Visualizing Topic Modeling with Latent Dirichlet Allocation

Kai Bernardini

Abstract

Latent Dirichlet Allocation (LDA) is an unsupervised, generative approach to topic modeling that learns latent semantic topics from a collection of documents. LDA posits that words used to convey semantic information about a particular topic are similar across different documents. Such topics can be represented as a latent random variable, and are discovered by identifying groups of words in the corpus that frequently occur together within documents. In this way, LDA models documents as a random mixture over latent topics, with each topic being characterized by its own particular distribution over words. In this report, an overview of vector space models for text are discussed and the applications of LDA to news topic modeling are explored. Other topics discussed are PCA and K-means for visualization.

CONTENTS

I. Introduction

With the widespread availability of large scale computing and seemingly endless amounts of data, statistical machine learning problems have become not only pervasive, but have also exploded in both complexity and scope. This revolution has breathed new life into the longstanding goal of creating artificial general intelligence.[1] This goal can be (informally) summarized as a program that is indistinguishable from [or superior to] a human operator with respect to some task [3]. In the early days of artificial intelligence, the field rapidly solved problems that are mentally difficult for humans to solve quickly, but are relatively straight-forward to describe with a series of formal, mathematical rules. Ironically, such mathematically formal tasks that are among the most difficult mental undertakings for humans, are easy for a computers. On the other hand, tasks that are both difficult to describe mathematically and relatively simple for humans to solve tend to be incredibly difficult for computers. That is, the biggest challenge to artificial general intelligence is solving tasks that are trivial for people to perform but hard *to describe formally*. In particular, they consist of a class of problems that we as humans solve intuitively– almost automatically without any mathematical formalism.[2] One such class of problems is *topic modeling*. As a motivating example, suppose you are presented with a news article and are tasked with describing what topics are discussed. For a human being, its not a challenge to figure out which topic a news article belongs to, but how can we teach a computer to understand the same topics? After all, this is a subjective question, and is difficult to make formal. This is where topic modeling comes into the picture. Topic models are unsupervised machine learning models that seek to discover abstract *topics* from a document.

This class of problems are important, as with the massive amount of digital multimedia data available, it is imperative to develop methods for retrieval, organization, and management of large corpora. In particular, for each document in a corpus, it would be useful if appended to each document was a short summary that could quickly tell a human or a machine what the document is generally about. Going further, it would also be useful for these short descriptions to have representations that are consistent across the entire collection. Such a standard would allow us to measure how closely related one document is to another without ever reading either document.[8] We may even use it to visualize where a particular document stands relative to all other documents in the corpus in terms of content similarity. One can see that having these short descriptions would be invaluable for searching, indexing and ranking

similar documents in a large collection of text data.[7]

II. Vector Space models for Text

A. Bag of Words

The latter can be achieved using *vector space models* for text. The simplest one is the unigram model sometimes called the *bag of words model*. In this setup, every single document can be represented as $w \in \mathbb{R}^n$ with each dimension corresponding to a particular word in a fixed vocabulary of n words. Each value w_i corresponds to the number of times word i appears in document w. This vector space representation allows us to compare documents, as if each document is represented as a vector in a vector space, we can simply take two vectors $w, w' \in \mathbb{R}^n$ and compute a similarity score

$$\delta(w, w') = \frac{\langle w, w' \rangle}{||w||||w'||}$$

Notice that this value will be large if w is *close* to w' in some sense, with the innerprodcut $\langle \cdot, \cdot \rangle$ defining precisely what notion of "close" is. A typical choice of inner product is the dot product. This is exactly cosine similarity where

$$\text{CosSim}(w, w') = \frac{\sum_{i=1}^{n} w_i w_i'}{\sqrt{\sum_{i=1}^{n} w_i^2}\sqrt{\sum_{i=1}^{n} w_i'^2}}.$$

The notion of "nearness" induced by this inner product is the similarity of two documents is large when the angle between the two vectors is small. Notice that as a result of this representation, the model assumes that the order of words in a document can be neglected. In the language of probability theory, this is an assumption of *exchangeability* for the words in a document. A finite collection of random variables $\{z_1, \ldots, z_N\}$ is said to be *exchangeable* if the joint distribution is invariant to permutation. That is, for any permutation π on integers from 1 to N, then

$$p(z_1, \ldots, z_N) = p(z_{\pi(1)}, \ldots, z_{\pi(N)}).$$

An infinite sequence of random variables is *infinitely exchangeable* if every finite subsequence is exchangeable.

