Yuening Hu, **Jordan Boyd-Graber**, Brianna Satinoff, and Alison Smith. **Interactive Topic Modeling**. *Machine Learning*, 2014, 56 pages.


```
@article{Hu:Boyd-Graber:Satinoff:Smith-2014,
Publisher = {Springer},
Author = {Yuening Hu and Jordan Boyd-Graber and Brianna Satinoff and Alison Smith},
Url = {docs/mlj_2013_itm.pdf},
Journal = {Machine Learning},
Title = {Interactive Topic Modeling},
Volume = {95},
Year = {2014},
Pages = {423--469},
}
```


Links:
- Journal [http://dx.doi.org/10.1007/s10994-013-5413-0]
- Frontend Code [http://www.cs.umd.edu/~ynhu/code/itm-release.zip]
- Backend Code [http://www.cs.umd.edu/~ynhu/code/tree-TM.zip]

Mach Learn (2014) 95:423–469
DOI 10.1007/s10994-013-5413-0

Interactive topic modeling

Yuening Hu · Jordan Boyd-Graber · Brianna Satinoff · Alison Smith

Received: 17 August 2012 / Accepted: 20 August 2013 / Published online: 19 October 2013

Abstract Topic models are a useful and ubiquitous tool for understanding large corpora. However, topic models are not perfect, and for many users in computational social science, digital humanities, and information studies—who are not machine learning experts—existing models and frameworks are often a "take it or leave it" proposition. This paper presents a mechanism for giving users a voice by encoding users' feedback to topic models as correlations between words into a topic model. This framework, interactive topic modeling (ITM), allows untrained users to encode their feedback easily and iteratively into the topic models. Because latency in interactive systems is crucial, we develop more efficient inference algorithms for tree-based topic models. We validate the framework both with simulated and real users.

Keywords Topic models · Latent Dirichlet Allocation · Feedback · Interactive topic modeling · Online learning · Gibbs sampling

Editors: Winter Mason, Jennifer Wortman Vaughan, and Hanna Wallach.

Y. Hu (✉) · B. Satinoff · A. Smith
Computer Science, University of Maryland, College Park, Maryland, USA
e-mail: ynhu@cs.umd.edu

B. Satinoff
e-mail: bsonrisa@cs.umd.edu

A. Smith
e-mail: amsmit@cs.umd.edu

J. Boyd-Graber
iSchool and UMIACS, University of Maryland, College Park, Maryland, USA
e-mail: jbg@umiacs.umd.edu

1 Introduction

Understanding large collections of unstructured text remains a persistent problem.[1] Unlike information retrieval, where users know what they are looking for, sometimes users need to *understand* the high-level themes of a corpus and *explore* documents of interest. Topic models offer a formalism for exposing a collection's themes and have been used to aid information retrieval (Wei and Croft 2006), understand scientific ideas (Hall et al. 2008), and discover political perspectives (Paul and Girju 2010). Topic models have also been applied outside text to learn natural scene categories in computer vision (Li Fei-Fei Perona 2005); discover patterns in population genetics (Shringarpure and Xing 2008); and understand the connection between Bayesian models and cognition (Landauer et al. 2006; Griffiths et al. 2007).

Topic models, exemplified by Latent Dirichlet Allocation (Blei et al. 2003), are attractive because they discover groups of words that often appear together in documents. These are the namesake "topics" of topic models and are multinomial distributions over the vocabulary words.[2] The words which have the highest probability in a topic evince what a topic is about. In addition, each document can be represented as an admixture of these topics, a low-dimensional shorthand for representing what a document is about.

The strength of topic models is that they are unsupervised. They do not require any *a priori* annotations. The only input a topic model requires is the text divided into documents and the number of topics you want it to discover, and there are also models and heuristics for selecting the number of topics automatically (Teh et al. 2006).

This is clearer with an example. A "*technology*"[3] topic has high probability for the words "computer, technology, system, service, site, phone, internet, machine"; a topic about "*arts*" has high probability for the words "play, film, movie, theater, production, star, director, stage". These topics are "good" topics since they have semantic coherence, which means the top words are meaningful enough for users to understand the main theme of this topic. Judging "good" topics is not entirely idiosyncratic; Chang et al. (2009) showed that people mostly agree on what makes a good topic.

However, despite what you see in topic modeling papers, the topics discovered by topic modeling do not always make sense to end users. From the users' perspective, there are often "bad" topics. These bad topics can confuse two or more themes into one topic; two different topics can be (near) duplicates; and some topics make no sense at all. This is a fundamental problem, as even if everything went perfectly, the objective function that topic models optimize does not always correlate with human judgments of topic quality (Chang et al. 2009). A closely related issue is that topic models—with their bag-of-words vision of the world—sometimes simply lack the information needed to create the topics end-users expect.

There has been a thriving thread of research to make topic models better fit users' expectations by adding more and more information to topic models, including modeling perspective (Paul and Girju 2010; Lin et al. 2006), syntax (Gruber et al. 2007;

[1] This work significantly revises, extends, and synthesizes our previous work (Hu et al. 2011; Hu and Boyd-Graber 2012a; Hu and Boyd-Graber 2012b).

[2] To avoid confusion, we distinguish Platonic exemplars between words themselves ("words" or "word types") and instances of those words used in context ("tokens"). We italicize and put topics in quotes: e.g., "*politics*", or "*business*".

[3] Topic models do not name the topics that they discover; for discussion, it's convenient to talk about topics as if they were named. Automatic techniques for labeling (Lau et al. 2011) exist, however. In addition, we italicize the topic names to distinguish these abstractions from the actual words in a topic, which are in quotes.

Wallach 2006), authorship (Rosen-Zvi et al. 2004; Dietz et al. 2007), etc. That there is such an effort also speaks to a weakness of topic models. While topic models have captured the attention of researchers in digital humanities (Nelson 2010; Meeks 2011; Drouin 2011; Templeton 2011) and information retrieval (Wei and Croft 2006), for users who are not machine learning experts, topic models are often a "take it or leave it" proposition. There is no understandable, efficient, and friendly mechanism for improving topic models. This hampers adoption and prevents topic models from being more widely used.

We believe that users should be able to give feedback and improve topic models without being machine learning experts, and that this feedback should be **interactive** and simple enough to allow these non-expert users to craft models that make sense for them. This is especially important for users, such as those in the social sciences, who are interested in using topic models to understand their data (Hopkins 2012; Grimmer 2010), and who have extensive domain knowledge but lack the machine learning expertise to modify topic modeling algorithms. In Sect. 2, we **elaborate our user-centric manifesto** for interactive topic models (ITM) and give two examples of how users can use topic models that adapt to meet their needs and desires.

For example, consider a political scientist who must quickly understand a single US congress's action on "*immigration and refugee issues*" over a two-year congressional term. She does not have the time to read every debate and every bill; she must rely on automated techniques to quickly understand what is being discussed. Topic modeling can partially answer her questions, but political scientists often want to make distinctions that are not easily encoded in topic models. For instance, the Policy Agendas Codebook,[4] a common mechanism for analyzing the topical content of political discourse, separates "*immigration*" into "*migrant and seasonal workers, farm labor issues*" and "*immigration and refugee issues*". If our political scientist is faced with a topic model that conflates "*migrant and seasonal workers, farm labor issues*" and "*immigration and refugee issues*" into a single "*immigration*" topic, her investigation using topic models ends there, and she must turn to—for example—supervised techniques (Boydstun et al. 2013). Interactive topic modeling allows her research with topic models to continue. We will return to this example throughout the paper.

We address the political scientist's dilemma through three major contributions: proposing a mechanism for interactive refinement of topic models, extending Yao et al.'s (2009) SparseLDA to tree-based topic models, and evaluating the efficacy of interactive topic modeling. In addition, we generalize Andrzejewski et al.'s (2009) tree-based topic model to encode more general correlations and propose techniques to automatically suggest constraints for interactive topic modeling. We summarize these contributions further in the following paragraphs, which also outline the article's contents.

First, we need a mechanism for encoding user feedback into interactive topic modeling. We focus on one specific kind of knowledge—the correlations among words within a topic (Petterson et al. 2010)—that can create topics that improve topic coherence (Newman et al. 2009; Newman et al. 2010; Mimno et al. 2011). We focus on tree-structured priors (Boyd-Graber et al. 2007; Andrzejewski et al. 2009; Jagarlamudi and Daumé 2010; Séaghdha and Korhonen 2012), which are appealing because tree-based priors preserve conjugacy, making inference using Gibbs sampling (Heinrich 2004) straightforward. In Sect. 3, we **review tree-structured priors for topic models**, including the generative process, how to model the correlations into the prior of topic models, and the inference scheme with mathematical derivations and discussions.

[4] http://www.policyagendas.org/page/topic-codebook.

Effectively incorporating feedback into models allows users to understand the underlying modeling assumptions. In this influential treatise, *The Design of Everyday Things*, Norman (2002) argues that systems must afford quick, consistent feedback to user inputs for users to build mental models of how complicated systems function. Because the efficiency of inference is so critical to aiding user interactivity (Ceaparu et al. 2004), we **propose an efficient tree-based topic modeling framework**, and show that it dramatically improves the efficiency of inference (Sect. 4).

Once we have encoded the insights and needs from a user, the next step is to support interactivity. In Sect. 5, we describe the mechanics underneath our **interactive topic modeling** framework, which retains aspects of a topic model that a user is satisfied with but relearns the aspects the user is not satisfied with. We investigate several techniques for incorporating user feedback that draw on techniques from online learning by treating user feedback as additional data we must learn from.

We then **examine how users can use these models to navigate and understand datasets** (Sect. 6). We show that our model can accommodate diverse users in a crowd-sourced evaluation on general-interest data and can also help users in a task inspired by computational social science: quickly sifting through political discourse—an attempt to recreate the situation of a political scientist studying "*immigration and refugee issues*" in a laboratory setting.

We additionally **propose techniques to suggest and guide users** during the interactive topic modeling process in Sect. 7. Section 8 concludes with a discussion of how interactive topic modeling can be applied to social science applications, data beyond text, and larger data.

2 How users can benefit from interactive topic models

Users of topic models are the ultimate judge of whether a topic is "good" or "bad". In this section, we discuss what it means to have effective topic models, effective topic modeling software, and how these technologies interact with users. After identifying a set of goals that interactive topic models should fulfill, we present two real scenarios where interactive topic models can help a user cope with a surfeit of data.

Are you a good topic or a bad topic? A user might echo one of the many complaints lodged against topic models: these documents should have similar topics but do not (Daumé 2009); this topic should have syntactic coherence (Gruber et al. 2007; Boyd-Graber and Blei 2008); this topic makes no sense at all (Newman et al. 2010); this topic shouldn't be associated with this document but is (Ramage et al. 2009); these words shouldn't be the in same topic but are (Andrzejewski et al. 2009); or these words should be in the same topic but are not (Andrzejewski et al. 2009).

