diagonal" matrix.

THEOREM 3.2 (GENERALIZED PSEUDOINVERSE [14]). *For any $n\times k$ matrix $W = \{W_{wc}\}$ such that $\Gamma = \min_{\mathbf{y}\neq 0} \frac{\|W\mathbf{y}\|_1}{\|\mathbf{y}\|_1} > 0$, there exists a generalized pseudoinverse $W^* = \{x_{cw}\}$ such that $\max |b'_{cw}| < \frac{1}{\Gamma}$. This pseudoinverse can be found in polynomial time.*

PROOF. Desired generalized pseudoinverse (if it exists) can be found by solving the following linear program,

$$\begin{cases} \sum\limits_{w} x_{c'w} W_{wc''} = \delta_{c'c''} & \text{for } 1 \leq c', c'' \leq k \\ -\gamma \leq x_{cw} \leq \gamma & \text{for } 1 \leq c \leq k,\, 1 \leq w \leq n \\ \min \gamma \end{cases},$$

for the existential part of the proof we refer to [14] $\square$

This theorem gives us the following way to do supervised learning. Use training data to learn topic distributions $\tilde{W}$ (by simply observing the term histogram for each topic). Then compute pseudoinverse $\tilde{W}^*$, using linear program above. Finally, learn mixture coefficients of unseen document d by applying pseudoinverse $\tilde{W}^*$ to the document signature $\mathbf{d}$.

ALGORITHM 1 (SUPERVISED LEARNING).
Input: Collection of documents $\mathfrak{C}$, with a prelabeled subset $\mathfrak{C}_0 \subseteq \mathfrak{C}$
Output: Classification c_d for each document d
Description:

1. *For each topic c, compute $\bar{W}_c$ as a word distribution in documents labeled with topic c.*
2. *Compute pseudoinverse $\bar{W}^*$ using linear program from theorem 3.2*
3. *For each unlabeled document d, compute $\bar{\mathbf{p}} = W^*\tilde{\mathbf{d}}$ and assign it to the topic c such that $\bar{\mathbf{p}}_c = \|\bar{\mathbf{p}}\|_\infty$*

We would like to emphasize here that the number of words in document needed to learn mixture coefficient is *independent* of the size of the dictionary. It only depends on desired confidence parameters and independence coefficient Γ.

Remark 1. It is also possible to use for projection a pseudoinverse obtained from Singular Value Decomposition (or any other). However the value of maximal element will be dependent on n in general case thus making tail inequalities not immediately inapplicable.

3.2 Finding "mixture model"

Note that equation (1) holds not only for matrix W, but essentially for arbitrary matrix V as long as for each document d, $\mathbf{d}$ can be represented in the form Vq. Indeed then:

$$V^*\mathbf{d} = V^*Vq = q.$$

Also theorem 3.1 applies to an arbitrary matrix with bounded maximal element. Thus in order to find term distribution it suffices to find matrix V which would satisfy the following: Columns of V are linear combinations of true W, and V has $\Gamma(V)$ bounded.

One source of vectors which are linear combinations of W, is a *co-occurrence matrix $\mathcal{R}$.*

DEFINITION 2 (COOCURENCE MATRIX). *Let $\tilde{\mathcal{R}}_{wv}$ denotes the number of times words w and v have occurred together in the same document. The $n \times n$ matrix $\mathcal{R} = \{\mathbf{E}\ \mathcal{R}_{wv}\ \}$ is then called co-occurrence matrix and the matrix $\tilde{\mathcal{R}} = \{\mathcal{R}_{wv}\}$ is called observed co-occurrence matrix.*

Note that as number of documents $|\mathfrak{C}|$ grows comparatively to the size of the word dictionary $|\mathfrak{D}|$, the observed co-occurrence matrix uniformly approaches to its expectation. In [14] authors argued that an algorithm, that tries to maintain large independence coefficient, by greedily choosing columns of normalized $\tilde{\mathcal{R}}$, will result in V such that $\Gamma(V) \geq f(\Gamma(W), k)$. However their algorithm while being feasible to run, still would take prohibitively long time (order of weeks on a desktop) on a large dataset.