De Finetti's representation theorem states that exchangeable observations are conditionally independent relative to some latent variable. I.e., the joint distribution of an infinitely exchangeable sequence of random variables is as if a random parameter were drawn from some distribution and then the random variables in question were *independent* and *identically distributed*, conditioned on that parameter. The key property is an epistemic probability distribution could then be assigned to this variable.

B. TFIDF

A basic extension to the bag of words model is used by modern Internet search engines and is caleld is *tf-idf*, short for term frequency-inverse

document frequency. Tf-idf is a numerical statistic that tries to weight each word based on how important it is, where the underlying assumption is rare words could hold important information. As with the bag of words model, a vocabulary is fixed, and vectors are formed by weighting vectors according to the number of occurrences of that word in the document. Then, the term frequency count is compared to an inverse document frequency count, which measures the number of occurrences of a word in the entire corpus and is normalized. The end result is a datamatrix X where the columns of X contain the *tf-idf* values for each of the documents in the corpus.[7] Thus the *tf-idf* scheme reduces documents of arbitrary length to fixed-length lists of numbers. Further, these matrices tend to be very sparse, allowing for massive feature spaces to be stored in memory. A typical example is a vocabulary of say 150,000 can be selected and generated from 1 million documents. If the matrix was represented in memory as a dense matrix, assuming 8 bytes (64 bits) per entry, you would need $8 \times 150000 \times 1000000 = 1.2 \times 10^{12}$ bytes which is 1200 gigabytes. For reference, modern laptops typically have between 8 and 16 gigabytes of random accesses memory. A key problem for this methodology is it provides little to no reduction in description length. I.e., it transforms a document of arbitrary length into an n dimensional vector. If the number of words in the document is smaller than n, than it provides no compression, and even increases the size. Another problem is such representations reveals little in the way of inter–document statistical structure. To address these shortcomings, Blei et. al. introduced Latent Dirichlet Allocation (probabilistic latent semantic indexing with a Dirichlet prior)[7].

III. LDA

Latent Dirichlet Allocation (LDA) is a model that aims to find to find short descriptions of the members of a collection that enable efficient processing of large corpa while preserving the essential statistical relationships that are useful for basic tasks such as classification, novelty detection, summarization, and similarity. To make this precise, let a *word* be the basic unit of *discrete data*. This is a singular word from the available (ordered) vocabulary indexed by $\{1, \ldots, V\}$. Each element of the vocabulary is represented as a one-hot encoded unit basis vector. In agreement with the original paper, this report uses superscript to denote the entry in the vector. I.e., the vth word in the vocabulary is represented by a V- dimensional vector w such that $w^v = 1$ and $w^u = 0$ for $u \neq v$. A *document* is a sequence of words denoted by $\mathbf{w} = (w_1, w_2, \ldots, w_N)$, where w_n is the nth word in the sequence. Note that N varies between documents, and does not need to be fixed (in fact, LDA

assumes N is random!). A *corpus* is a collection of M documents denoted by $\mathcal{D} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M\}$. The core concept of LDA is documents are described by a random mixture over latent topics, where each discrete topic is parameterized by a distribution over words. The original LDA paper assumes that for each document $\mathbf{w} \in \mathcal{D}$, the number of tokens in the document is random, and distributed $N \sim \text{Poisson}(\xi)$. It also assumes that each topic $z_n \sim \text{Multinomial}(\theta)$ where $\theta \sim \text{Dir}(\alpha)$. LDA further assumes each document is constructed via the following generative process:

1) For each of the N words w_n:
    a) Choose a topic z_n.
    b) Choose a word w_n from $p(w_n \,|\, z_n, \beta)$ (multinomial conditioned on the topic z_n).

where k, the dimensionality of the Dirichlet distribution, is known and fixed $\implies$ the dimensionality of the topic variable z is also known. Each word probability is parameterized by $\beta_{k \times V}$ where

$$\beta_{ij} = p(w^j = 1 \,|\, z^i = 1)$$

which is fixed, but unknown. Since the number of words in a document is assumed to be independent of θ and $\mathbf{z}$, N is an ancillary variable, meaning it irrelevant to the computation of the posterior distributions of the other variables conditioned on a fixed document. The choice of distribution for N is also not important, and can be swapped out with another distribution to fit the needs of the specific application.