Many of these complaints can be corrected by encouraging topic models through correlations. Good topics can be encouraged by having correlations that link together words the model separated. A bad topic can be discouraged by correlations that split topic words that should not have been together. This is the approach of Andrzejewski et al. (2009), who used tree-based priors (Boyd-Graber et al. 2007) to encode expert correlations on topic models. A nice property of these correlations is that they form "soft" constraints for the model, which means the results will match users' expectations if and only if the correlations are supported by the underlying statistical model of the text (For an example of when data override correlations, see the "*Macintosh*" vs. "*Windows*" discussion in Sect. 6.2).

Moreover, Andrzejewski et al.'s (2009) assumption of *a priori* correlations is inefficient. Users attempting to curate coherent topics need to see where the topic models go wrong before they can provide useful feedback. This suggests that topic models should be **interactive**—users should be able to see the output, give feedback, and continue refining the output.

Another reason to prefer an interactive process is that users invest effort in understanding the output of a topic model. They might look at every topic to determine which topics are good and which are bad. After a user gives feedback, a model should not throw away the topics the user did not have a problem with. This saves user cognitive load, will lead to a faster end-result by not relearning acceptable topics, and can perhaps improve overall quality by maintaining topics that are acceptable to users.

Additionally, any model involving interaction with users should be as **computationally efficient** as possible to minimize users' waiting time. In particular, Thomas and Cook (2005), summarizing a decade of human-computer interaction research, argue that a system should respond to a user on the order of one second for a simple response, such as clicking a web link, and on the order of ten seconds for a response from a complex user-initiated activity. For an iterative model, this requirement demands that we make inference as efficient as possible and to minimize the total number of iterations.

To summarize, we want a system that can help users who may or may not have prior knowledge of topic modeling, such as a news reporter or political analyst, to obtain or update topics easily, and this system should be:

- Simple enough to allow novices to encode feedback and update topics
- Fast enough to get the updates quickly
- Flexible enough to update the topics iteratively
- "Smart" enough to keep the "good" topics and improve the "bad" topics.

We use Andrzejewski et al.'s (2009) tree-structured correlations to meet the first requirement and combine it with a framework called interactive topic modeling (ITM) to satisfy the other three requirements.

Before discussing the details of how our interactive topic modeling (ITM) works, we begin with two examples of a system that meets the above requirements. While we gloss over the details of our interactive topic modeling system for now, these examples motivate why a user might want an interactive system.

2.1 Example A: joining two topics separated by a naïve topic model

For the first task, we used a corpus of about 2000 New York Times editorials from the years 1987 to 1996. We started by finding 20 initial topics using Latent Dirichlet Allocation (Blei et al. 2003, LDA), as shown in Table 1 (left). In the sequel, we will refer to this model as "vanilla LDA" as it lacks any correlations and embellishments from users.

Topic 1 and Topic 20 both deal with "*Russia*" (broadly construed). Topic 20 is about the "*Soviet Union*", but Topic 1 focuses on the development of the democratic, capitalist "*Russian Federation*" after the collapse of the Soviet Union. This might be acceptable for some users; other users, however, may view both as part of a single historical narrative.

At this point, two things could happen. If the user was not a machine learning expert, they would throw up their hands and assume that topic modeling is not an appropriate solution. A machine learning expert would sit down and come up with a new model that would capture these effects, such as a model where topics change (Wang et al. 2008).

However, both of these outcomes are suboptimal for someone trying to understand what's going on in this corpus *right now*. Our system for creating interactive topic models allows

Table 1 Five topics from 20 topics extracted topic model on the editorials from the New York times before adding a positive correlation (*left*) and after (*right*). After the correlation (words in *red* and *blue* on *left*) was added, which encouraged Russian and Soviet terms to be in the same topic (in *red* and *blue*), non-Russian terms gained increased prominence in Topic 1 (in *green*), and "Moscow" (which was not part of the correlation) appeared in Topic 20 (in *green*) (Color table online)

Topic	Words	Topic	Words
1	election, yeltsin, russian, political, party, democratic, russia, president, democracy, boris, country, south, years, month, government, vote, since, leader, presidential, military	1	election, democratic, south, country, president, party, africa, lead, even, democracy, leader, presidential, week, politics, minister, percent, voter, last, month, years
2	new, york, city, state, mayor, budget, giuliani, council, cuomo, gov, plan, year, rudolph, dinkins, lead, need, governor, legislature, pataki, david	2	new, york, city, state, mayor, budget, council, giuliani, gov, cuomo, year, rudolph, dinkins, legislature, plan, david, governor, pataki, need, cut
3	nuclear, arms, weapon, defense, treaty, missile, world, unite, yet, soviet, lead, secretary, would, control, korea, intelligence, test, nation, country, testing	3	nuclear, arms, weapon, treaty, defense, war, missile, may, come, test, american, world, would, need, lead, get, join, yet, clinton, nation
4	president, bush, administration, clinton, american, force, reagan, war, unite, lead, economic, iraq, congress, america, iraqi, policy, aid, international, military, see	4	president, administration, bush, clinton, war, unite, force, reagan, american, america, make, nation, military, iraq, iraqi, troops, international, country, yesterday, plan
⋮		⋮	
20	soviet, lead, gorbachev, union, west, mikhail, reform, change, europe, leaders, poland, communist, know, old, right, human, washington, western, bring, party	20	soviet, union, economic, reform, yeltsin, russian, lead, russia, gorbachev, leaders, west, president, boris, moscow, europe, poland, mikhail, communist, power, relations

a user to create a positive correlation with all of the clearly "*Russian*" or "*Soviet*" words ({"boris", "communist", "gorbachev", "mikhail", "russia", "russian", "soviet", "union", "yeltsin"}, shown in red and blue in Table 1). This yields the topics in Table 1 (right).[5] The two "*Russia*" topics are combined into Topic 20. This combination also pulled in other relevant words that are not near the top of either topic before: "moscow" and "relations"(green in Topic 20, right). Topic 20 concerns the entire arc of the transition from the "*Soviet Union*" to the "*Russian Federation*", and Topic 1 is now more about "*democratic movements*" in countries other than Russia. The other 18 topics stay almost the same, allowing our hypothetical user to continue their research.

2.2 Example B: splitting a topic with mixed content

The National Institutes of Health (NIH), America's foremost health-research funding agency, also has challenging information needs. They need to understand the research that they are funding, and they use topic modeling as one of their tools (Talley et al. 2011). After running a 700-topic model, an informant from the NIH reported that Topic 318 conflated two distinct concepts: the "*urinary system*" and the "*nervous system*", shown on the left of Table 2.

[5]Technical details for this experiment that will make sense later: this runs inference forward 100 iterations with tree-based topic model (Sects. 3 and 4) and **doc** ablation strategy discussed in Sect. 5.

Table 2 One topic from 700 topics extracted by topic model on the NIH proposals before (*left*) adding a negative correlation (between "bladder" and "spinal_cord") and after (*right*). After the correlation was added to push urinary system terms (*red*) away from the motor nerves terms (*blue*), most urinary terms went away (*red*), and some new terms related with motor nerves appeared (*green*) (Color table online)

Topic	Words	Topic	Words
318	bladder, sci, spinal_cord, spinal_cord_injury, spinal, urinary, urinary_tract, urothelial, injury, motor, recovery, reflex, cervical, urothelium, functional_recovery	318	sci, spinal_cord, spinal_cord_injury, spinal, injury, recovery, motor, reflex, urothelial, injured, functional_recovery, plasticity, locomotor, cervical, locomotion

This was unsatisfactory to the users, as these are discrete systems and should not be combined. To address this error, we added a negative correlation between "bladder" and "spinal_cord" and applied our model to update their results. Then, Topic 318 was changed to a topic about "*motor nerves*" only (as shown on the right of Table 2): in addition to "bladder", other words associated with the urinary system disappeared; the original words related with "spinal_cord" all remained in the same topic; and more related words (in green)—"injured", "plasticity" and "locomotor"—appeared in this topic. The updated topic matched with NIH experts' needs.

These two real-world examples show what is possible with ITM. More specifically, they show that topic models do not always satisfy users' needs; effective topic modeling requires us to provide frameworks to allow users to improve the outputs of topic models.

2.3 Improvement or impatience?

The skeptical reader might wonder if the issues presented above are *problems* that are being solved by interactive topic modeling. It could be merely that users are impatient and are looking at topic models before they are fully converged. If this is the case, then interactive topic modeling is only a placebo that makes users feel better about their model. The result is that they run inference longer, and end up at the same model. Interactive topic models can do more.

From users' perspective, topic models are often used before the models converge: not only because users despise waiting for the results of inference, but also because normal users, non-machine learning experts, lack the intuition and tools to determine whether a model has converged (Evans 2013). Thus interactive topic modeling might encourage them to more effectively "buy in" to the resulting analysis, as users have more patience when they are actively involved in the process (Bendapudi and Leone 2003; Norman 1993). As machine learning algorithms enter mainstream use, it is important not to overlook the **human factors** that connect to usage and adoption.

From models' perspective, interactive topic modeling allows models to **converge faster** than they would otherwise. As we show in Sect. 7.2, interactivity can improve ill-formed topics faster than through additional rounds of inference alone.

In addition, interactive topic models can also allow users to escape from **local minima**. For example, the example from Sect. 2.2 was obtained from an inference run after tens of thousands of iterations that, by all traditional measures of convergence, was the best answer that could be hoped for. By adding correlations, however, we discover topics that are more coherent and escape from local minima (Sect. 2.2).

In the next section, we dig deeper into the modeling assumptions of topic models and describe our new model that makes the interactive topic model refinement possible.

3 Tree-based topic models

Latent Dirichlet Allocation (LDA) discovers topics—distributions over words—that best reconstruct documents as an admixture of these topics. This section begins by reviewing vanilla LDA. Starting with this simple model also has the advantage of introducing terminology we'll need for more complicated topic models later.

Because vanilla LDA uses a symmetric Dirichlet prior for all words, it ignores potential relations between words.[6] This lacuna can be addressed through tree-based topic models, which we review in this section. We describe the generative process for tree-based topic models and show how that generative process can, with appropriately constructed trees, encode feedback from users for interactive topic modeling.

3.1 Background: vanilla LDA

We first review the generative process of vanilla LDA with K topics of V words (Blei et al. 2003 and Griffiths and Steyvers 2004 offer more thorough reviews):

- For each topic $k = \{1 \dots K\}$
  - draw a V-dimensional multinomial distribution over all words: $\pi_k \sim \textbf{Dirichlet}(\beta)$
- For each document d
  - draw a K-dimensional multinomial distribution that encodes the probability of each topic appearing in the document: $\theta_d \sim \textbf{Dirichlet}(\alpha)$
  - for nth token w of this document d
    - draw a topic $z_{d,n} \sim \text{Mult}(\theta_d)$
    - draw a token $w_{d,n}|z_{d,n} = k, \pi \sim \text{Mult}(\pi_k)$

where α and β are the hyperparameters of the Dirichlet distribution. Given observed documents, posterior inference discovers the latent variables that best explain an observed corpus.