We use heuristics to both speed up the algorithm and reduce the required sample complexity. First, we don't compute the full co-occurrence matrix, but only columns corresponding to the words which have occurred at least t times and are not in the stoplist. t is the parameter of the system and is set to approximately 1/10 of the dictionary size. This serves as both a speeding up and an error reduction measure: don't analyze words, if there is no enough statistics about them. Resulting matrix is called *truncated* co-occurrence matrix[5] Second, our independence coefficient is heuristically approximated as a minimal L_1 distance between columns.

Now we are ready to present the algorithm which finds the model.

ALGORITHM 2 (FINDING "MIXTURE MODEL").
Input: Observed co-occurrence matrix $\tilde{\mathcal{R}}$, number of topics $k \geq 2$, parameter t.
Output: Topical space V.
Description:

```
1. Remove columns from $\mathcal{R}$ corresponding to stoplist words, or having $L_1$ norm less than $t$. Normalize (w.r.t $L_1$ norm) remaining columns.
2. Let $(w_1, w_2) = \operatorname{argmax} \|\mathcal{R}_{w_1} - \mathcal{R}_{w_2}\|_1$, set $H_1$ to $\mathcal{R}_{w_1}$ and $H_2$ to $\mathcal{R}_{w_2}$.
3. For each $c$ in between 3 and $k$:
   (a) Find word $w$, such that column $\mathcal{R}_w$ would maximize $\min_{1 \leq i \leq c} \|H_i - \mathcal{R}_w\|_1$
   (b) Set $V_c = \mathcal{R}_w$ and iterate.
```

Our algorithm creates matrix V, which (in the limit) spans the same subspace as W, and that is sufficient for our LP algorithm to learn coefficients.

While due to heuristics used, our algorithm potentially might fail to find k sufficiently independent vectors. If it succeeds (and this is checked during the final pseudoinverse lookup step), it still enjoys the same type of guarantees as original algorithm. In our practice we have never encountered the case when independence coefficient Γ would be below $1/3$.

3.3 Joining parts together

We start with presenting an algorithm, and then show simple iterative modification which allows to significantly decrease required sample complexity.

ALGORITHM 3 (UNSUPERVISED CLASSIFICATION).
Input: Documents $\mathfrak{C}$, parameter t.
Output: Topical subspace H. D_d and P_d -estimated term distribution vectors and mixing coefficients.
Description:

```
1. Compute the matrix $\tilde{D}$ of documents signatures $\tilde{D}$.
2. Compute the observed co-occurrence matrix $\tilde{\mathcal{R}}(\tilde{D})$
3. Find the matrix $H$ - our alternative mixture model, using algorithm 2. Report $H$ as our approximate topical subspace.
4. Let $H^*$, be matrix found via LP from theorem 3.2
5. For each document $d$, with signature vector $\tilde{\mathbf{d}}$:
   (a) Compute $\bar{\mathbf{p}} = H^* \tilde{\mathbf{d}}$, and $\bar{\mathbf{d}} = V \bar{\mathbf{p}}$, and report those as mixture coefficients and term distributions respectively.
   (b) Classification: Assign $d$ to the distribution with the highest mixture coefficient.
```

[5]Note that we still keep full set of rows, including stopwords.

Because of the choice of H, has a pseudoinverse with bounded maximal element, then following the analysis of [14] this algorithm will provably recover term distributions as the number of documents goes to infinity. We refer reader to [14] for the proof details.

Furthermore, for binary classification problem as we show later the algorithm, even with the heuristics used, the algorith succeeds in *actual classification* (and not only recovering term distribution vectors) with high probability. For the general classification problem, while our algorithm certainly succeeds in many experiments, it is still an open problem to prove related result.

