THE COOPER UNION
ALBERT NERKEN SCHOOL OF ENGINEERING

Unsupervised Topic Clustering of Text Corpora

by

Daniel Gitzel

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering

September 2016

Professor Sam Keene, Advisor

THE COOPER UNION FOR THE
ADVANCEMENT OF SCIENCE AND ART

ALBERT NERKEN SCHOOL OF ENGINEERING

This thesis was prepared under the direction of the Candidate's Thesis Advisor and has received approval. It was submitted to the Dean of the School of Engineering and the full Faculty, and was approved as partial fulfillment of the requirements for the degree of Master of Engineering.

Dean, School of Engineering Date

Prof. Sam Keene, Advisor Date

Acknowledgements

It is with immense gratitude that I acknowledge the support and help of my Professor, Sam Keene, without his guidance and persistent help this work would not have been possible. I would also like to thank my classmates, Christopher Curro and Ethan Lusterman, for my discussions with them as well as help in typesetting the final work. Lastly, I would like to thank my parents for their support throughout this endeavor.

VR LW EHJLQV,
D WZLVWHG MRXUQHB RI PDUYHO DQG ZRQGHU;
D SULPH ULGGOH OLHV XQGHU DQG JXDUGV:
DQ HPSWB SURPLVH RU SOXQGHU?

KDSED YLJQQ CMJQX OJWNK DSKQS BWLWU

ILGFX OXGXB YOAZW DAWDQ ZJAXJ RAJFU OFLTG

Abstract

This work describes an application of Latent Dirichlet Allocation (LDA) text modeling to find clusters in various text corpora. Specific optimizations and applications are made to model medical data consisting of patient observations. The LDA topic model is first explored with a set of labeled (supervised) data to evaluate performance and demonstrate the viability of an unsupervised system. The supervised data is clustered and this reduced representation is used to classify the documents. The class labels are then compared to the original labels to evaluate the topic model. The performance on the unsupervised set is evaluated through observation of the clusters and the most-likely topic distributions, as well as the log-likelihood score of the topic model.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Topic Modeling

Text is a ubiquitous and important source of information. Finding a compact representation of the contents of a document that is both human-readable and informative remains an important task. The goal of *topic modeling* is to create such representations by discovering latent topic structures in collections of documents. These representations are useful for document classification and retrieval tasks, making topic modeling an important machine learning problem.

Effectively using collections of text requires interacting with them in a more structured way: finding documents similar to those of interest, and exploring the collection through its underlying topics.

The central problem is that this structure—the index of ideas contained in the articles, and which other articles are about the same kinds of ideas—is not readily available in most modern collections. The size and growth rate of these collections preclude one from building it by hand. To develop the necessary tools for exploring and browsing modern digital libraries, automated methods of organizing, managing, and delivering their contents are required.

Topic models are systems which seek to uncover the underlying semantic structure of a document collection. Topic models may be as simple as indexing systems or hand-labeling. More sophisticated models incorporate word frequency [1], singular-value decomposition [2], or probablistic modelling of the word frequencies [3]. The system described here is based on a hierarchical Bayesian analysis of the texts. Topic models have been applied to many kinds of documents including email [4], scientific abstracts [5], and news articles [6]. By discovering patterns of word occurrences in and connecting documents

that exhibit similar patterns, topic models have emerged as a powerful new technique for finding useful structure in an otherwise unstructured collection.

Probabilisic topic models are often used to analyze and extract semantic topics from large text collections. Many of the existing topic models are based on the assumption that each document is represented as a mixture of topics, where each topic defines a probability distribution over words. The mixing proportions of the topics are document specific, but the probability distribution over words, defined by each topic, is the same across all documents.

The most common approach to topic modeling is to build a generative probabilistic model of a *bag-of-words* in a document. These models treat documents as a *bag-of-words*, a collection of terms or words, and discard their relative positioning. While this destroys any existing grammatical and semantic structure, this simplifying assumption does not preclude the development of functional models. They are deemed *generative models* since they generate observable values from a conditional probability density given a set of hidden parameters formed through Baye's rule.

Directed graphical models, such as Latent Dirichlet Allocation (LDA) [7], have been extensively used for this. Non-parametric extensions of these models have also been quite successful [8]. Even though exact inference in these models is hard, efficient inference schemes, including stochastic variational inference, online inference, and collapsed Gibbs sampling have been developed that make it feasible to train and use these methods.

These probabilistic models may be interpreted as graphical models (in plate notation). Latent topic variables have directed connections to observed variables: the words in a given document. As mentioned above, exact inference,

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
computer	chemistry	cortex	orbit	infection
methods	synthesis	stimulus	dust	immune
number	oxidation	fig	Jupiter	AIDS
two	reaction	vision	line	infected
principle	product	neuron	system	viral
design	organic	recordings	solar	cells
access	conditions	visual	gas	vaccine
processing	cluster	stimuli	atmospheric	antibodies
advantage	molecule	recorded	Mars	HIV
important	studies	motor	field	parasite

Table 1: Five topics from a 50-topic LDA model fit to *Science* from 1980–2002. Excerpted from [10].

in these models is intractable, so one has to variational inference methods to compute the posterior distribution over topics.

Note that any mixture model cannot make predictions for words that are "sharper" than the distributions predicted by any of the individual topics, since distributions predicted by individual active features are multiplied together to give the distributions predicted by many active features. This allows individual features to be fairly general but their intersection to be much more precise. For example, the topics "government", "coverup", and "moon" to combine to give the very high probability to a word "NASA".

With these statistical tools one can automate the categorizing of digital archives to facilitate efficient browsing and exploring. For example, an analysis of academic journals may reveal the internal structure of topics and their most relevant articles. Analysis of email topics are useful for spam detection [9].

Table 1 demonstrates five topics—groups of highly probable words—that were discovered automatically from this collection using the latent Dirichlet allocation (LDA) topic model. This algorithm has no prior knowledge of the existence of the illustracted themes, such as computer science or astrophysics.

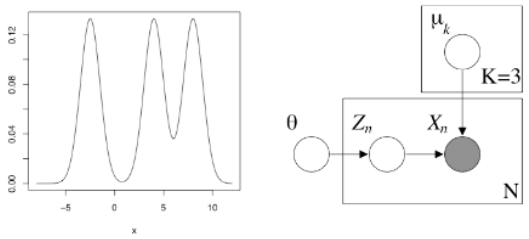

Figure 1: Example mixture distribution of three Gaussians and plate notation for the model.

The topics are automatically discovered from analyzing the corpus. Note, that while the algorithm clusters words into topics it cannot synthesize explantory topic names.

1.2 Hierarchical Bayesian Models

Hierarchical Bayesian models make use of two important concepts in deriving the posterior distribution in that they assume hyper-parameters which are distributed according to their hyper-priors.

Directed graphical models are used to describe the independence assumptions of the models. The graphical model provides a compact description of the factorization of a joint distribution: *nodes* denote random variables; *edges* denote possible dependence between random variables; and *plates* denote replication of a substructure. The indexes of the variables within a plate is noted on the bottom-right corner.

Graphical models can be used to describe *latent variable models*. Latent variable modeling is a method of developing complicated structured distributions, where the data interact with *latent* or *unobserved* random variables. In the graphical model notation, observed random variables are shaded, and latent random variables are unshaded.

For example, the distribution in Figure 1 is the *mixture distribution* formed by combining three unit-variance Gaussian distributions with means $\mu_1 = -2.5$, $\mu_2 = 4$, and $\mu_3 = 8$. A data point is drawn by first choosing a latent variable $Z \in \{1, 2, 3\}$ from a multinomial, and then drawing the data point from $\mathcal{N}(\mu_z, 1)$. This example is illustrated as a graphical model in Figure 1.

The central task of the latent variable modeling for data analysis is *posterior inference*, where we determine the distribution of the latent variables conditional on the observations. Loosely, posterior inference can be thought of as a reversal of the generative process which the graphical model illustrates. For example in the Gaussian mixture with fixed means, we would like to determine the posterior distribution of the indicator Z given a data point x. If $x = 1$, then the posterior $p(Z \mid X = 1, \mu_1, \mu_2, \mu_3)$ is $(0.16, 0.83, 0.01)$.

Traditionally, the structure of the graphical model informs the ease or difficulty of posterior inference. In the models of the subsequent chapters, however, inference is difficult despite a simple graph structure. Thus, we resort to approximate posterior inference.

Typically, the parameters of the model are not observed—in this case the means in the Gaussian mixture—and part of the posterior inference problem is to compute their posterior distribution conditional on the data. One option is to adopt the *empirical Bayes* perspective and find point estimates of the parameters based on maximum likelihood. Such estimates can be found, for example, with the expectation-maximization (EM) algorithm, or approximate variant of it.

Alternatively, we may take a more fully Bayesian approach, placing a prior distribution on the parameters and computing a proper posterior distribution. This is called *hierarchical Bayesian modeling* because it necessitates the speci-

fication of a distribution of the parameters, which itself must have parameters called *hyperparameters*.

In a hierarchical Bayesian model, we may still use the empirical Bayes methodology, and find point estimates of the hyperparameters by maximum likelihood. This is often sensible because it affords the advantages of exhibiting uncertainty on the parameters, while avoiding the unplesant necessity of choosing a fixed hyperparameter or further extending the hierarchy.

1.2.1 Exponential family distributions

All the random variables we will consider are distributed according to *exponential family* distributions. Distributions in this family have the form:

$$p(x \mid \boldsymbol{\eta}) = h(x) \exp(\boldsymbol{\eta} \cdot \mathbf{t}(x) - A(\boldsymbol{\eta})) \tag{1}$$

$$\boldsymbol{\eta} = (\eta_1, \eta_2, \ldots, \eta_s)^T \tag{2}$$

where $\boldsymbol{\eta}$ is the *natural parameter*, a vector of dimension s, $\mathbf{t}(x)$ is a vector-valued function representing the sufficient statistics and $A(\boldsymbol{\eta})$ is the *log partition function*:

$$A(\boldsymbol{\eta}) = \log \int h(x) \exp(\boldsymbol{\eta} \cdot \mathbf{t}(x)) \, \mathrm{d}x \tag{3}$$

The forms of the derivatives of $A(\boldsymbol{\eta})$ are determined by the dimensions of $\mathbf{t}(x)$.

Consider the Gaussian (normal) distribution with unknown mean, μ and variance σ:

$$\mathcal{N}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right)$$

It may be written in the canonical form from Equation 2 with parameters:

$$\begin{aligned}
\boldsymbol{\eta} &= \left(\frac{\mu}{\sigma^2}, \frac{1}{2\sigma^2}\right)^T \\
h(x) &= \frac{1}{\sqrt{2\pi}} \\
\mathbf{t}(x) &= (x, x^2)^T \\
A(\boldsymbol{\eta}) &= \frac{\mu^2}{2\sigma^2} + \ln|\sigma| \\
&= -\frac{\frac{2}{1}}{4\eta_2} + \frac{1}{2}\ln\left|\frac{1}{2\eta_2}\right|
\end{aligned}$$

If $x \in \mathbb{R}$ then the derivatives of the *log partition function* are:

$$\begin{aligned}
A'(\boldsymbol{\eta}) &= \mathbb{E}_{\boldsymbol{\eta}}[\mathbf{t}(X)] \\
A''(\boldsymbol{\eta}) &= \mathbb{E}_{\boldsymbol{\eta}}[\mathbf{t}(X^2)] - (\mathbb{E}[\mathbf{t}(X)])^2 \\
&= \mathrm{Var}(\mathbf{t}(X))
\end{aligned}$$

If $\mathbf{t}(x)$ is a multidimensional, one-hot vector (i.e. all zeros with a single one) then the corresponding exponential family distribution is multinomial as shown in Equations 4–7. More information on the exponential family of distribution can be found in [11].

$$\boldsymbol{\eta} = (\ln p_1, \ln p_2, \ldots, \ln p_k)^T \text{ where } \sum_{i=1}^{k} p_i = 1 \tag{4}$$

$$h(x) = \frac{\Gamma(n)}{\prod_{i=1}^{k}\Gamma(x_i)} \tag{5}$$

$$\mathbf{t} = (x_1, x_2, \ldots, x_k)^T \text{ where } \sum_{i=1}^{k} x_i = 1 \tag{6}$$

$$A(\boldsymbol{\eta}) = 0 \tag{7}$$

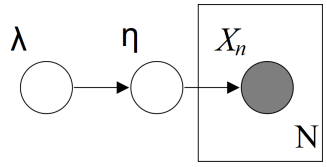

Figure 2: The plate notation of IID data $X_{1:N}$ from $p(x \mid \boldsymbol{\eta})$, where $\boldsymbol{\eta}$ is distributed by $p(\boldsymbol{\eta} \mid \boldsymbol{\lambda})$ for a hyper-parameter λ. Note that the observed random variables are shaded, and latent random variables are unshaded.