However, the generative process makes it clear that LDA does not know what individual words *mean*. Tokens are mere draws from a multinomial distribution. LDA, by itself, has no way of knowing that a topic that gives high probability to "dog" and "leash" should also, all else being equal, also give high probability to "poodle". That LDA discovers interesting patterns in its topic is possible due to document co-occurrence, a consequence of how English (or any natural language) works. Any semantic or syntactic information not captured in document co-occurrence patterns will be lost to LDA.

3.2 Tree-structured prior topic models

To correct this issue, we turn to tree-structured distributions. Trees are a natural formalism for encoding lexical information. WordNet (Miller 1990), a resource ubiquitous in natural language processing, is organized as a tree based on psychological evidence that human lexical memory is also represented as a tree. The structure of WordNet inspired Abney and Light (1999) to use tree-structured distributions to model selectional restrictions. Later, Boyd-Graber et al. (2007) also used WordNet as a tree structure and put it into a fully Bayesian model for word sense disambiguation. Andrzejewski et al. (2009) extended this statistical formalism to encode "must link" (positive correlations) and "cannot link" (negative correlations) correlations from domain experts.

[6] Strictly speaking, all pairs of words have a weak negative correlation when a Dirichlet is a sparse prior. However, the semantics of the Dirichlet distribution preclude applying a negative correlation to **specific** words.

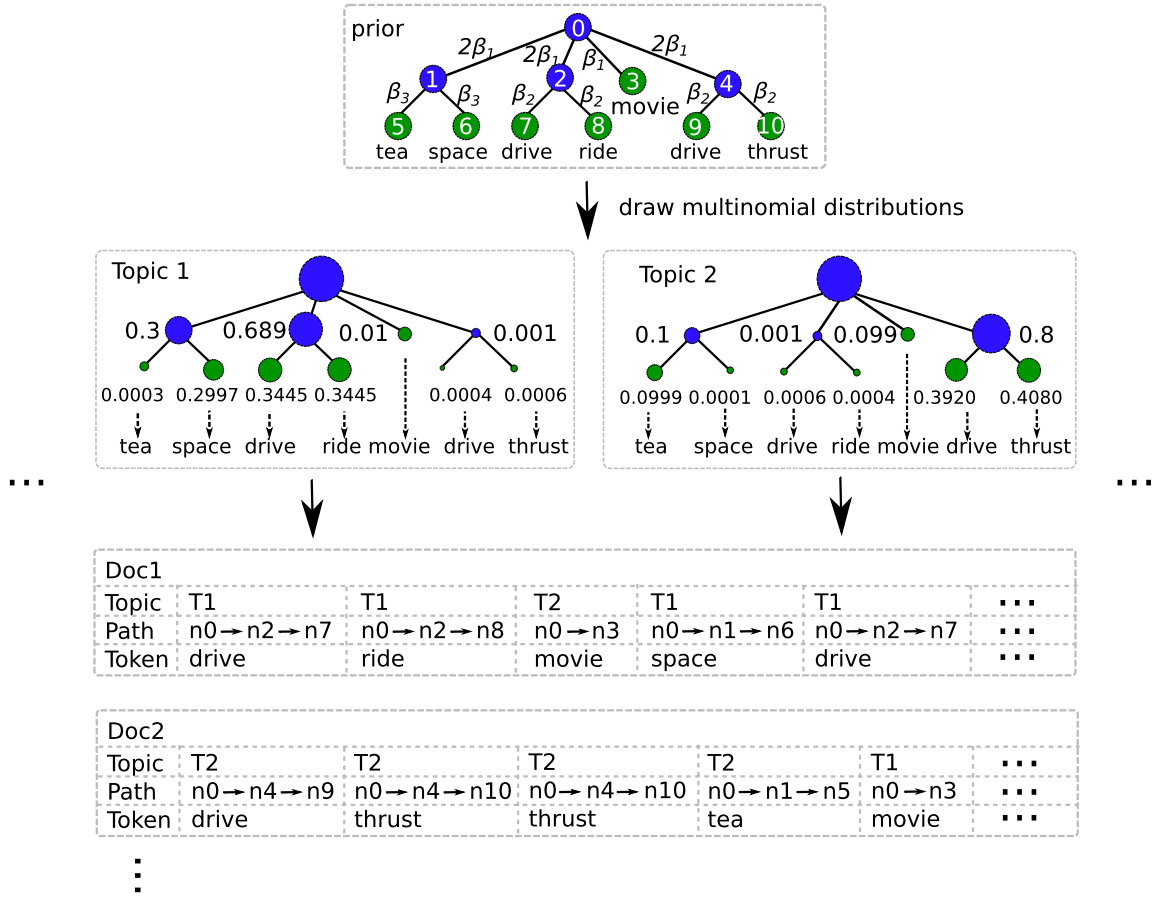

Fig. 1 Example of the generative process for drawing topics (*first row* to *second row*) and then drawing token observations from topics (*second row* to *third row*). In the *second row*, the size of the children nodes represents the probability in a multinomial distribution drawn from the parent node with the prior in the *first row*. Different hyperparameter settings shape the topics in different ways. For example, the node with the children "drive" and "ride" has a high transition prior β_2, which means that a topic will always have nearly equal probability for both (if the edge from the root to their internal node has high probability) or neither (if the edge from the root to their internal node has low probability). However, the node for "tea" and "space" has a small transition prior β_3, which means that a topic can only have either "tea" or "space", but not both. To generate a token, for example the first token of doc1 with topic indicator $z_{0,0} = 1$, we start at the root of topic 1: first draw a child of the root n_0, and assume we choose the child n_2; then continue to draw a child of node n_2, and we reach n_7, which is a leaf node and associated with word "drive"

We adopt Andrzejewski et al. (2009)'s formalism for these two correlations to encode feedback to topic models. The first are positive correlations (PC), which encourage words to appear in the same topic; the second are the negative correlations (NC), which push words into different topics. In the generative story, words in positive correlations have high probability to be selected in the same topic. For example, in Fig. 1, "drive" and "ride" are positively correlated, so they will have similar probabilities in all topics. In Topic 1, the edge going to the parent of "drive" and "ride" has high probability, so both words have similar high probability; in Topic 2, in contrast, the parent has low probability, so both leaves have lower probability. On the other hand, if two words are negatively correlated, when one word has high probability to appear in one topic ("space" in Topic 1 in Fig. 1), the other word turns to unlikely be selected in the same topic ("tea" in Topic 1 in Fig. 1).

These tree-structured distributions replace multinomial distributions for each of the K topics. Instead, we now have a set of multinomial distributions arranged in a tree (we will go deeper and describe the complete generative process that creates these distributions soon).

Each leaf node is associated with a word, and each of the V words must appear as at least (and possibly more than) one leaf node.

One tree structure that satisfies this definition as a very flat tree with only one internal node with V leaf nodes, a child for each word. This is identical to vanilla LDA, as there is only a one V-dimensional multinomial distribution for each topic. The generative process for a token is simple: generate a word according to $w_{d,n} \sim \text{Mult}(\pi_{z_{d,n},\text{root}})$.

Now consider the non-degenerate case. To generate a token $w_{d,n}$ from topic k, we traverse a tree path $l_{d,n}$, which is a list of nodes starting at the root: we start at the root $l_{d,n}[0]$, select a child $l_{d,n}[1] \sim \text{Mult}(\pi_{k,l_{d,n}[0]})$; if the child is not a leaf node, select a child $l_{d,n}[2] \sim \text{Mult}(\pi_{k,l_{d,n}[1]})$; we keep selecting a child $l_{d,n}[i] \sim \text{Mult}(\pi_{k,l_{d,n}[i-1]})$ until we reach a leaf node, and then emit the leaf's associated word. This walk along the tree, which we represent using the latent variable $l_{d,n}$, replaces the single draw from a topic's multinomial distribution over words. The rest of the generative process remains the same as vanilla LDA with θ_d, the per-document topic multinomial, and $z_{d,n}$, the topic assignment for each token. The topic assignment $z_{d,n}$ selects which tree generates a token.

The multinomial distributions themselves are also random variables. Each transition distribution $\pi_{k,i}$ is drawn from a Dirichlet distribution parameterized by β_i. Choosing these Dirichlet priors specifies the direction (i.e., positive or negative) and strength of correlations that appear. Dirichlet distributions produce sparse multinomials when their parameters are small (less than one). This encourages an internal node to prefer few children. When the Dirichlet's parameters are large (all greater than one), it favors more uniform distributions.

An example of this generative process is in Fig. 1. To generate a topic, we draw multinomial distributions for each internal node. The whole tree induces a multinomial distribution over all paths in a topic. For any single node's distribution, the smaller the Dirichlet hyperparameter is, the more sparsity we get among children. Thus, any paths that pass through such a node face a "bottleneck", and only one child will be a likely next step in the path once a path touches a node. For example, if we set β_3 to be very small, "tea" and "space" will not appear together in a topic. Conversely, for nodes with a large hyperparameter, the transition is selected almost as if it were chosen uniformly at random. For example, "drive" and "thrust" are nearly equally likely once a path reaches Node 4.

In Fig. 1, words like "movie" are connected to the root directly, while words like "drive" and "thrust" are connected through an internal node, which indicates the correlation between words. In the next section, we show how these trees can be constructed to reflect the desires of users to make topic modeling more interactive.

3.3 Encoding correlations into a tree

So how do we go from user feedback, encoded as correlations (an unordered set of words with a direction of correlations) to a final tree? Any correlations (either positive or negative) without overlapping words can easily be encoded in a tree. Imagine the symmetric prior of the vanilla LDA as a very flat tree, where all words are children of the root directly with the same prior, as shown in Fig. 2 (top). To encode one correlation, replace all correlated words with a single new child node of the root, and then attach all the correlated words children of this new node. Figure 2 (left middle) shows how to encode a positive correlation between "constitution" and "president".