A simple modification of the algorithm which allows to significantly reduce sample complexity. Namely, we observe that our algorithm uses only few columns of co-occurrence matrix to build topical subspace, which, effectively means that we use only small fraction of documents to construct probability distributions, and hence significantly increase our sampling error.

To address this problem we use iterative modification of the algorithm. Classification results of the first iteration, are used as a pre-classified data, to train our *supervised learning* algorithm.

ALGORITHM 4 (ITERATIVE CLASSIFICATION).
Input: Collection of documents $\mathfrak{C}$.
Output: Classification c_d for every document d.
Description:

```
1. Using unsupervised algorithm compute intermediate classification.
2. Run supervised algorithm as if intermediate classification is our training data and obtain new intermediate classification on the full collection.
3. Iterate previous step 4 more times[6].
4. Report current intermediate classification as the final.
```

Remark 1. This procedure allows significantly reduce sample complexity for a given accuracy. However it becomes insignificant as the amount of available data grows, and the sampling error of the co-occurrence matrix decreases.

[6]Our choice of constant 4 was somewhat arbitrary. However experiments show that effect of further iterations is negligible.

3.4 Binary classification

In what follows for simplicity we assume that all documents have the same size s, each document is relevant to only one topic and each topic is has the same number of documents relevant to it[7]. We show that if two topic distributions are far apart e.g. $\|W_1 - W_2\|_1 \geq \gamma$. Then it is possible to do fully unsupervised classification of the collection, with an error approaching zero as the number of documents grows.

THEOREM 3.3 (BINARY CLASSIFICATION). *Suppose W is a mixture W of two topics, such that $\|W_1 - W_2\|_1 \geq \Gamma_0$, and ε, δ are two constants, If $|\mathfrak{C}| \geq f(\Gamma_0, \varepsilon, \delta)|\mathfrak{D}|$ and the size s of each document is at least $g(\Gamma_0, \varepsilon, \delta)$ for some appropriate polynomial functions f and g, then algorithm 3 with probability $1 - \delta$ will correctly classify at least $1 - \varepsilon$ fraction of all the documents.*

PROOF. First, since $\tilde{\mathcal{R}}$ uniformly approaches $\mathcal{R}$ as number of documents grows and since number of documents can be a function of the size of the dictionary, we can assume without loss of generality that $\mathcal{R} = \tilde{\mathcal{R}}$.

Now, condition $\|W_1 - W_2\|_1 \geq \Gamma_0$ implies that there are two words w_1 and w_2, such that $W_{1w_1} - W_{2w_1} \geq \frac{\Gamma_0}{2n}$ and $W_{2w_2} - W_{1w_2} \geq \frac{\Gamma_0}{2n}$. This gives an obvious lower bound on a distance between columns of the co-occurrence matrix chosen by the algorithm 2, and that in turn lower bounds $\Gamma(H)$. Now recall that all documents have underlying vectors either W_1 or W_2 and thus their representation in the space H would be of the form (α_1, α_2), where one of the components is nonpositive and another at least 1. The rest is a simple corollary of the lemma 3.1. $\square$

While we believe that similar approach of considering convex hull for co-occurrence matrix might help to do classification in multi-topic problem, and empirical evaluation supports it, even with the use of heuristic. It is an open problem if such a theorem could be proved for $k > 2$.

3.5 Multiple word occurrences

One obvious discrepancy between the model and the textual data lies in the way words occur in the document. Namely, if particular word w has appeared in a document, then it is somewhat likely to appear again, then some other word also relevant to the same topic. In the mixture model all consequent appearances have the same probability as the first one. Especially this is noticeable in the small documents, like paper abstracts. To accommodate this in our algorithm, we ignore second and all consequent appearances of the same word in the single document[8].

4. EXPERIMENTS

For our experiments we have used two datasets. The first one is arXiv collection of approximately 250.000 scientific abstracts on 12 different categories, which we used as our primary dataset. Our second collection is 20 Newsgroup, which contains 20,000 posts to one of the 20 groups.