The range of a k-dimensional Dirichlet random variable θ is the $k-1$-simplex where a k dimensional vector θ is contained in the $k-1$ dimensional simplex if

$$\theta_i \geq 0 \text{ and } \sum_{i=1}^{k} \theta_i = 1$$

. The probability density function p is defined as

$$p(\theta \,|\, \alpha) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1}, \qquad (1)$$

where the parameter α is a k- dimensional vector with components $\alpha_i > 0$, and where

$$\Gamma(z) = \int_0^{\infty} x^{z-1} e^{-x} dx$$

is the Gamma function.

For fixed parameters α and β, the joint distribution of a topic mixture θ, a set of N topics $\mathbf{z}$, and a set of N words $\mathbf{w}$ is given by:

$$p(\theta, \mathbf{z}, \mathbf{w} \,|\, \alpha, \beta) = p(\theta \,|\, \alpha) \prod_{n=1}^{N} p(z_n \,|\, \theta) p(w_n \,|\, z_n, \beta), \qquad (2)$$

where $p(z_n \,|\, \theta)$ is θ_i for unique i with $z_n^i = 1$. The LDA PDF is obtained by marginalizing out the topic variables and endowing θ with a Dirichlet distribution. I.e., integrating over θ and summing over z, the marginal distribution $p(\mathbf{w} \,|\, \alpha, \beta)$ of a document is

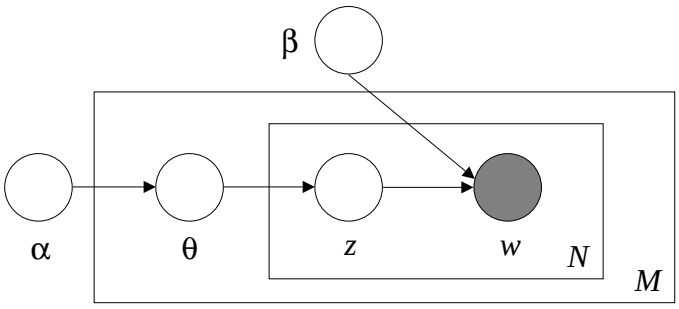


Fig. 1. Graphical model representation of LDA. The boxes are "plates" representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document.[7]

$$\int p(\theta \mid \alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n \mid \theta) p(w_n \mid z_n, \beta) \right) d\theta.$$

To obtain the probability of a particular corpus, take the product of the marginal probabilities of single document $p(\mathcal{D} \mid \alpha, \beta)$ given by

$$\prod_{d=1}^{M} \int p(\theta_d \mid \alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn} \mid \theta_d) p(w_{dn} \mid z_{dn}, \beta) \right) d\theta_d.$$

At its core, LDA is a conditionally independent hierarchical Bayseian model represented as graphical model for topic discovery (see Figure 5). LDA involves three levels of representation, and notably the topic node is sampled *repeatedly* within the document. Under this model, documents are a mixture of topics with:

1) The parameters α and β are corpus-level parameters, assumed to be sampled once in the process of generating a corpus. This can be thought of as the *training phase*.
2) The variables θ_d are document-level variables, sampled once per document.
3) The variables z_{dn} and w_{dn} are word level variables and are sampled once for each word in each document[7].

In LDA, words are assumed to be generated by latent topics via fixed conditional distributions. The latent topics are also assumed to be infinitely exchangeable *within a fixed document*. By de Finetti's theorem, the probability of a sequence of words and latent topics can be written

$$p(\mathbf{w}, \mathbf{z}) = \int p(\theta) \left(\prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n) \right) d\theta,$$

where θ is the random parameter of a multinomial distribution over topics. In particular, for the word distribution $p(w \mid \theta, \beta)$:

$$p(w \mid \theta, \beta) = \sum_{z} p(w \mid z, \beta) p(z \mid \theta).[7]$$

This gives an alternative representation of the underlying random process– instead of a three-level hierarchical Bayesian model, it can be reduced to a two-level model. In particular, each document $\mathbf{w}$ is generated by:

1) $\theta \sim \text{Dir}(\alpha)$.
2) For each word w_n for $n = 1, \ldots, N$:
    a) Choose a word w_n from $p(w_n \mid \theta, \beta)$.