When words overlap in different correlations, one option is to treat them separately and create a new internal node for each correlation. This creates multiple paths for each word. This is often useful, as it reflects that tokens of the same word in different contexts can have different meanings. For example, when "drive" means a journey in a vehicle, it is associated

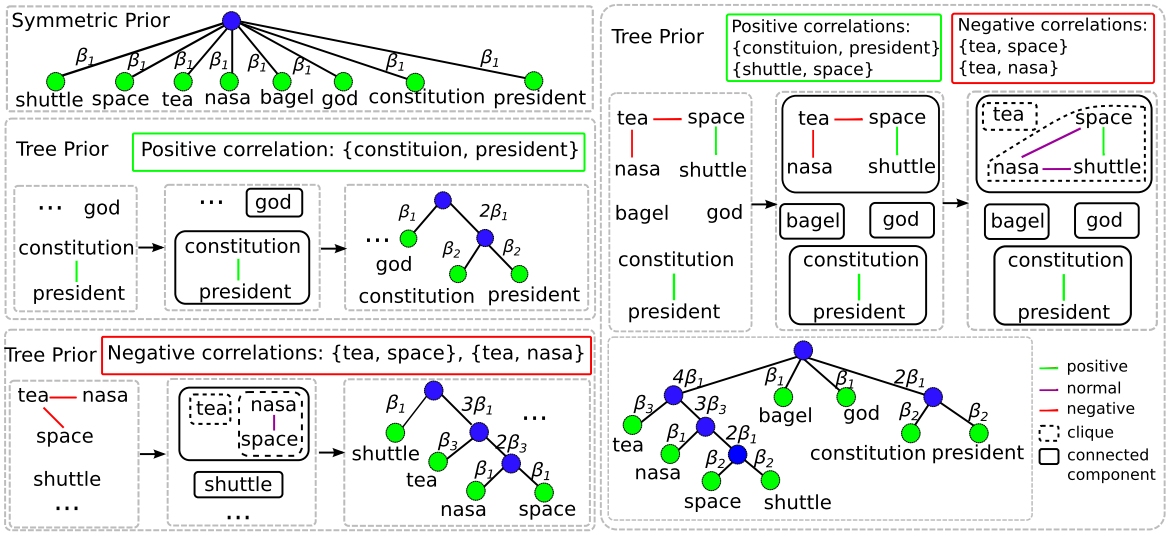

Fig. 2 Given a flat tree (*left top*) as in vanilla LDA (with hyperparameter $\beta = 0.01$ for the uncorrelated words), examples for adding single/multiple positive (*in green*)/negative (*in red*) correlations into the tree: first, generate a graph; detect the connected components; flip the negative edges of components; then detect the cliques; and finally construct a prior tree. *Left middle*: add one positive correlation, connect "constitution" and "president" to a new internal node, and then link this new node to the root, set $\beta_2 = 100$; *Left bottom*: add two negative correlations, set $\beta_3 = 10^{-6}$ to push "tea" away from "nasa" and "space", and use $\beta_1 = 0.01$ as the prior for the uncorrelated words "nasa" and "space". *Right*: two positive correlations and two negative correlations. A positive correlation between "shuttle" and "space" combined with a negative correlation between "tea" and "space" implies a negative correlation between "shuttle" and "tea", so "nasa", "space", "shuttle" will all be pushed away from "tea"; {"space", "shuttle"} and {"constitution", president"} are pulled together by β_2 (Color figure online)

with "ride", but when "drive" refers to the act of applying force to propel something, it is more connected with "thrust". As in lexical resources like WORDNET (Miller 1990), the path of a word in our tree represents its meaning; when a word appears in multiple correlations, this implies that it has multiple meanings. Each token's path represents its meaning.

Another option is to merge correlations. For positive correlations, this means that correlations are transitive. For negative correlations, it is a little more complex. If we viewed negative correlations as completely transitive—placing all words under a sparse internal node—that would mean that only one of the words could appear in a topic. Taken to the illogical extreme where every word is involved in a negative correlation, each topic could only have one word.

Instead, for negative correlations, Andrzejewski et al. (2009) view the negative correlations among words as a **correlation graph**; each word is a node and an edge is a negative correlation a user wants represented by the final tree structure. To create a tree from this graph, we find all the connected components (Harary 1969; Hopcroft and Tarjan 1973) in the graph, and then for each component, we run the Bron and Kerbosch (1973) algorithm to find all the cliques (a clique is a subset of vertices where every two vertices are connected by an edge) on the complement of the component (where nodes without an edge in the primal graph are connected in the complement and *vice versa*). To construct the tree, each component will be a child of the root and each clique will be child of the component. This whole process is shown with examples in Fig. 2 (bottom left): if we want to split "tea" and "space", "tea" and "nasa" at the same time (Fig. 2), one graph component has three nodes for "tea", "space" and "nasa" and two edges between "tea" and "space", "tea" and "nasa". After flipping the edges, there is only one edge left, which is between "space" and "nasa".

So there are two cliques in this component: one includes "space" and "nasa", and the other is "tea". Figure 2 shows an example of going from constraints to tree.

Figure 2 (right) shows a more complex example when there are overlapping words between positive and negative correlations. We first extract all the connected components; for components with negative edges, we do a flip (remove the negative edges, and add all possible normal edges); remove the unnecessary edges: in Fig. 2, there should be a normal edge between "tea" and "shuttle", but "tea" should be away from "space", while "space" and "shuttle" have a positive edge, so the edge between "tea" and "shuttle" is removed; then we detect all the cliques, and construct the prior tree as shown in Fig. 2 (right).

3.4 Relationship to other topic models

Like Pachinko allocation (Li and McCallum 2006) and hierarchical LDA (Blei et al. 2010), tree-structured topic models add additional structure to topics' distributions over words, allowing subsets of the vocabulary to cluster together and expose recurring subtopics. However, tree-structured topic models add structure on individual groups of words (not the entire vocabulary) and are derived from an external knowledge source.

This same distinction separates tree-structured topic models from the Pólya urn model (Mimno et al. 2011), although their goals are similar. While the Pólya urn model does provide additional correlations, these correlations are learned from data and are not restricted to sets of words. Another related model uses Pólya trees (Lavine 1992). Like the tree-based topic models, Pólya tree also provides a prior tree structure, which imposes a Beta prior on the children and generates binary trees. Tree-based topic models generalize the concept by allowing a "bushier" tree structure.

Adding correlations to topic models can come to either the vocabulary distribution (topics) or the document distribution over topics (allocation), as in correlated topic models (Blei and Lafferty 2005; Mimno et al. 2008) and Markov random topic fields (Daumé 2009). While correlated topic models add a richer correlation structure, they have been shown not to improve perceived topic coherence (Chang et al. 2009).

One fundamental advantage of tree-based topic models is that inference using Gibbs sampling remains straightforward because these models retain conjugacy. We develop MCMC for tree-based topic models inference in the next section.

3.5 Inference

We use Gibbs sampling for posterior inference (Neal 1993; Resnik and Hardisty 2010) to uncover the latent variables that best explain observed data. For vanilla LDA, the latent variables of interest are the topics, a document's distribution over topics, and the assignment of each token to a topic.

Gibbs sampling works by fixing all but one latent variable assignment and then sampling that latent variable assignment from the conditional distribution of holding all other variables fixed. For vanilla LDA, a common practice is to integrate out the per-document distribution over topics and the topic distribution over words. Thus, the only latent variable left to sample is the per-token topic assignment (Griffiths and Steyvers 2004),

$$p(z_{d,n} = k|\mathbf{Z}_{-}, w_{d,n}) \propto (\alpha_k + n_{k|d}) \frac{\beta + n_{w_{d,n}|k}}{\beta V + n_{\cdot|k}} \tag{1}$$

where d is the document index, and n is the token index in that document; $n_{k|d}$ is topic k's count in the document d; α_k is topic k's prior; $\mathbf{Z}_{-}$ are the topic assignments excluding the

current token $w_{d,n}$; β is the prior for word $w_{d,n}$[7]; $n_{w_{d,n}|k}$ is the count of tokens with word $w_{d,n}$ assigned to topic k; V is the vocabulary size, and $n_{\cdot|k}$ is the count of all tokens assigned to topic k.

For inference in tree-based topic models, the joint probability is

$$p(\mathbf{Z}, \mathbf{L}, \mathbf{W}, \pi, \theta; \alpha, \beta) = \prod_k \prod_i \underbrace{p(\pi_{k,i}|\beta_i)}_{\text{transition}} \Bigg[\prod_d p(\theta_d|\alpha) \Bigg[\prod_n \underbrace{p(z_{d,n}|\theta_d)}_{\text{assignment}} \underbrace{p(l_{d,n}|\pi, z_{d,n})}_{\text{path}} \underbrace{p(w_{d,n}|l_{d,n})}_{\text{token}}\Bigg]\Bigg], \tag{2}$$

where i is an internal node in the tree. The probability of a path, $p(l_{d,n}|\pi, z_{d,n})$, is the tree structured walk described in Sect. 3.2, and the probability of a token being generated by a path, $p(w_{d,n}|l_{d,n})$ is one if the path $l_{d,n}$ terminates at leaf node with word $w_{d,n}$ and zero otherwise.

Because the conjugate prior of the multinomial is the Dirichlet, we can integrate out the transition distributions π and the per-document topic distributions θ in the conditional distribution

$$\begin{aligned}
&p(z_{d,n} = k, l_{d,n} = \lambda|\mathbf{Z}_-, \mathbf{L}_-, \mathbf{W}; \alpha, \beta) \\
&= \frac{p(z_{d,n} = k, l_{d,n} = \lambda, \mathbf{Z}_-, \mathbf{L}_-, w_{d,n}; \alpha, \beta)}{p(\mathbf{Z}_-, \mathbf{L}_-; \alpha, \beta)} \\
&= p(w_{d,n}|\lambda) \underbrace{\frac{\int_\theta p(z_{d,n} = k, \mathbf{Z}_-|\theta) p(\theta; \alpha) d\theta}{\int_\theta p(\mathbf{Z}_-|\theta) p(\theta; \alpha) d\theta}}_{\text{topic assignment}} \\
&\quad \times \underbrace{\prod_{(i \to j) \in \lambda} \frac{\int_{\pi_{k,i}} p((i \to j) \in \lambda, \mathbf{L}_-|\mathbf{Z}_-, z_{d,n}, \pi_{k,i}) p(\pi_{k,i}; \beta_i) d\pi_{k,i}}{\int_{\pi_{k,i}} p(\mathbf{L}_-|\mathbf{Z}_-, \pi_{k,i}) p(\pi_{k,i}; \beta_i) d\pi_{k,i}}}_{\text{path}} \\
&= \mathbb{I}\big[\Omega(\lambda) = w_{d,n}\big] \underbrace{p(z_{d,n} = k|\mathbf{Z}_-; \alpha)}_{\text{topic assignment}} \underbrace{p(l_{d,n} = \lambda|\mathbf{L}_-, \mathbf{Z}_-, z_{d,n}; \beta)}_{\text{path}}
\end{aligned} \tag{3}$$

where $\mathbf{Z}_-$ are the topic assignments, $\mathbf{L}_-$ are the path assignments excluding the current token $w_{d,n}$, and the indicator function ensures that the path $\lambda_{d,n}$ ends in a path consistent with the token $w_{d,n}$. Using conjugacy, the two integrals—canceling Gamma functions from the Dirichlet normalizer that appear in both numerator and denominator—are

$$p(l_{d,n} = \lambda|\mathbf{L}_-, z_{d,n} = k, w_{d,n}; \beta) = \frac{\prod_{(i,j) \in \lambda} \frac{\Gamma(n_{i \to j|k} + \beta_{i \to j} + 1)}{\Gamma(\sum_{j'} (n_{i \to j'|k} + \beta_{i \to j'}) + 1)}}{\prod_{(i,j) \in \lambda} \frac{\Gamma(n_{i \to j|k} + \beta_{i \to j})}{\Gamma(\sum_{j'} (n_{i \to j'|k} + \beta_{i \to j'}))}} \tag{4}$$

$$p(z_{d,n} = k|\mathbf{Z}_-; \alpha) = \frac{\frac{\Gamma(n_{k|d} + \alpha_k + 1)}{\Gamma(\sum_{k'} (n_{k'|d} + \alpha_{k'}) + 1)}}{\frac{\Gamma(n_{k|d} + \alpha_k)}{\Gamma(\sum_{k'} (n_{k'|d} + \alpha_{k'}))}} \tag{5}$$

[7] Vanilla LDA applies the same prior for all word types, so we omit the subscript and use β instead of β_w in Vanilla LDA.

where $\beta_{i\to j}$ is the prior for edge $i \to j$, and $n_{i\to j|k}$ is the count of the number of paths that go from node i to node j in topic k. All the other terms are the same as in vanilla LDA: $n_{k|d}$ is topic k's count in the document d, and α is the per-document Dirichlet parameter.