1.2.2 Conjugate Priors

As previously stated, in a hierarchical Bayesian model, one must specify the prior distribution of the parameters. This section describes a family of priors which facilitate computations in such a model.

If X is a random variable distributed according to an exponential family with natural parameter $\boldsymbol{\eta}$ and log partition function $A(\boldsymbol{\eta})$, then a *conjugate prior* of $\boldsymbol{\eta}$, with natural parameter $\boldsymbol{\lambda}$, has the form:

$$p(\boldsymbol{\eta} \mid \boldsymbol{\lambda}) = h(\boldsymbol{\eta}) \exp(Q - A(\boldsymbol{\lambda})) \tag{8}$$

Where Q is:

$$Q = \lambda_1^T + \lambda_2(-A(\boldsymbol{\eta})) \tag{9}$$

Where λ_1 and λ_2 are decompositions of $\boldsymbol{\lambda}$ such that λ_1 is of dimension $\dim(\boldsymbol{\eta})$ and λ_2 is a scalar. The conjugate distribution is an important mathematical trick, because the corresponding posterior will have the same form. Consider the simple model illustrated in Figure 2 where $X_{1:N}$ are independent, identically distributed (IID) from the exponential family distribution $p(x_n \mid \boldsymbol{\eta})$ and $p(\boldsymbol{\eta} \mid \boldsymbol{\lambda})$ is the conjugate prior.

The *posterior* of $\boldsymbol{\eta}$ is:

$$\begin{aligned} p(\boldsymbol{\eta} \mid \lambda, x_{1:\,N}) &\propto p(\boldsymbol{\eta} \mid \boldsymbol{\lambda}) p(x_{1:\,N} \mid \boldsymbol{\eta}) \\ &\propto h(\boldsymbol{\eta}) \exp(q) \prod_{n=1}^{N} \exp(\boldsymbol{\eta} \cdot t(x_n) - A(\boldsymbol{\eta})) \\ &= h(\boldsymbol{\eta}) \exp\left(\lambda_1 + (\lambda_2 + N)(-A(\eta)) + \sum_{n=1}^{N} t(x_n) \cdot \boldsymbol{\eta}\right) \end{aligned}$$

which is of the same form as the prior, $p(\boldsymbol{\eta} \mid \boldsymbol{\lambda})$, in Equation 8 with posterior parameters $\hat{\boldsymbol{\lambda}} = (\hat{\lambda}_1, \hat{\lambda}_2)$ which become:

$$\hat{\lambda}_1 = \lambda_1 + \sum_{n=1}^{N} \mathbf{t}(x_n) \tag{10}$$

$$\hat{\lambda}_2 = \lambda_2 + N \tag{11}$$

Conditioned on any amount of data, the posterior can be fully defined by the prior parameters, the sum of the sufficient statistics, and the data points. The convenience of the conjugate prior also extends to computing the marginal distribution:

$$p(x \mid \boldsymbol{\lambda}) = \int p(x \mid \boldsymbol{\eta}) p(\boldsymbol{\eta} \mid \boldsymbol{\lambda}) \, \mathrm{d}\boldsymbol{\eta} \tag{12}$$

Using Equation 8 as the conjugate:

$$\begin{aligned} p(x \mid \boldsymbol{\lambda}) &= h(x) \int \exp(\boldsymbol{\eta} \cdot \mathbf{t}(x) - A(\boldsymbol{\eta})) \cdot h(\boldsymbol{\eta}) \exp(Q - A(\boldsymbol{\lambda}) d\boldsymbol{\eta}) \\ &= h(x) \exp(-A(\boldsymbol{\lambda})) \int h(\boldsymbol{\eta}) \exp((\lambda_1 + \mathbf{t}(x)) \cdot \boldsymbol{\eta} + (\lambda_2 + 1)(-A(\boldsymbol{\lambda}))) \, \mathrm{d}\boldsymbol{\eta} \\ &= h(x) \exp(A(\lambda_1 + \mathbf{t}(x), \lambda_2 + 1) - A(\boldsymbol{\lambda})) \end{aligned}$$

If the log partition function $A(\cdot)$ is easy to compute, then the marginal distribution will, by extension, also be simple.

2 Problem Statement and Related Works

2.1 Term Frequency, Inverse Document Frequency

Term Frequency, Inverse Document Frequency or (tf-idf) is a statistical method to reduce a corpus of documents into a matrix of words (features). The metric is proportional to the number of appearances of a word and inversely proportional to the number of appearances across the entire corpus. This measure was proposed early on, during the first attempts to analyze textual data by machine [1]. While simple, the metric has proved effective and is still used in modern analysis [12, 13].

2.1.1 Term Frequency

There are various approaches to measuring term frequency used throughout the literature:

Weighting Scheme	Term Frequency Weight
Boolean	$\{0, 1\}$
Raw Frequency	$f_{w,d}$
Log Normalization	$1 + \log f_{w,d}$
Augmented Normalization	$K + (1 - K)\frac{f_{w,d}}{f_{\max}}$

Table 2: Term Frequency weighting schemes

Where $f_{w,d}$ is the frequency of word w in document d.

$$f_{w,d} = \frac{\text{count}(w)}{|d|} \tag{13}$$

The simplest choice is simply to use the *raw frequency* of a word in a document, the number of times that the term t appears in a document d. Boolean frequencies are 1 if the term appears at all or 0 if it is absent. Augmented normalization prevents longer documents from biasing the frequency.

2.1.2 Inverse Document Frequency

The *inverse document frequency* is a measure of how much information a word provides. This metric measures the occurrence of a word across all of the documents in the corpus.

Weighting Scheme	Inverse Document Frequency Weight
Unary	1
Inverse Frequency	$\log \frac{N}{n_w}$
Smooth Inverse Frequency	$\log\left(1 + \frac{N}{n_w}\right)$
Max Inverse Frequency	$\log(1 + \max(n_w))$
Probabilistic Inverse Frequency	$\log \frac{N - n_w}{n_w}$

Table 3: Inverse Document Frequency weights

Where N is the number of documents in the corpus and n_w is the inverse document frequency and D is the corpus:

$$\text{idf}(w, D) \equiv n_w = \log \frac{N}{1 + |d \in D : w \in d|} \tag{14}$$

Where the denominator $|d \in D : w \in dr||$ represents the number of documents which contain word w. Combining 13 and 14 and one of the proposed weighting schemes results in the metric used throughout the following sections.

2.2 Latent Semantic Indexing

Latent Semantic Indexing (LSI) is an automatic indexing method for textual information developed by Deerwester et. al. in [2]. The basic premise of this method is to use the tf-idf representation of the corpus and perform a singular-value decomposition (SVD). By selecting a small number of singular values for each document a lower-dimensional representation of the corpus may be found.

Documents are represented as td-idf vectors and similarity is computed via the cosine similarity in the latent space. Clustering in the latent space produces

the notion of "topics" or latent semantic meaning (hence the name of this process).

The primary drawback of this method is that it has difficulty dealing with *synonymy* and *polysemy*. Synonymy is the property of a language that allows multiple words or symbols to refer to the same thing. Polysemy describes how a single word or symbol may have multiple meanings. Moreover, the latent space representation does not allow for mixtures of topics. The clustering assumes that each document belongs only to one cluster of tf-idf terms.

2.3 Probabilistic Latent Semantic Indexing

Probablistic Latent Semantic Indexing pLSI is a statistical adaptation of LSI developed first by Hofmann in [3]. Rather than using singular values to decompose the tf-idf vector space, pLSI considers a mixture of multinomial distributions:

$$P(d, w) = P(d)P(w \mid d)$$
$$P(w \mid d) = \sum_{z \in Z} p(w \mid z)p(z \mid d)$$

Where d is a document and w is a word and z is a class label or topic. Since $|Z| \ll |D| \leq |W|$, the set of topics strictly reduces the dimension of the corpus.

$$P(d, w) = \sum_{z \in Z} P(z)P(d \mid z)P(w \mid z) \tag{15}$$

Which is symmetric in topics, documents, and words.

This model is fit using expectation-maxmization (EM) in two steps. The E-step first computes the posterior probabilities for the latent variables:

$$P(z \mid d, w) = \frac{P(z)P(d \mid z)P(w \mid z)}{\sum_{z' \in Z} P(z')P(d \mid z')P(w \mid z')} \tag{16}$$

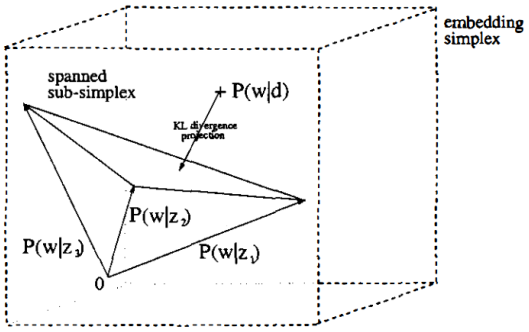

Figure 3: Probability sub-simplex spanned by model embedded with tf-idf space [3]

And the M-step updates the parameters:

$$P(w \mid z) \propto \sum_{d \in D} f(w,d) P(z \mid d, w)$$

$$P(d \mid z) \propto \sum_{w \in W} f(w,d) P(z \mid d, w)$$

$$P(z) \propto \sum_{d \in D} \sum_{w \in W} f(w,d) P(z \mid d, w)$$

Where $f(w,d)$ is the term-frequency defined in 13. The class-conditional multinomial distribution $P(\cdot \mid z)$ over the vocabulary can be represented as point on the $M-1$ dimensional hyper-simplex. Where M is the size of vocabulary and K is the number of topics. The convex hull or sub-simplex defined by the set of K points. As stated earlier since $K \ll M$ the sub-simplex is embedded within the higher dimensional space.

2.4 Mixture of Unigrams

The mixture of unigrams model is closely related to the the latent Dirichlet allocation. While it is not widely used in text modeling, understanding this model makes the details of LDA simpler to comprehend. LDA is strictly superior to this model as it is better able to encompass polysemy: the ability for a word or symbol to have multiple meanings—usually dependent on context. In

some interesting cases, words can have opposite meanings. For example "peruse" can mean to read closely "the lawyer perused the terms of the contract" or to browse or glance "I perused the ice cream selection at the store".

This model, as those described previously, also relies on tf-idf vectors. For further details on the following explanation, the reader is referred to [14, 15]. Assume, for the purposes of explanation, that there are two topics—space exploration and an account of Christian missions in the Americas—($K = 2$), twenty articles ($N = 20$) and each article consists of n_i words drawn from a vocabularly of eight words ($V = 8$). The model makes three key assumptions:

1. Documents in a topic are likely to share a set of words
2. We assume as a prior, that each topic is equiprobable for any document
3. Similary, each word can appear in any document with equal probability

This can be described by the following relations:

$$
\begin{aligned}
w_{ij} \mid z_i &\sim \mathcal{M}(1, \phi_{z_i}), j = 1, \ldots, n_i \text{ IID} \\
z_{1:\,N} &\sim^{\text{IID}} \mathcal{M}(1, \theta) \text{ independent} \\
\theta &\sim \mathcal{D}(\alpha_{1:\,K}) \\
\phi_{1:\,K} &\sim \mathcal{D}(\beta_{1:\,V})
\end{aligned}
$$

Where $\mathcal{M}$ and $\mathcal{D}$ denote multinomial and Dirichlet distributions, respectively, and w_{ij} is the jth word of document i. The topic of the ith document is z_i—either 1 or 2 in this example with two topics. The hyper-parameters $\alpha_{1:\,K} = (1, 1)$ and $\beta_{1:\,V} = (1, 1, 1, 1, 1, 1, 1, 1)$ are the parameters for the Dirichlet distributions expressing the priors described in the assumptions above. Note that $\mathcal{D}(1, 1)$ is equivalent to a Beta distribution with shape parameters $\alpha = 1, \beta = 1$.

