

Applications of Topic Models

Jordan Boyd-Graber

Department of Computer Science, UMIACS, Language Science
University of Maryland¹
jbg@umiacs.umd.edu

Yuening Hu

Google, Inc.²
ynhu@google.com

David Mimno

Information Science
Cornell University
mimno@cornell.edu

now

the essence of knowledge

Boston — Delft



Documents

Document Sharing Hub

Foundations and Trends[®] in Information Retrieval

Published, sold and distributed by:

now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:

now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

J. Boyd-Graber, Y. Hu and D. Mimno. *Applications of Topic Models*. Foundations and Trends[®] in Information Retrieval, vol. 11, no. 2-3, pp. 143–296, 2017.

This Foundations and Trends[®] issue was typeset in L^AT_EX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-68083-308-9

© 2017 J. Boyd-Graber, Y. Hu and D. Mimno

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording or otherwise, without prior written permission of the publishers.

Photocopying. In the USA: This journal is registered at the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923. Authorization to photocopy items for internal or personal use, or the internal or personal use of specific clients, is granted by now Publishers Inc for users registered with the Copyright Clearance Center (CCC). The 'services' for users can be found on the internet at: www.copyright.com

For those organizations that have been granted a photocopy license, a separate system of payment has been arranged. Authorization does not extend to other kinds of copying, such as that for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. In the rest of the world: Permission to photocopy must be obtained from the copyright owner. Please apply to now Publishers Inc., PO Box 1024, Hanover, MA 02339, USA; Tel. +1 781 871 0245; www.nowpublishers.com; sales@nowpublishers.com

now Publishers Inc. has an exclusive license to publish this material worldwide. Permission to use this content must be obtained from the copyright license holder. Please apply to now Publishers, PO Box 179, 2600 AD Delft, The Netherlands, www.nowpublishers.com; e-mail: sales@nowpublishers.com

**Foundations and Trends® in
Information Retrieval**
Volume 11, Issue 2-3, 2017
Editorial Board

Editors-in-Chief

Maarten de Rijke
University of Amsterdam
The Netherlands

Mark Sanderson
Royal Melbourne Institute of Technology
Australia

Editors

Ben Carterette
University of Delaware

Charles L.A. Clarke
University of Waterloo

Claudia Hauff
Delft University of Technology

Diane Kelly
University of Tennessee

Doug Oard
University of Maryland

Ellen M. Voorhees
National Institute of Standards and Technology

Emine Yilmaz
University College London

Fabrizio Sebastiani
ISTI-CNR

Ian Ruthven
University of Strathclyde

Jaap Kamps
University of Amsterdam

James Allan
University of Massachusetts, Amherst

Jamie Callan
Carnegie Mellon University

Jian-Yun Nie
University of Montreal

Jimmy Lin
University of Maryland

Leif Azzopardi
University of Glasgow

Marie-Francine Moens
Catholic University of Leuven

Mark D. Smucker
University of Waterloo

Rodrygo Luis Teodoro Santos
Federal University of Minas Gerais

Ryen White
Microsoft Research

Soumen Chakrabarti
Indian Institute of Technology Bombay

Tie-Jan Liu
Microsoft Research

Yiqun Liu
Tsinghua University

Editorial Scope

Topics

Foundations and Trends® in Information Retrieval publishes survey and tutorial articles in the following topics:

- Applications of IR
- Architectures for IR
- Collaborative filtering and recommender systems
- Cross-lingual and multilingual IR
- Distributed IR and federated search
- Evaluation issues and test collections for IR
- Formal models and language models for IR
- IR on mobile platforms
- Indexing and retrieval of structured documents
- Information categorization and clustering
- Information extraction
- Information filtering and routing
- Metasearch, rank aggregation, and data fusion
- Natural language processing for IR
- Performance issues for IR systems, including algorithms, data structures, optimization techniques, and scalability
- Question answering
- Summarization of single documents, multiple documents, and corpora
- Text mining
- Topic detection and tracking
- Usability, interactivity, and visualization issues in IR
- User modelling and user studies for IR
- Web search

Information for Librarians

Foundations and Trends® in Information Retrieval, 2017, Volume 11, 5 issues. ISSN paper version 1554-0669. ISSN online version 1554-0677. Also available as a combined paper and online subscription.

Applications of Topic Models

Jordan Boyd-Graber
Department of Computer Science, UMIACS, Language Science
University of Maryland¹
jbg@umiacs.umd.edu

Yuening Hu
Google, Inc.²
ynhu@google.com

David Mimno
Information Science
Cornell University
mimno@cornell.edu

¹Work completed while at University of Colorado

²Work completed while at Yahoo!

Contents

1	The What and Wherefore of Topic Models	2
1.1	Tell Me about Your Haystack	2
1.2	What is a Topic Model	5
1.3	Foundations	6
1.4	Latent Dirichlet Allocation	11
1.5	Inference	13
1.6	The Rest of this Survey	19
2	Ad-hoc Information Retrieval	21
2.1	Document Language Modeling	23
2.2	Topic-based Document Language Models	26
2.3	Query Expansion	27
2.4	Beyond Relevance—Search Personalization	34
2.5	Summary	36
3	Evaluation and Interpretation	38
3.1	Displaying Topics	38
3.2	Labeling Topics	40
3.3	Displaying Models	43
3.4	Evaluation, Stability, and Repair	45
3.5	Summary	47

4	Historical Documents	49
4.1	Newspapers	50
4.2	Historical Records	54
4.3	Scholarly Literature	56
4.4	Summary	58
5	Understanding Scientific Publications	60
5.1	Understanding Fields of Study	62
5.2	How Fields Change	64
5.3	Innovation	66
5.4	Summary	68
6	Fiction and Literature	69
6.1	Topic Models in the Humanities	69
6.2	What is a Document?	71
6.3	People and Places	72
6.4	Beyond the Literal	76
6.5	Comparison to Stylometric Analysis	78
6.6	Operationalizing “Theme”	78
6.7	Summary	79
7	Computational Social Science	81
7.1	Topic Models for Qualitative Analysis	84
7.2	Sentiment Analysis	84
7.3	Upstream and Downstream Models	86
7.4	Understanding Stance and Polarization	87
7.5	Social Networks and Media	88
7.6	Summary	91
8	Multilingual Data and Machine Translation	92
8.1	Document-level Alignment from Multilingual Corpora	94
8.2	Word-level Alignment from Lexical Data	96
8.3	Alignment from Parallel Corpora and Lexical Information	98
8.4	Topic Models and Machine Translation	99
8.5	The Components of Statistical Machine Translation	100
8.6	Topic Models for Phrase-level Translation	102

- 8.7 Topic Models for Sentence-level Language Modeling 106
 - 8.8 Reordering with Topic Models 110
 - 8.9 Beyond Domain Adaptation 111
 - 8.10 Summary 112
- 9 Building a Topic Model 113**
 - 9.1 Designing a Model 114
 - 9.2 Implementing the Model 117
 - 9.3 Debugging and Validation 123
 - 9.4 Communicating Your Model 125
 - 9.5 Summary 126
- 10 Conclusion 127**
 - 10.1 Coping with Information Overload 127
 - 10.2 Deeper Representations 128
 - 10.3 Automatic Text Analysis for the People 129
 - 10.4 Coda 131
- References 132**

Abstract

How can a single person understand what's going on in a collection of millions of documents? This is an increasingly common problem: sifting through an organization's e-mails, understanding a decade worth of newspapers, or characterizing a scientific field's research. Topic models are a statistical framework that help users understand large document collections: not just to find individual documents but to understand the general themes present in the collection.

This survey describes the recent academic and industrial applications of topic models with the goal of launching a young researcher capable of building their own applications of topic models. In addition to topic models' effective application to traditional problems like information retrieval, visualization, statistical inference, multilingual modeling, and linguistic understanding, this survey also reviews topic models' ability to unlock large text collections for qualitative analysis. We review their successful use by researchers to help understand fiction, non-fiction, scientific publications, and political texts.

J. Boyd-Graber, Y. Hu and D. Mimno. *Applications of Topic Models*. Foundations and Trends® in Information Retrieval, vol. 11, no. 2-3, pp. 143–296, 2017.

DOI: 10.1561/15000000030.

1

The What and Wherefore of Topic Models

Imagine that you are an intrepid reporter with an amazing scoop: you have twenty-four hours of exclusive access three decades of e-mails sent within a corrupt corporation. You know there's dirt and scandal there, but it has been well-concealed by the corporation's political friends. How are you going to understand this haystack well enough to explain it to your devoted readers under such a tight deadline?

1.1 Tell Me about Your Haystack

Unlike the vignette above, interacting with large text data sets is often posed as a needle in a haystack problem. The poor user—faced with documents that would take a decade to read—is looking for a single needle: a document (or at most a handful of documents) that matches what the user is looking for: a “smoking gun” e-mail, the document that best represents a concept [Salton, 1968] or the answer to a question [Hirschman and Gaizauskas, 2001].