With additional cancellations, we can remove the remaining Gamma functions to obtain the conditional distribution[8]

$$p(z = k, l_w = \lambda | \mathbf{Z}_-, \mathbf{L}_-, w) \propto (\alpha_k + n_{k|d}) \prod_{(i\to j)\in\lambda} \frac{\beta_{i\to j} + n_{i\to j|k}}{\sum_{j'} (\beta_{i\to j'} + n_{i\to j'|k})}. \tag{6}$$

The complexity of computing the sampling distribution is $O(KLS)$ for models with K topics, paths at most L nodes long, and at most S paths per word. In contrast, computing the analogous conditional sampling distribution for vanilla LDA has complexity $O(K)$. As a result, tree-based topic models consider correlations between words at the cost of more complex inference. We describe techniques for reducing the overhead of these more expensive inference techniques in the next section.

4 Efficient inference for tree-based topic models

Tree-based topic models consider correlations between words but result in more complex inference. SparseLDA (Yao et al. 2009) is an efficient Gibbs sampling algorithm for LDA based on a refactorization of the conditional topic distribution. However, it is not directly applicable to tree-based topic models. In this section, we first review SparseLDA (Yao et al. 2009) and provide a factorization for tree-based models within a broadly applicable inference framework that improves inference efficiency.

4.1 SparseLDA

The SparseLDA (Yao et al. 2009) scheme for speeding inference begins by rearranging vanilla LDA's sampling equation (1) as

$$\begin{aligned} p(z = k | Z_-, w) &\propto (\alpha_k + n_{k|d}) \frac{\beta + n_{w|k}}{\beta V + n_{\cdot|k}} \\ &= \underbrace{\frac{\alpha_k \beta}{\beta V + n_{\cdot|k}}}_{s_{\text{LDA}}} + \underbrace{\frac{n_{k|d}\beta}{\beta V + n_{\cdot|k}}}_{r_{\text{LDA}}} + \underbrace{\frac{(\alpha_k + n_{k|d}) n_{w|k}}{\beta V + n_{\cdot|k}}}_{q_{\text{LDA}}}. \end{aligned} \tag{7}$$

Following their lead, we call these three terms "buckets". A bucket is the *total* probability mass marginalizing over latent variable assignments (i.e., $s_{\text{LDA}} \equiv \sum_k \frac{\alpha_k \beta}{\beta V + n_{\cdot|k}}$, similarly for the other buckets). The three buckets are: a smoothing-only bucket s_{LDA} with Dirichlet prior α_k and β which contributes to every topic in every document; a document-topic bucket r_{LDA}, which is only non-zero for topics that appear in a document (non-zero $n_{k|d}$); and the topic-word bucket q_{LDA}, which is only non-zero for words that appear in a topic (non-zero $n_w|k$). The three buckets sum to one, so this is a "reshuffling" of the original conditional probability distribution.

[8] In this and future equations we will omit the indicator function that ensures paths end in the required token $w_{d,n}$ by using l_w instead of l. In addition, we omit the subscript $_{d,n}$ from z and l, as all future appearances of these random variables will be associated with single token.

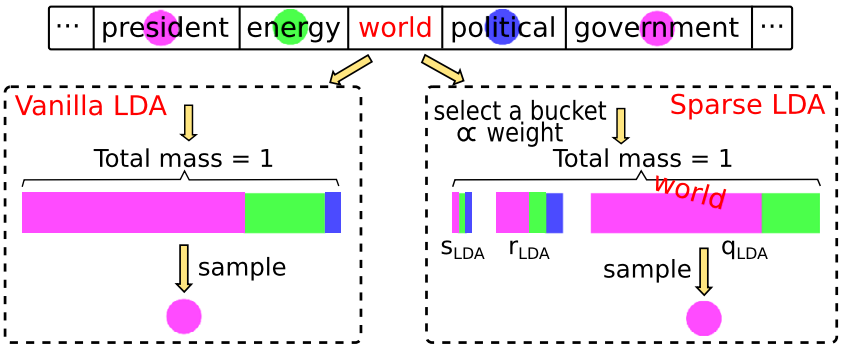

Fig. 3 Comparison of inference between vanilla LDA and SparseLDA: vanilla LDA computes the probability for each topic (sums to 1) and sample a topic; SparseLDA computes the probability for each topic in the three buckets separately (the total is still 1), so it will select a bucket proportional to its weight, and then sample a topic within the selected bucket. Because s_{LDA} is shared by all tokens and r_{LDA} is shared by all tokens in a document, both of them can be cached to save computation. q_{LDA} only includes topics with nonzero count (only the *pink* and *green topics* in this example), which is very sparse in practice, saving computation (Color figure online)

Then the inference is changed to a two-step process: instead of computing the probability mass for each topic, we first compute the probability for each topic in each of the three buckets; then we randomly sample *which* bucket we need and then (and only then) select a topic *within* that bucket, as shown in Fig. 3. Because we are still sampling from the same conditional distribution, this does not change the underlying distribution.

At this point, it might seem counterintuitive that this scheme should improve inference, as we sample from $3K$ (three buckets for each topic) possible outcomes rather than K outcomes as before. However, the probability of a topic within a bucket is often zero in the document-topic r and topic-word q buckets. Only topics that appear in a document have non-zero contributions to the document-topic bucket r, and only (topic, path) pairs assigned to a word have non-zero contributions to the topic-word bucket q for the word. While all topics have non-zero contribution to s, this smoothing-only bucket is typically much smaller (i.e., have smaller probability mass) than the other buckets. We can also efficiently update the bucket total probabilities through constant-time updates (in contrast to $O(K)$ construction of the conditional distribution in traditional LDA sampling schemes). The topic-word bucket q_{LDA} has to be computed specifically for each token, but only for the (typically) few words with non-zero counts in a topic, which is very sparse. Because q_{LDA} often has the largest mass and has few non-zero terms, this speeds inference.

Yao et al. (2009) proposed further speedup by sampling topics within a bucket in *descending* probability. The information needed to compute a probability within a bucket is stored in an array in decreasing order of probability mass. Thus, on average, after selecting one of the three buckets, only a handful of topics need to be explicitly considered. To maintain (topic, count) tuples in sorted order within a bucket more efficiently, the topic and the count are packed into one integer (count in higher-order bits and topic in lower-order bits). Because a count change is only a small shift in the overall ordering, a bubble sort (Astrachan 2003) returns the array to sorted order in $O(n)$.

4.2 Efficient sampling for tree-based topic models

While tree-based topic models are more complicated than vanilla LDA, our model enjoys much of the same sparsity: each topic has a limited number of words seen in a corpus, and each document has only a handful topics. In this section, we take advantage of that

sparsity to extend the sampling techniques for SparseLDA to tree-based topic models. This is particularly important for interactive topic models, as users can be annoyed by even the slightest latency (Nah 2004), and users faced with long wait times may perceive the content to be of a lower quality, have trouble remembering what they were doing or think that an error has occurred (Ceaparu et al. 2004).

To match the form of (7), we first define

$$\begin{aligned} N_{k,\lambda} &= \prod_{(i\to j)\in\lambda} \sum_{j'} (\beta_{i\to j'} + n_{i\to j'|k}) \\ S_\lambda &= \prod_{(i\to j)\in\lambda} \beta_{i\to j} \\ O_{k,\lambda} &= \prod_{(i\to j)\in\lambda} (\beta_{i\to j} + n_{i\to j|k}) - \prod_{(i\to j)\in\lambda} \beta_{i\to j}. \end{aligned} \tag{8}$$

We call $N_{k,\lambda}$ the normalizer for path λ in topic k, S_λ the smoothing factor for path λ, and $O_{k,\lambda}$ the observation for path λ in topic k. Notice $N_{k,\lambda}$ and $O_{k,\lambda}$ are path and topic specific, and S_λ is specific for each path. Then we can refactor (6), yielding buckets analogous to SparseLDA's,

$$\begin{aligned} p(z=k, l=\lambda|Z_-, L_-, w) &\propto (\alpha_k + n_{k|d}) \prod_{(i\to j)\in\lambda} \frac{\beta_{i\to j} + n_{i\to j|k}}{\sum_{j'} (\beta_{i\to j'} + n_{i\to j'|k})} \\ &\propto (\alpha_k + n_{k|d}) N_{k,\lambda}^{-1} [S_\lambda + O_{k,\lambda}] \\ &\propto \underbrace{\frac{\alpha_k S_\lambda}{N_{k,\lambda}}}_{s} + \underbrace{\frac{n_{k|d} S_\lambda}{N_{k,\lambda}}}_{r} + \underbrace{\frac{(\alpha_k + n_{k|d}) O_{k,\lambda}}{N_{k,\lambda}}}_{q}. \end{aligned} \tag{9}$$

$$s \equiv \sum_{k,\lambda} \frac{\alpha_k S_\lambda}{N_{k,\lambda}} \qquad r \equiv \sum_{k,\lambda} \frac{n_{k|d} S_\lambda}{N_{k,\lambda}} \qquad q \equiv \sum_{k,\lambda} \frac{(\alpha_k + n_{k|d}) O_{k,\lambda}}{N_{k,\lambda}} \tag{10}$$

We use the same bucket names without the subscript "LDA" from SparseLDA. Unlike SparseLDA, each bucket sums over the probability of not only the topics but also paths. However, the sampling process is much the same as for SparseLDA: select *which* bucket and then select a topic and path combination *within* the bucket. The resulting algorithm is Algorithm 1. Figure 4 shows a specific example of this proposed inference. However, the correlations introduced by the tree-based structure complicate inference.