The following is a summary of the definitions used in this model:

- N the number of documents
- V the number of unique words (or vocabulary)
- K the number of topics
- w_{ij} jth word of the ith document
- z_i the topic of the ith document
- n_i the length of the ith document (in words)
- α parameter of the document-topic distribution
- β parameter of the topic-word distribution

Assume that the vocabulary and word frequencies of the documents is as described in Table 4 and that the initial document-topic vector is $z = (2, \ldots, 2)$, that is to say we initially assign all the documents, arbitrarily, to Topic 2.

#	Topic	space	orbit	launch	mission	god	christian	church	faith
1	1	1	0	0	2	0	3	2	5
2	1	0	0	0	3	1	0	6	1
3	2	2	2	9	1	0	0	0	0
4	2	1	1	0	4	0	0	0	1
5	1	0	0	1	1	l	3	0	9
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

Table 4: The vocabulary matrix for the example text corpus. The first column indicates the document; the second the topic label; the remaining columns are the frequencies of the each word.

As we observe the words in each document, we update each of the distributions. From Table 4, we see that "orbit" should be strongly associated with Topic 1, space exploration, all documents in Topic 1 contain it and no documents in

Topic 2 do.

$$
\begin{aligned}
p(\phi_{1:\,K} \mid z_{1:\,N}, w) &\propto p(w \mid \phi_{1:\,K}, z_{1:\,N}) p(\phi_{1:\,K}) \\
&= \prod_{i=1}^{N} \prod_{j=1}^{n_i} \mathcal{M}(w_{ij} \mid \phi_{z_i}) \times \prod_{k=1}^{K} \mathcal{D}(\phi_k \mid \beta_{1:\,V}) \\
&\propto \prod_{i=1}^{N} \prod_{v=1}^{V} \phi_{z_i,v}^{f_{i,v}} \times \prod_{k=1}^{K} \prod_{v=1}^{V} \phi_{k,v}^{\beta_v - 1} \\
&= \prod_{i=1}^{N} \prod_{v=1}^{V} \phi_{k,v}^{e_{k,v}} \times \prod_{k=1}^{K} \prod_{v=1}^{V} \phi_{k,v}^{\beta_v - 1} \\
&= \prod_{i=1}^{N} \prod_{v=1}^{V} \phi_{k,v}^{e_{k,v} + \beta_v - 1} p(\phi_{1:\,K} \mid z_{1:\,N}, w) \\
&= \prod_{k=1}^{K} \mathcal{D}(\phi_k \mid \beta_1 + e_{k,1}, \ldots, \beta_V + e_{k,V}
\end{aligned}
$$

Where $f_{i,v}$ is the frequency of the vth word in ith document and $e_{k,v}$ is the frequency the vth word is assined to the kth topic.

Having associated a word with a topic, we now update the associations of topics with documents. Using the previous example word "orbit" we will update the third document—with three occurences of the word—to the first topic. This would change the document-topic vector to $z = (2, 2, 1, \ldots, 2)$. This update is computed as follows:

$$
\begin{aligned}
p(z_i = k \mid w, \phi_{1:\,K}, \theta) &\propto p(w \mid z_k, \phi_{1:\,K}) p(z_i = k \mid \theta) \\
&\propto p(z_i = k \mid \theta) \prod_{j=1}^{n_i} p(w_{ij} \mid z_i, \phi_{1:\,K}) \\
&= \theta_k \prod_{v=1}^{V} \phi_{k,v}^{f_{i,v}} \\
p(z_i \mid w, \phi_{1:\,k}, \theta) &= \mathcal{M}(z_i \mid 1, s^{(i)})
\end{aligned}
$$

Where,

$$s^{(i)} \propto \left(\theta_1 \prod_{v=1}^{V} \phi_1^{f_{i,v}}, \theta_2 \prod_{v=1}^{V} \phi_2^{f_{i,v}}, \ldots, \theta_K \prod_{v=1}^{V} \phi_K^{f_{i,v}} \right) \tag{17}$$

Having done the above for each word in each document, we update the proportions of each topic in the corpus. Our naive assumption of 100% Topic 2 will be updated to take into account any assignments we have made in the previous step.

$$\begin{aligned}
p(\theta \mid z_{1:\,N}) &\propto p(z_{1:\,N} \mid \theta) p(\theta) \\
&= \prod_{i=1}^{N} p(z_i \mid \theta) \prod_{k=1}^{K} \mathcal{D}(\theta_k \mid \alpha) \\
&\propto \prod_{k=1}^{K} \theta_k^{d_k} \prod_{k=1}^{K} \theta_k^{\alpha-1} \\
&= \prod_{k=1}^{K} \theta_k^{d_k+\alpha-1} \\
p(\theta \mid z_{1:\,N}) &= \prod_{k=1}^{K} \mathcal{D}(\theta_k \mid \alpha_1 + d_1, \alpha_2 + d_2, \ldots, \alpha_K + d_K)
\end{aligned}$$

Where d_k is the number of documents assigned to Topic k. Computing these updates over and over is an instance of *Gibbs sampling* and the document-topic and word-topic distributions will converge to a representation reflecting the articles' word frequencies.

2.5 Problem Statement

An electroencephalograph or EEG is a monitoring technique that records electrical activity in the brain [16]. The procedure is typically used for the diagnosis of epilepsy, as wells as, sleep disorders, coma, and other diseases of the brain. EEGs are performed in an attempt to link activity in the brain to physiological conditions such as seizures or sleep disturbances [17].

Using the textual data outlined in Section 2.7, this work attempts to use topic modeling to cluster patient observations into meaningful lower-dimensional representations. Ideally, these groups would share similar medical profiles, such as symptoms, medication, or diagnosis. This would allow for simpler comparison of patient profiles and may aid in treatment or in diagnosis. The clusters may also serve exploratory or instructional purposes allowing physicians to draw new inferences from the dataset.

2.6 Supervised Test Data Sets

2.6.1 20-newsgroups

The 20-newsgroups dataset is a collection of approximately 20,000 newsgroup documents (or forum posts) evenly distributed across twenty different newsgroups [18]. In use for over twenty years in machine learning literature for text classification and clustering, this dataset has become a standard supervised test set.

Group	Topics
comp	graphics, ms-windows.misc, sys.ibm.pc.hardware, sys.mac.hardware, windows.x
rec	autos, motorcycles, sport.baseball, sport.hockey
sci	crypt, electronics, med, space
talk	politics.misc, politics.guns, politics.mideast, religion.misc
alt	atheism
soc	religion.christian
misc	forsale

Table 5: 20-newsgroups groups and topics. Each group may contain various topics and sub-topics.

Testing the topic model on this large dataset is useful, as it demonstrates the performance of the model on a very large corpus with a relatively sparse vocabulary. Given that these are relatively short forum posts common words are often repeated throughout the corpus. This provides an interesting challenge

for the model as there are fewer informative words per document making topic clustering more difficult. The results are more sensitive to the size of the initial vocabulary as well as to the stemming and stop word policy used in preparing the features.

2.6.2 NIPS Abstracts

Neural Information Processing Systems (NIPS) is one of the top machine learning conferences in the world [19]. This conference covers various topics including deep learning, computer vision, cognitive science, and reinforcement learning. This dataset contains the entire corpus of submissions for the 2015 NIPS Conference (over 400 total papers). The data is structured as a single comma-separated-value (CSV) file and includes: the title of the paper, abstract, poster/oral presentation flag, and the raw text converted from the PDF submission.

Analysis of the raw text is hampered by the incomplete conversion of tables as well as captions and special characters such as ligatures. The abstract information was scraped from the NIPS website as raw text and does not suffer from PDF conversion issues.

This corpus presents an analogue to the medical texts used in the unsupervised portion of this analysis. Both are small corpora consisting of short documents with highly technical vocabularies. The preliminary analysis on this supervised training set can be meaningfully extrapolated to the performance on the medical data, absent the judgement of a trained medical professional. That is to say the relative performance of the algorithm should be comparable across both corpora.

2.7 Medical Data Set

The unsupervised medical data set is drawn from [20]. The corpus consists of nearly 4,000 documents each corresponding to the notes of an attending physician or nurse during the course of an EEG scan. The corpus was collected by Temple University Hospital over the course of an 11-year period from 2002–2013. While the dataset also contains over 20,000 EEG scans, the focus here to infer patient clusters solely from the *textual data* provided. This represents a subset of the patients in the original data; while several patients underwent multiple EEG scans, not all had recored observations. Each document notes the patient's symptoms, medication (if applicable) and any notes from the duration of the brain scan. The document is partitioned accordingly.

Listing 1: Sample Patient Record

```
CLINICAL HISTORY: A 30-year-old woman, with epilepsy, 27 weeks
   pregnant. The patient was admitted with a change in mental
   status. She had a seizure on the 9th lasting 45 seconds and
   then another event today, question encephalopathy, question
   postictal, question behavior.

MEDICATIONS: Metoprolol, heparin, Keppra, Ativan.

INTRODUCTION: Digital video EEG was performed in the lab using
   standard 10-20 system of electrode placement with one channel
    of EKG. The patient had been sedated, had a recent seizure
   and was quite somnolent.

DESCRIPTION OF THE RECORD: The initial sections of the EEG
   demonstrate a low voltage slow pattern with slow delta. With
   stimulation, a scant 8 Hz alpha rhythm was noted. The
   technologist was able to awaken the patient for
   hyperventilation which seems to produce some rhythmic,
   sharply contoured slowing in the right hemisphere,
   particularly in the right mid to anterior temporal region. As
    the patient becomes drowsy bursts of high amplitude frontal
   delta were noted.

Photic stimulation did not activate the record. Heart rate 78
   BPM.

IMPRESSION: This is an abnormal EEG due to:
* Abnormal background with a low voltage slow pattern.
* Excess theta.
* Rhythmic right delta particularly in the right mid temporal
   region notable with hyperventilation.
* FIRDA.

CLINICAL CORRELATION: This EEG demonstrates a diffuse
   disturbance of cerebral function which may be due to a
   postictal state or medications. In addition, there are focal
   features with rhythmic, sharply contoured slowing in the
   right temporal region. Based on these findings, EEG
   monitoring was requested.
```

3 System Description

3.1 Tokenization

Tokenization is the process of splitting a sequence of characters into individual words and symbols. This process is difficult to solve in general; this work solves the problem for specific data sets with customized solutions. For example the test data in the 20-news-groups contains untokenized forum input. Since the data is not curated or filtered, it contains various special characters, substitutions, colloquialisms, and misspellings. Moreover, the documents vary in length from empty strings to over 60,000 characters.

Listing 2: Sample Forum Post

```
Archive-name: graphics/resources-list/part1
Last-modified: 1993/04/17


Computer Graphics Resource Listing : WEEKLY POSTING [ PART 1/3 ]
============================
Last Change : 17 April 1993


Many FAQs, including this Listing, are available on the archive
   site pit-manager.mit.edu (alias rtfm.mit.edu) [18.172.1.27]
   in the directory pub/usenet/news.answers. The name under
   which a FAQ is archived appears in the Archive-name line at
   the top of the article. This FAQ is archived as graphics/
   resources-list/part[1-3]
```

```
There's a mail server on that machine. You send a e-mail message
    to mail-server@pit-manager.mit.edu containing the keyword "
   help" (without quotes!) in the message body.

You can see in many other places for this Listing. See the item:

0. Places to find the Resource Listing
...
```

[h]

The splitting of words is done with a custom regular expression that searches for special characters, apostrophes, number, and punctuation.

Listing 3: Sample regular expression for tokenization

```
  '(?!(d|m|t|ll|ve)\W)|[.,\-_!?:;()0-9@=+^*`~#&|\t\n+
```

Applying this regular expression to the corpus generates a sequence of separated words, or tokens.

3.1.1 Stop Words

The sequence of separated words is then parsed for any *stop words*. These are words which are filtered out prior to any further processing. They consist of common English words such as articles, prepositions, and conjunctions. This work uses a custom set of words based on the list provided in the package [21]. While tf-idf features would typically filter out these common words, explicit stop words allows for the use of a larger vocabulary without re-introducing these words into the model.