This process defines the marginal distribution of a document as a continuous mixture distribution:

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

where $p(w_n \mid \theta, \beta)$ are the mixture components and $p(\theta \mid \alpha)$ are the mixture weights.

A. Inference

In order to use LDA as a Latent Variable model, we need to compute the posterior distribution of the hidden variables given a document:

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

Although the posterior distribution is intractable for exact inference, a wide variety of approximate inference algorithms can be considered for LDA. The one presented in the original paper uses a convexity-based variational algorithm for inference.

B. Convexity-Based Variational inference

Variational Bayesian methods describe a collection of techniques for numerically estimating common intractable integrals that usually arise in Bayesian inference. The main principle behind convexity-based variational inference is to utilize Jensen's inequality to bound the log likelihood via a family of lower bounds, indexed by a set of *variational parameters*. The variational parameters are chosen to have the tightest bound after a fixed number of iterations using some optimization procedure.[7]

The original paper relaxes the original graphical model by introducing a surrogate to the original distribution. They add a simple modifications to the original graphical model in which a subset of the nodes and edgers are deleted. The original calculation is intractable due to the "problematic coupling" between θ and β. This coupling arises due to the edges between θ, $\mathbf{z}$, and $\mathbf{w}$. By dropping such edges and nodes, and endowing the resulting simplified graphical model with free variational parameters, this parameterizes a new family of distributions on the latent variables, characterized by the following variational distribution:.[7]

$$q(\theta, \mathbf{z} \mid \gamma, \phi) = q(\theta \mid \gamma) \prod_{n=1}^{N} q(z_n \mid \phi_n), \qquad (3)$$

where the Dirichlet parameter γ and the multinomial parameters $(\phi_1, \ldots, \phi_N)$ are the free variational parameters that parameterize the surrogate distribution. The next step is to set up an optimization problem that estimates the variational parameters γ and ϕ. As shown in [7], finding a tight lower bound on the log likelihood is equivalent to the following optimization problem:

$$(\gamma^*, \phi^*) = \arg\min_{(\gamma,\phi)} \mathrm{D}(q(\theta, \mathbf{z} \mid \gamma, \phi) \parallel p(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta)). \qquad (4)$$

Hence, optimizing values of the variational parameters are estimated by minimizing the relative entropy between the variational distribution and the true posterior $p(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta)$. This minimization can be achieved via an iterative fixed-

$\phi_{ni}^0 \leftarrow 1/k$ for all i and n
$\gamma_i \leftarrow \alpha_i + N/k$ for all i
repeat
  for Zip$(n = 1, k = 1)$ **to** N, K
    for $i = 1$ **to** k
      $\phi_{ni}^{t+1} \leftarrow \beta_{iw_n} \exp(\Psi(\gamma_i^t))$
    normalize ϕ_n^{t+1} to sum to 1.
  $\gamma^{t+1} \leftarrow \alpha + \sum_{n=1}^{N} \phi_n^{t+1}$
until convergence

Fig. 2. A variational inference algorithm for LDA.

point method. In particular, the original paper [7] shows that by computing the derivatives of the relative entropy and setting them equal to zero, equations similar to those derived in Expectation Maximization for maximum a posteriori estimates are derived:

$$\phi_{ni} \propto \beta_{iw_n} \exp\{\mathrm{E}_q[\log(\theta_i) \mid \gamma]\} \quad (5)$$

$$\gamma_i = \alpha_i + \textstyle\sum_{n=1}^{N} \phi_{ni}. \quad (6)$$

They show further that the expectation in the multinomial update can be computed as follows:

$$\mathrm{E}_q[\log(\theta_i) \mid \gamma] = \Psi(\gamma_i) - \Psi\left(\textstyle\sum_{j=1}^{k} \gamma_j\right), \quad (7)$$

where Ψ is the first derivative of the $\log \Gamma$. The optimizing parameters $(\gamma^*(\mathbf{w}), \phi^*(\mathbf{w}))$ are document-specific. Further, the Dirichlet parameters $\gamma^*(\mathbf{w})$ provide a representation of a document in the topic simplex, meaning every document can be described as a distribution over topics. Each iteration of variational inference for LDA requires $\mathrm{O}((N+1)k)$ operation, with the whole process being $\mathrm{O}(N^2 k)$.