One of the benefits of SparseLDA was that s is shared across tokens in a document and thus need not be recomputed. This is no longer possible, as $N_{k,\lambda}$ is distinct for each path in tree-based LDA. This negates the benefit of caching the smoothing-only bucket s, but we recover some of the benefits by caching and updating the normalizer $N_{k,\lambda}$ rather than the bucket s. We split the normalizer to two parts: the "root" normalizer from the root node (shared by all paths) and the "downstream" normalizer,

$$N_{k,\lambda} = \underbrace{\sum_{j'} (\beta_{root\to j'} + n_{root\to j'|k})}_{\text{root normalizer}} \cdot \underbrace{\prod_{(i\to j)\in\lambda'} \sum_{j'} (\beta_{i\to j'} + n_{i\to j'|k})}_{\text{downstream normalizer}} \tag{11}$$

Algorithm 1 EFFICIENT SAMPLING

for token w in this document **do**
  sample $\leftarrow$ rand() $*$ $(s + r + q)$
  if sample $< s$ **then**
    return (k, λ) sampled from s
  sample $\leftarrow$ sample $- s$
  if sample $< r$ **then**
    return (k, λ) sampled from r
  sample $\leftarrow$ sample $- r$
  return (k, λ) sampled from q

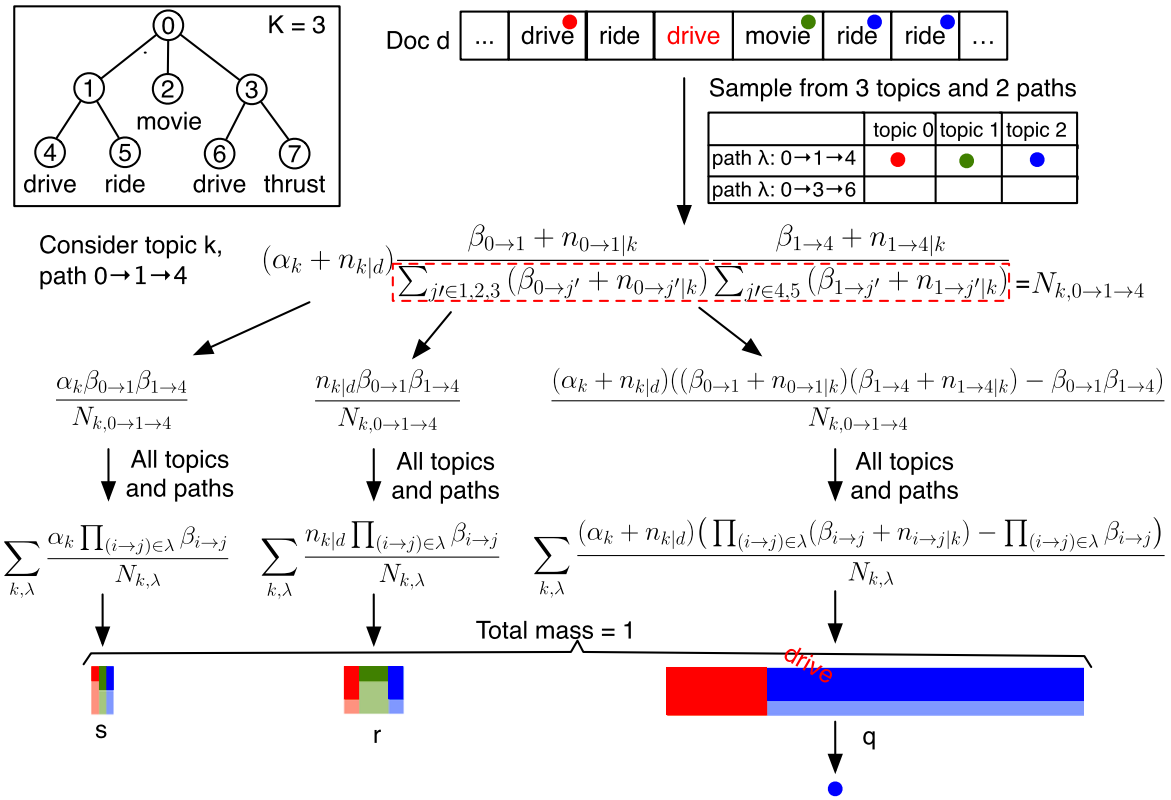

Fig. 4 An example of efficient inference for tree-based topic models: *color* denotes different topics; and the *shade* denotes the paths; like SparseLDA, we need to compute the three buckets, but instead of just considering all topics, we need to consider all topics and paths. First select a bucket proportional to the probability mass, and then sample a topic and a path within the selected bucket. The normalizer N_{k_λ} changes for each path, as s and r are not shared by multiple tokens. Because r only includes the terms where $n_{k|d}$ is non-zero, and q only includes the terms where any of the edges from i to k in the path λ has a non-zero count $n_{i\to j|k}$, which implies the part $\prod_{(i\to j)\in\lambda}(\beta_{i\to j} + n_{i\to j|k}) - \prod_{(i\to j)\in\lambda}\beta_{i\to j}$ is non-zero (only the *red* and *blue topics* in this example). Both are sparse in practice, so it reduces computation time (Color figure online)

where λ' denotes the path excluding the root. The root normalizer only considers the children of the root, and it is shared by all tokens. As a result, we can cache it and update it in constant time. The downstream normalizer considers the remaining part of the normalizer, and it is needed only for correlated words (i.e., words that have been placed in correlations); in many situations it is reasonable to assume that these are relatively few (compared to the overall size of the vocabulary). This normalizer splitting saves memory and improves computation efficiency.

A second problem is that the normalizer $N_{k,\lambda}$ is coupled; changing the transition count $n_{i\to j|k}$ in one path changes the normalizers of all cousin paths (paths that share at least one node i). Take Fig. 2 (left middle) as an example: the paths for "constitution" and "president" are coupled, because they share an edge. When we change the count for each edge along the path of "constitution", the count of the shared edge is changed, so that both downstream normalizers will be changed. For this problem, we precompute which paths share downstream normalizers; all paths are partitioned into cousin sets, defined as sets for which changing the count of one member of the set changes the downstream normalizer of other paths in the set. Thus, when updating the counts for path λ, we only recompute $N_{k,\lambda'}$ for all λ' in the cousin set.

SparseLDA's computation of q, the topic word bucket, benefits from topics with unobserved (i.e., zero count) words. In our case, any non-zero path—a path with *any* non-zero edge—contributes to the probability mass of bucket q (notice a path might have zero path count but non-zero edges). To quickly determine whether a path contributes, we introduce an **edge-masked count** (EMC) for each path. Higher order bits encode whether edges have been observed and lower order bits encode the number of times the path has been observed. For example, in Fig. 2 (left bottom), if we use 8 bits for EMC and observed the path ending in "space" seven times and "nasa" zero times, the EMC for "space" is $\overline{11}100111$, and the EMC for "nasa" is $\overline{11}000000$, since the first two edges of the path ending at "nasa" have been traversed.

4.3 Sorting paths

Encoding the path in this way allows us to extend SparseLDA's sorting strategy to consider latent variable assignments in *decreasing* order of probability mass. Unlike SparseLDA, our latent space is richer; we must sample both a path l and a topic z. Considering fewer possible assignments can speed sampling at the cost of the overhead of maintaining sorted data structures.

Sorting topic and path prominence for a word (ST) can improve our ability to sample from q. If we rank the topic and path pairs for a word in the decreasing order of edge-masked count (EMC), the order serves as a proxy of ranking the topic and path pairs by their probability mass. That is, when sampling a topic and path from q, we sample based on the decreasing EMC, which roughly correlates with path probability. Thus, we will on average choose our sample from the conditional distribution more quickly.

Recall that SparseLDA packed the topic and count into one integer to sort more efficiently. We cannot directly extend this because we need to pack topic, path, and EMC together, and EMC is already a packed integer. Instead, we pack topic and path into one integer, and sort an integer pair (EMC, topic-path integer) together according to the value of EMC.

Using Fig. 2(left bottom) as example, if we use 8 bits for EMC and 8 bits for packing topic and path, and assume we observe the path of "space" (path index 3) seven times and "nasa" (path index 4) zero times in topic 5, the integer pair for "space" is $(\overline{11}100111, \overline{0101}0011)$ and for "nasa" is $(\overline{11}000000, \overline{0101}0100)$. Like SparseLDA, since we only need to update the count by either increasing one or decreasing one, we can use bubble sort to maintain sorted order.

Sorting topics' prominence within a document (SD) can improve sampling from the document-topic bucket r; when we need to sample within a bucket, we consider paths in decreasing order of the document-topic count $n_{k|d}$, so we can identify a topic and path more quickly if the bucket r is selected.

Algorithm 2 EFFICIENT CRB SAMPLING

```
for token w in this document do
    sample ← rand() ∗ (s′ + r + q)
    if sample < s′ then
        compute s
        sample ← (s + r + q)/(s′ + r + q) ∗ sample
        if sample < s then
            return (k, λ) sampled from s
        sample ← sample − s
    sample ← sample − s′
    if sample < r then
        return (k, λ) sampled from r
    sample ← sample − r
    return (k, λ) sampled from q
```

4.4 Efficient sampling with coarse-to-refined buckets

While refactoring and caching the normalizers as described in Sect. 4.2 improves efficiency, the gains are disappointing. This is because while the smoothing only bucket s is small, recomputing it is expensive because it requires us to consider all topics and paths (10). This is not a problem for SparseLDA because s is shared across all tokens.

However, when the counts of each edge per topic are all zero, the prior on bucket s gives an obvious upper bound,

$$s = \sum_{k,\lambda} \frac{\alpha_k \prod_{(i\to j)\in\lambda} \beta_{i\to j}}{\prod_{(i\to j)\in\lambda} \sum_{j'} (\beta_{i\to j'} + n_{i\to j'|k})} \leq \sum_{k,\lambda} \frac{\alpha_k \prod_{(i\to j)\in\lambda} \beta_{i\to j}}{\prod_{(i\to j)\in\lambda} \sum_{j'} \beta_{i\to j'}} = s'. \tag{12}$$

A sampling algorithm can take advantage of this upper bound by not explicitly calculating s, which we call sampling with Coarse-to-Refined Bucket (CRB). Instead, we use a larger s' as proxy, and only compute the smaller refined bucket s if and only if we hit the coarse bucket s' (Algorithm 2). No accuracy is sacrificed for efficiency in this algorithm. As shown in Algorithm 2, when we sample a bucket, if it is not the coarse bucket s', we sample a topic and a path based on the other two buckets (these are always explicitly computed, but their sparsity helps); when we choose the coarse bucket s', we will explicitly compute the refined bucket s and sample based on the correct probabilities. This approximation does not sacrifice accuracy, as we always sample from the true distribution if our sample lands in the approximation gap $s' - s$, but we gain efficiency as samples often do not land in the smoothing bucket s or even in its coarse approximation. This whole process is shown in Fig. 5.

4.5 Measuring inference time efficiency

In this section, we compare the running time[9] of our proposed sampling algorithms FAST and FAST-CRB against the unfactored Gibbs sampler (NAÏVE) and in addition examine the effect of sorting.

[9] Mean of five chains on a 6-Core 2.8-GHz CPU, 16 GB RAM.

