ABSTRACT

The human language is inherently unstructured. Human authors never follow the strict rules of grammar because their human readers possess cognitive abilities to interpret the semantic meaning of unstructured human written text. Since unstructured text possess ambiguities and uncertainties, using probabilities to model human language is a natural choice. The investigation of a widely used probabilistic model is the motivation of my survey here. This probabilistic model known as Latent Dirichlet Allocation (LDA), has seen widespread adoption in the field of information retrieval. As prelude to the introduction of LDA, I shall review the Latent Semantic Analysis (LSA) and Probabilistic Latent Semantic Analysis (PLSA) for historical interests. Finally, I briefly discuss Hierarchical Latent Dirichlet Allocation.

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]: Clustering

General Terms
Probabilities and Statistics, Algorithms and Experimentation

Keywords
Bayesian Models, Gibbs Sampling, Singular Value Decomposition

1. INTRODUCTION

Information Retrieval (IR) as a field emerged in computer science when authors started storing their documents in digital format. IR importance increased after wide spread usage of internet. Since there are millions of webpages on the internet, users need a quick and simple way of finding web pages that are relevant to their information need. To fulfill the information need, internet search engines emerged to provide search service for users. Traditionally, these search engines were built using deterministic mathematical approaches such as set theory, logic, linear algebra and combinatorics. However, unlike computer programming languages which follows strict grammars and syntax, the common human language is unstructured with great amount of ambiguity. The unstructured nature of human language makes it difficult for search engines to understand the semantic meaning of web pages. To overcome the difficulty of understanding human languages, an IR related field, Natural Language Processing (NLP) emerged. Despite NLP progress, their scalability for understanding the huge amount of text on the web is still intractable. To make computation tractable for understanding text documents, IR researchers have treated documents using a simple approach known as bag-of-words. Using the bag-of-words approach, IR researchers have been able to cluster documents based on similarities between them.

A notable technique which uses the bag-of-words assumption for document classification of a text corpus is Latent Dirichlet Allocation (LDA) proposed by Blei et al [2]. The success of LDA in the research community extends far beyond the field of Information Retrieval. LDA has also seen wide application in related fields such as data mining and multimedia classification. My survey here is motivated by the importance of LDA and the bayesian principles used in LDA. To gain a thorough understanding of LDA, it is necessary to appreciate the predecessors of LDA, namely Latent Semantic Analysis (LSA) [4] and Probabilistic Latent Semantic Analysis (PLSA) [9].

Despite the success of these clustering techniques, it is not clear how many clusters should be assigned to for a text corpus. The naive approach performs the clustering repeatedly by varying the number of clusters and observing the scores of precision, recall and perplexity. To address the issue of variable clusters, Blei et al proposed the Hierarchical Latent Dirichlet Allocation (hLDA) based on principles of nonparametric bayesian techniques [1]. Besides being able to vary the number of clusters, hLDA also model subclusters which meant that documents belonging to a cluster can be further classified into specialized clusters. A close look at the the Computer Science article in Figure 1 also reflects this phenomenon.

The paper is organized as follows, we give a derivation of LSA and show a basic folding-in technique for answering IR queries in Section 2. Section 3 introduces PLSA for modeling
2. LATENT SEMANTIC ANALYSIS

Latent Semantic Indexing is a linear algebraic technique that uses Singular Value Decomposition (SVD) to factorize a given matrix of term-document occurrence [4]. I will describe the factorization process of SVD, then show an example of how SVD can be used to improve retrieval results.

2.1 Singular Value Decomposition

Definition 1. Suppose we have $t$ number of terms and $d$ number of documents in our text corpus. We can form a term by document matrix $X$ of size $t \times d$. SVD decomposes $X$ into the product of three matrices $X = TSD^T$, where $T$ is a $t \times d$ matrix, $S$ is a $d \times d$ matrix and $D^T$ is a $d \times d$ matrix. $T$ and $D^T$ are orthogonal matrices while $S$ is a diagonal matrix.

Lemma 1. Multiplying matrix $X$ by its own transpose, always yields a square matrix. $XX^T$ is a square matrix of size $t \times t$. $X^T X$ is a square matrix of size $d \times d$.

Theorem 1. The square matrices $XX^T$ and $X^T X$ can always be factorized into their corresponding eigenvectors and eigenvalues. We will not discuss the proof in detail here but interested readers might want to refer to [11] for a detailed explanation.