IV. Training via Parameter Estimation

This section briefly summarizes fitting a model via parameter estimation. In particular, starting with a training corpus of documents

$$\mathcal{D} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M\}$$

fitting the model requires finding parameters α and β that (approximately) maximize the (marginal) log likelihood of the data:

$$\ell(\alpha, \beta) = \sum_{d=1}^{M} \log p(\mathbf{w}_d \mid \alpha, \beta).$$

With variational inference, it is possible to find an approximate empirical Bayes estimates for the LDA model via an alternating *variational EM* procedure that maximizes a lower bound with respect to the variational parameters γ and ϕ, and then, for fixed values of the variational parameters, maximizes the lower bound with respect to the model parameters α and β. The original paper derives the following form for the EM algorithm:

1) (E-step) For each document, find the optimizing values of the variational parameters $\{\gamma_d^*, \phi_d^* : d \in \mathcal{D}\}$. This is done as described in the previous section.
2) (M-step) Maximize the resulting lower bound on the log likelihood with respect to the model parameters α and β. This

corresponds to finding maximum likelihood estimates with expected sufficient statistics for each document under the approximate posterior which is computed in the E-step.[7] where the process is repeated until convergence to the desired lowerbound. Thee M-step update for the conditional multinomial parameter β can be written out analytically:

$$\beta_{ij} \propto \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi^*_{dni} w^j_{dn}. \tag{8}$$

Most implementations for the M-step update for Dirichlet parameter α use a quasi-Newton method as the Hessian can be inverted in linear time.[7]

V. Topic Modeling for Articles

The example considered here is the NewsGroup20 dataset. The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. Each newsgroup corresponds to a different topic, some of which are very closely related to each other (e.g. comp.sys.ibm.pc.hardware and comp.sys.mac.hardware), while others are highly unrelated (e.g misc.forsale and soc.religion.christian). The goal is to cluster news documents together that have similar (or ideally the same) newsgroup label. The text data is cleaned by converting all text to lowercase, removing punctuation and tokenizing

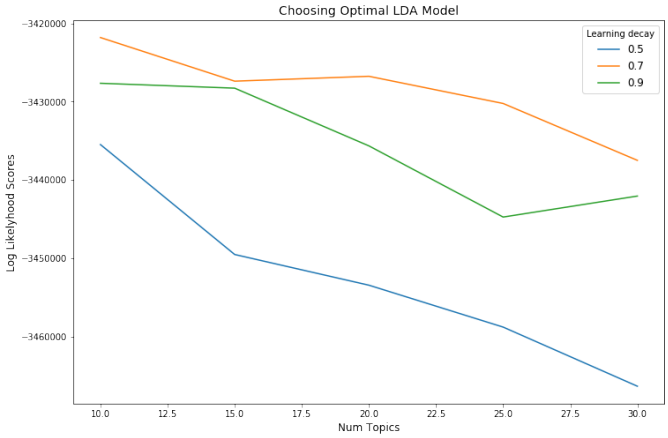


Fig. 3. A model with higher log-likelihood and lower perplexity ($\exp(-1. * \text{log-likelihood per word})$) is considered to be good

and stemming (lemmatization) words. From there, a datmatrix X constructed by using a TFIDf vectorizer on the cleaned text. The sci-kit-learn implementation of LDA topic model algorithm requires a document word matrix as the main input, and is used to construct the model. As with any latent generative model, we can diagnose model performance by looking at perplexity and log-likelihood as a function of the number of topics, and learning decay rate. 15 LDA models are trained, and scored using a grid search over the number of topics: $\{10, 15, 20, 25, 30\}$ and the learning decay: $\{.5, .7, .9\}$

The model with the highest log likelihood has a learning decay of 0.7 and 10 latent topics.