These questions are important. The discipline of information retrieval is built upon systematizing, solving, and evaluating this problem. Google's search service is built on the premise of users typing a few

keywords into a search engine box and seeing quick, consistent search results. However, this is not the only problem that confronts those interacting with large text datasets.

A different, but related problem is *understanding* large document collections, common in science policy [Talley et al., 2011], journalism, and the humanities [Moretti, 2013a]. The haystack has more than one precious needle. At the risk of abusing the metaphor, *sometimes you care about the straw*. Instead of looking for a smoking gun alerting to you some crime that was committed, perhaps you are looking for a sin of omission: did this company never talk about diversity in its workforce? Instead of a single answer to a question, perhaps you are looking for a diversity of responses: what are the different ways that people account for rising income inequality? Instead of looking for one document, perhaps you want to provide population level statistics: what proportion of Twitter users have ever talked about gun violence?

At first, it might seem that answering these questions would require building an extensive ontology or categorization scheme. For every new corpus, you would need to define the buckets that a document could fit into, politely ask some librarians and archivists to put each document into the correct buckets, perhaps automate the process with some supervised machine learning, and then collect summary statistics when you are done.

Obviously, such laborious processes are possible—they have been done for labeling congressional speeches¹ and understanding emotional state [Wilson and Wiebe, 2005]—and remain an important part of social science, information science, library science, and machine learning. But these processes are not always possible, fast, or even the optimal outcome if we had infinite resources. First, they require a significant investment of time and resources. Even creating the *list* of categories is a difficult task and requires careful deliberation and calibration. Even if it were possible, a particular question might not warrant the time or effort: the oeuvre of a minor author (only of interest to a few), or the tweets of a day (not relevant tomorrow).

¹www.congressionalbills.org/

Table 1.1: Five topics from a twenty-five topic model fit on Enron e-mails. Example topics concern financial transactions, natural gas, the California utilities, federal regulation, and planning meetings. We provide the five most probable words from each topic (each topic is a distribution over all words).

Topic	Terms
3	trading financial trade product price
6	gas capacity deal pipeline contract
9	state california davis power utilities
14	ferc issue order party case
22	group meeting team process plan

This survey explores the ways that humans and computers make sense of document collections through tools called topic models. Topic models allow us to answer big-picture questions quickly, cheaply, and without human intervention. Once trained, they provide a framework for humans to understand document collections both directly by “reading” models or indirectly by using topics as input variables for further analysis. For readers already comfortable with topic models, feel free to skip this chapter; we will mostly cover the definitions and implementations of topic models.

The intended audience of this book is a reader with some knowledge of document processing (e.g., knows what “tokens” and “documents” are), basic understanding of some probability (e.g., what a distribution is), and interested in many application domains. We discuss the information needs of each application area, and how those specific needs affect models, curation procedures, and interpretations.

By the end of the book (Chapter 9), we hope that readers will be excited enough to attempt to embark on building their own topic models. In this chapter, we go deeper into more of the implementation details. Readers who are already topic model experts will likely not learn much technically, but we hope our coverage of diverse applications will expose a topic modeling expert to models and approaches they had not seen before.

Yesterday, SDG&E filed a motion for adoption of an electric procurement cost recovery mechanism and for an order shortening time for parties to file comments on the mechanism. The attached email from SDG&E contains the motion, an executive summary, and a detailed summary of their proposals and recommendations governing procurement of the net short energy requirements for SDG&E's customers. The utility requests a 15-day comment period, which means comments would have to be filed by September 10 (September 8 is a Saturday). Reply comments would be filed 10 days later.

Topic	Probability
9	0.42
11	0.05
8	0.05

Figure 1.1: Example document from the Enron corpus and its association to topics. Although it does not contain the word “California”, it discusses a single California utility’s dissatisfaction with how much it is paying for electricity.

1.2 What is a Topic Model

Returning to our motivating example, consider the e-mails from Enron, the prototypical troubled corporation of the turn of the century. A source has provided you with a trove of emails, and your editor is demanding an article by yesterday. You know that wrongdoing happened, but you do not know who did it or how it was planned and carried out. You have suspicions (e.g., around the California energy spot market), but you are curious about other skeletons in the closet and you are highly motivated to find them.

So you run a topic model on the data. True to its name, a topic model gives you “topics”, each of which is a ranking of all the distinct words in the e-mails by relevance to a topic. Taking the top five most relevant words in each topic results in collections of words that make sense together (Table 1.1). For example, one topic seems to have words relating to finance and trading. Another seems to involve to gas pipelines, their capacity, and deals or contracts relating to those pipelines. This all makes sense: Enron was an energy trading company. Others seem to involve language used in any business, such as meetings and plans.

$$\begin{array}{ccc}
 \left[\begin{array}{c} M \times K \end{array} \right] & \times & \left[\begin{array}{c} K \times V \end{array} \right] \approx \left[\begin{array}{c} M \times V \end{array} \right] \\
 \text{Topic Assignment} & & \text{Topics} \qquad \qquad \text{Dataset}
 \end{array}$$

Figure 1.2: A matrix formulation of finding K topics for a dataset with M documents and V unique words. While this view of topic modeling includes approaches such as latent semantic analysis (LSA, where the approximation is based on SVD), we focus on probabilistic techniques in the rest of this survey.

The first half of a topic model connects topics to a jumbled “bag of words”. When we say that a topic is about X , we are manually assigning a *post hoc* label (more on this in Chapter 3.1). It remains the responsibility of the human consumer of topic models to go further and make sense of these piles of straw (we discuss labeling the topics more in Chapter 3).

Making sense of one of these word piles by itself can be difficult. The second half of a topic model links topics to individual documents. For example, the document in Figure 1.1 is about a California utility’s reaction to the short-term electricity market and exemplifies Topic 9 from Table 1.1. Considering examples of documents that are strongly connected to a topic, along with the words associated with the topic, can give us a more complete representation of the topic. If we get a sense that Topic 9 is of interest, we can explore deeper to find other documents.

1.3 Foundations

You might notice that we are using the general term “topic model”. There are many mathematical formulations of topic models and many algorithms that learn the parameters of those models from data. Although we will focus on particular models and algorithms, we choose

our terminology to emphasize that the similarities between formulations, models, and algorithms are often greater than their differences.

Topic modeling began with a linear algebra approach [Deerwester et al., 1990] called latent semantic analysis (LSA): find the best low rank approximation of a document-term matrix (Figure 1.2). While these approaches have seen a resurgence in recent years [Anandkumar et al., 2012, Arora et al., 2013], we focus on probabilistic approaches [Hofmann, 1999a, Papadimitriou et al., 2000, Blei et al., 2003], which are intuitive, work well, and allow for easy extensions (as we see later in many of our later chapters).

The two foundational probabilistic topic models are latent Dirichlet allocation [Blei et al., 2003, LDA] and probabilistic latent semantic analysis [Hofmann, 1999a, pLSA]. We describe the former in significant detail in Chapter 1.4, but we want to take a moment to address some of the historical connection between these two models.

pLSA was historically first and laid the foundation for LDA. pLSA was used extensively in many applications such as information retrieval. However, this survey focuses on LDA because more researchers have not just *used* LDA—they have also *extended* it. LDA is not just widely used, but it is also widely modified. Because of these prolific modifications, we focus on the mechanics of LDA, which many researchers have used as the foundations of new models. However, as we explain below (Chapter 1.5.4), the similarities between pLSA and LDA outweigh the differences.

In any technical field it is common for general terms to take on specific, concrete meanings, and this can be a source of confusion. In topic modeling the word “topic” takes on the specific meaning of a probability distribution over words, while still alluding to the more general meaning of a theme or subject of discourse. Because other areas of information retrieval have similarly developed specific meanings for the word “topic”, we distinguish them here. The most common definition is a specific information need, as in the TREC evaluation corpora developed by NIST [Voorhees and Harman, 2005]. TREC topics are generally much more specific than topic model topics, and may relate to particular aspects or perspectives on a subject. An example from

the 2003 TREC Robust Track is “Identify positive accomplishments of the Hubble telescope since it was launched in 1991” [Voorhees, 2003]. Similarly to information retrieval, the related field of topic detection and tracking also has a specific technical definition of “topic” [Allan, 2002]. In TDT, a “topic” is usually closer to an event or an individual story. In contrast, topic models tend to identify more abstract latent factors. For example, a TDT topic might include an earthquake in Haiti, whereas a topic model might represent the same event as a combination of topics such as Haiti, natural disasters, and international aid.

There has been some work on using topic models to detect emerging events by searching for changes in topic probability [AlSumait et al., 2008]. But these methods tend to identify mainly the fact that an event has occurred, without necessarily identifying the specific features of that event. Other work has found that more lexically specific methods than topic models are best for identifying memes and viral phrases [Leskovec et al., 2009].

1.3.1 Probabilistic Building Blocks

In probabilistic models we want to find values for unobserved model variables that do a good job of explaining the observed data. The first step in inference is to turn this process around, and assert a way to generate data given model variables. Probabilistic models thus begin with a generative story: a recipe listing a sequence of random events that creates the dataset we are trying to explain. Figure 1.3 lists some of the key players in these stories, how they are parameterized and what samples drawn from these distributions look like. We will briefly discuss them, as we will use them to build a wide variety of topic models later.