3.2 Stemming and Lemmatization

Stemming is a linguistic operation that removes the suffixes from individual words [22].*Lemmatization* is a closely related operation that reduces a word to its linguistic root. These associative maps are useful when working with bag-of-words models such as LDA as they reduce the number of related or redundant entries in the vocabulary matrix. For example the words "fish", "fishing", and "fisher" will all be stemmed to the root "fish". Lemmatization takes this process further: "is", "was", and "be" will all be mapped to the morphological root "be". Note how stemming only truncates prefixes and suffixes, while lemmatizing is a custom map. The system described here uses the Snowball stemmer [23].

3.2.1 Snowball Stemmer

Snowball is a string processing language and library incorporated into Python [23]. The basic algorithm in Snowball is the Porter stemming algorithm [24,25]. As with nearly all stemmers, the Porter stemmer is based on the formal written rather than the spoken language. This assumption may affect the parsing of less formal corpora as well as slang expressions and other colloquialisms.

This stemmer contains both algorithms to remove specific types of suffixes as well as a dictionary mapping conjunctions of verbs that change their forms. The stemming algorithm recognizes various suffix forms including:

- *a-suffixes*, are attached, such as "loving" + "-ly" = "lovingly"
- *i-suffixes*, are inflectional, such as " fit" + "-ed" = "fitted" (doubled t) or "love" + "-ed" = "loved" (drop final e)
- *d-suffixes*, are derivational, "little" + "-ness" = "littleness"

Suffixes may be combined or cascaded to form new words: *i-suffixes* can follow or precede *d-suffixes*, but the latter is much rarer. For example: "love" + "-ing" (*d-suffix*) + "-ly" (*i-suffix*) = "lovingly"; "devote" + "-ed" (*i-suffix*) + "-ness" (*d-suffix*) = "devotedness".

Note that suffixes can serve multiple purposes. For example, "-ly" is a standard way of turning an adjective into an adverb ("great" to "greatly") but can also turn a noun into an adjective ("love" to "lovely"). Since the order of the suffixes and their usage is so varied, words are often conflated and mapped to the same root. This is often desirable, as is reduces the size of the vocabulary, but may also result in errors. Consider the pairs of words presented in Table 6.

Verb	Adjective	Root
prove	provable	prov-
probe	probable	prob-?

Table 6: An example of over-stemming. Note that while the pairs of words are morphologically identical, stemming "probable" and "probe" to the same root would be an error.

To prevent these types of errors, most stemmers also incorporate a dictionary of common words and roots or exceptions to specific suffix rules. Despite these conflations and over-stemming issues, stemming remains a useful and effective way to reduce vocabulary sizes and to associate words across their various conjugations and forms.

3.3 Feature Selection

Beyond the applications of tokenization, stemming, and lemmatization, feature selection for the tf-idf family of models consists selecting the vocabulary size. Determining the vocabulary size was done via an exploratory iterative process

optimizing for model log-likelihood and perplexity metrics. Beyond a certain size, increasing the vocabulary size provides little added benefit as increasing the number of tf-idf terms adds less and less informative words.

For each dataset, vocabulary size was chosen in proportion to the size of the corpus. This was done to reduce overfitting as well as noise from less informative words in each document. The 20-newsgroups dataset provided a testing ground to investigate the effect of vocabulary size on the generated topics. Vocabularies of 10,000, 5,000, 1,000 and 100 words were used to train a 6-topic model on a dataset to 4,000 documents. The actual number of topics in this data was reduced to be six by removing fourteen of the original twenty topics. The following topics were used here: "comp.graphics", "rec.sport.baseball", "rec.autos", "sci.med", "sci.space", "soc.religion.christian".

Notice how the reduction of vocabulary size not only affects the topic location in the latent space, but may cause the merging or splitting of clusters. For example, in reducing the vocabulary from 10,000 to 5,000 words the cluster (car, orbit, launch) becomes (car, game, good) indicating a separation of the notion of "transport" and the merging of the cars topic with sports. While the (space, nasa, center) is further refined into (space, launch, develop); this is an improvement, since it is now separate from the cars/auto cluster.

Interestingly, the 1,000 word representation reintroduces the baseball/sports topic as (game, team, hit) while maintaining an excellent separation between cars/auto and space. However, the medicine forum is reduced to the rather simplistic (time, problem, people) which does provide a description of medicine, albeit a crude one.

Further reduction of the vocabulary (i.e. lower-dimensional representations) of the document space, lead to cruder topics and more collisions. It is interesting

Size	Topic Vectors
10,000	car, orbit, launch god, christian, people game, good, team image, file, graphic food, doctor, pain space, nasa, center
5,000	car, game, good god, christian, jesus people, thing, time imag, file, program food, pain, doctor space, launch, develop
1,000	car, drive, engin god, christian, people game, team, hit imag, file, graphic time, problem, people space, launch, orbit
500	car, time, problem god, jesus, church game, team, hit imag, file, graphic people, thing, question space, launch, orbit
100	car, time, good god, christian, people point, game, day imag, file, graphic univers, center, object space, launch, nasa

Table 7: Topic vectors for various vocabulary sizes

to note however that certain topics remain cohesive and unchanged such as (image, file, graphic) and (god, christian, people) suggesting that these particular clusters are very compact in the latent space. One would expect such clusters to be "far" from the others and have a clearly defined boundary.

3.4 LDA Algorithm

3.4.1 Notation and Terminology

As in previous sections describing related methods, this section will use the terminology of text collections referring to "words", "documents" and "corpora" as specific entities. The abstract notion of a "topic" is also introduced as a latent variable of such a collection. Note that while the applications presented herein are concerned specifically with textual data and the clustering of latent topics, there are numerous other applications of this technique not limited to clustering or even to textual data [6, 7].

The following definitions are used throughout the following section:

- A *word* is the basic unit of discrete data, a single item from the vocabulary. Each word is a unit-basis vector of dimension V.
- The *vocabulary* is the unique set of all words across the entire corpus represented as an M by V matrix.
- A *document* is a sequence of N words denoted by $\mathbf{w} = (w_1, w_2, \ldots, w_N)$.
- A *corpus* is a collection of M documents denoted by $\mathbf{C} = (\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_M})$.

3.4.2 Model Details

The latent Dirichlet allocation (LDA) is a generative probabilistic model of a corpus. The model was developed and further developed in [7, 10, 26]. The

model's primary assumption is that documents may be represented as random mixtures over latent topics. Each topic is itself a distribution over the corpus' vocabulary.

The model assumes the following generative process for each document in the corpus:

1. Choose an ancillary distribution for $N \sim \mathcal{P}(\xi)$
2. Choose $\boldsymbol{\theta} \sim \mathcal{D}(\boldsymbol{\alpha})$
3. For each of the N words $\mathbf{w}_n$:
   (a) Choose topic $z_n \sim \mathcal{M}(\boldsymbol{\theta})$
   (b) Choose a word w_n from $p(w_n \mid z_n, \boldsymbol{\beta})$, multinomial conditioned on the topic z_n

Where $\mathcal{P}$ is a Poisson distribution, $\mathcal{D}$ is a Dirichlet distribution and $\mathcal{M}$ is a multinomial distribution.

Three simplifying assumptions were made above. First, the dimension of the Dirichlet distribution, k, is assumed to be known and fixed. This also fixes the dimension of the topic variable, z. Second, the word probabilities in the $k \times V$ matrix $\boldsymbol{\beta}$, where $\boldsymbol{\beta}_{ij} = p(w_j = 1 \mid z_i = 1)$, is also assumed to be fixed quantity. Lastly, the assumption of an ancillary Poisson distribution for the document length distribution is not critical and was chosen here for simplicity. A more realistic distribution may substituted as needed.

A k-dimensional Dirichlet random variable $\boldsymbol{\theta}$ can take values in the $(k-1)$ hyper-simplex and has the following probability density over this hyper-simplex:

$$p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{j=1}^{k} \Gamma(\alpha_j)} \prod_{m=1}^{k} \theta_m^{\alpha_m - 1} \tag{18}$$

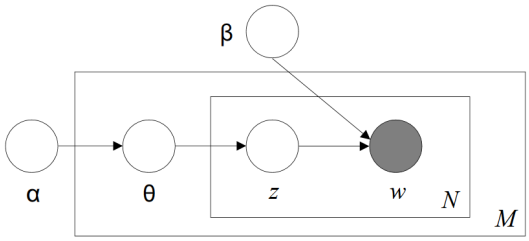

Figure 4: Plate notation for LDA algorithm. The outer plate represents the document level. The inner plate represents the repeated selection of topics and words for a document. Figure from [7]

Where the parameter $\boldsymbol{\alpha}$ is a k-vector and its components $\boldsymbol{\alpha}_i > 0$, and $\Gamma(\cdot)$ is the Gamma function. The Dirichlet is chosen here because it is well-behaved over hyper-simplexes, is in the exponential family, has finite sufficient statistics, and is the conjugate to the multinomial distribution. Each of the properties makes the computation of the inference and parameter estimation algorithms viable.

Given the parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$ the joint distribution for a topic mixture $\boldsymbol{\theta}$, a set of N topics $\mathbf{z}$, and a set of N words $\mathbf{w}$ is given by:

$$p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{w} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}) = p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) \prod_{n=1}^{N} p(z_n \mid \boldsymbol{\theta}) p(w_n \mid z_n, \boldsymbol{\beta}) \tag{19}$$

Where $p(z_n \mid \boldsymbol{\theta})$ is just $\boldsymbol{\theta}_i$ for the unique index i such that $z_n^i = 1$. The marginal distribution of a document is given by:

$$p(\mathbf{w} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}) = \int p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n \mid \boldsymbol{\theta}) p(w_n \mid z_n, \boldsymbol{\beta}) \right) \mathrm{d}\boldsymbol{\theta} \tag{20}$$

And the probability of a corpus is:

$$p(\mathbf{C} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{d=1}^{M} \int p(\theta_d \mid \boldsymbol{\alpha}) \left(\prod_{n=1}^{N_d} \sum_{z_d n} p(z_d n \mid \theta_d) p(w_d n \mid z_d n, \boldsymbol{\beta}) \right) \mathrm{d}\theta_d \tag{21}$$

The plate notation for this model is shown in Figure 4.

The latent multinomial variables are deemed "topics", but the author makes no claim to their comprehensibility to a human researcher nor to their intelligibility, beyond their probabilistic definition.

As in the multinomial mixture model, the LDA model also allows for each document to become associated to more than one topic. This is due to the sampling of the topic node repeatedly within the document, see Figure 4. While documents may exhibit multiple topics, note that the hyper-simplex restriction forces the sum of this topic-vector to be equal to one. This produces a slightly negative correlation between topics and might lead to truncation effects since all topic are effectively scaled to the largest component.

The plate notation in Figure 4 is common the in the Bayesian statistical modeling world for more depth on this topic, the reader is referred to [14, 15]. The hyper-parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$ are estimated using an empirical Bayes approach.

3.4.3 Geometric interpretation

An interesting way of viewing the LDA model output is by observing the geometry of the latent space. Figure 5 demonstrates the simple example of a corpus with three unique words and four topics. This results in a surface of multinomial distributions over a (V-1)-simplex (or two-dimensional triangle). In this case, the topics represented are $z_1 = (1, 0, 0), z_2 = (0.1, 0.05, 0.85), z_3 = (0.04, 0.91, 0.05), z_4 = (0.4, 0.35, 0.25)$.

This same geometric analysis may be used to compare LDA to the previous models discussed. Taking the perspective in Figure 5 and shifting to a top-down projection, this analysis is seen in Figure 6. This projection allows for the simultaneous comparison of mixture of unigrams, pLSI and LDA models:

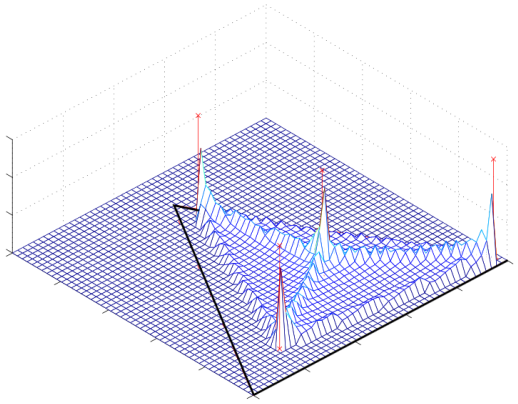

Figure 5: Example unigram density $p(\mathbf{w} \mid \boldsymbol{\theta}, \boldsymbol{\beta})$ for three words and four topics. This results in a surface over a 2-simplex (i.e. an equilateral triangle). Each vertex is a deterministic distribution that assigns probability one to one of the words. The four points marked on the surface are the locations of the multinomial distributions $p(\mathbf{w} \mid z)$ for each of the four topics [7].