We now show how to derive the matrices $T$, $S$ and $D^T$.

$$X = TSD^T$$

$$X^T = (TSD^T)^T = DS^T T^T$$

$$XX^T = (TSD^T) (DS^T T^T) = TS(D^T D) S^T T^T$$

If we constrain $D$ to be orthogonal, then $D^T = D^{-1}$. Hence,

$$XX^T = TS(D^T D) S^T T^T = TS S^T T^T$$

If we constrain $S$ to be a diagonal matrix, then $S = S^T$.

Hence,

$$XX^T = TSS^T T^T = TS^T T^T$$

Doing the same manipulations for $X^T X$ and constraining $T$ to be orthogonal allow us to obtain

$$X^T X = D S^T T^T S D T^T = DS^T D^T$$

From Theorem 1, obtaining the eigenvectors of $XX^T$ and $X^T X$ gives us $T$ and $D^T$ respectively. Since LSA have been mentioned in class, I will skip the examples of LSA.

2.2 Folding-in

Suppose we have a query $q$ and we want to retrieve the relevant documents in the reduced semantic space. First we have to map $q$ into the reduced semantic space $D^T$. The square matrices $T_q$, $S_q$ and $D_q$ are orthogonal matrices while $S_q$ is a diagonal matrix.
term document matrix \( X^{(t+m)\times(d+q)} \) can be factorized in this manner,

\[
X^{(t+m)\times(d+q)} = T^{(t+m)\times m} S_{s,x} D_{x \times (d+q)}^T
\]

3. PROBABILISTIC LATENT SEMANTIC ANALYSIS

**Definition 2.** Suppose we have \( M \) number of documents and \( V \) number of words in the vocabulary. Let \( d_m \) be the index for each document and \( w_v \) be the occurrence of word \( v \) in the document \( d_m \). We define a joint probability model \( P(d_m, w_v) \) as the probability of observing word \( v \) in the document \( d_m \). As we have shown earlier in LSA, it is possible to represent words and documents in latent space. PLSA achieves the same effect by introducing a latent variable \( z_k \) that denotes the topic of each word in a document, where \( K \) is the number of topics in the corpus and \( 1 \leq k \leq K \). PLSA makes several assumptions to define the model.

1. Given a document \( d_m \) in the corpus, we are able to infer the topic \( z_k \) of the document, \( P(z_k|d_m) \).

2. Given a topic, we are able to infer which word \( v \) is likely to appear, \( P(w_v|z_k) \).

With the introduction of this latent topic variable \( z \), we can define the joint probability of observing a word in the document as \( P(d_m, w_v, z_k) \), where

\[
P(d_m, w_v, z_k) = P(w_v|z_k)P(z_k|d_m)P(d_m)
\]

Since \( z_k \) is latent and we can only observe \( d_m \) and \( w_v \), then we need to marginalize \( z_k \) out of the equation by summing over all possible values that \( z_k \) can take.

\[
P(d_m, w_v) = \sum_{k=1}^{K} P(d_m, w_v, z_k) = \sum_{k=1}^{K} P(w_v|z_k)P(z_k|d_m)P(d_m)
\]

\[
= \sum_{k=1}^{K} P(w_v|z_k)P(z_k|d_m)P(d_m)
\]

We will like to define the probability of the entire corpus \( P(d, w) \) where

\[
P(d, w) = \prod_{m=1}^{M} \prod_{v=1}^{V} P(d_m, w_v) = \prod_{m=1}^{M} \prod_{v=1}^{V} \sum_{k=1}^{K} P(w_v|z_k)P(z_k|d_m)P(d_m)
\]

Before we proceed on, we will like to show its relation to LSA. Suppose we assume that in our corpus in Table 1, our documents consists of two main topics, then what we want to find is the conditional probability tables here as shown in Table 2. These conditional probabilities bear similarities to the LSA formulation for a semantic space of two dimensions. These values should be filled in to maximize the log equivalent of likelihood

![Table 1: An example for Bayesian Networks](image)