Gaussian If you know any probability distribution already, it is (probably) the Gaussian. This distribution does not have a role in the most basic topic models that we will discuss here, but it will later (e.g., Chapter 7). We include it because it is a useful point of comparison against the other distributions we *are* using (since it is perhaps the easiest to understand and best known). A Gaussian is a distribution over all real numbers (e.g., 0.0, 0.5, -4.2 , π , ...). You can ask it to spit

Distribution	Density	Example Parameters	Example Draws
Gaussian	$\frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$\mu = 2, \sigma^2 = 1.1$	$x = 2.21$
Discrete	$\prod_i \phi_i^{\mathbb{1}[w=i]}$	$\phi = \begin{bmatrix} 0.1 \\ 0.6 \\ 0.3 \end{bmatrix}$	$w = 2$
Dirichlet	$\frac{\prod_{i=1}^K \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^K \alpha_i)} \prod_{i=1}^K \theta_i^{\alpha_i-1}$	$\alpha = \begin{bmatrix} 1.1 \\ 0.1 \\ 0.1 \end{bmatrix}$	$\theta = \begin{bmatrix} 0.8 \\ 0.15 \\ 0.05 \end{bmatrix}$

Figure 1.3: Examples of probability distributions used in the generative stories of topic models. In the case of the discrete draw, $w = 2$ denotes that the second element (the one with probability 0.6) was drawn.

out a number, and it will give you some real number between negative infinity and positive infinity. But not all numbers have equal probability. Gaussian distributions are parameterized by a mean μ and variance σ^2 . Most samples from the distribution will be near the mean μ ; how close is determined by the variance: higher variances will cause the samples to be more spread out.

Discrete While Gaussian distributions are over a continuous space, documents are combinations of discrete symbols, usually word tokens.² Thus, we need a distribution over discrete sets.

A useful metaphor for thinking about discrete distributions is a weighted die. The number of faces on the die is its dimension, and each face is associated with a distinct outcome. Each face has its own probability of how likely that outcome is; these probabilities are the parameters of a discrete distribution (Figure 1.3).

Topic models are described by discrete distributions (sometimes called multinomial distributions) that describe the connection between words and topics (the first half) and topics and documents (the second half). A distribution over words is called a topic distribution; each of

²An emerging trend in natural language processing research is to view words as embedded in a continuous space. We discuss these “representation learning” approaches and their connection to topic modeling in Chapter 10, but even then models are still defined over a discrete set of words.

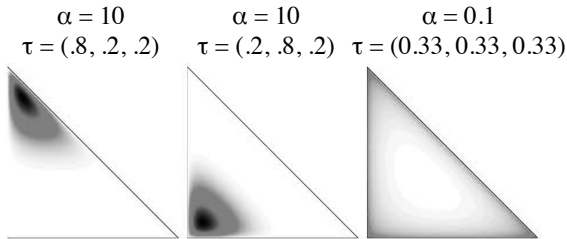


Figure 1.4: Given different Dirichlet parameters, the Dirichlet distribution can either be informative (left, middle) or sparse (right). Sparse distributions encourage distributions to favor a few elements but do not care which ones. This is consistent with our intuitions of how documents are written: they are only about a few things, and topics contain only a handful of words.

the topics gives higher weights to some words more than others (e.g., in Topic 9 from the Enron corpus, “state” and “california” have higher probability than other words). Each document also has an “allocation” for each topic: documents are about a small handful of topics, and most documents have very low weights for most of the possible topics.

Dirichlet Although discrete distributions are the star players in topic models, they are not the end of the story. We often begin with Dirichlet distributions. Just as Gaussians produce real numbers and discrete distributions produce symbols from a finite set, Dirichlet distributions produce probability vectors that can be used as the parameters of discrete distributions. Like the Gaussian distribution, they have parameters analogous to a mean and variance. The mean is called the “base measure” τ and is the expected value of the Dirichlet distribution: the values you would get if you averaged many draws from the Dirichlet. The concentration parameter α_0 controls how far away individual draws

are from the base measure. We often combine these parameters into a single value for each dimension: $\alpha_k = \alpha_0 \tau_k$.

If α_0 is very large, then the draws from a Dirichlet will be very close to τ (Figure 1.4, left). If α_0 is small, however, the discrete distributions become sparse (Figure 1.4, right). A sparse distribution is a distribution where only a few values have high probability and all other values are small.

Because topic models are meant to reflect the properties of real documents, modeling sparsity is important. When a person sits down to write a document, they only write about a handful of the topics that they could potentially use. They do not write about every possible topic, and the sparsity of Dirichlet distributions is the probabilistic tool that encodes this intuition.

There are several important special cases of the Dirichlet distribution. If the base measure τ is the same for every dimension, we call the resulting distribution *symmetric*. This case is appropriate when we do not expect any one element to be, on average, more likely than any other element across all samples from the distribution. In the symmetric case the distribution has only one parameter, the concentration α_0 . If the base measure is uniform and the concentration parameter α_0 is equal to the number of dimensions K (or, equivalently, $\alpha_k = 1.0$ for all k), the distribution is uniform, placing equal probability on all K -dimensional probability distributions.

1.4 Latent Dirichlet Allocation

We now have all the tools we need to tell the complete story of the most popular topic model: latent Dirichlet allocation [Blei et al., 2003, LDA]. Latent Dirichlet allocation³ posits a “generative process” about how the data came to be. We assemble the probabilistic pieces to tell this

³The name LDA is a play on LSA, its non-probabilistic forerunner (latent semantic analysis). Latent because we use probabilistic inference to infer missing probabilistic pieces of the generative story. Dirichlet because of the Dirichlet parameters encoding sparsity. Allocation because the Dirichlet distribution encodes the prior for each document’s allocation over topics.

story about generating topics and how those topics are used to create diverse documents.

Generating Topics The first part of the story is to create the topics. The user specifies that there are K distinct topics. Each of the K topics is drawn from a Dirichlet distribution with a uniform base distribution and concentration parameter λ : $\phi_k \sim \text{Dir}(\lambda \mathbf{u})$. The discrete distribution ϕ_k has a weight for *every* word in the vocabulary.

However, when we summarize topics (as in Figure 1.1), we typically only use the top (most probable) words of a topic. The lower probability words are less relevant to the topic and thus are not shown.

Document Allocations Document allocations are distributions over topics for each document. This encodes what a document is about; the sparsity of the Dirichlet distribution’s concentration parameter α_0 ensures that the document will only be about a few topics. Each document has a discrete distribution over topic: $\theta_d \sim \text{Dir}(\alpha \mathbf{u})$.

Words in Context Now that we know what each document is about, we create the words that appear in the document. We assume⁴ that there are N_d words in document d . For each word n in the document d , we first choose a **topic assignment** $z_{d,n} \sim \text{Discrete}(\theta_d)$. This is one of the K topics that tells us which topic the word token is from, but not what the word is.

To select which word we will see in the document, we draw from a discrete distribution again. Given a word token’s topic assignment $z_{d,n}$, we draw from that topic to select the word: $w_{d,n} \sim \phi_{z_{d,n}}$. The topic assignment tells you what the word is about, and then this selects which distribution over words we use to generate the word.

For example, consider the document in Figure 1.1. To generate it, we choose a distribution over all of the topics. This is θ . For this document, the distribution favors Topic 9 about California. The value for this topic

⁴We can model this in the generative story as well, e.g., with a Poisson distribution. However, we often do not care about document *lengths*—only what the document is about—so we can usually ignore this part of the story.

is higher than any other topic. For each word in the document, the generative process chooses a topic assignment z_n . For this document, any topic is theoretically possible, but we expect that most of those will be Topic 9.

Then, for each token in the document, we need to choose which word type will appear. This comes from Topic 9's distribution over words (multiple topics have word distributions shown in Figure 1.1). Each is a discrete draw from the topic's word distribution, which makes words like "California", "state", and "Sacramento" more likely.

It goes without saying that the generative story is a fiction [Box and Draper, 1987]. Nobody is sitting down with dice to decide what to type in on their keyboard. We use this story because it is *useful*. This fanciful story about randomly choosing a topic for each word can help us because if we assume this generative process, we can work backwards to find the topics that explain how a document collection was created: every word, every document, gets associated with these underlying topics.

This simple model helps us order our document collection: by assuming this story, we can discover *topics* (which certainly do not exist) so we can understand the common themes that people use to write documents. As we will see in later chapters, slight tweaks of this generative story allow us to uncover more complicated structures: how authors prefer specific topics, how topics change, or how topics can be used across languages.

1.5 Inference

Given a generative model and some data, the process of uncovering the hidden pieces of the probabilistic generative story is called *inference*. More concretely, it is a recipe for generating algorithms to go from data to *topics that explain a dataset*.