- The mixture of unigrams model assumes that, for each document, one of the corners of the simplex is chosen and all the words of the document are drawn from the corresponding distribution

- The pLSI model allows each word in the training documents to come from a different topic. The topics are drawn from empirical distributions within the boundaries of the simplex. There is one distribution per document and the training distribution defines the empirical distribution.

- The LDA model allows for both observed and unobserved documents to have words generated from a randomly chosen topic. The topics are drawn from a distribution with a randomly chosen parameter. This parameter is sampled once per document from a smooth distribution on the topic simplex.

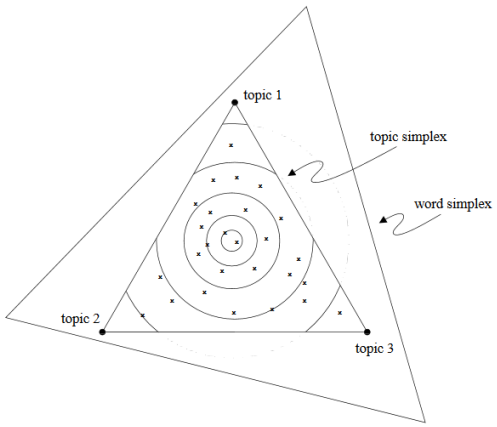

Figure 6: Example topic simplex for three words and three topics. As in Figure 5, each vertex corresponds to a distribution where each word has probability one. The inner simplex illustrates the topics, which are different distributions over words. The unigrams models places each document at a vertex of the inner simplex; the pLSI model at one of the empirical distributions marked by an "x"; and the LDA model places each document within a smooth distribution in the inner simplex illustrated by the contour lines [7].

By introducing the additional parameter $\boldsymbol{\theta}$ on a per-document basis the LDA model improves upon the previous topic models in that it allows for a smooth, continuous distribution of topics in the latent space.

3.4.4 Inference and Parameter Estimation

The efficient inference scheme presented here was adapted from [27, 28].

The key problem that needs to be solved in order to use LDA is computing the posterior distribution of the hidden variables, given a document. Loosely, this can be thought of the reversal of the generative process of creating a document by drawing words out of the topic distributions mentioned previously. The posterior is of the form:

$$p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{w} \mid \boldsymbol{\alpha}, \boldsymbol{\beta})}{p(\mathbf{w} \mid \boldsymbol{\alpha}, \boldsymbol{\beta})} \tag{22}$$

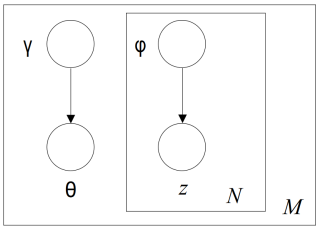

Figure 7: Plate notation for approximate LDA posterior. The edges between $\boldsymbol{\theta}$, $\mathbf{z}$, and $\mathbf{w}$ are severed and the free parameters $\boldsymbol{\gamma}, \boldsymbol{\varphi}$ are introduced [7]

However, due to the coupling between $\boldsymbol{\theta}, \boldsymbol{\beta}$ the denominator of this expression is intractable. To compute it one would have to sum over all possible configurations of the interdependent N topics assignment variables in z.

The solutions presented in [27, 28], use convexity-base variational inference to obtain a simpler distribution containing free *variational parameters*. These parameters are then fit so that they closely approximate the true posterior. Figure 7 shows this more tractable model.

This simplified graphical model is characterized by the following variational distribution:

$$q(\boldsymbol{\theta}, \mathbf{z} \mid \boldsymbol{\gamma}, \boldsymbol{\varphi}) = q(\boldsymbol{\theta} \mid \boldsymbol{\gamma}) \prod_{n=1}^{N} q(z_n \mid \varphi_n) \tag{23}$$

Where $\boldsymbol{\gamma}, \boldsymbol{\varphi}$ are the free variational parameters. Using this variational distribution, finding the values for $\boldsymbol{\gamma}, \boldsymbol{\varphi}$ is an optimization problem. This optimization problem is solved with an iterative method shown in A. The Kullback-Leibler (KL) divergence between the variational distribution and the true posterior $p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. Note that this variational distribution is conditional on $\mathbf{w}$. [h]

3.4.5 Model Smoothing

The dimensions of the vocabulary matrix for large corpora creates a sparsity problem. For example, the 20-new-groups corpus contains over 20,000 docu-

Listing 4: Variational inference for LDA

```
theta = [[i/k for i in k] for n in N]
gamma = [alpha(i) + N/k for i in k]
while not converged:
    for n in N:
        for i in k:
            theta[n][i] = beta[n][i])*exp(di_gamma(gamma[i]))
        theta[n] = [theta[n][i]/sum(theta[n]) for i in k]
    gamma = alpha + sum(theta[n])
    check(converged)
```

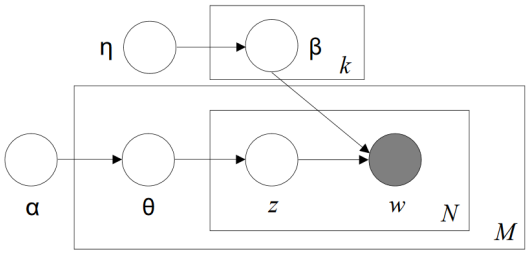

Figure 8: Plate notation for smoothed LDA model. The hyper-parameters $\boldsymbol{\alpha}, \eta$ are used to decouple $\boldsymbol{\beta}$ from $\boldsymbol{\theta}$. Figure from [7]

ments and is analyzed with a vocabulary of 10,000 unique words. If a subset of the documents is used for training, new documents introduced to the model are likely to contain words not not observed in the training set. This is further exacerbated if the 10,000 features are stripped of common words using a combination of tf-idf and stop-words. This will leave only "informative" words in each document. The most common approaches [29,30] smooths the maximum likelihood estimates of the multinomial and assigns a positive, non-zero probability to all vocabulary items even if they are not observed in the training set. The solution used here simply extends the graphical model by treating $\boldsymbol{\beta}$ as a $k \times V$ random matrix where each each row is drawn independently drawn from a Dirichlet with a single scalar parameter η. Iterating over nodes in Figure 8

This final model has only two hyper-parameters $\boldsymbol{\alpha}, \eta$ and variational expectation-maximization approach is used to find maximum likelihood estimates of these parameters (see Appendix A).

4 Results

4.1 Supervised Datasets

Both supervised datasets were processed in the same manner. The data files were first loaded and decoded in the UTF-8 format to support special characters. Each document was then cleaned, stopped and stemmed. This involved removing punctuation and special characters, as well as converting the entire document to lower case. The aforementioned list of the stop words was used in conjunction with the English Snowball stemmer. Finally, all words shorter than three characters were removed from the vocabulary.

Next, TF-IDF features were generated for each corpus. The 20-newsgroups dataset was processed with 10,000 features and the NIPS abstracts were processed with 5,000 features. The number of topics was varied in both datasets to observe the differences in the topic clusters—especially as the number of topics began to exceed the ground truth. The subsequent merging and splitting of these clusters suggests a latent topic hierarchy is present within both corpora.

4.1.1 20-newgroups

The 20-newsgroups data was run with a testing subset of six topics. These topics were chosen for their subjective conceptual proximity and divergence. To see if this subjective semantic distance was well captured by the model the number of topics was varied and the consequent topic agglomeration or fission was observed.

A clear hierarchy in the topics is evident by the way that topics tend to merge as the number of clusters is reduced from ground truth. First, the “auto”

#	Topic Vectors
1	time, god, space
2	space, imag, car god, peopl, christian
3	space, imag, program god, peopl, christian car good time
4	space, car, launch god, christian, peopl game, good, time imag, file, program
5	space, car, launch god, christian, peopl game, good, team imag, file, program medic, food, health
6	car, orbit, launch god, christian, peopl game, good, team imag, file, graphic food, doctor, pain space, nasa, center
7	car, time, orbit god, peopl, christian game, team, hit imag, file, graphic doctor, pain, day space, nasa, launch food, msg, eat
8	space, launch, orbit god, peopl, christian game, good, team imag, file, graphic pain, doctor, day space, post, health food, msg, eat car, drive, engin
9	space, launch, orbit god, christian, peopl game, team, hit imag, file, graphic pain, doctor, day space, health, medic food, msg, eat water, request, send car, time, good

Table 8: Topic vectors for various number of LDA topics (#). The ground truth number of topics is six. Note how the topics merge or split as the number of topics is decreased or increased, respectively.

topic merges with “space travel”, next “medicine” vanishes, then scientific and religious topics merge, until finally we are left with an interesting, if somewhat existential topic: “time, god, space”. The topic model clearly attempts to merge similar topics first.

Similarly, when increasing the number of topics discordant clusters are spun off first. First, a “diet” topic (food, msg, eat) is spun out of medicine—which becomes (pain, doctor, day)—then, “auto” becomes more defined and a strange “throwaway?” topic is created (space, post, health). Perhaps this topic represents some kind of spatial interpretation of health care? Further increasing the number of clusters generates a “water management” topic (water, request, send) and refines the previous throwaway topic to (space, health, medic) which is now a clear subdivision of the medicine topic.

The existence of these topic hierarchies is interesting and is highly suggestive of the quality of the underlying latent representation. Further, this structure presents an opportunity for further refinement and modeling in future work.

To test the accuracy of this clustering method, a simple logistic-regression classifier was trained with the document-topic matrix (the latent space topic affinities of each document). The latent representation proved highly useful and achieved a 70% accuracy rate over the corpus. A more complex classification algorithm may improve this result. But, for the purposes of this paper the result demonstrates the validity of the latent representation and is remarkable! Consider that the feature space was reduced from a 10,000 word vocabulary to a six-topic vector.

Visualizing the latent space defined by the topics is challenging. Since this space is already a highly reduced vector-space, further reducing the number of dimensions for easy visualization would be detrimental to the discovered

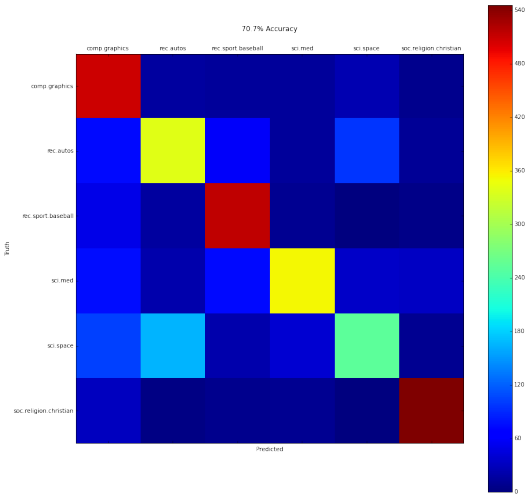

Figure 9: Confusion matrix for topic classifier. Note that the primary mode of confusion is between the autos and space topic. The semantic interpretation suggests that as both as vehicle related topics the confusion is not entirely without reason.

clusters. At the same time, visualizing higher-dimensional spaces is difficult and non-intuitive. To overcome this limitation, a manifold embedding into a two-dimensional space was used to attempt to preserve the distances in the higher-dimensional latent space. The TSNE (t-distributed stochastic neighbor embedding) was used to reduce the six-dimensional space. This embedding was then clustered with the HDBSCAN (hierarchical density-based spatial clustering of applications with noise) algorithm [31, 32]. This clustering algorithm is commonly used in scientific literature and operates in an agglomerative fashion, marking points belonging to a neighborhood (or cluster) as well as those that are outliers in the space. Results of both the embedding as well as the clustering are shown in Figures 10 and 11.