<table>
<thead>
<tr>
<th>Terms</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>networks</td>
<td>d1 d2 d3 d4 d5 d6</td>
</tr>
<tr>
<td>learning</td>
<td>1 0 0 0 0 0</td>
</tr>
<tr>
<td>bayesian</td>
<td>1 0 0 0 0 0</td>
</tr>
<tr>
<td>graphical</td>
<td>0 1 1 0 0 0</td>
</tr>
<tr>
<td>models</td>
<td>0 1 1 0 0 0</td>
</tr>
<tr>
<td>information</td>
<td>0 0 0 1 0 1</td>
</tr>
<tr>
<td>retrieval</td>
<td>0 0 0 1 0 1</td>
</tr>
<tr>
<td>data</td>
<td>0 0 0 0 1 1</td>
</tr>
<tr>
<td>mining</td>
<td>0 0 0 0 1 0</td>
</tr>
</tbody>
</table>

![Table 2: The Conditional Probability Tables](image)

| P(w_v|z_k) | w_1 w_2 w_3 w_4 w_5 w_6 w_7 w_8 w_9 |
|-----------|-----------------|

values, hence, the log likelihood \( \mathcal{L} \) is often expressed in this manner,

\[
\mathcal{L} = \log P(d, w)
\]

\[
= \log \prod_{m=1}^{M} \prod_{v=1}^{V} P(w_v|z_k)P(z_k|d_m)P(d_m)
\]

\[
= \sum_{m=1}^{M} \sum_{v=1}^{V} n(d_m, w_v) \log \sum_{k=1}^{K} P(w_v|z_k)P(z_k|d_m)P(d_m)
\]

where \( n(d_m, w_v) \) is the number of times word \( w_v \) occur in document \( d_m \). Maximizing the likelihood here by filling in the conditional probability tables is essentially an optimization problem. For maximizing likelihood, the Expectation Maximization algorithm often used for computing these value [5]. We shall skip the full derivation here but interested readers may refer to [10]. We shall follow the equations as described in [9] to compute these values. In the E step of the algorithm, we first compute \( P(z_k|d_m, w_v) \).

\[
P(z_k|d_m, w_v) = \frac{P(z_k)P(d_m|z_k)P(w_v|z_k)}{\sum_{l=1}^{K} P(z_l)P(d_m|z_l)P(w_v|z_l)}
\]
In the M step of the algorithm, we compute the quantities $P(w_v|z_k), P(d_m|z_k), P(z_k)$.

$$P(w_v|z_k) = \frac{\sum_{m=1}^{M} n(d_m, w_v)P(z_k|d_m, w_v)}{\sum_{m=1}^{M} \sum_{n=1}^{N} n(d_n, w_n)P(z_k|d_n, w_n)}$$

$$P(d_m|z_k) = \frac{\sum_{v=1}^{V} n(d_m, w_v)P(z_k|d_m, w_v)}{\sum_{n=1}^{N} n(d_n, w_n)P(z_k|d_n, w_n)}$$

$$P(z_k) = \frac{\sum_{m=1}^{M} \sum_{v=1}^{V} n(d_m, w_v)P(z_k|d_m, w_v)}{\sum_{m=1}^{M} \sum_{n=1}^{N} n(d_n, w_n)P(z_k|d_n, w_n)}$$

After running the EM algorithm, we get such values in Table 3.

| $P(w_v|z_k)$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ |
|--------------|-------|-------|-------|-------|-------|-------|
| $z_1$        | 0.31  | 0.25  | 0.25  | 0.25  | 0.25  | 0.14  |
| $z_2$        | 0.13  | 0.25  | 0.13  | 0.25  | 0.29  | 0.29  |
| $z_3$        | 0.47  | 0.53  |

Since PLSA is a probabilistic technique, the results of small samples may not always be consistent. Therefore, we ran experiments on a corpus of a larger size. Using text corpus from wikipedia articles for Computer science, Information theory, Cryptography, Cryptanalysis, History of cryptography, Database, Data mining and Information retrieval. We run PLSA on these articles. Then we rank the words and obtain the following sets of words for each topic. Refer to Table 4 for the results.