There are many flavors of algorithms for posterior inference: message passing [Zeng et al., 2013], variational inference [Blei et al., 2003], gradient descent [Hoffman et al., 2010], and Gibbs sampling [Griffiths and Steyvers, 2004]. All of these algorithms have their advocates and

reasons you should use them. In this survey, we focus on Gibbs sampling, which is simple, intuitive, and—with some clever tricks specific to topic models—fast [Yao et al., 2009]. (We discuss variational inference in Chapter 9.)

We present the results of Gibbs sampling without derivation, which—along with the history of its origin in statistical physics—are well described elsewhere.⁵ We use a variety of Gibbs sampling called *collapsed* Gibbs sampling, which allows inference to side-step some of the pieces of the generative story: instead of explicitly representing the parameters of a discrete distribution, distinct from any observations drawn from that distribution, we represent the distribution solely through those observations. We can then recreate the topic and document distributions through simple formulas.

1.5.1 Random Variables

Topic Assignments Since every individual token is assumed to be generated from a single topic, we can consider the *topic assignment* of a token as a variable. For example, an instance of the word “compilation” might be in a Computer topic in one document and in an Arts topic in another document. Because each token has its own topic assignment, the same word might be assigned to *different* topics in the *same* document. To estimate *global* properties of the topic model we use aggregate statistics derived from token-level topic assignments.

Document Allocation The document allocation is a distribution over the topics for each document; in other words, it says how popular each topic is in a document. If we count up how often a document uses a topic, this gives us its popularity. We define $N_{d,i}$ as the number of times document d uses topic i . This is larger for more popular topics; however, it is not a probability because it is larger than one. We make it a probability by dividing by the number of words in a document

$$\frac{N_{d,i}}{\sum_k N_{d,k}}, \quad (1.1)$$

⁵We recommend Resnik and Hardisty [2009] for additional information on derivation.

but this is problematic because it can sometimes give us zero and ignores the influence of the Dirichlet distribution; a better estimate is⁶

$$\theta_{d,i} \approx \frac{N_{d,i} + \alpha_i}{\sum_k N_{d,k} + \alpha_k}. \quad (1.2)$$

This must never become zero because we do not want it to rule out the possibility that a topic is used in a particular document (hence, each α must be non-zero). This helps the sampler explore more of the possible combinations.

Topics Each topic is a distribution over words. To understand what a topic is about, we look at the profile of all of the tokens that have been assigned to that topic. We estimate the probability of a word in a topic as

$$\phi_{i,v} \approx \frac{V_{i,v} + \beta_v}{\sum_w V_{i,w} + \beta_w}, \quad (1.3)$$

where β is the Dirichlet parameter for the topic distribution.

1.5.2 Algorithm

The collapsed Gibbs sampling algorithm for learning a topic model is only based on the topic assignments, but we will use our estimates for the topics ϕ_k and the documents θ_d discussed above. We begin by setting topic assignments randomly: if we have K topics, each word has equal chance to be associated with any of the topics. These topics will be quite bad, looking like noisy copies of the overall corpus distribution. But we will improve them one word at a time.

The algorithm proceeds by sweeping over all word tokens in turn over and over. At each iteration we change the topic assignments for each word in a way that reflects the underlying probabilistic model of the data. On average, each pass over the data makes the topics slightly better until the model reaches a steady state. There is no easy way to tell when such a steady state has been reached, but eventually the topics will “converge” to reasonable themes and you can consider yourself done.

⁶To be technical, Equation 1.1 is a maximum likelihood estimate and Equation 1.2 is the maximum *a posteriori*, which incorporates the influence of both the prior and the data.

The equation for the probability of assigning a word to a particular topic combines information about words and about documents⁷

$$p(z_{d,n} = i \mid \dots) = \theta_d \phi_{ji} = \left(\frac{N_{d,i} + \alpha_i}{\sum_k N_{d,k} + \alpha_k} \right) \left(\frac{V_{i,w_{d,n}} + \beta_v}{\sum_w V_{i,w} + \beta_w} \right). \quad (1.4)$$

Computing this value for each topic will result in a probability distribution over the topic assignment for this word token, given all the other topic assignments. The next step is to randomly choose one of those indices with probability proportional to the vector value. You now assign that word to the topic, update $N_{d,\cdot}$ and $V_{\cdot,w_{d,n}}$, and move on to the next word and repeat. The two terms provide two “pressures”, for global and local coherence. Sparsity in the topic-word distributions encourages tokens of the same word type to be assigned to a small number of topics, regardless of where they occur. Sparsity in the document-topic distributions encourages tokens in the same document to be assigned to a small number of topics, regardless of what type they are. For example, knowing that a word is “compilation” narrows down the number of potential topics considerably, but leaves ambiguity: is it *program* compilation or a *music* compilation? Knowing that the word occurs in a document with many other words in the Arts topic resolves this ambiguity, leaving the Arts topic as the most probable assignment.

At the very end of the algorithm, we can use the estimates of each topic (Equation 1.3) to summarize the main themes of the corpus and the estimates of each document’s topic distribution (Equation 1.2) to start exploring the collection automatically (Chapter 2) or with a human in the loop (Chapter 3).

The algorithm that we have sketched here is the foundation of many of the more advanced models that we will discuss later in the survey. While we will not describe the algorithms in detail, we will occasionally reference this sketch to highlight challenges or difficulties in implementing topic models.

⁷To be theoretically correct, it is important not to include the count associated with the token you are sampling in these counts, which becomes more clear if the probability is written as $p(z_{d,n} = j \mid z_{d,1} \dots z_{d,n-1}, z_{d,n+1} \dots z_{d,N_d}, w_{d,n})$ to show the dependence on the topic assignments of *all other* tokens but not this token.

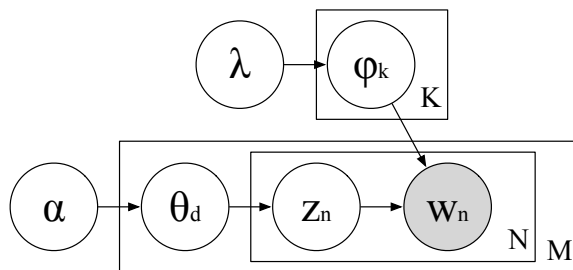


Figure 1.5: Plate diagram for LDA. Nodes show random variables, lines show (possible) probabilistic dependence, rectangles show repetition, and shading shows observation.

1.5.3 Plate Diagrams

Plate diagrams provide a shorthand for quickly explaining which random variables are associated with each other. If you look up many of the references used in this survey, you will likely see plate diagrams (we also use a plate diagram later in Figure 2.1b).

Let's begin with a plate diagram for LDA (Figure 1.5). You can compare these to the generative story in Chapter 1.4. All of the random variables are there, each in its own circle. The lines between random variables tell more of the story. You can see that if a random variable is conditioned on another, there is a line going from the variable that is *conditioned on* **to** the variable that is *conditionally dependent*. For example, a word depends on the token assignment $z_{d,n}$ and a topic ϕ_k , so we draw lines from both.

You can think about the rectangular boxes as repetition. The letter in the bottom right of the box shows how often what is inside the box is replicated. There is a box for each document (there are M in total) and each token (the box of words is inside the box for documents).

When a variable is shaded, this means that it is observed. These are the data we start with. The unshaded variables must either be inferred (e.g., topics ϕ) or are hyperparameters that must be set or inferred (e.g., Dirichlet parameter α).

Plate diagrams allow a reader to quickly see a “family resemblance” between related models, and once someone has become fully immersed in topic models, it is often possible to at a glance understand a model from its plate diagram. However, plate diagrams are imperfect; they lack some of the key information you need to understand the model. For instance, the exact probabilistic relationship between variables is underspecified.

1.5.4 What is so Great about Dirichlet?

Now that we have described what LDA is, we can return to its history. What is the innovation that separates LDA from pLSA, its predecessor? Naïvely, the difference is changing an “s” to a “d” (i.e., changing pLSA to LDA). The deeper story is about as consequential.

Instead of having a Dirichlet prior over θ , pLSA assumes that θ is a discrete parameter. In practice, this means that documents are not encouraged to focus on a limited number of topics and often “spread out” to have small weights for many different topics. In theory, this means that there is not as sound a generative story for how a document came to be: you cannot run the generative process forward from scratch if you must have θ as a parameter to start with.

These differences are relatively minor. LDA has slightly easier inference—particularly when it comes to tweaking the model—which has caused it to become the more popular of the two models. Thus, we will focus on comparing models to LDA. This is not to diminish from pLSA and its unquestionable place in the literature, but it helps us present a more unified narrative for our reader.

1.5.5 Implementations

Hopefully the previous algorithm sketch has convinced you that implementing topic models is not a Herculean task; most skilled programmers can complete a reasonable implementation of topic models in less than a day. However, we would suggest not trying to implement basic LDA if you just want the output of a topic model, many solid implementations can help users get to useful results more quickly, particularly as topic models often require extensive preprocessing.