4.1.2 NIPS Abstracts

The NIPS dataset provided a second method to test the clustering of the model. Unlike the previous dataset, no target labels are provided, although the class of the document can be reasonably inferred from the abstract and

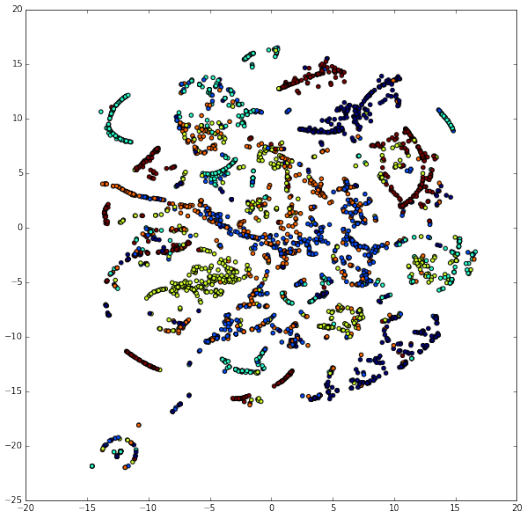

Figure 10: TSNE manifold embedding for 20-newsgroups document-topic matrix. The TSNE manifold embedded in a two-dimensional space. Each point represents a document and is colored by its ground truth topic.

Figure 11: HDBSCAN clustering for the 20-newsgroups. This hierarchical clustering model attempts to agglomerate the document-topic matrix in two dimensions.

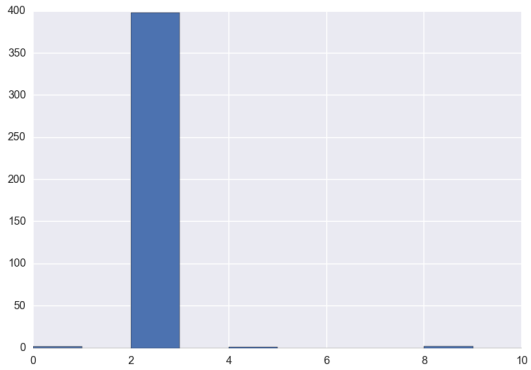

Figure 12: Distribution of most likely topic for ten topics. The distribution peaks at the second topic with over 99% of the values.

the indexing terms of the full-text. This provides a semi-supervised training set to observe clustering.

As a first pass the number of topics was naively selected to be ten clusters. While ten topics were produced by the model, the relative proportions of each topic per document was very unevenly distributed. Across nearly every document one topic was heavily favored. Reducing the number of topics to five had no effect on the distribution. For comparison the *most* and *least* likely topics across all documents were plotted. Both suggested that the true number of topics lay below the naive assumption. A two-topic model provided the better balance between the topics in the distribution. Of course, the model is only useful if more than one topic is present. This test draws an interesting conclusion: there is a fundamental limit to the resolution of this model. Beyond a certain point, documents may become too *similar* to differentiate in the latent space.

After reducing the number of expected clusters, the NIPS abstracts partition incredibly cleanly. Not only is the partition readily apparent in the TSNE embedding in Figure 16 the predicted class labels fall into a natural cluster as do the HDBSCAN agglomerative labels. The agreement of the three

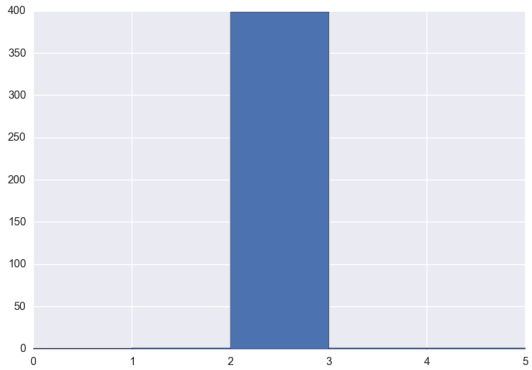

Figure 13: Distribution of most likely topic for five topics. Reducing the number to five topics has not helped as one topic still dominates the distribution.

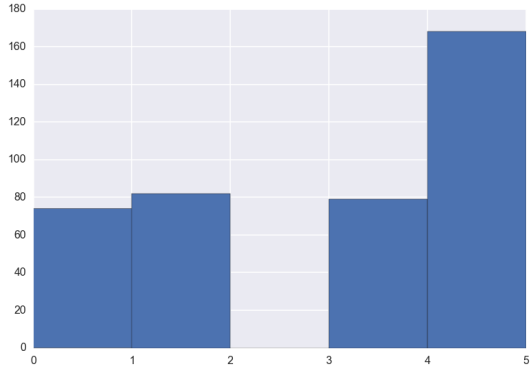

Figure 14: Distribution of least likely topic for five topics.

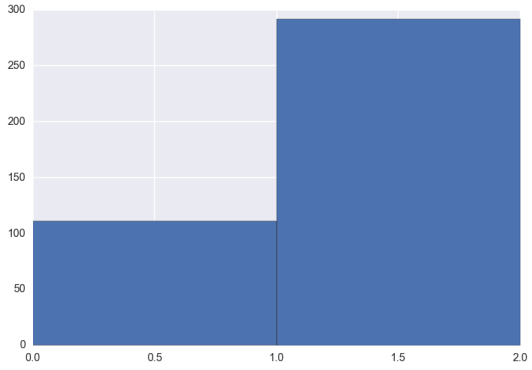

Figure 15: Distribution of least likely topic for two topics. Finally, a reasonable partition of the latent space.

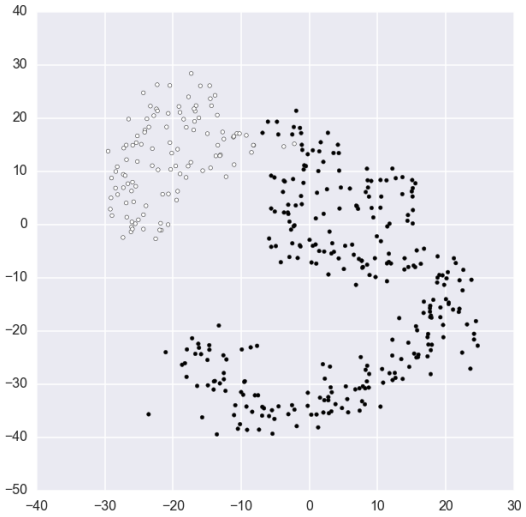

Figure 16: TSNE clustering of NIPS topics. Color indicates the most likely topic. Note that both the latent space and the color clustering are well separated.

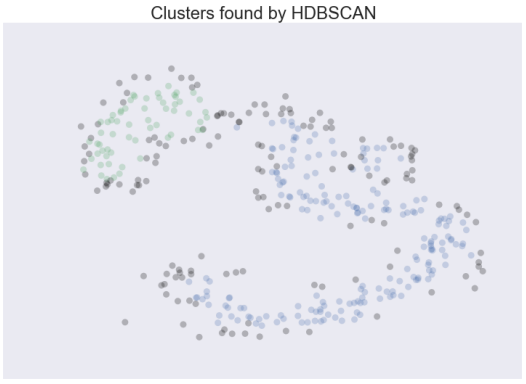

Figure 17: HDBSCAN clustering of NIPS topics. Color indicates agglomerative clustering class. Note the strong separation and similarity to Figure 16

separate methods is supported by the semantic construction of the topics. Topic #1 is (model, network, learn, train, imag, predict, neural, problem, algorithm, state)—a neural network image recognition topic—and Topic #2 is (algorithm, problem, propos, method, distribut, optim, learn, model, data, estim)—a Bayesian statistics and modeling topic. Interestingly, the model understood the semantic difference between the words "model" and "learn" within these two different contexts.

4.2 Medical Dataset

Having established the viability of this particular topic model in two supervised scenarios, the model is now applied to the aforementioned medical data set. Starting again with the naive assumption of ten topics, the LDA algorithm is run for 500 iterations. The log-likelihood is computed at the end of each iteration and convergence is usually obtained by iteration 100. The log-likelihood of each topic model serves as a rough estimate of how "good a fit" the number of topics is for the data. In Table 9 the minimum log-likelihood occurs when two topics are present in the model.

Plotting the *most likely topic* distribution supports the results in Table 9, the distribution is bi-modal with two sharp peaks and very small in all other topics (see Figures 18 and 19). For completeness, cluster graphs are shown for the case of two and three topics.

Even with the agreement between two separate metrics, a non-expert would have a difficult time justifying the choice in the number of topics. The following figures present a geometric argument by embedding the clusters in a two-dimensional space, but an expert opinion would be far more compelling.

Number of Topics	Log-likelihood
10	-2442210
6	-2435820
3	-2414470
2	-2388243
1	-2400736

Table 9: Log-likelihood for various numbers of topics on the medical data. Note how the function reaches a minimum a two topics. This suggests the true number of topics in the data set is two.

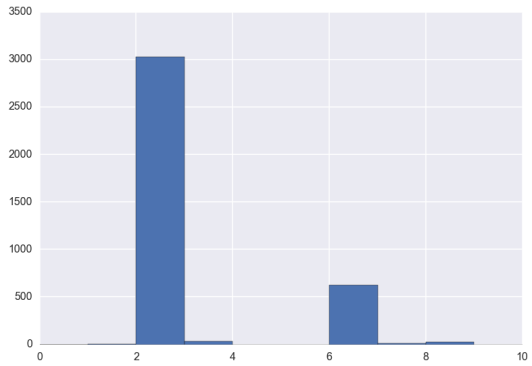

Figure 18: Distribution of most likely topic for ten topics. Note the two prominent peaks in the distribution.

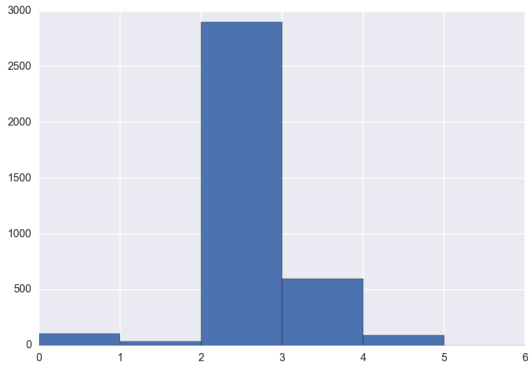

Figure 19: Distribution of most likely topic for six topics. Note that two prominent peaks in the distribution are still present, but reshuffled, due to the random initialization of the topics.

#	Topic Vectors
2	seizur slow sleep stimul photic activ abnorm left discharg wake activ slow background left seizur pattern wave burst record correl
3	photic stimul alpha sleep hyperventil rhythm wake drowsi featur channel activ background pattern slow burst left delta demonstr bedsid correl seizur slow activ left abnorm record discharg tempor sleep wave

Table 10: Clusters produced by two and 3-topic models.

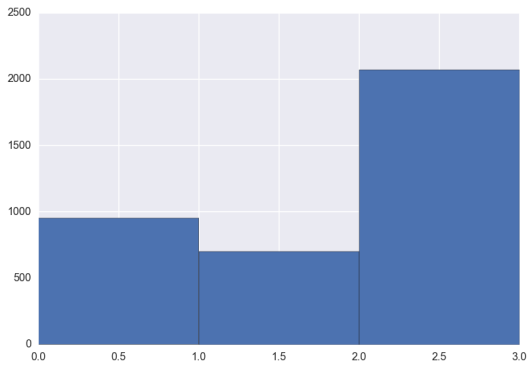

Figure 20: Distribution of most likely topic for three topics. Note that two prominent peaks in the distribution are still present, but are now less prominent.

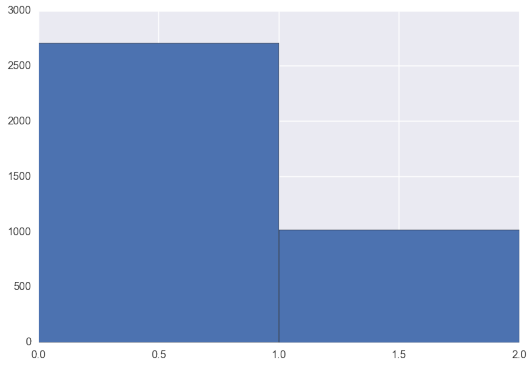

Figure 21: Distribution of most likely topic for two topics. Note that the third topic was merged into the larger of the two topics.