<table>
<thead>
<tr>
<th>Table 3: The Conditional Probability Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(w_v</td>
</tr>
<tr>
<td>$z_1$</td>
</tr>
<tr>
<td>$z_2$</td>
</tr>
<tr>
<td>$z_3$</td>
</tr>
</tbody>
</table>

4.1 Probabilities and Identities

$$\Gamma(x) = (x-1)!$$

$$\int_0^1 f^{a-1}(1-f)^{b-1} df = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

$$\int_0^1 \prod_{i=1}^{N} f_i^{a_i-1} df = \prod_{i=1}^{N} \frac{\Gamma(a_i)}{\Gamma(\sum_{i=1}^{N} a_i)}$$

4.1.1 Dice Toss Toy Example

Suppose I have a dice of $I$ sides. I toss the dice and the probability of landing on side $i$ is $p(s = i|f) = f_i$. I throw the dice $N$ number of times and obtain a set of results $s = (s_1, s_2, s_3, \ldots, s_N, \ldots, s_N)$. We can specify the probability of observing this set of results in this manner,

$$p(s|f) = \prod_{n=1}^{N} p(s_n|f)$$

$$= f_1^{n_1} f_2^{n_2} f_3^{n_3} \cdots f_I^{n_I}$$

$$= \prod_{i=1}^{I} f_i^{n_i}$$

where $n_i$ denotes the number of times side $i$ appear. Notice that the $\prod_{n=1}^{N}$ has been changed to $\prod_{i=1}^{I}$. The $\prod_{n=1}^{N}$ has been expressed as $\prod_{i=1}^{I}$.

Suppose that $f$ is a dirichlet distribution with $a$ as hyper-parameters. Then we express the probability distribution of $f$ as,

$$p(f|a) = \frac{\Gamma(\sum_{i=1}^{I} a_i)}{\prod_{i=1}^{I} \Gamma(a_i)} \prod_{i=1}^{I} f_i^{a_i-1}$$

If we want to update the parameter $f$ base on the observation of $s$, then we can express $f$ in this manner.

$$p(f|s, a) = \frac{p(s|f,a)p(f|a)}{\int_0^1 p(s|f,a)p(f|a) df}$$

$$= \frac{\prod_{n=1}^{N} p(s_n|f,a) p(f|a) df}{\prod_{i=1}^{I} \prod_{n=1}^{N} p(s_n|f,a) p(f|a) df}$$

Notice that after updating $f$ base on $s$ observations, $f$ is still a dirichlet distribution. This property is known as conjugate priors. Base on this property, estimating the parameters $f_i$ after observing $n$ trials is a simple counting procedure. Suppose I want to obtain $f_z$ from $f = \sum_{z=1}^{Z} f_z$.
\( (f_1, f_2, \ldots, f_x, \ldots, f_l) \),

\[
E(f_x|s, a) = \int_0^1 f_x p(f|s, a) \, df
\]

\[
= \int_0^1 f_x \frac{\Gamma(\sum_{i=1}^I n_i + a_i)}{\prod_{i=1}^I \Gamma(n_i + a_i)} \prod_{i=1}^I f_x^{n_i + a_i - 1} \, df
\]

\[
= \frac{\Gamma(\sum_{i=1}^I n_i + a_i)}{\prod_{i=1}^I \Gamma(n_i + a_i)} \int_0^1 f_x^{n_x + a_x} \prod_{i=1, i \neq x}^I f_x^{n_i + a_i - 1} \, df
\]

\[
= \frac{\Gamma(\sum_{i=1}^I n_i + a_i)}{\prod_{i=1}^I \Gamma(n_i + a_i)} \prod_{i=1, i \neq x}^I \Gamma(n_i + a_i)
\]

\[
= \frac{\Gamma(\sum_{i=1}^I n_i + a_i)}{\prod_{i=1}^I \Gamma(n_i + a_i)} \prod_{i=1, i \neq x}^I \Gamma(n_i + a_i) + \sum_{i=1}^I n_i + a_i
\]

Now I show the Collapsed Gibbs Sampling for this toy example. Suppose we want to obtain \( f \) but do not have the counts of occurrence \( n_i \). We can generate the samples \( s_n \) using \( f \) by deriving a Markov chain, \( p(s_n|s_{-n}, a) \), where \( s_n \) represents all samples except \( s_n \). Using the above results,

\[
p(s_n = i|s_{-n}, a) = \frac{\sum_{i=1}^I n_i + a_i}{\sum_{i=1}^I n_i + a_i}
\]

\[
= E(f_i|s_{-n}, a)
\]

I will show later in Algorithm 1 how we adjust the parameters by excluding the results of \( s_n \) in \( s_{-n} \).

### 4.2 LDA Model

![Figure 2: Latent Dirichlet Allocation](image)

Refer to Figure 2 for a graphical description of LDA in plate notation form.