Mallet is fast and is a widely used implementation in Java [McCallum, 2002]. This is where you should probably start, in our biased opinion. It runs in Java, uses highly-optimized Gibbs sampling implementations, and can work from a variety of text inputs. It is well documented, mature, and runs well on a multi-core machine, allowing it to process up to millions of documents. Variational inference is the other major option [Blei et al., 2003, Langford et al., 2007], but often requires a little more effort for new users to get a first result.

However, not all users are comfortable with Java; many implementations are available on other platforms and in many programming languages.⁸ Many of these implementations are well-built, but check whether they have all of the features of mature implementations like Mallet so that you know what (if anything) you're missing.

However, if your corpus is truly large, consider techniques that can be parallelized over large computer clusters. These techniques can be based on variational inference [Narayanamurthy, 2011, Zhai et al., 2012] or on sampling [Newman et al., 2008].

While these implementations allow you to run *specific* topic models, other frameworks allow you to specify arbitrary generative models. This enables quick prototyping of topic models and integrating topic models with other probabilistic frameworks like regression or collaborative filtering. Examples of these general frameworks include Stan [Stan Development Team, 2014], Theano [Theano Development Team, 2016], and Infer.net [Minka et al., 2014].

If you cannot find the specific model that you want among these existing software packages, the flexibility and simplicity of topic models and inference makes it relatively simple to adapt topic models to model specific phenomena (as we describe in following chapters).

1.6 The Rest of this Survey

In each of the following chapters, we focus on an application of topic models, gradually increasing the complexity of the underlying models.

⁸So many that change so quickly; thus, we are reluctant endorse specific ones here.

The chapters do occasionally refer to each other, but a reader should be able to read each of the chapters independently.

The next chapter returns to the distinction between high level overviews and finding a needle in a haystack. We show how a high level overview can help users and algorithms find documents of interest. We show how a high level overview can help algorithms (Chapter 2) and users (Chapter 3) find documents of interest.

These tools help enable new applications of topic models: how understanding newspapers (Chapter 4) reveals the march of history, how the corpus of writers of fiction (Chapter 6) illuminates societal norms, how the writings of science reveal innovation (Chapter 5), or how politicians' speeches (Chapter 7) reveal schisms in political organizations.

Finally, the survey closes with thoughts about how interested researchers can start building their own topic models (Chapter 9) and how topic models may change in the future (Chapter 10).

2

Ad-hoc Information Retrieval

Topic models explore and summarize document collections outside the context of any specific information need, when we do not necessarily know what we are looking for. This approach to information retrieval stands in contrast to traditional IR systems, which retrieve relevant documents given users' explicit information needs. Where IR systems might look for the “needle in the haystack”, topic models will tell you about the overall proportion of hay and needles, and perhaps inform you about the mice that you did not know were there. But topic models can also be useful in situations when we do have a specific information need, but we do not quite know how to search for it. Despite their differences in purpose, there are strong mathematical and conceptual connections between these two approaches. In this chapter we consider the use of topic modeling in IR to balance specific user queries with more open-ended discovery.

In the most direct sense, topic models can be used as a simple indexing method. Users can find topics that assign high probability to a particular query term, and then find documents with a high probability of these topics. Such topic-based search may additionally provide some level of query disambiguation, since it may be clear from topic-word

distributions that one or another topic is more relevant to the user's information need. More sophisticated approaches blur the boundary between query-driven retrieval and unsupervised topic modeling. Erlin [2017] searches for passages related to epistemology in English and German books by “seeding” topic models with words thought to be relevant to that subject. This approach can be successful, but does not guarantee that relevant topics will be found, or that topics will match the intended subject.

In the more formal *ad-hoc* retrieval setting, users start with an information need expressed in queries. Many IR systems treat both the queries and documents as “bags of words”, and retrieve and rank the documents by measuring the word overlap between queries and documents. However, the ability of this direct and simple matching is always limited. Words with similar meaning or in different forms should also be considered as matched instead of being ignored. **Language modeling** has been one of the most popular frameworks to capture such semantic relationships. But humans would also like to use background knowledge to interpret and understand the queries and “add” missing words [Wei, 2007], which provides another approach called **query expansion** to improve retrieval and ranking results.

Both directions can be pursued by learning and discovering the semantic relations between words and, further, the semantic relations between queries and documents. Topic models provide semantic relations between query words and documents [Deerwester et al., 1990, Hofmann, 1999b] by describing each topic using probabilistically-weighted words and modeling each document as a distribution over all topics. This adds a layer of abstraction between a document and the exact words present in that document.

Appealing to the generative “story” of a model, we want to recover the words that *could* have been available to an author based on the words that were chosen. Such semantic relations can be applied to smoothing the language models, or introducing related words in query expansion. This chapter focuses on how to apply topic models in document language modeling [Lu et al., 2011, Wei and Croft, 2006] and query

expansion [Park and Ramamohanarao, 2009, Andrzejewski and Buttler, 2011] to further improve ranking results of information retrieval.

2.1 Document Language Modeling

The language modeling approach [Ponte and Croft, 1998, Song and Croft, 1999, Croft and Lafferty, 2003] is one of the main frameworks for using topic models in IR systems, since it is an effective probabilistic framework for studying information retrieval problems [Ponte and Croft, 1998, Berger and Lafferty, 1999]. A statistical language model estimates the probability of word sequences, denoted as $p(w_1, w_2, \dots, w_n)$. In practice, the statistical language model is often approximated by n-gram models. A unigram model assumes each word in the sequence is independent,

$$p(w_1, w_2, \dots, w_n) = p(w_1)p(w_2) \cdots p(w_n) \quad (2.1)$$

A trigram model assumes the probability of the current word only depends on the previous two words, and it is represented as

$$p(w_1, \dots, w_n) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \cdots p(w_n|w_{n-2}, w_{n-1}). \quad (2.2)$$

Language models are used in information retrieval to estimate the similarity of documents and queries [Zhai and Lafferty, 2001a]. Each document is treated as a sample from a particular language model, which we estimate based on the terms present in the document. Given a language model trained from a document, we can then calculate the probability of any other sequence of words under that model. For a unigram language model, this probability is just the product of the individual term probabilities (or, equivalently, the sum of their log probabilities). We can therefore easily find the score for a given user query under every document's language model, and rank the documents by that score.

Given a sample document d , the simplest way to estimate the associated language model is the maximum likelihood principle. The resulting distribution is the one that places the largest possible probability on

the observed document. The probability of generating a word w is the proportion

$$p_{\text{ml}}(w | d) = \frac{n_{d,w}}{n_{d,\cdot}} \quad (2.3)$$

where $n_{d,w}$ is the term frequency of word w in document d , and $n_{d,\cdot}$ is the total number of tokens in document d . Then the probability of generating the given query q is

$$p(q | d) = \prod_{w \in q} p(w | d) = \prod_{w \in q} \frac{n_{d,w}}{n_{d,\cdot}}. \quad (2.4)$$

Then the documents are ranked based on this probability $p(q | d)$. Higher probability implies the corresponding document is more relevant to the given query [Song and Croft, 1999]. However, a document often contains limited number of words and maximum likelihood estimation gives zero probability to those unseen words. If a query contains any word not in the document, the probability of generating the whole query given this document is zero, which may throw out perfectly good documents.

This data sparsity problem can be fixed by smoothing, which allocates some non-zero probability to the missing terms. Another solution—which also provides other benefits—is topic models. They provide a unique way to extract the word probabilities given the corpus, which can be used to smooth document language models. We summarize two simple smoothing methods, and then show how topic models fit into this smoothing framework.

There are two major directions for smoothing: **interpolation** [Jelinek and Mercer, 1980, Mackay and Peto, 1995, Ney et al., 1994, Ponte and Croft, 1998, Zhai and Lafferty, 2001a] and **backoff** [Katz, 1987, Song and Croft, 1999]. The interpolation-based method discounts the counts of the seen words and distribute the extra counts to both seen words and unseen words. An alternative backoff smoothing strategy trusts the maximum likelihood estimation for high count words, discounts and redistributes mass only for the less common words [Zhai and Lafferty, 2001a].

Here we review two popular and simple interpolation smoothing methods, which are further extended with topic models to smooth document language models.

Jelinek-Mercer The Jelinek-Mercer method [Jelinek and Mercer, 1980] is a linear interpolation of the maximum likelihood model in a document with the model based on the whole corpus, and a coefficient λ combines the two parts:

$$p(w|d) = (1 - \lambda)p_{\text{ml}}(w|d) + \lambda p(w|\mathcal{C}), \quad (2.5)$$

where $\mathcal{C}$ denotes the whole corpus. This simple mixture solves the data sparsity problem. For terms that occur in the document d , the maximum likelihood estimator (Equation 2.3) is not accurate given the limited size of a document, thus it is smoothed with the more reliable corpus level probability. For a missing term w in the document d , the probability of generating word w is not zero any more, but falls back to the corpus level probability $p(w|\mathcal{C})$. This smoothing method has been explored and successfully applied in information retrieval tasks [Ponte and Croft, 1998, Song and Croft, 1999].