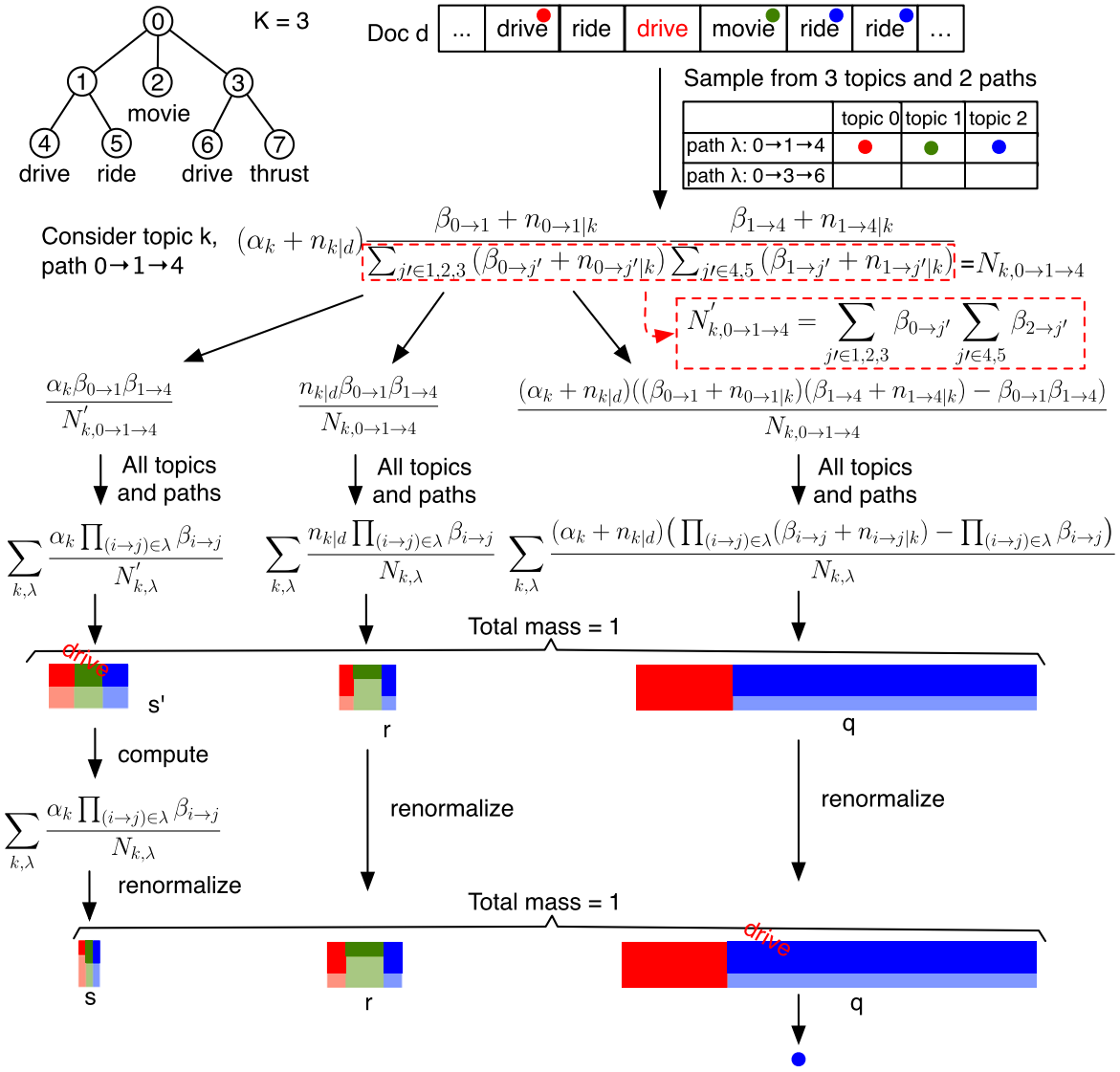

Fig. 5 An example of sampling with coarse-to-refined buckets. Computing the exact smoothing-only s bucket in Fig. 4 needs to go over all topics and paths, which is time-consuming. Instead, we use an upper bound of s initially. We call this the coarse bucket s'; if the current token doesn't land in this coarse bucket, we can just sample a topic and a path in the other two buckets as before; only when the token lands in this coarse bucket do we compute the actual bucket s. We compute the true normalized distribution then resample a topic and a path

The first corpus we use is the 20 Newsgroups corpus (20NEWS),[10] which contains 18770 documents (originally 18846 documents, short documents are deleted) divided into 20 constituent newsgroups, 9743 words, and 632032 tokens. In addition, we use editorials from the New York Times (NYT) from 1987 to 1996, including 13284 documents, 41554 words, and 2714634 tokens.

For both datasets, we rank words by average tf-idf and choose the top V words as the vocabulary. Tokenization, lemmatization, and stopword removal was performed using the Natural Language Toolkit (Bird et al. 2009). WORDNET 3.0 generates the correlations between words. We use WORDNET 3.0 to generate correlations between words. WORDNET organizes words into sets of synonyms called synsets. For each synset, we generate a subtree

[10] http://people.csail.mit.edu/jrennie/20Newsgroups/.

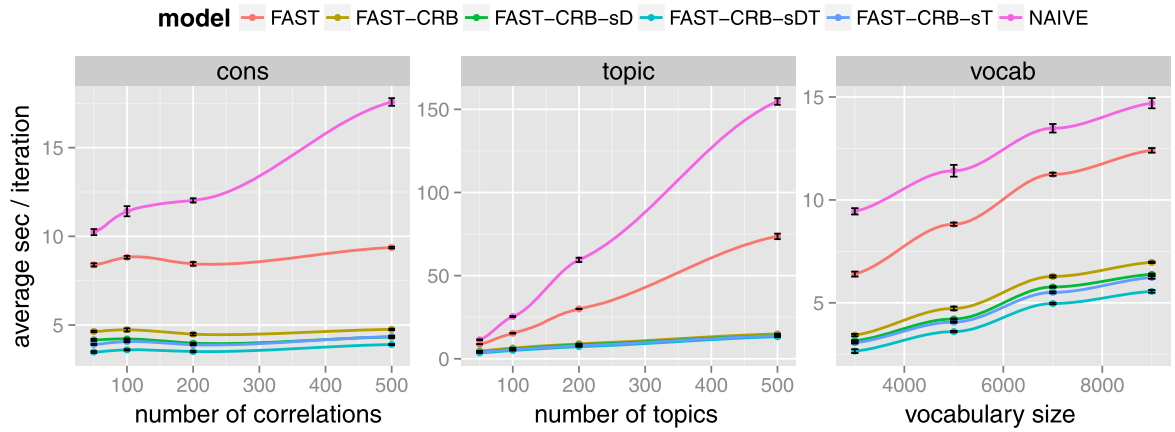

Fig. 6 20 newsgroups' average running time per iteration (Sec) over 100 iterations, averaged over 5 seeds. Experiments begin with 50 topics, 100 correlations, vocab size 5000 and then vary one dimension: number of correlations (*left*), number of topics (*middle*), and vocabulary size (*right*)

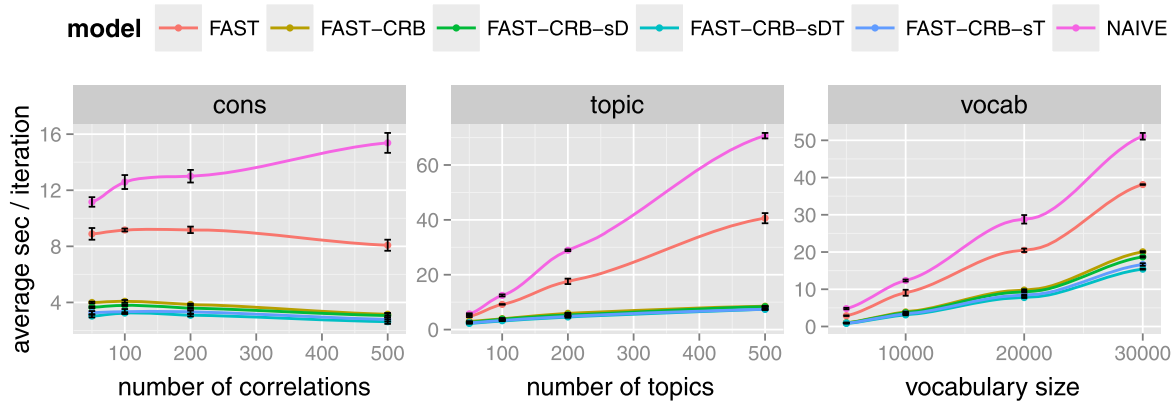

Fig. 7 New York Times' average running time per iteration (Sec) over 100 iterations, averaged over 5 seeds. Experiments begin with 100 topics, 100 correlations, vocab size 10000 and then vary one dimension: number of correlations (*left*), number of topics (*middle*), and vocabulary size (*right*)

with all words in the synset—that are also in our vocabulary—as leaves connected to a common parent. This subtree's common parent is then attached to the root node. The generated correlations include {"drive", "ride", "riding"}, {"drive", "repel"}, etc., which represents three senses of word "drive".

The hyperparameters for all experiments are $\alpha = 0.1$, $\beta = 0.01$ for uncorrelated words, $\beta = 100$ for positive correlations and $\beta = 10^{-6}$ for negative correlations. However, sampling hyperparameters often (but not always) undoes the correlations (by making β for correlations comparable to β for uncorrelated words), so we keep the hyperparameters fixed.

We compared the FAST and FAST-CRB against NAÏVE (Figs. 6 and 7) on different numbers of topics, various vocabulary sizes and different numbers of correlations. For both datasets, FAST is consistently faster than NAÏVE and FAST-CRB is consistently faster than FAST. Their benefits are clearer as distributions become sparse (e.g., the first iteration for FAST is slower than later iterations). Gains grow as the topic number increases, but diminish with larger vocabulary size. While both sorting strategies reduce time, sorting topics and paths for a word (ST) helps more than sorting topics in a document (SD), and combining the two is (with one exception) better than either alone.

Although 20NEWS is smaller than NYT, inference on 20NEWS is slower than on NYT for different number of topics and correlations. This is because NYT has many words with

Table 3 The total number of non-zero paths for correlated words averaged over the number of tokens with correlated words (*first row*), and the same value for uncorrelated words (*second row*), as the number of correlations increases. When the number of correlations increases, the averaged value for correlated words doesn't change much while the averaged value for uncorrelated words decreases. It is because the number of non-zero paths for uncorrelated words decreases as more correlations are added to the model

	C50	C100	C200	C500
Correlated	1.306	1.494	1.904	1.735
Uncorrelated	14.419	14.294	13.858	11.516

high tf-idf score but low frequency. When we filter the dataset using the vocabulary ranked by tf-idf, a lot of high frequency words are filtered out, resulting in fewer remaining tokens in NYT than in 20NEWS. In addition, 20NEWS has many more words with multiple paths, and this sometimes prevents the techniques of this section from speeding inference.