With the first dataset our algorithm was tested as both unsupervised learning algorithm and dimensionality reduction step. In both cases it has shown outstanding results. Particularly on 4 topic classifcation problem, algorithm outperform LSI by far, and performed better than SVM with 400 (100x4) training documents. More details on this dataset below.

In the second dataset (20 Newsgroup), we do binary classification and compare it with results of [24]. While due the scarcity of data (recall that we need to build huge co-occurence matrix), we weren't ablt to run our algorithm on 5-topic classification problems, we did experiments with those sets in a setting where for each topic our algorithm was given "a hint"- a descriptive keyword, as an additional noise reduction measure.

For our tests we have used SVM^{struct} implementation of support vector machines due Joachims [12].Our algorithm was implemented using MatLab [9] and Python [10].

To build dictionary we have used words which have occurred at least 5 times and we used neither stemming nor stopword removal.

4.1 arXiv

Due to large amount of computer resources required to perform LSI on the full arXiv, we randomly selected 1/10th of documents from each category and used that to do comparison with LSI. For the next few experiments our threshold t is set to $t = 500$.

As our first experiment we classify 3780 documents on hep-ph (high energy physics - phenomenology) against 4370 documents on astro-ph (astrophysics). For each document we produce 2-dimensional vector of document relevancies to semantic dimensions. Illustrative representation of both algorithms outcome is presented on the figure 1. For LSI 2-nd and 3-rd singular vectors were used for a projection[11] While both sets are approximately linearly separable, as one can see clusters are very distinct for our method, whereas in LSI they are barely distinguishable and additional clustering is needed.

Second we do four topic classification between astro-ph, hep-ph, cond-mat (condensed matter, 4690 documents) and math (mathematics, 2300 documents). Our algorithm fully successes in recovering topics, and performs almost as good as trained SVM! For LSI further classification is needed. In order to minimize additional error, by introducing extra unsupervised method, we have trained SVM to do final discrimination between topics. Comparison results are presented in the table 1, we also included classification results for trained on a full dictionary SVM as a reference. Note that our method performs almost as good as SVM with 300 training documents per topic (e.g. 1200 total)!

Now we switch to the full arXiv. For all experiments to follow t is set to 5000. In the first experiment we do unsupervised clustering for 4 largest categories, and compare performance gain from iteration step. Quite expectedly effect for full arXiv is nearly not as dramatic as for 16.000 document subset. This supports our claim that iterations are helpful to reduce required amount of data, but they don't give any structural improvements .

[7]Neither of these conditions are crucial but they are helpful to maintain clarity.

[8]We also did experiments where all words were counted accordingly to their actual frequencies or tf.idf [8] measures, those produced very comparable, yet slightly worse results. We don't presented this comparison in the current paper.

[9]*http://www.mathworks.com*.

[10]*http://www.python.org*.