Bayesian Smoothing using Dirichlet Priors A language model can be viewed as a discrete distribution, thus it can be smoothed by applying the Dirichlet distribution as the conjugate prior [Mackay and Peto, 1995]. We made a similar observation in the previous chapter comparing Equation 1.1 and Equation 1.2; the same intuition can be extended through multiple layers of discrete distributions with Dirichlet priors to create a smoothing model. Intuitively, this smoothing adds an extra prior count for each word to smooth the probability of unseen words,

$$p(w | d) = \frac{n_{d,w} + \beta p(w|\mathcal{C})}{\sum_{v \in V} n_{d,v} + \beta}, \quad (2.6)$$

where the Dirichlet prior is decided by concentration parameter β and the corpus-level probabilities $p(v|\mathcal{C})$,

$$(\beta p(v_1|\mathcal{C}), \beta p(v_2|\mathcal{C}), \dots, \beta p(v_n|\mathcal{C})). \quad (2.7)$$

2.2 Topic-based Document Language Models

Topic models, which model each document as a mixture of topics and each topic as a mixture of words, offer an interesting framework to model documents in information retrieval. Popular topic models such as probabilistic latent semantic analysis (pLSA) and latent Dirichlet allocation (LDA) have been both explored to improve document language models.

Hofmann [1999b] introduces pLSA to learn the relationship between query words and documents, and the conditional probability of a query word w given a document d is computed as marginalizing all topics k ,

$$p_{\text{TM}}(w | d) = \sum_k p(w | k)p(k | d) \quad (2.8)$$

Following this idea, Wang et al. [2013] further add regularizations—changing the shape of distributions to be more or less spread out—on document topic representations which is useful for retrieval. Instead of using pLSA, Wei and Croft [2006] apply the same idea to learn the topic-smoothed document-word distribution using LDA. Vosecky et al. [2014] also explore LDA for document language models on twitter search.

Because the posterior estimates for topics are smoothed by the Dirichlet priors, topic models learn a better and smoothed semantic relationship between document words and documents. As a result, even though this approach loosens the connection between query words and documents, it is a good approach to complement the original document language models. Thus Wei and Croft [2006] further propose to combine the LDA-based document model with the original document model (Equation 2.5) through a linear interpolation,

$$p(w | d) = \lambda' \left((1 - \lambda)p_{\text{m1}}(w | d) + \lambda p(w | \mathcal{C}) \right) + (1 - \lambda')p_{\text{TM}}(w | d) \quad (2.9)$$

where λ' is the coefficient which combines the LDA-based document model with the general smoothed language model.

Following Wei and Croft [2006], Lu et al. [2011] further evaluate the performance of applying topic models into the document language model framework. Instead of combining with the language model with

Jelinek-Mercer smoothing (Equation 2.5), Lu et al. [2011] smooth the document language model with Bayesian smoothing (Equation 2.7), and the final linear combination with topic models

$$p(w|d) = \lambda \frac{n_{d,w} + \beta p(w|\mathcal{C})}{\sum_{v \in V} n_{d,v} + \beta} + (1 - \lambda) p_{\text{TM}}(w|d) \quad (2.10)$$

While using different smoothing strategies, both approaches apply topic models to connect the query words with documents through hidden topics. As the example shown in Wei and Croft [2006], given a query “buyout leverage”, a relevant document talks about “Farley Unit Defaults On Pepperell Buyout Loan” without the exact word “leverage”, thus the ranking for this relevant document is very low. However, topic models connect this document with two topics that have strong connections with the term “leverage”: one economic topic contain words like “million”, “company” and “bankruptcy”, and the other money market topic is connected to “bond”. Since a better semantic relationship between the query and the document is learned, this relevant document is ranked much higher and the retrieval performance improves.

2.3 Query Expansion

The document language models in information retrieval [Ponte and Croft, 1998] attempt to model the query generation process based on the document models. However, a big problem is that these models abandon modeling the query-document relevance explicitly [Lavrenko and Croft, 2001], which is important in traditional information retrieval tasks.

In fact, queries, which are normally brief and using informal language from users, diverge significantly from the language in documents [Müller and Gurevych, 2009]. This semantic gap or lexical gap can result in perceived poor query-document relevance, even though the document is quite relevant from the users’ view point. For example, after typing in “apple products” to a search engine, a frustrated user might append “computer” to the query after faced with a screen of fruit-based search results.

Query expansion tries to automatically simulate a similar process to prevent this frustration. Query expansion normally analyzes the relationships between the query words and other words and tries to find potential related words so that the original query is better represented; thus improving better query-document relevance. For example, without much context, the query “dtd amc” is hard to understand [Jiang et al., 2016]. Through query expansion, it is possible to build up the relationship between “dtd” and “disneyland downtown”, which is more helpful in document retrieval. Next, this section reviews the classic query expansion frameworks in information retrieval, and the related works about using topic models for query expansion are introduced in the next section.

2.3.1 Learning Query-Word Relationships for Query Expansion

There are two main steps for query expansion. The first step is to find the relationships between queries and words and select the top related words to expand the query. The second step is to apply the expanded queries for ranking and compute the final ranking relevance scores. We start with the first step. Two major directions have been explored: query language models [Zhai and Lafferty, 2001b] and relevance models [Lavrenko and Croft, 2001].

Query Language Model To learn the query-word relationship, Zhai and Lafferty [2001b] build up a query language model to estimate the probability $p(w | q)$ of a word w given a query q . However, it is not easy to learn a good query language model since the query content is too limited.

Zhai and Lafferty [2001b] propose to use both the query content and the relevant documents $\mathcal{F}$ (sometimes referred as feedback documents or clicked documents) to estimate the query language model. Let $\hat{\theta}_{\mathcal{F}}$ be the estimated query language model based on the relevant documents and $\hat{\theta}_Q$ is the original query language model estimated purely based on queries, the combined query model $\hat{\theta}_{Q'}$ is

$$\hat{\theta}_{Q'} = (1 - \lambda)\hat{\theta}_Q + \lambda\hat{\theta}_{\mathcal{F}} \quad (2.11)$$

Estimating $\hat{\theta}_Q$ is obvious based on query words, and $\hat{\theta}_{\mathcal{F}}$ can also be simply estimated by a unigram language model θ which generates each word in $\mathcal{F}$ independently. However, most documents contain not only the query relevant information, but also the background information. As a result, Zhai and Lafferty [2001b] propose to generate a relevant document by a mixture model, which combines a language model $p(w | \theta)$ with a collection language model $p(w | \mathcal{C})$, and the log-likelihood of the relevant document is,

$$\log p(\mathcal{F} | \theta) = \sum_i \sum_w n_{d_i, w} \log((1 - \lambda')p(w | \theta) + \lambda'p(w | \mathcal{C})) \quad (2.12)$$

where $n_{d_i, w}$ is count of word w in document d_i . By combining information from relevant documents, the query language model is more robust, thus query-word relationships are better represented.

Relevance Model Unlike the query language model approach, Lavrenko and Croft [2001] assume both the query and the relevant documents are random samples from an unknown relevance model R . Given the query q , they approximate the probability $p(w | R)$ based on the observed query q as

$$p(w | R) \approx p(w | q) = \frac{p(w, q)}{p(q)}. \quad (2.13)$$

To estimate the joint probability $p(w, q)$, Lavrenko and Croft [2001] assumes the word w and the query q are sampled independently from the same distribution, e.g., from a unigram distribution, then the joint probability is

$$p(w, q) = \sum_{d \in \mathcal{C}} p(d)p(w, q | d) = \sum_{d \in \mathcal{C}} p(d)p(w | d)p(q | d). \quad (2.14)$$

Then the discrete distribution $p(w | q)$ for a given query q is

$$p(w | q) = \frac{p(w, q)}{p(q)} = \sum_{d \in \mathcal{C}} p(w | d)p(d | q). \quad (2.15)$$

Both the query language model and the relevance model capture the relationship between the query and other words, based on which the top related words can be selected for query expansion.

2.3.2 Ranking Relevance with Query Expansion

Given the related words for query expansion, the next question is how to apply the expanded queries for computing the final ranking relevance score. The combination can happen either before or after the relevance score is computed.

Zhai and Lafferty [2001b] combine the expanded query language model $\hat{\theta}_F$ with the original query language model $\hat{\theta}_Q$ as one query language model $\hat{\theta}_{Q'}$ (Equation 2.11). Given a query q generated from the expanded query model $p(q|\hat{\theta}_{Q'})$, and a document d generated from a document model $p(d|\hat{\theta}_D)$, they compare how similar the topic distributions are between these two “documents” are.

In contrast to Zhai and Lafferty [2001b], Lavrenko and Croft [2001] compute the relevance score using the original query and the expanded query respectively, and then linearly combine the two scores. Thus the final query-document relevance $\hat{s}_d(q)$ is computed as,

$$\hat{s}_d(q) = \lambda s_d(e) + (1 - \lambda) s_d(q) \quad (2.16)$$

where $s_d(q)$ is the relevance between the original query q and documents d , and $s_d(e)$ is relevance between the expanded query terms e and document d .