A. Visualization and Results

In order to visualize the topics, it is important to first introduce basic tools for dimensionality reduction, and clustering.

B. PCA

Consider an $n \times d$ real matrix A. The rank of A is the dimension of its column space. The dimension of a space is the smallest number of linearly independent vectors needed to span the space. Hence, the dimension of the column space of A is the smallest number of vectors that suffice to span the columns of A. Then the rank of A is the size of the smallest set $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p\}$ such that every $\mathbf{a}_i$ can be expressed as:

$$\mathbf{a}_i = c_{i1}\mathbf{u}_1 + c_{i2}\mathbf{u}_2 + \cdots + c_{ip}\mathbf{u}_p \quad i = 1, \ldots, n.$$

The largest value that a matrix rank can take is $\min(n, d)$. However it can happen that the rank of a matrix is less than $\min(m, n)$. Representing a matrix $A \in \mathbb{R}^{n \times d}$ requires mn values. However, if A has rank k, it can be factored as $A = UV$ where $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{k \times d}$. This only requires $k(n+d)$ values, which could be much smaller than nd.

In many real world applications, it is possible to approximate a data matrix A with a low-rank matrix $A^{(k)}$. To make precise what is meant by "one matrix approximating the other", the Frobenius norm will be used. This is just the usual ℓ_2 norm, treating the matrix as a vector in $\mathbb{R}^{n \times d}$.

The definition of the Frobenius norm of A, denoted $\|A\|_F$, is:

$$\|A\|_F = \sqrt{\sum a_{ij}^2}.$$

To quantify when one matrix is "close" to another, the usual notion of distance in Euclidean space is used as follows:

$$\text{dist}(A, B) = \|A - B\|_F.$$

(where the Euclidean space is the nd-dimensional space of $n \times d$ matrices.)

When $k < \text{rank } A$, the rank-k approximation to A is the closest rank-k matrix to A that satisfies

$$A^{(k)} = \arg \min_{\{B \mid \text{rank } B = k\}} \|A - B\|_F.$$

This can also be considered the best rank-k approximation to A in a least-squares sense.

Theorem 5.1 (The Singular Value Decomposition): Every $n \times m$ matrix A has a decomposition

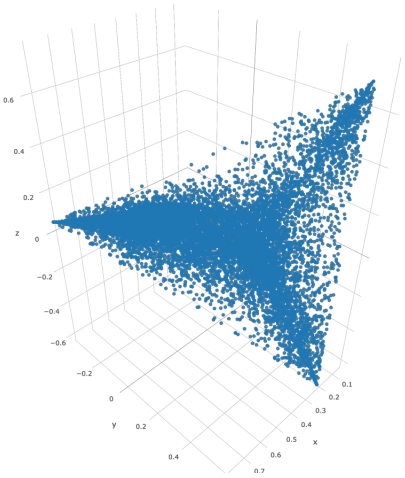
$$A = U\Sigma V^T$$

where

- V is an orthogonal $m \times m$ matrix;
- Σ is a diagonal $n \times m$ matrix; and
- U is an orthogonal $n \times n$ matrix.

For an $n \times m$ matrix A, the singular values of A are the square roots of the eigenvalues of the symmetric matrix $A^T A$. The singular values are written $\sigma_1, \cdots, \sigma_m$ where $\sigma_i \geq \sigma_{i+1}$. An important result from Linear Algebra is the spectral theorem

which guarantees the existence of eigenvalues for all symmetric matrices. In particular, it guarantees that a matrix is orthogonally diagonalizable if and only if the matrix is symmetric. In the context of visualizing LDA, the resulting transformed datamatrix $\mathbf{X}$ is a $n \times k$ matrix where k is the number of topics. We can further reduce the dimensionality by using PCA– a continuous latent variable model that is obtained by taking the top ℓ principal components and using them to project the datmatrix to a lower dimensional space. The basis vectors correspond to linear combinations of word distributions over topics Using PCA with 3 components on the transformed topic matrix, we get the following:



This is of course not very illuminating because we already knew the range of possible values was inside a simplex. It is reassuring though, that the resulting transformed space is also contained in a simplex. One way to visualize similar objects within the reduced space is to run a vector quantization algorithm such as k-means on the resulting transformed datamatrx.