[11]Recall that first singular vector corresponds to stopwords

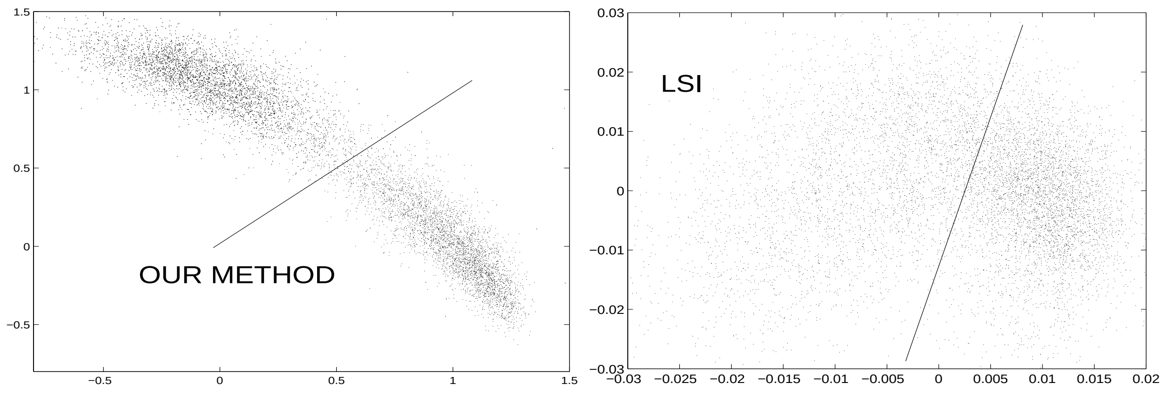

Figure 1: Binary classification between hep-ph (4370 docs) and astro-ph (3780 docs). Points represent documents after projection to topical subspace. Each line approximately separate documents of different categories. Note that for I-LP this line is always $x = y$, for LSI one have to do additional clustering to locate it.

#	LSI + SVM	I-LP+ SVM	SVM	I-LP
0	N/A	N/A	N/A	**90.2%**
1	56.0%	86.5%	49.8%	-
10	63.8%	90.0%	67.8%	-
30	60.5%	90.4%	82.0%	-
100	61.0%	90.6%	87.7%	-
300	68.5%	90.8%	91.2%	-
1000	67.9%	90.8%	93.3%	-

Table 1: SVM vs. LSI+SVM vs I-LP. accuracy of 4-way classification on a subset of arXiv of $\approx$ 15.000 documents. First column contains number of labeled samples per topic, supplied to SVM step. The last column is performance of our method, which does not need any training data. Rows 2-4 contain averaged over 10 runs accuracy.

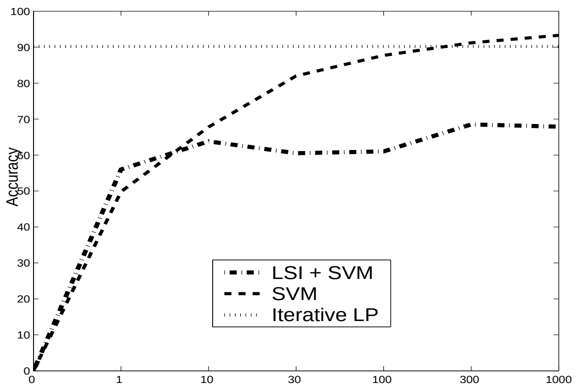

Figure 2: LSI vs I-LP+ SVM vs. SVM, 4-way classification on a subset of arXiv of $\approx$ 15.000 documents. X-axis contains number of labeled examples per category supplied to SVM (note that there is no SVM step for I-LP, so its performance is a constant).

Next we do classification in the datasets with 5 and 6 topics, for that we add category gr-qc (general relativity - quantum cosmology) and nucl-th (nuclear theory) to the test set. While accuracy has declined it is still remained very high. Particularly, considering unsupervised nature of the method and closely related categories (e.g. quantum cosmology vs. astro-physics). Detailed classification results are presented in the table 3

Our last experiment on arXiv is a test of I-LPas a dimensionality reduction step. To measure quality of our projections we compare performance of supervised method (SVM) on a full feature set with SVM trainied on projection subspace. For that we try 4-, 6- and 10- categories test cases. The 4-category dataset contains the same categories as above. For the 6-category we add: gr-qc, nucl-th. For the last one we add four more: quant-ph, nlin, physics and hep-th. The largest dataset contains ≈ 232.000 documents. Results are presented in the table 4.

Remark 1. Unsupervised classification of 6 and more topics proved to be difficult for our algorithm, the classification had less than 50% accuracy on one or more topics. One such example is presented in the table 3. While we suspect insufficient amount of data to be the major reason, it might also be the point where our heuristic algorithm starts to break, and/or mixture model becomes less accurate description of textual data. Future exploration of those reasons is a very interesting next step.

iter #	16.000 docs	160.000 docs
0	72.1%	87.5%
1	83.8%	90.9%
2	87.5%	91.3%
5	90.2%	91.3%
10	90.7%	91.3%

Table 2: Effect of iterations as amount of available data grows. In both cases classification is done on the same topics (hep-ph, cond-mat, math, astro-ph).