2.3.3 Applying Topic Models For Query Expansion

Topic models capture the semantic relationships of words through learning the latent topics, which are presented as distributions over different words. Such semantic relationships among words provide a unique way to match or expand words at the semantic level rather than by a direct spelling matching. For example, given a short query “diabetes”, topic models can easily find the related words such as “insulin”, “glucose”, “coronary” and “metformin” etc., as they often occur in the same context [Zeng et al., 2012]. As a result, topic models have been successfully applied into query expansion [Yi and Allan, 2009, Park and Ramamohanarao, 2009, Zeng et al., 2012].

Smoothing Query Language Model The most intuitive way to use topic models for query expansion is to extract the words’ relevance from

topics directly as Yi and Allan [2009]. They train a topic model, from which the probability $p_{\text{TM}}(k | q)$ of a topic k in a query q is learned. Then the query-word relevance $p(w | q)$ is computed based on topics:

$$p(w | q) = \sum_k p_{\text{TM}}(w | k) p_{\text{TM}}(k | q). \quad (2.17)$$

This query-word relevance $p(w | q)$ from topic models smooths the original query language model through linear interpolation. However, queries are normally too short to learn meaningful topics, thus the quality of query-word relevance is relatively limited. To improve the quality of extracted topics, Yi and Allan [2009] also train topic models from the relevant documents (e.g., top documents retrieved by a query), and extract the query-word relationships based on the Equation 2.17 for query expansion.

Improving Relevance Model In addition to this direct approach, Yi and Allan [2009] also apply topic models to improve the relevance model in Equation 2.15 for query expansion. In this approach, topic models capture the document-word relationship $p(w | d)$ given the query q as

$$p_{\text{TM}}(w | d, q) = \sum_k p(w | k) p(k | d, q) \quad (2.18)$$

where,

$$p(k | d, q) = \frac{p(k | d) p(q | k)}{p(q | d)} \approx p(k | d) p(q | k) \quad (2.19)$$

where $p(k | d)$ is the topic probability in document d , and $p(q | k)$ is the probability of generating a query q given the topic k . Then the topic-based document-word relationship $p_{\text{TM}}(w | d, q)$ is applied to smooth the document-word relationship $p(w | d)$ in the relevance model (Equation 2.15) through a linear interpolation,

$$p(w | q) = \sum_{d \in C} (\lambda p(w | d) + (1 - \lambda) p_{\text{TM}}(w | d, q)) p(d | q) \quad (2.20)$$

where λ is a constant weight to combine the original relevance model and the topic-based relevance model. Because the topic models capture the word relationships on a semantic level, this improved relevance model better captures the query-word relationships and improve query expansion.

Learning Pair-wise Word Relationships Park and Ramamohanarao [2009] also apply topic models for query expansion but in a different way. They model the pair-wise relationships between words through topic models and then apply it for query expansion. More specifically, based on the topics extracted from topic models, they compute the probabilistic relationships of each word pair (w_x, w_y) ,

$$\begin{aligned} p(w_x | w_y, \alpha) &= \sum_k p(w_x, k | w_y, \alpha) \\ &= \sum_k p(w_x | k, \alpha) p(k | w_y, \alpha) \end{aligned} \quad (2.21)$$

where α is the concentration parameter of the Dirichlet prior for document-topic distributions and $p(w_x | k, \alpha)$ is the probability of word w_x in topic k which can be learned from topic models, and $p(k | w_y, \alpha)$ is

$$p(k | w_y, \alpha) = \frac{p(w_y | k, \alpha) p(k | \alpha)}{\sum_{k'} p(w_y | k', \alpha) p(k' | \alpha)}$$

where $p(w_y | k, \alpha)$ is the probability of word w_y in topic k . Park and Ramamohanarao [2009] also show that $p(k | \alpha) = \frac{\alpha_k}{\sum_j \alpha_j}$. As a result, we have,

$$p(k | w_y, \alpha) = \frac{p(w_y | k, \alpha) \alpha_k}{\sum_{k'} p(w_y | k', \alpha) \alpha_{k'}}$$

The final probabilistic relationships of each term pair can be represented as

$$p(w_x | w_y, \alpha) = \frac{\sum_k p(w_x | k, \alpha) p(w_y | k, \alpha) \alpha_k}{\sum_{k'} p(w_y | k', \alpha) \alpha_{k'}} \quad (2.22)$$

Once this term relationship is obtained, they choose the top related terms as the expanded terms e for the given query q , and the final document ranking score is computed as Equation 2.16.

Interactive Feedback Relevance feedback involves the users in the retrieval process to improve the ranking result set [Rocchio, 1971]. The basic idea is to ask users to give feedback on the relevance of documents in an initial set of retrieval results, and users' feedback on relevance is

Table 2.1: Given query “euro opposition”, seven topics are selected and shown to a user. The user selected Topic 79 as the feedback topic (Example from Andrzejewski and Buttler [2011]).

Topic	Terms
196 (debate)	Tory Euro sceptics, social chapter, Liberal Democrat, mps, Labour, bill, Commons
404 (ratification)	ratification Masstricht treaty, Poul Schluter, Poul Rasmussen, Danish, vote, Denmark, ec
79 (Emu)	economic monetary union, Masstricht treaty, member states, European, Europe, Community, Emu
377 (George)	President George Bush, White House, Mr Clinton, administration
115 (power)	de regulation bill, Sunday trading, Queen Speech, law, legislation, government, act
446 (years)	chairman chief executive, managing director, finance director, Sir, board, group, company
431 (cabinet)	Mr John Major, prime minister, Mr Major, party, tory, government, Conservative

further used to improve the ranking results. This process can go through one or more iterations.

Andrzejewski and Buttler [2011] present a new framework for obtaining and exploiting user feedback at the latent topic level. They learn the latent topics from the whole corpus and construct meaningful topic representations. At query time, they decide which latent topics are potentially relevant and present the topic representations along keyword search results. When a user select a latent topic, the original query is expanded with the top words in this selected topic, and the search results are refined. Andrzejewski and Buttler [2011] use the query “euro opposition” as an example: users want to find documents about opposition to the introduction of the single European currency. 500 topics are learned using the corpus and relevance judgements. Seven of the 500 topics are selected to show to users as shown in Table 2.1 and the user select the Topic 79 as the user feedback. Using the top terms in Topic 79 as the expanded query terms, the ranking of the relevant documents improves.

This direction is related to topic labeling and interactive visualization for topic models, which will be further discussed in Chapter 3.

2.4 Beyond Relevance—Search Personalization

Traditional search systems retrieve documents based on the queries only, regardless of who submitted the queries. As more Web pages become available, queries are normally too short to express users' needs, and users may prefer different results even the input queries are the same [Jansen et al., 2000, Dou et al., 2007]. As the examples shown in Dou et al. [2007], the query “mouse” may mean “rodents” for biologists, while the programmers may use the same query to search for computer peripherals. Even for queries without ambiguity, for example “online shopping”, some users may prefer “www.amazon.com” while others may prefer “www.ebay.com”.

Understanding users' preference and context helps meet their information needs. Thus, IR systems must adapt the ranking results [Pitkow et al., 2002, Micarelli et al., 2007]. This is referred as personalized search, or search personalization.

There are multiple ways to do search personalization, and two of major directions are normally referred as contextualization [Melucci, 2012] and individualization [Pitkow et al., 2002]. The former is to consider users' conditions in a search activity, for example, time and location. The latter focuses more on users' individual characteristics and activities, which is also described as the users' profile. Topic models have been investigated to model users' preference on the topic space [Song et al., 2010, Carman et al., 2010].

Modeling Users' Preference via the Output Topics Song et al. [2010] apply topic models to model users' preference from users' search history. The idea is very similar to smoothing query language models $p(w | q)$ by topic models, as explained in Equation 2.17. The estimation of $p(w | q)$ is split into two parts $p(w | k)$ and $p(k | q)$.

For each users' query, they concatenate the clicked documents (or the top n ranked documents if no click happened) as a big preference

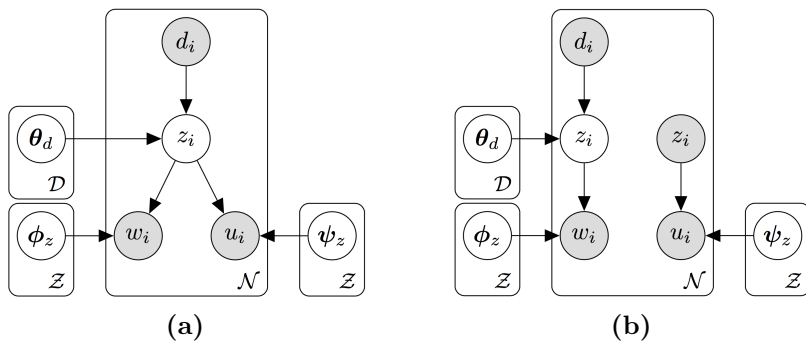


Figure 2.1: The plate diagrams for personalized retrieval in Figure 2.1a and the actual simplified model used for parameter estimation in Figure 2.1b (Both figures from Harvey et al. [2013].)

document. Then the topic model pLSA is applied on the preference collection to extract the latent topics as users’ preference ($p(w | k)$ in the Equation 2.17). The second part is to estimate the query-topic distribution $p(k | q)$ in Equation 2.17. However, the queries are too short to estimate the query topics directly. Instead, they first estimate a language model θ_q from the big preference document of query q and compare the cosine similarity against each topic to estimate the query-topic distribution $p(k | q)$,

$$p(k | q) = \frac{p(k, q)}{\sum_k p(k, q)} \approx \frac{\text{sim}(\theta_k, \theta_q)}{\sum_k \text{sim}(\theta_k, \theta_q)}. \quad (2.23)$$

This can then direct the user to documents that match the user’s interests (e.g., that match those topics well).