**Definition 3.** Suppose we have \( M \) documents and \( N \) words in document \( m \). Let \( \theta_m \) represent the dirichlet distribution of topics in each document \( m \) and \( \alpha \) are the hyper-parameters of the dirichlet distribution. Let \( w_{m,n} \) be the index of the word \( n \) in document \( m \) and \( z_{m,n} \) be the topic \( k \) of each word \( w_{m,n} \). Let \( \phi_k \) represent the dirichlet distribution of words over topic \( k \) with \( \beta \) as hyper-parameters.

Formally, the LDA model can be described mathematically in this manner,

\[
\alpha = (\alpha_1, \ldots, \alpha_K)
\]

\[
p(\theta_m|\alpha) = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \theta_m^{\alpha_k-1}
\]

\[
= \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \theta_m^{\alpha_k-1}
\]

\[
\beta = (\beta_1, \ldots, \beta_V)
\]

\[
p(\phi_k|\beta) = \frac{\Gamma(\sum_{v=1}^V \beta_v)}{\prod_{v=1}^V \Gamma(\beta_v)} \prod_{v=1}^V \phi_k^{\beta_v-1}
\]

The discrete values \( z_{m,n} \) depends on the topic distribution of each document \( m \).

\[
p(z_{m,n} = k|\theta_m) = \theta_{m,k}
\]

The probability of observing each word \( w_{m,n} \) in document \( m \) depends on the discrete topic \( z_{m,n} \) and word distribution \( \phi \).

\[
p(w_{m,n} = v|z_{m,n} = k, \phi) = \phi_{k,v}
\]

The joint probability of observing a word is,

\[
p(w_{m,n} = v, z_{m,n} = k, \phi, \theta_m|\alpha, \beta) = p(w_{m,n}|z_{m,n}) p(z_{m,n}|\theta_m) p(\phi|\beta) p(\theta_m|\alpha)
\]

The joint probability of observing a corpus is

\[
p(w, z, \phi, \theta_m|\alpha, \beta)
\]

\[
= \prod_{m=1}^M \prod_{n=1}^N p(w_{m,n}, z_{m,n}, \phi, \theta_m|\alpha, \beta)
\]

\[
= \prod_{m=1}^M p(\theta_m|\alpha) \prod_{n=1}^N p(z_{m,n}|\theta_m) p(w_{m,n}|z_{m,n}, \phi)
\]

What we want to solve for are the parameters \( \theta \) and \( \phi \). There are many ways of solving this model. I present a collapsed Gibbs Sampling approach to find these parameters [7, 8].

### 4.3 Gibbs Sampling

Gibbs Sampling is commonly used when it is hard to derive an analytical expression of a single variable from the joint probability distribution. Our main objective is to obtain,

\[
\theta = (\theta_{1,1}, \theta_{1,2}, \ldots, \theta_{m,k}, \ldots, \theta_{M,K})
\]

\[
\theta_{m,k} = \frac{n(m,k) + \alpha_k}{\sum_{k=1}^K n(m,k) + \alpha_k}
\]

\[
\phi = (\phi_{1,1}, \phi_{1,2}, \ldots, \phi_{K,V})
\]

\[
\phi_{k,v} = \frac{n(k,v) + \beta_v}{\sum_{v=1}^V n(k,v) + \beta_v}
\]

where \( n_{k,v} \) is the number of times word \( v \) has been assigned topic \( k \) and here \( n_{k,v} \) is the number of times topic \( k \) appear in document \( m \). But \( n(m,k) \) and \( n(k,v) \) are unknown to us now.

The Gibbs Sampling algorithm allows us to reduce the parameter estimation problem to a simple counting and sampling process. We will like to derive a sampling, counting
and update procedure for estimating the parameters and latent variables.