	astro-ph	cond-mat	hep-ph	math	Precision
Cluster 1	38529	486	407	52	97.61%
Cluster 2	3056	45531	3274	803	86.46%
Cluster 3	2755	444	33710	96	91.10%
Cluster 4	362	1075	534	22416	91.92%
Recall:	86.19%	95.78 %	88.89 %	95.93 %	91.3 %

	astro-ph	cond-mat	gr-qc	hep-ph	math	Precision
Cluster 1	28197	166	63	122	22	98.69%
Cluster 2	813	41368	316	1926	413	92.27%
Cluster 3	15177	5130	9716	9844	1565	36.63%
Cluster 4	472	234	89	25748	83	96.70%
Cluster 5	43	638	809	286	21284	92.30%
Recall:	63.08%	87.02 %	88.38 %	67.89 %	91.09 %	76.77 %

	astro-ph	cond-mat	gr-qc	hep-ph	math	nucl-th	Precision
Cluster 1	28610	182	84	92	31	28	98.56%
Cluster 2	2339	24422	1655	3945	2431	1902	66.56%
Cluster 3	9289	1549	8504	11767	1031	512	26.04%
Cluster 4	366	212	45	20584	74	4440	80.03%
Cluster 5	30	235	502	98	19666	24	95.68%
Cluster 6	4068	20936	203	1440	134	1561	5.5%
Recall:	64.00 %	51.38 %	77.36 %	54.27 %	84.16 %	18.41%	59.7%

Table 3: Confusion and precision/recall tables for 4-, 5-, and 6-ways (bottom) classifications. The rightmost bottom number is the total accuracy (e.g. fraction of documents classified correctly.) Datasets are of the sizes $\approx 154,000$ and $\approx 165,000$ and $\approx 173,000$ abstracts respectively. Order of clusters in the table has changed to maintain a 'diagonal shape' of the table, as our algorithm by itself has no knowledge of true cluster order.

	4 topics ($\approx 153,500$ documents)		6 topics ($\approx 173,000$ documents)		10 topics ($\approx 232,300$ documents)	
#	I-LP+SVM	SVM	I-LP+SVM	SVM	I-LP+SVM	SVM
5	89.6%	65.6%	68.8%	51.3%	50.4%	39.7%
10	90.9%	71.9%	74.7%	61.0%	55.6%	47.2%
30	91.0%	82.1%	74.9%	71.7%	58.1%	59.8%
100	91.7%	88.2%	78.7%	81.2%	60.7%	66.6%
300	91.8%	91.4%	80.2%	84.9%	65.7%	70.1%
1000	92.2%	93.7%	80.5%	87.9%	66.7%	75.4%
3000	92.3%	95.3%	79.6%	90.5%	67.6%	-

Table 4: LP-algorithm as a dimensionality reduction step. SVM algorithm trained on the full dictionary is compared against SVM algorithm trained on the topical subspace. The test cases contain 4, 6 and 10 underlying topics. The number in the first column is number of labeled documents *per category* supplied to SVM. Accuracy for the first 3 rows is averaged over 10 runs. Each number represent accuracy for corresponding test. For the last case SVM has running time has exceeded 20 hours, there no data were collected

.

4.2 20 Newsgroup

20 Newsgroup collection consists of 20 groups, each containing approximately 1000 messages. Below is the list of groups, where we have used the same numbering as in [24]

```
NG1: alt.atheism, NG2:comp.graphics,
NG3: comp.os.ms-windows.misc,
NG4: comp.sys.ibm.pc.hardware,
NG5: comp.sys.mac.hardware,
NG6: comp.windows.x,
NG7: misc.forsale, NG8:rec.autos,
NG9: rec.motorcycles,
NG10: rec.sport.baseball,
NG11: rec.sport.hockey,
NG12: sci.crypt, NG13: sci.electronics,
NG14: sci.med, NG15: sci.space,
NG16: soc.religion.christian,
NG17: talk.politics.guns,
NG18: talk.politics.mideast,
NG19: talk.politics.misc,
NG20: talk.religion.misc
```

While 1,000 per group might seem like a lot of documents, it is not quite enough to build $n \times n$ co-occurrence matrix, especially if more than two topics are involved. Therefore we only compare our algorithm on binary classification with results of [24] for p-QR, p-k-means and k-means. We also note that in [24] experiments were run on 100 messages subsets, which render them to be not directly comparable with ours.