C. K-means

K-means clustering is a heuristic for vector quantization that attempts to minimize the distortion over all points and their respective clusters.The algorithm uses an iterative approach to compute a locally optimum solution. It starts by choosing an initial set of k centroids. Call this initial set T_0. As a remark, since T partitions $\mathbb{R}^d$, this means that once such a partition is found, novel datapoints can be uniquely assigned clusters. This induces a Voronoi partition of $\mathbb{R}^d$ into k convex cells each corresponding to a unique centroid $c_i \in T$. For each cluster, a new centroid is calculated by taking the mean value along each dimension of data points belong to the cluster. This will results in a new set of centroids. Each data point is assigned a new cluster based on the new centroids. This processes continues until the distance between successive clusters goes to zero, or after a fixed number of iterations. More specifically, K-Means guarantees a local minimum to the cost function defined on each cluster $A \subset \mathbb{R}^d$. Let the J cost for a particular cluster A be

$$J_T(A_j) = \sum_{n=1}^{N} \chi_j(x_n)||x_n - c_j||^2$$

for cluster A_j with cluster center c_j Then the J value for a k clusters is

$$J_T = \sum_{j=1}^{k} J_T(A_j) = \sum_{n=1}^{N}\sum_{j=1}^{k} \chi_j(x_n)||x_n - c_j||^2$$

where T is the set of cluster centers that induce a partition $\mathbb{R}^d$, x_n is one of the N observed data points, and $c_j \in T$ and

$$\chi_j(x_n) = \begin{cases} 1 & x_n \in A_j \\ 0 & x_n \notin A_j \end{cases}$$

Lemma 5.2: For a $X \in \mathbb{R}^d$ a random variable and $x \in \mathbb{R}^d$,

$$\mathbb{E}||X - x||^2 = \mathbb{E}||X - \mathbb{E}X||^2 + ||x - \mathbb{E}X||^2$$

Proof: Expand out

$$\mathbb{E}||X - \mathbb{E}X||^2 + ||X - \mathbb{E}X||^2$$

to get

$$\mathbb{E}||X - \mathbb{E}X||^2 + ||x - \mathbb{E}X||^2$$

$$= \mathbb{E}[||X||^2 + ||\mathbb{E}X||^2 - 2X\mathbb{E}X]$$
$$+ [||x||^2 + ||\mathbb{E}X||^2 - 2x\mathbb{E}X]$$
$$= \mathbb{E}||X||^2 + ||x||^2 - 2x\mathbb{E}X$$
$$= E||X - x||^2$$

■

Proposition 5.3: For any cluster $A_j \subset \mathbb{R}^d$ and any $x \in \mathbb{R}^d$,

$$J(A_j, x) = J(A_j, \text{Ave}(A) + |A| \cdot ||x - \text{Ave}(A_j)||^2$$

Proof: Let X be a uniform random sample from A. Then by the Lemma above, we can represent the cost as a sum of

$$\mathbb{E}||X - x||^2 = \sum_{a \in A_j} \frac{1}{|A_j|}||a - x||^2 = \frac{1}{|A_j|}J(A_j; x)$$

and

$$\mathbb{E}||X - \mathbb{E}X||^2 = \frac{1}{|A|}J(A_j; \text{Ave}(A))$$

Since

$$\mathbb{E}||X - x||^2 = \mathbb{E}||X - \mathbb{E}X||^2 + ||x - \mathbb{E}X||^2$$

,

$$\implies J(A_j, x) = J(A_j, \text{Ave}(A_j) + |A_j| \cdot ||x - \text{Ave}(A_j)||^2$$

■

The actual algorithm is performed as follows:

1) Initialize centers $c_1, \ldots c_k \in \mathbb{R}^d$ and induced clusters $A_1, \ldots, A_k \subset \mathbb{R}^d$ with $\forall i \neq j$, $A_i \cap A_j = \varnothing$ and $\cup_{j=1}^{k} A_j = \mathbb{R}^d$
2) While $|J_{T_a} - J_{T_b}| > \epsilon, \epsilon > 0$, assign each datapoint $x_i \in D$ to a cluster (choose the closest center)
3) pick new cluster center c_j based on Ave(A_j)