4.3.1 Collapsed Gibbs Sampling

Following the approach in [7, 8], we integrate the \( \phi \) and \( \theta \) parameters to obtain the following,

\[
p(z, w|\alpha, \beta) = p(w|z, \beta)p(z|\alpha)
\]

\[
p(w|z, \beta) = \int p(w, \phi|z, \beta) \, d\phi
\]

\[
p(w|z, \phi)p(\phi|z, \beta) \, d\phi
\]

\[
p(w|z, \phi)p(\phi|\beta) \, d\phi
\]

\[
p(z|\alpha) = \int p(z, \theta|\alpha) \, d\theta
\]

\[p(z, \theta|\alpha) \propto \Gamma(\sum_{k=1}^{K}\beta_k) \prod_{k=1}^{K} \phi_{z_k|\theta_k}^{\beta_k-1} \prod_{k=1}^{K} \Gamma(\phi_{z_k|\theta_k}) \prod_{k=1}^{K} \phi_{z_k|\theta_k}^{\beta_k-1} \prod_{k=1}^{K} \Gamma(\phi_{z_k|\theta_k}) \prod_{k=1}^{K} \phi_{z_k|\theta_k}^{\beta_k+n_{z_k|w}} \]

\[p(z|\alpha) \propto \frac{\Gamma(\sum_{k=1}^{K}\beta_k) \prod_{k=1}^{K} \phi_{z_k|\theta_k}^{\beta_k-1} \prod_{k=1}^{K} \Gamma(\phi_{z_k|\theta_k}) \prod_{k=1}^{K} \phi_{z_k|\theta_k}^{\beta_k+n_{z_k|w}}}{\sum_{w=1}^{W} \beta_w + n_w} \]

Equation 10 follows Equation 6 and 7 earlier in the dice toss example. Now with Equation 8, 9 and 10, we are ready to look at Algorithm 1. Algorithm 1 outputs the parameters that tells us the topic distribution in documents and topic distribution in words. Refer to Table 4 for a summarize comparison with PLSA.

5. HIERARCHICAL LATENT DICHRILET ALLOCATION

The LDA presented in previous section fails to address one issue: how many topics to use for a text corpus. Blei proposed a Hierarchical Latent Dirichlet Allocation (hLDA) based on nonparametric Bayesian statistics [1]. In nonparametric Bayesian statistics, the probability of observing a word in a document follows the probability distribution of stochastic processes instead of dirichlet distributions. The stochastic processes proposed in hLDA is a combination of two classical stochastic processes. The first stochastic process known as the Chinese restaurant process allows the number of topics to grow infinitely as many as required. The second stochastic process known as the stick breaking process allows for the division of topics into specialized topics to form a hierarchy of topics. Putting this two stochastic processes together, Blei call it the nested Chinese restaurant process [1]. Figure 3 shows an example of the words associated with the hierarchical topics of hLDA. I took this example from [3].

![Figure 3: Hierarchical Latent Dirichlet Allocation](image-url)
Algorithm 1 LDA

\[
\begin{align*}
\text{Initialize } & z[M][N], \text{ nkv}[K, V] \text{ and } nmk[M, K] \text{ arrays to 0} \\
\text{Initialize } & \phi[V] \text{ and } \alpha[K] \text{ to 1} \\
\text{docs}[M][N] & \leftarrow \text{ read in documents} \\
\text{for } & m \leftarrow 1 \text{ to } M \text{ do} \\
\text{for } & n \leftarrow 1 \text{ to } N \text{ do} \\
& k \leftarrow \text{ a random number between 1 and } K \\
& v \leftarrow \text{ docs}[m][n] \\
& \text{nkv}[k, v] \leftarrow \text{nkv}[k, v] + 1 \\
& \text{nmk}[m, k] \leftarrow \text{nmk}[m, k] + 1 \\
& z[m][n] \leftarrow k \\
\text{end for} \\
\text{end for} \\
\text{while run for 1000 iterations do} \\
\text{for } & m \leftarrow 1 \text{ to } M \text{ do} \\
\text{for } & n \leftarrow 1 \text{ to } N \text{ do} \\
& k \leftarrow z[m][n] \\
& v \leftarrow \text{ docs}[m][n] \\
& \{\text{The subtraction here corresponds to } z_{\_i}\} \\
& \text{nkv}[k, v] \leftarrow \text{nkv}[k, v] - 1 \\
& \text{nmk}[m, k] \leftarrow \text{nmk}[m, k] - 1 \\
\text{for } & k \leftarrow 1 \text{ to } K \text{ do} \\
& p[k] \leftarrow \frac{\alpha[k] + \text{nmk}[m, k]}{\sum_{k' = 1}^{K} \alpha[k'] + \text{nmk}[m, k']} * \frac{\alpha[k] + \text{nkv}[k, v]}{\sum_{v' = 1}^{V} \alpha[v'] + \text{nkv}[k, v']} \\
\text{end for} \\
\text{end for} \\
& k \leftarrow 2 \text{ to } K \text{ do} \\
& p[k] \leftarrow p[k] + p[k - 1] \\
\text{end for} \\
& u \leftarrow \text{ a random number between 0 and 1} \\
& u \leftarrow u \ast p[k] \\
\text{for } & k \leftarrow 1 \text{ to } K \text{ do} \\
& \text{if } p[k] > u \text{ then} \\
& \text{break and use current } k \\
\text{end if} \\
\text{end for} \\
\text{end for} \\
\text{end while} \\
\text{Initialize } \phi[K, V] \text{ and } \theta[M, K] \\
\text{for } & k \leftarrow 1 \text{ to } K \text{ do} \\
\text{for } & v \leftarrow 1 \text{ to } V \text{ do} \\
& \phi[k, v] \leftarrow \frac{nkv[k, v] + \beta[v]}{\sum_{v' = 1}^{V} nkv[k, v'] + \beta[v']} \\
\text{end for} \\
\text{for } & m \leftarrow 1 \text{ to } M \text{ do} \\
& \theta[m, k] \leftarrow \frac{nmk[m, k] + \alpha[k]}{\sum_{k' = 1}^{K} nmk[m, k'] + \alpha[k']} \\
\text{end for} \\
\text{return } \phi[K, V] \text{ and } \theta[M, K].
\end{align*}
\]