In our experiments we have removed all headers from all messages, and all words that have occurred less than 5 times.

For non-related groups (NG1/NG2 and NG1/NG15) our algorithm gives 96% and 93% accuracy respectively (best in [24] is 89% and 73% respectively). For related groups accuracy has dropped. For NG2/NG3 and NG8/NG9 we have 75.5% and 87% (vs. 62.3 and 75.9), and for the final two examples our classification is close to meaningless[12]. Whereas there might be multiple reasons for this drop, we suspect the primary one to be insufficient amount of data - with high probability noise influence our choice of columns in co-occurrence matrix. To support this claim, we do the following experiment. For each group a *single* descriptive word is picked, and these words are given as a part of the input. The algorithm then chooses corresponding to these words columns of co-occurrence matrix as a first approximation to the topic subspace (instead of greedy search). After that it runs iteration in usual mode. Here is the list of words we have used as hints:

```
NG1: ATHEISM, NG2:GRAPHICS, NG3: WINDOWS
NG4: IBM, NG5: MAC, NG6: SUN
NG8:CAR,  NG9: BIKE, NG10: BASEBALL,
NG11: HOCKEY,  NG15: SPACE, NG18: ISRAEL
NG19: FBI
```

Classification results for this experiment are given in the last column of the table 5. Quite expectedly, all problematic (NG10/NG11 and NG18/NG19) binary cases were resolved, and the rest observed only a slight increase in accuracy. For hinted-I-LP, we also ran 5-way classification tests. The algorithm certainly succeeds on the first test (unrelated topics, 88% accuracy). For the second test, while it outperforms all other algorithms - the results are not as impressive (computer related topics, 54% accuracy). This again confirms our statement that for closely related topics our algorithm needs more statistical data in order to discriminate between them.

[12]Note that random classification gives 50% accuracy.

5. CONCLUSION AND FURTHER DIRECTIONS

We presented a new technique for large scale unsupervised text classification, which to the best of our knowledge outperforms all unsupervised methods. While this is very applicable result by itself, this suggests that there is a lot of structure still hidden in the high-dimensional textual data. And we believe that our algorithm is important first step towards exploring understanding such structure.

Another distinctive feature of our algorithm and analysis of [14] is that they allow guarantees about underlying term distributions and possibly about classification accuracy within the model and thus potentially could be used to *measure* suitability of the model for a given task. Particularly, success of our method shows that MCMM is indeed a good approximation for natural textual data.

Our approach poses a lot of new questions of both technical and theoretical nature. First, we used heuristic to build a mixture model. While it has worked surprisingly well for classification, it would be very interesting to see if our mixture is indeed related to the underlying in general case. Alternatively, can one find in some sense "the best" mixture model in feasible time? It is our belief that for single-label classification problem, there is a suitable definition of the "best" model, where it could be proven to be close to underlying model. However perhaps, additional assumptions about the model might be needed.

Another interesting question is to see if one can apply our method to linearly dependent topics, such as simultaneous clustering by authors and content, and/or hierarchical clustering.

From the theoretical point of view, improving actual bounds on the sample complexity presented is [14] and particularly matching them with our experimental results, is a very important open question. One approach here is to assume that distributions have particular shape - for example power-law. Finally, as it was mentioned in [14] Γ is similar to the smallest singular value. Can one construct analogues of other singular values and build something similar to SVD, but with respect to L_1 norm? Would it describe textual data better than traditional approach? What one could say about data partitioning when similarity is measured using L_1 norm, in general case?