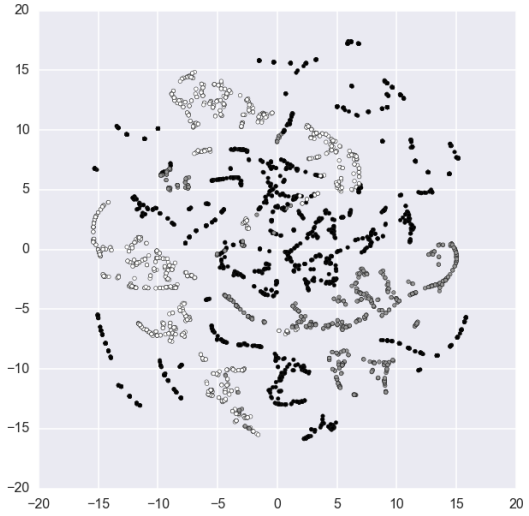

Figure 22: TSNE clustering of medical data 3-topic latent space. Color indicates most likely LDA topic label. Note the strong separation of Topics #1 and #2 but the interspersion of Topic #3. The third topic may or may not belong to the others.

The third topic that is merged when moving to the 2-topic model appears to deal with a sleep disorder given the keywords: alpha, sleep, rhythm, wake, and drowsy. The argument could be made that, despite the log-likelihood and cluster metrics, this is a legitimate medical topic. The other two topics appear to deal with different types of seizures, one is: (slow, sleep, seizure); the other is: (active, seizure, wave, burst). At the semantic level they appear to describe two different types of psychophysiological disturbances.

Again, the results shown in Figures 22 and 23 are very compelling, but not medically conclusive. Further clustering via HDBSCAN proved fruitless. Due to the high density of the overall embedding, clustering produced nonsensical agglomerations throughout the xy-plane.

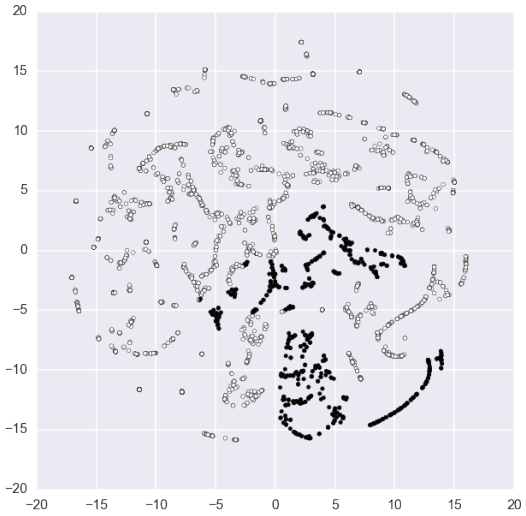

Figure 23: TSNE clustering of medical data 2-topic latent space. Color indicates most likely LDA topic label. Note the strong separation between the clusters at the bottom of the figure.

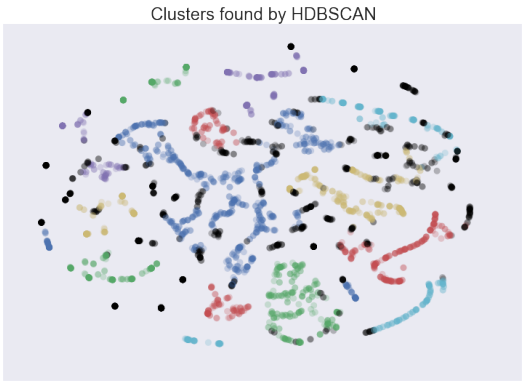

Figure 24: HDBSCAN clustering of medical data 2-topic latent space. Color indicates discovered agglomerative clusters. Due to the density of the data and lack of separation, clusters are mostly nonsense. Note, however, the discovery of a similar cluster at the bottom of the figure similar to Figure 23.

5 Future Improvements

Future improvements to this simple topic model center primarily around the feature space provided to the clustering algorithm. As demonstrated in Section 3.3 after the number the topics, the primary sensitivity of the model is the input features.

5.1 Lemmatization

Future improvements in lemmatization focus on the use and development of custom dictionaries for specific technical terms and vocabularies. Much like the Snowball stemmer's use of stem-to-word dictionaries and language-specific rules, custom logic can be developed for medical terminology and literature. Specifically the stemming of common drugs, symptoms and conditions as well as the parsing of various medical abbreviations could improve performance. To implement such a system, however, expert knowledge would be required. Such a system is not novel, a *BioLemmatizer* already exists for the field of molecular biology and drug-development [33]. Another improvement could detect and merge common misspellings these were especially prevalent in the 20-newsgroups data, since it was unedited.

5.2 N-grams

The addition of n-grams to the feature set would improve this model by adding context sensitivity. At the moment, the LDA topic model is a continuous bag of words; that is to say, each document is modeled as a vector of interchangeable features. However, written English is highly context specific, assuming that words are interchangeable is very simplistic! Consider the simple case of negation: "take the red pill not the blue pill" the order and positioning of

each word is very important to the meaning of the sentence. Reducing such a construction removes a great deal of information in the process.

5.3 Feature Selection

More granular feature representation beyond bag of words may also help in the clustering process. If the documents could be standardized to a consistent form useful numerical data could be extracted. For example, the number of previous seizures or the number of days a patient has taken a specific medication. The standardization of this data pushes the feature engineering to the health care professionals making observations. While free-form input can still be accommodated by LDA additional structured data could be easier to classify and interpret.

5.4 Supervision

Lastly the addition of supervised data points would be of great use to in training the clustering system. Knowing the patients' ultimate diagnosis or outcome could greatly help in refining the modeling process. This would provide a reasonable way to measure the system's accuracy beyond simple subjective measures or exploratory clustering. The addition of these labels would incur significant cost however, a trained professional would have to label each individual case.

6 Conclusion

In summary, this work has introduced a viable application and specialization of the LDA topic model to an unsupervised clustering task. Specifically, the viability of the processing pipeline present was determined through a series of

test data sets. Each test set was used to analyze the sensitivity of the topic model to its parameters: the number of topics, and the size of the vocabulary. A fast implementation of this algorithm was used to sample and tune the hyper-parameters of the models as shown in Appendix A. Both qualitative analysis of the semantics of the clusters, as well as quantitative measures were used to analyze the topics generated by the model.

In order to test the unsupervised dataset, the same proven pipeline was used. While the analysis and justifications for the parameter settings for this dataset were looser and more qualitative, the end goal of this work is not to produce a patient classifier or diagnostic, but to provide a fast, reliable, and alternative method to explore patient data. In this respect, it is the opinion of the author that the model may be used to provide useful insights into the patient dataset. An end user with a greater degree of medical knowledge, in addition to personal knowledge of the patients would have a better capacity to judge the output of this particular model. Nevertheless, some confidence is due here given the promising results on the other test data sets, and the clustering of two seizure phenomenon and well as a sleep disturbance. As stated in [16, 17] EEG are administered precisely to diagnose seizures in epileptic patients as well as tracking sleep disturbances through phase tracking. The clustering of seizure topics and separation of the sleep analysis by the model shows great promise.

A Appendix

A.1 Variational Inference

Using the following variational distribution as a substitute for the posterior distribution $p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta)$:

$$q(\theta, \mathbf{z} \mid \boldsymbol{\gamma}, \boldsymbol{\varphi}) = q(\theta, \mid \boldsymbol{\gamma}) \prod_{n=1}^{N} q(z_n \mid \varphi_n) \tag{24}$$

The hyper-parameters α, η are set via the process described [27]. First the log-likelihood is bound using Jensen's inequality.

$$\begin{aligned}
\log p(\mathbf{w} \mid \alpha, \beta) &= \log \int sum_z p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) \, \mathrm{d}\theta \\
&= \log \int \sum_z \frac{p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) q(\theta, \mathbf{z})}{q(\theta, \mathbf{z})} \, \mathrm{d}\theta \\
&\geq \int \sum_z q(\theta, \mathbf{z}) \log p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) \, \mathrm{d}\theta - \int \sum_z q(\theta, \mathbf{z}) \log q(\theta, \mathbf{z}) \, \mathrm{d}\theta \\
\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\varphi}, \alpha, \beta) &\equiv \mathrm{E}_q[\log p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta)] - \mathrm{E}_q[\log q(\theta, \mathbf{z})]
\end{aligned}$$

Where $\mathcal{L}$ provides a lower bound on the log-likelihood for the distribution $q(\theta, \mathbf{z} \mid \boldsymbol{\gamma}, \boldsymbol{\varphi})$.

A.2 Parameter Estimation

This appendix demonstrates a solution to the problem of obtaining empirical Bayes estimates of the model parameters α, β. This problem is solved by using the variational lower bound as a substitute for the intractable log likelihood. The variational parameters $\boldsymbol{\gamma}, \boldsymbol{\varphi}$ are fixed to the values found by variational inference. The approximate Bayes estimates are found by maximizing the lower bound with respect to the model parameters.

Assuming the log likelihood over the corpus $\mathbf{C}$ is the sum of the individual log likelihoods for individual documents, then the overall lower bound is the sum of the individual lower bounds. The overall approach is based on a variational expectation-maximization procedure. In the expectation step, the total variational bound is maxmized with respect to the parameters $\boldsymbol{\gamma}, \boldsymbol{\varphi}$. In the maximization step, the bound is maximized with respect to the model parameters α, β.

To maximize with respect to β,

$$\mathcal{L}_\beta = \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{k}\sum_{j=1}^{V}\boldsymbol{\varphi} + d_{ni}w_{dn}^{j}\log\beta_{ij} + \sum_{i=1}^{k}\lambda_i\left(\sum_{j=1}^{V}\beta_{ij} - 1\right)$$

Setting the derivative equal to zero,

$$\beta_{ij} \propto \sum_{d=1}^{M}\sum_{n=1}^{N_d}\varphi_{dni}w_{dn}^{j}$$

Similarly for α,

$$\mathcal{L}_\alpha = \sum_{d=1}^{M}\left\{\log\Gamma\left(\sum_{j=1}^{k}\alpha_j\right) - \sum_{i=1}^{k}\log\Gamma(\alpha_i) + \sum_{i=1}^{k}\left[(\alpha_i - 1)(\Psi(\gamma_{di})) - \Psi\left(\sum_{j=1}^{k}\gamma_{d_j}\right)\right]\right\}$$

Where $\Psi(\cdot)$ is the digamma function, the derivative of $\Gamma(\cdot)$. Again taking the derivative,

$$\frac{\partial\mathcal{L}}{\partial\alpha_i} = M\left[\Psi\left(\sum_{j=1}^{k}\alpha_j\right) - \Psi(\alpha_i)\right] + \sum_{d=1}^{M}\left[\Psi(\gamma_{di}) - \Psi\left(\sum_{j=1}^{k}\boldsymbol{\gamma}_{d_j}\right)\right]$$

Note that the same procedure can be used to find an estimate for $bm\eta$.

B Appendix

This appendix contains the code listings for analysis of both the supervised and unsupervised data. The same processing pipeline is used for all datasets with

minor adjustments for the number of topics and the size of the vocabulary. Adjustments to the stemming algorithm, stop words, and tokenizer also vary between datasets. For example the tokenizer for the 20-newsgroups dataset is much more aggressive: removing a greater number of special characters and other symbols.