6. CONCLUSION

I have presented in detail, the formulation of LDA and the derivation of the Gibbs Sampling technique for solving LDA. The derivation is necessary because the math equations are hard to understand without working through it ourselves. I believe my derivation of LDA Gibbs Sampling is much more thorough than [7] and [8]. This will be useful for other researchers who are relatively new to Bayesian parameter learning.

In earlier section, LSA and PLSA was presented because of their incremental advances leading to the invention of LDA. Then I have briefly discussed hLDA. Although the mathematical formulation of LSA, PLSA and LDA are different, they are fundamentally performing the same task, grouping similar terms and documents together. Whether it is algebraic method or probabilistic method, all mathematical techniques aim to achieve the similar objective of representing unstructured data into structured and well-organized knowledge. The semantics of linear algebra is significantly much simpler than PLSA and LDA. PLSA also is much simpler than LDA. However, probabilistic representation is much simpler to understand when we want to interpret the results. Having prior knowledge of the text corpus and using the bayesian formalism in PLSA and LDA allow us to process the text corpus quickly. One notable difference between LSA and LDA is that LDA allows the number of topics to exceed the number of documents which expands the semantic space rather than reducing it. However, the results of expanding semantic space requires further validation.

One question remains in my mind. Since the mathematics of LDA is so much harder, what makes it different from PLSA? The difference is not obvious since both techniques uses probabilities for defining the topics of documents and words. Girolami addressed this issue in [6] and explained that LDA introduces dirichlet priors which avoids an overfitting problem in the text corpus. Another issue that I have not addressed is how does such document classification improve IR results? Wei and Croft have shown that LDA does improve IR results [12]. Finally, I have not give an in-depth explanation of hLDA. hLDA builds on the field of nonparametric Bayesian statistics which requires a thorough theoretical foundation in this area. hLDA was proposed a year later after LDA but the research community have readily adopted LDA instead of hLDA. This is because the research community have not fully understood how to interpret the topics discovered by LDA, hence, the usefulness of hLDA is still limited by our understanding of text.

7. REFERENCES


**APPENDIX**

I wrote these programs myself.

**A. RUNNING PROGRAMS**

**A.1 PLSA for Toy Example**

1. Run Matlab.
2. Run the script EM from the directory \matlab\.
3. Run the script a few times to observe the random results generated.

This random results obtained motivated me to use a larger text corpus in the next section.

**A.2 PLSA for wikipedia articles**

1. Run Matlab.
2. Run the script PLSA from the directory \matlab\.
3. This will take some time.

**A.3 LDA**

I wrote LDA with reference to an existing C++ implementation of Gibbs Sampling from [8].

1. Run TopicModel\LDAGibbsSampling\bin\Release \LDAGibbsSampling.exe
2. This is quite fast.

**A.4 LDA Gutenberg corpus**

1. Run TopicModel\Gutenberg\bin\Release \Gutenberg.exe <no. of topics>

The results are shown here.