Using this algorithm with 10 clusters (as we originally had 10 latent topics), the resulting clus-

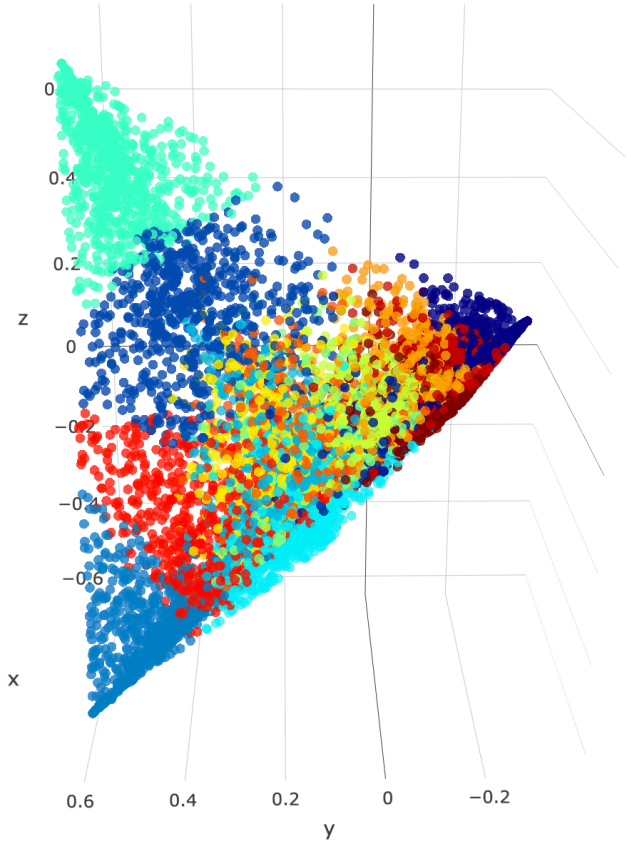


Fig. 4. A model with higher log-likelihood and lower perplexity ($\exp(-1. * \text{log-likelihood per word})$) is considered to be good

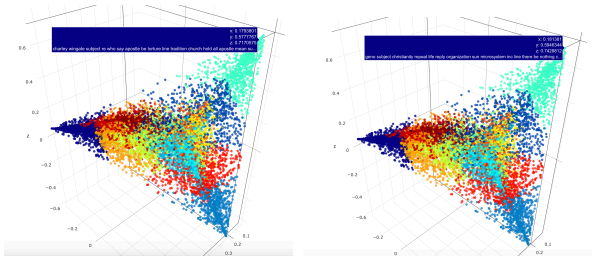


Fig. 5. A model with higher log-likelihood and lower perplexity ($\exp(-1. * \text{log-likelihood per word})$) is considered to be good

ters can be used to color the PCA-transformed simplex values.

In this example, it works out that values that are near each other inside of the simplex indeed do refer to similar topics.

REFERENCES

[1] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. New York: Springer series in statistics, 2001.

[2] Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning. Vol. 1. Cambridge: MIT press, 2016.

[3] Turing, Alan M. "Computing machinery and intelligence." In Parsing the Turing Test, pp. 23-65. Springer, Dordrecht, 2009.

[4] Kosuth, Joseph, "One and Three Chairs.", MOMA, NY 1965

[5] Lake, Brenden M., Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. "Building machines that learn and think like people." Behavioral and Brain Sciences 40 (2017).

[6] Santoro, Adam, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. "One-shot learning with memory-augmented neural networks." arXiv preprint arXiv:1605.06065 (2016).

[7] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Journal of machine Learning research 3, no. Jan (2003): 993-1022.

[8] Linstead, Erik, Cristina Lopes, and Pierre Baldi. "An application of latent dirichlet allocation to analyzing software evolution." In Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on, pp. 813-818. IEEE, 2008.

[9] Landauer, Thomas K. Latent semantic analysis. John Wiley & Sons, Ltd, 2006.

[10] Ng, Andrew, Jordan, Michael, and Weiss, Yair, On Spectral Clustering: Analysis and an Algorithm

[11] Zhu, Xiaojin, Course notes, Advanced Natural Language Processing