B.1 LDA Algorithm Pipeline

Listing 5: Algorithm Pipeline

```
%matplotlib inline
import matplotlib.pyplot as plt, matplotlib.cm as cm
import codecs, re, pandas as pd, numpy as np, json
from time import time

# Data
from sklearn.datasets import fetch_20newsgroups

# NLP-Tools
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.lancaster import LancasterStemmer
from nltk.stem.regexp import RegexpStemmer
from sklearn.feature_extraction.text import TfidfVectorizer,
   CountVectorizer
from sklearn.decomposition import NMF, LatentDirichletAllocation

# Classification
from sklearn.linear_model import LogisticRegression,
   SGDClassifier
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
    classification_report

# Input data
stopword_file = '/afs/ee.cooper.edu/user/g/i/gitzel/masters/
    lda_repos/lda-textmine/python/stoplist.txt'
stemming = True
minlength = 3

categories = ['comp.graphics', 'rec.sport.baseball', 'rec.autos'
    , 'sci.med', 'sci.space', 'soc.religion.christian']

# Model parameters
n_features = 500
n_topics = len(categories)
n_top_words = 10
n_topics = 6
n_lda_iter = 10

class Document(object):
    def __init__(self, raw, target):
        self._raw_text = raw
        self._target = target
        self._clean_text = ''
        self._features = []
        self._topic = []
```

```
@property
def raw_text(self):
    return self._raw_text
@raw_text.setter
def raw_text(self, value):
    self._raw = value
@property
def clean_text(self):
    return self._clean_text
@clean_text.setter
def clean_text(self, value):
    self._clean_text = value
@property
def features(self):
    return self._features
@features.setter
def features(self, value):
    self._features = value
@property
def topic(self):
    return self._topic
@topic.setter
def topic(self, value):
    self._topic = value
@property
def target(self):
```

```
        return self._target
    @target.setter
    def target(self, value):
        self._target = value

class DocumentEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, Document):
            return obj.__dict__
        return super(DocumentEncoder, self).default(obj)

# Helper Functions
def print_top_words(model, feature_names, n_top_words):
    for topic_idx, topic in enumerate(model.components_):
        print("Topic␣#%d:" % topic_idx)
        print("␣".join([feature_names[i] for i in topic.argsort()
            [:-n_top_words - 1:-1]]))

def top_words(model, feature_names, n_top_words):
    return [tuple([feature_names[i] for i in topic.argsort()[:-
        n_top_words - 1:-1]]) for topic in model.components_]

def load_stopwords(stopword_filename):
    stopwords = set()
    with codecs.open(stopword_filename, 'r', 'utf-8') as sf:
        for line in sf:
            if len(line.split()) != 1:
```

```
                print('ignoring␣line␣with␣more␣than␣one␣stopword:\
                    n"{0}"'.format(line))
                continue
            stopwords.add(line.strip())
    return stopwords

def stem(document, stemmer, stopwords, exp, minlength):
    clean = ''
    for token in re.split(exp, document.raw_text.lower()):
        if not token or token in stopwords:
            continue
        if len(token) < minlength:
            continue
        token = stemmer.stem(token)
        clean += '␣' + token
    document.clean_text = clean

# Load the 20 newsgroups dataset and vectorize it. We use a few
    heuristics
# to filter out useless terms early on: the posts are stripped
    of headers,
# footers and quoted replies, and common English words, words
    occurring in
# only one document or in at least 95% of the documents are
    removed.

print("Loading␣dataset...")
```

```
t0 = time()
dataset = fetch_20newsgroups(categories=categories, shuffle=True
    , random_state=1,
                             remove=('headers', 'footers', 'quotes'
                                 ))
documents = [Document(datum, dataset.target_names[target]) for
    datum, target in zip(dataset.data, dataset.target)]
print("done␣in␣%0.3fs." % (time() - t0))

# Stopwords and Stemming
print("Cleaning␣dataset...")
t0 = time()
stopwords = load_stopwords(stopword_file)
stemmer = SnowballStemmer('english', ignore_stopwords=True)
stemmer.stopwords = stopwords
exp = "'(?!(d|m|t|ll|ve)\W)|[.,\-_!?:;()0-9@=+^*`~#$%&|␣\t\n
    \>\<\"\\\/\[\]{}]+"

for document in documents:
    clean_sample = ''
    for token in re.split(exp, document.raw_text.lower()):
        if not token or token in stopwords:
            continue
        if stemming:
            token = stemmer.stem(token)
        if len(token) < minlength:
            continue
```

```
        clean_sample = clean_sample + ' ' + token
    document.clean_text = clean_sample
print("done in %0.3fs." % (time() - t0))

# tf (raw term count)
print("Extracting tf features...")
tf_vectorizer = CountVectorizer(
    max_df=0.95, stop_words='english', max_features=n_features)
t0 = time()
tf = tf_vectorizer.fit_transform([doc.clean_text for doc in
    documents])
print("done in %0.3fs." % (time() - t0))

# Print TF Features
# Print the features of the first document to check parsing
    output.
feature_map = {feature: word for word, feature in tf_vectorizer.
    vocabulary_.iteritems()}
doc_index = 0
print 'Document: {}'.format(documents[doc_index].raw_text)
tf_array = tf.toarray()
for i, feature in enumerate(tf_array[doc_index]):
    if feature > 0:
        print 'Feature #{}\t{:.4f}\t{}'.format(i, feature,
            feature_map[i])

for i, doc in enumerate(documents):
```

```
    doc.features = list(tf_array[i])
# print '{} total features'.format(len(tf_vectorizer.
    vocabulary_))

print("Fitting␣LDA␣model␣with␣%d␣tf␣features␣and␣%d␣topics..." %
     (n_features, n_topics))
lda = LatentDirichletAllocation(n_topics=n_topics, max_iter=
    n_lda_iter,
                                learning_method='online',
                                   learning_offset=20.,
                                random_state=0)
t0 = time()
lda.fit(tf)

topic_word = lda.components_
doc_topic = lda.transform(tf)
print("done␣in␣%0.3fs." % (time() - t0))
print("log-likelihood:␣%0.2f" % lda.score(tf))

print("\nTopics␣in␣LDA␣model:")
tf_feature_names = tf_vectorizer.get_feature_names()
print_top_words(lda, tf_feature_names, n_top_words)
lda_topics = top_words(lda, tf_feature_names, 3)

for i, document in enumerate(documents):
    topic_vector = doc_topic[i]
```

```
    document.topic = list(topic_vector / topic_vector.sum()) #
        normalize distribution

# Join target and predicted topic matrices
target_names = dataset.target_names
target = pd.DataFrame(dataset.target, columns=['target topic'])
    # target matrix

df_y_hat = pd.DataFrame(doc_topic, columns=lda_topics) #
    document-topic matrix
joined = target.join(df_y_hat) # join target values
joined.groupby('target topic').mean() # group by target value,
    averaging the topic weightings

# Predict generated topic
lr = LogisticRegression()
lr.fit(joined[lda_topics], joined['target topic'])
prediction = lr.predict(joined[lda_topics])

score = accuracy_score(joined['target topic'], prediction) *
    100.
matrix = confusion_matrix(joined['target topic'], prediction)

fig = plt.figure(figsize=(15,15))
ax = fig.add_subplot(111)
cax = ax.matshow(matrix)
plt.title('{:.1f}% Accuracy'.format(score))
```

```
fig.colorbar(cax)
ax.set_xticklabels([''] + target_names)
ax.set_yticklabels([''] + target_names)
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.show()

# print raw matrix
pd.DataFrame(matrix, columns=target_names, index=target_names)

# TSNE Feature Reduction (for scatter-plot)
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2, random_state=0)
X_tsne = tsne.fit_transform(doc_topic)

plt.figure(1, figsize=(10,10), dpi=100)
plt.scatter(X_tsne[:,0], X_tsne[:,1], c=dataset.target)
# fig.c
plt.legend()
plt.show()

# HDBSCAN clustering
import seaborn as sns
import sklearn.cluster as cluster

sns.set_context('poster')
```

```
sns.set_color_codes()
plot_kwds = {'alpha' : 0.25, 's' : 80, 'linewidths':0}

def plot_clusters(data, algorithm, args, kwds):
    start_time = time()
    labels = algorithm(*args, **kwds).fit_predict(data)
    end_time = time()
    palette = sns.color_palette('deep', np.unique(labels).max()
       + 1)
    colors = [palette[x] if x >= 0 else (0.0, 0.0, 0.0) for x in
        labels]
    plt.scatter(data.T[0], data.T[1], c=colors, **plot_kwds)
    frame = plt.gca()
    frame.axes.get_xaxis().set_visible(False)
    frame.axes.get_yaxis().set_visible(False)
    plt.title('Clusters␣found␣by␣{}'.format(str(algorithm.
       __name__)), fontsize=24)
# plt.text(-0.5, 0.7, 'Clustering took {:.2f} s'.format(
    end_time - start_time), fontsize=14)
    plt.legend()

import hdbscan
plot_clusters(X_tsne, hdbscan.HDBSCAN, (), {'min_cluster_size'
    :25})
```

References

[1] H. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM Journal*, pp. 309–317, October 1957.

[2] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman, "Indexing by latent semantic analysis," *J. of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[3] T. Hofmann, "Probabilistic latent semantic indexing," *Research and Development in Information Retrieval*, pp. 50–57, 1999.

[4] M. Dredze, H. Wallach, D. Puller, and F. Pereira, "Generating summary keywords for emails using topics," in *13th International Conference on Intelligent User Interfaces*, 2008.

[5] T. Griffiths and M. Steyvers, "Finding scientific topics," in *Proceedings of the National Academy of Science*, 2004.

[6] D. J. Hu, "Lda for text, images, and music," *Research Exam*, 2009.

[7] D. M. Blei, A. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, pp. 993–1022, 2003.

[8] D. M. Blei, T. L. Griffiths, and M. I. Jordan, "The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies," *J. ACM*, vol. 57, no. 2, 2010.

[9] I. Biro, J. Szabo, and A. A. Benczur, "Latent dirichlet allocation in web spam filtering," in *Proceedings of the 4th international workshop on adversarial information retrieval on the web*, 2008, pp. 29–32.

[10] D. M. Blei and J. D. Lafferty, "Topic models," in *Text Mining: Classification, Clustering, and Applications*, A. Srivastava and M. Sahami, Eds. Chapman and Hall, 2009.

[11] L. Brown, "Fundamentals of statistical exponential families," *Institute of Mathematical Statistics*, 1986.

[12] G. Salton and M. McGill, Eds., *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[13] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. ACM Press, 1999.

[14] P. Lee, *Bayesian Statistic: An Introduction*. Wiley, 2004.

[15] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[16] E. Niedermeyer and F. da Silva, *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Lippincot, Williams and Wilkins, 2004.

[17] B. Ahou-Khalil and K. Musilus, *Atlas of EEG and Seizure Semilogy*. Elsevier Health Sciences, 2006.

[18] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 331–339.

[19] B. Hamner. (2015) Nips 2015 papers. [Online]. Available: https://www.kaggle.com/benhamner/nips-2015-papers

[20] A. Harati, S. Lopez, I. Obeid, M. Jacobson, S. Tobochnik, and J. Picone, "The tuh eeg corpus: A big data resource for automated eeg interpretation," in *Proceedings of the IEEE Signal Processing in Medicine and Biology Symposium*, 2014, pp. 1–5.

[21] L. Dierickx. (2015) python stopwords 0.13 package. [Online]. Available: https://pypi.python.org/pypi/stopwords

[22] J. Dawson, "Suffix removal for word conflation," *Bulletin of the Association for Literary and Linguistic Computing*, vol. 2, pp. 33–46, 1974.

[23] Y. Shibukawa. (2015) python snowballstemmer 1.2.1 package. [Online]. Available: https://pypi.python.org/pypi/snowballstemmer

[24] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, pp. 130–137, 1980.

[25] P. Willet, "The porter stemming algorithm: Then and now," *Program*, vol. 40, 2006.

[26] D. M. Blei, "Probabilistic topic models," *Commum. ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[27] K. Canini, L. Shi, and T. Griffiths, "Online inference of topics with latent dirichlet allocation," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, vol. 5, 2009.

[28] Y. Teh, D. Newman, and M. Welling, "A collapsed variational bayesian inference algorithm for latent dirichlet allocation," *Neural Information Processing Systems*, 2006.

[29] F. Jelink, *Statistical Methods of Speech Recognition*. Cambridge, MA: MIT Press, 1997.

[30] K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," in *IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999, pp. 61–67.

[31] L. McInnes. (2015) python hdbscan 0.4 package. [Online]. Available: https://pypi.python.org/pypi/hdbscan/0.4

[32] R. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2013, pp. 160–172.

[33] H. Liu, T. Christiansen, W. B. Jr., and K. Verspoor, "Biolemmatizer: a lemmatization tool for morphological processing of biomedical text," *J. Biomed Semantics*, April 2012.

[34] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *J. American Statistical Association*, vol. 101, pp. 1566–1581, 2006.

[35] M. Steyvers and T. Griffiths, "Probabilistic topic models," in *Latent Semantic Analysis: A Road to Meaning*, T. Landauer, D. McNamara, S. Dennis, and W. Kintsch, Eds., 2006.