Encoding Users into Topic Models Carman et al. [2010] also investigate topic models on large query logs for search personalization and propose a personalization topic model as shown in Figure 2.1a. The idea is given the topic distribution of the document, there will be words chosen at random to generate the query and users who chose to click that document.

As shown in Figure 2.1a (this uses plate diagram formalism, explained in Chapter 1.5.3), this model has three observed variables,

document d_i , query word w_i and user u_i . Given a topic k sampled from a discrete distribution θ_d , the corresponding query words w_i are sampled from a topic-word discrete distribution ϕ_k and the user u_i who submitted the corresponding query is sampled from a topic-user discrete distribution ψ_k .

To estimate the probability of $p(k | w_i, d_i, u_i)$, they further assume the conditional independence among the word w_i , the user u_i and document d_i given the topic k , and the model can be simplified as

$$p(k | w_i, d_i, u_i) = \frac{p(k, w_i, u_i | d_i)}{p(w_i, u_i | d_i)} \propto p(w_i | k)p(u_i | k)p(k | d_i). \quad (2.24)$$

By directly including the user in the topic model, this model assumes the user's topical interests for describing a document that the user clicked is equally important as the words to describe the document. This assumption is too strong [Carman et al., 2010]. As a result, Harvey et al. [2013] further propose to ignore the user during inference and simplify the model (Figure 2.1b). In this model, the topics are used to infer the topic-user distribution $p(u_i | k)$ once the Markov chain is converged. The intuition is to capture the idea that a user clicks on a document given a specific query due to his/her interests expressed over the topic space [Harvey et al., 2013].

Based on the estimates of this personalized topic model, the documents are ranked by the likelihood given the query and the user as follows,

$$\begin{aligned} p(d | q, u) &\propto p(d) \prod_{w \in q} p(w, u | d) \\ &\propto p(d) \prod_{w \in q} \sum_k p(w | k)p(u | k)p(k | d) \end{aligned} \quad (2.25)$$

By subtly incorporating users' profiles as part of the ranking algorithms, Harvey et al. [2013] significantly improved the personalized ranked document lists than the non personalized baselines.

2.5 Summary

Because topic models analyze documents on a semantic level, they offer an interesting and unique framework for modeling relationships between

words, and between documents and words. As a result, topic models have been successfully applied in smoothing language models, query expansion and search personalization.

There is also some work focusing on the diversification of search results [Dang and Croft, 2013, Santos et al., 2015]. The goal of search result diversification is to identify the different aspects of the ambiguous query, retrieve documents for each aspect and make the search results contain more varied documents [Dang and Croft, 2013]. Personalization and diversification are orthogonal, and can be combined in unified models [Vallet and Castells, 2012, Liang et al., 2014].

These successful applications on information retrieval are based on a good understanding of the outputs of topic models. In the next chapter, we will introduce how to visualize, label, and evaluate topics to help users better understand topics and how they are expressed in collections.

3

Evaluation and Interpretation

While the previous chapter focuses on algorithmic uses of topic models, one of the reasons for using topic models is that they produce human-readable summaries of the themes of large document collections. However, for users to use the results of topic models, they must be able to understand the models' output. This depends on model *visualization*, *interaction*, and *evaluation*.

We begin this chapter with a discussion of how best to show individual topics to users. From these foundations, we move to how we can display entire models—with many topics—to users. Finally, we close with how users can provide feedback through these interfaces to detect errors and improve the underlying model.

3.1 Displaying Topics

Recall from the previous chapters that topics are distributions over words; the words with the highest weight in a topic best explain what the topic is about. While the simplest answer—just show the most probable words—is a common solution, there are refinements that can improve

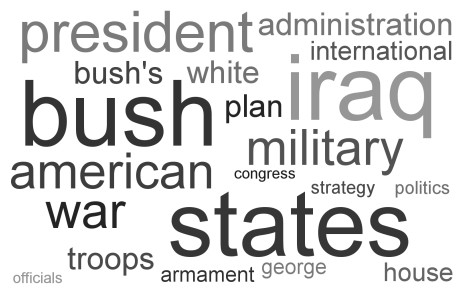


Figure 3.1: Word clouds use a 2D layout to show which words appear in a topic. Word size is related to its probability in the topic, showing which words are more prominent.

a user’s understanding of a collection by showing the relationships between words or explicitly showing words’ probability.

Word Lists Just showing a list of the most common words (a visualization that we will call “word list”) is very simple, and works well. Users can quickly understand how words are arranged, and it is an efficient use of space. Topics have been represented horizontally [Gardner et al., 2010, Smith et al., 2015] or vertically [Eisenstein et al., 2012, Chaney and Blei, 2012], with or without commas separating the individual words, or using set notation [Chaney and Blei, 2012]. Smith et al. [2015] go further by adding bars representing the probabilities of the word.

Word Clouds Word clouds (e.g., Figure 3.1) are another popular approach for displaying topics. Unlike word lists, they also use the size of words to convey additional information. Word clouds typically use the size of words to reflect the probability of the words. This uses more of a given visualization area to be used to display a topic.

However, word clouds have been criticized for providing poor support for visual search [Viégas and Wattenberg, 2008] and lacking contextual information between words [Harris, 2011]; users can sometimes draw false connections between words that are placed next to each other randomly in a word cloud. Another alternative is to use word associations

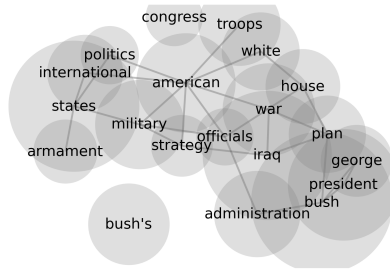


Figure 3.2: A topic-in-a-box visualization for topics—like a word cloud—shows words in a 2D context. However, it uses local co-occurrence (whether words appear together in a sentence) to decide which words to place next to each other.

to set the position of words [Smith et al., 2014]; Figure 3.2 places words that appear together next to each other in the visualization.

3.2 Labeling Topics

Throughout this survey, we have been referring to topics with *labels* such as Information Technology or Arts. These are convenient descriptors, but completely removed from the raw distribution over words. Thus, it is often useful to assign labels to topics within an interface.

In contrast to the previous *visualization* approaches, labeling focuses on showing not the original words of a topic but rather a clearer label more akin to what a human summary of the data would provide.

Approaches for automatic labeling can be divided into those that only use internal information from the topic model against those that also use external knowledge resources. While purely internal methods are more robust and consistent with the philosophy of unsupervised topic models, external resources often produce higher quality labels.

Of the techniques that use external resources, we further separate those that use direct supervision for labeling (i.e., knowing what constitutes a good labeling) from those that use general knowledge resources such as Wikipedia or knowledge bases.

Internal Labeling Mei et al. [2007b] propose an internal labeling method that takes prominent phrases from the topic and compares how consistent the phrase’s context is with the topic distribution. Phrases whose contexts closely resemble the topic often appear in regions of text that summarize the document, making them good candidates for labels. Mao et al. [2012] extend the technique to hierarchies, using the insight that parents’ labels should be consistent with their children’s.

Labeling with Supervised Labels Lau et al. [2010] use a supervised approach to rerank the words in a topic to ensure that the user sees the “best” word in the topic. Each candidate word forms a feature vector consisting of features such as the following:

- the conditional probability of a word given the other words in a topic (which implies topic coherence, as discussed in Chapter 3.4);
- whether the word is a hypernym of other words in the topic (e.g., “dog” in a topic that also contains “terrier” and “poodle”); and
- the original probability of the word in the topic.

While these can be used alone as an unsupervised reranking, Lau et al. [2010] use user-selected best topic words to weight which of these features are most important for selecting the best topic word. These weights are learned using support vector regression. Lau et al. [2011] extend their technique by adding candidates from Wikipedia to the set. The weakness of this approach is that Wikipedia may not have coverage of the topics in the collection; if Wikipedia ignores the theme captured by a topic model, then it will fail to find an appropriate label for that topic.

Labeling with Knowledge Bases Mao et al. [2012] align topic models with an external ontology of labels. They argue that labels should match topic words (as labeling with flat topics); a topic’s words should be consistent with a labels’ children in the hierarchy; and the topic’s labels should be unique.

Aletras et al. [2014] instead query the whole Web and then build a graph that includes the words in the titles of the retrieved webpages.