6. ACKNOWLEDGMENTS

Author would like to thank Jon Kleinberg and Thorsten Joachims for useful discussions and recommendations, Paul Ginsparg and Paul Houle for providing collection of arXiv abstracts, and also Aleksandrs Slivkins for help in preparation of this manuscript.

7. REFERENCES

[1] Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proc. STOC*, 2001.

[2] L. Baker and A. McCallum. Distributional clustering of words for text classification. In Proc. ACM SIGIR, 1998.

Newsgroups	p-QR(100)	p-Kmeans(100)	K-means(100)	I-LP	hinted I-LP
NG1/NG2	89.3%	89.6%	76.2%	96.9%	97.2%
NG2/NG3	62.3%	63.8%	61.6%	75.5%	82.3%
NG8/NG9	75.9%	77.6%	65.7%	87.6%	88.6%
NG10/NG11	73.3%	74.8%	62.0%	54.9%	90.3%
NG15/NG1	73.3%	74.8%	62.0%	93.1%	96.9%
NG18/NG19	63.9%	64.0%	63.7%	58.5%	86.3%
NG2/NG9/NG10/NG15/NG18	77.8%	70.1%	58.1%	- %	88.3%
NG2/NG3/NG4/NG5/NG6	41.7%	42.5%	37.2%	- %	54.7%

Table 5: Accuracy of classification on 20 Newsgroup dataset. Last column corresponds to the case where algorithm is given a hint: a relevant word for each topic it has to classify.

[3] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *FOCS*, 2002.

[4] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *J. of Machine Learning Research*, 3, 2003.

[5] E. Demaine and N. Immorlica. Correlation clustering with partial information. In *RANDOM-APPROX*, 2003.

[6] T. Fenner, A. Flaxman, and A. Frieze. High degree vertices and eigenvalues in the preferential attachment. In *Random*, 2003.

[7] G. H. Golub and C. V. Loan. *Matrix Computations(3rd edition)*. Johns Hopkins University Press, 1996.

[8] G.Salton and M. McGill. *Introduction to modern information retrieval*. McGraw-Hill, New York, 1983.

[9] T. Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999.

[10] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.*, 42(1-2):177–196, 2001.

[11] P. Husbands, H. Simons, and C. Ding. On the use of singular value decomposition for text retrieval. In *1st SIAM Computational Information Retrieval Workshop*, 2000.

[12] T. Joachims. Making large-scale svm learning practical. In *Advances in Kernel Methods - Support Vector Learning, B. Scholkopf, C. Burges and A. Smola(ed.)*. MIT-Press, 1998.

[13] Q. Ke and T. Kanade. Robust subspace computation using l_1 norm. Technical Report CMU-CS-03-172, Carnegie Mellon University, 2003.

[14] J. Kleinberg and M. Sandler. Using mixture models for collaborative filtering. In Proc. STOC, 2004.

[15] A. K. McCallum. Multi-label text classification with a mixture model trained by em. In Proc. NIPS, 1999.

[16] G. McLachlan and K. Basford. *Mixture Models, inference and applications to clustering*. Marcel Dekker, 1987.

[17] M. Mihail and C. Papadimitriu. On the eigenvalue power law. In *Proc. RANDOM*, 2002.

[18] A. Ng. Feature selection, l_1 vs l_2 regularization and rotational invariance. In *ICML*, 2004.

[19] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In Proc. NIPS, 2001.

[20] C. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In Proc. ACM PODS, 1998.

[21] M. Sahami, M. A. Hearst, and E. Saund. Applying the multiple cause mixture model to text categorization. In L. Saitta, editor, *Proc. ICML*, pages 435–443, Bari, IT, 1996. Morgan Kaufmann Publishers, San Francisco, US.

[22] E. Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 7:51–71, 1995.

[23] C. Swamy. Correlation clustering: Maximizing agreements via semidefinite programming. In *SODA*, 2004.

[24] H. Zha, X. He, C. Dong, and H. Simons. Spectral relaxation for k-means clustering. In *NIPS01*, 2001.