As more correlations are added, NAÏVE's time increases while FAST-CRB's decreases on the NYT dataset (Fig. 7). This is because the number of non-zero paths for uncorrelated words decreases as more correlations are added to the model. Since our techniques save computation for every zero path, the overall computation decreases as correlations push uncorrelated words to a limited number of topics (Table 3).

5 Interactive topic modeling

In this section, we describe interactive topic modeling, which combines the efficient tree-based topic model described in the previous section, with machine learning techniques to incorporate user feedback in an interactive exploration of a corpus. This framework will allow our hypothetical political scientist, attempting to understand "*immigration and refugee issues*", to find the information she needs in large corpora.

5.1 Making topic models interactive

As we argued in Sect. 2, there is a need for interactive topic models. Traditional topic models do not offer a way for non-experts to tweak the models, and those that do are "one off" interactions that preclude fine-grained adjustments and tweaks that solve users' problems but leave the rest of the model unmolested. This section proposes a framework for interactive topic refinement, interactive topic modeling (ITM).

Figure 8 shows the process at a high level. Start with vanilla LDA (without any correlations), show users topics, solicit feedback from users, encode the feedback as correlations between words, and then do topic modeling with the corresponding tree-based prior. This process can be repeated until users are satisfied.

Since it takes some effort for users to understand the topics and figure out the "good" topics and "bad" topics, to save users' effort and time, ITM should be smart enough to remember the "good" topics while improving the "bad" topics. In this section, we detail how interactively changing correlations can be accommodated in ITM.

A central tool that we will use is the strategic unassignment of states, which we call ablation (distinct from feature ablation in supervised learning). The state of a Markov Chain in MCMC inference stores the topic assignment of each token. In the implementation of a Gibbs sampler, unassignment is done by setting a token's topic assignment to an invalid topic (e.g., -1, as we use here) and decrementing any counts associated with that token.

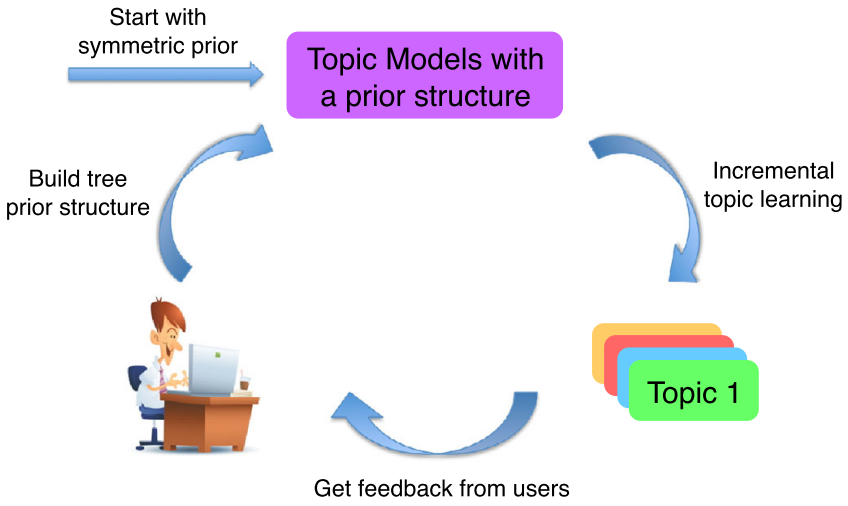

Fig. 8 Interactive topic modeling: start with a vanilla LDA with symmetric prior, get the initial topics. Then repeat the following process till users are satisfied: show users topics, get feedback from users, encode the feedback into a tree prior, update topics with tree-based LDA

	Previous	New
All	[bark:2, dog:3, leash:3 dog:2] [bark:2, bark:2, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]	[bark:-1, dog:-1, leash:-1 dog:-1] [bark:-1, bark:-1, plant:-1, tree:-1] [tree:-1,play:-1,forest:-1,leash:-1]
Doc	[bark:2, dog:3, leash:3 dog:2] [bark:2, bark:2, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]	[bark:-1, dog:-1, leash:-1 dog:-1] [bark:-1, bark:-1, plant:-1, tree:-1] [tree:2,play:2,forest:1,leash:2]
Term	[bark:2, dog:3, leash:3 dog:3] [bark:2, bark:2, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]	[bark:-1, dog:-1, leash:3 dog:-1] [bark:-1, bark:-1, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]
None	[bark:2, dog:3, leash:3 dog:2] [bark:2, bark:2, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]	[bark:2, dog:3, leash:3 dog:2] [bark:2, bark:2, plant:2, tree:3] [tree:2,play:2,forest:1,leash:2]

Fig. 9 Four different strategies for state ablation after the words "dog" and "bark" are added to the correlation {"leash", "puppy"} to make the correlation {"dog", "bark", "leash", "puppy"}. The state is represented by showing the current topic assignment after each word (e.g. "leash" in the first document has topic 3, while "forest" in the third document has topic 1). On the *left* are the assignments before words were added to correlations, and on the *right* are the ablated assignments. Unassigned tokens are given the new topic assignment −1 and are highlighted in *red* (Color figure online)

The correlations created by users implicitly signal that the model put certain words in the wrong place. In other models, this input is sometimes used to "fix", i.e., deterministically hold constant topic assignments (Ramage et al. 2009). Instead, we change the underlying model, using the current topic assignments as a starting position for a new Markov chain with some states strategically unassigned. How much of the existing topic assignments we use leads to four different options, which are illustrated in Fig. 9.

An equivalent (and equally important) way to think about how ablation works is as technique to handle the inertia of inference. Inference schemes for topic models can become caught in local optima (Sect. 2.3); because of the way topic models are used, users can often diagnose these local optima. Ablation allows the errors that trap inference in local optima to be forgotten, while retaining the unobjectionable parts of the model. Without ablation,

Algorithm 3 UNASSIGN(doc d, token w)

Get the topic of token w: k
Update topic count: $n_{k|d} \leftarrow n_{k|d} - 1$
for path λ of w in previous prior tree **do**
  for edge e of path λ **do**
    Update edge count: $n_{e|k} \leftarrow n_{e|k} - 1$
Forget the topic of token w

Algorithm 4 MOVE(doc d, token w)

Get the topic of token w: k
for path λ' of w in previous prior tree **do**
  for edge e' of path λ' **do**
    Update edge count: $n_{e'|k} \leftarrow n_{e'|k} - 1$
for path λ of w in current prior tree **do**
  for edge e of path λ **do**
    Update edge count: $n_{e|k} \leftarrow n_{e|k} + 1$

inertia would keep inference trapped in a local optimum. We now describe several candidate ablation strategies: all, doc, term, and none.

All We could revoke all state assignments, essentially restarting the sampler from scratch. This does not allow *interactive* refinement, as there is nothing to enforce that the new topics will be in any way consistent with the existing topics. Once the topic assignments of all states are revoked, all counts will be zero, retaining no information about the state the user observed.

Doc Because topic models treat the document context as exchangeable, a document is a natural context for partial state ablation. Thus if a user adds a set of words $\boldsymbol{S}$ to correlations, then we have reason to suspect that all documents containing any one of $\boldsymbol{S}$ may have incorrect topic assignments. This is reflected in the state of the sampler by performing the UNASSIGN (Algorithm 3) operation for each token in any document containing a word added to a correlation.

Term Another option is to perform ablation only on the topic assignments of tokens which have been added to a correlation. This applies the unassignment operation (Algorithm 3) only to tokens whose corresponding word appears in added correlations (i.e. a subset of the **Doc** strategy). This makes it less likely that other tokens in similar contexts will follow the words explicitly included in the correlations to new topic assignments.

None The final option is to move words into correlations but keep the topic assignments fixed, as described in Algorithm 4. This is arguably the simplest option, and in principle is sufficient, as the Markov chain should find a stationary distribution regardless of the starting position. However, when we "move" a token's count (Algorithm 4) for word that changes from uncorrelated to correlated, it is possible that there is a new ambiguity in the latent state: we might not know the path. We could either merge the correlations to avoid this problem (as discussed in Sect. 3.3), restricting each token to a unique path, or sample a new path. These issues make this ablation scheme undesirable.

The **Doc** and **Term** ablation schemes can be both viewed as online inference (Yao et al. 2009; Hoffman et al. 2010). Both of them view the correlated words or some documents as unseen documents and then use the previously seen documents (corresponding to the part of the model a user was satisfied with) in conjunction with the modified model to infer the latent space on the "new" data. Regardless of what ablation scheme is used, after the state of the Markov chain is altered, the next step is to actually run inference forward, sampling assignments for the unassigned tokens for the "first" time and changing the topic assignment of previously assigned tokens. How many additional iterations are required after adding correlations is a delicate tradeoff between interactivity and effectiveness, which we investigate further in Sect. 6.

The interactive topic modeling framework described here fulfills the requirements laid out in Sect. 2: it is simple enough that untrained users can provide feedback and update topics; it is flexible enough to incorporate that feedback into the resulting models; and it is "smart" enough—through ablation—to retain the good topics while correcting the errors identified by users. Interactive topic modeling could serve the goals of our hypothetical political scientist to explore corpora to identify trends and topics of interest.

6 Experiments

In this section, we describe evaluations of our ITM system. First, we describe fully automated experiments to help select how to build a system that can learn and adapt from users' input but also is responsive enough to be usable. This requires selecting ablation strategies and determining how long to run inference after ablation (Sect. 6.1).

Next, we perform an open-ended evaluation to explore what untrained users do when presented with an ITM system. We expose our system to users on a crowd-sourcing platform, and explore users' interactions, and investigate what correlations users create to cultivate topics interactively (Sect. 6.2).

Our final experiment simulates the running example of a political scientist attempting to find and understand "*immigration and refugee issues*" in a large legislative corpus. We compare how users—armed with either ITM or vanilla topic models—use these to explore a legislative dataset to answer questions about immigration and other political policies.

6.1 Simulated users

In this section, we use the 20 Newsgroup corpus (20NEWS) introduced in Sect. 4.5. We use the default split for training and test set, and the top 5000 words are used in the vocabulary.

Refining the topics with ITM is a process where users try reconcile mental models with the themes discovered by topic models. In this experiment, we posit that the users' mental model is defined by the twenty newsgroups that comprise the dataset, e.g. "*politics*", "*atheism*", or "*baseball*". These topics have natural associations with words. For example, the words "government" and "president" for "*politics*", and "skeptic" and "reason" for "*atheism*". As a user encounters more data, their mental models will become more defined; they may only have a handful of words in mind initially but will gather more words as they're exposed to data.

We can simulate these mental lexicons by extracting words from the 20NEWS dataset. For each newsgroup, we rank words with high information gain (IG)[11] for each category. We

[11] Computed by Rainbow toolbox, http://www.cs.umass.edu/~mccallum/bow/rainbow/.

then simulate the process of building more precise mental models by gradually adding more words with high IG.

Sorting words by information gains discovers words that should be correlated with a newsgroup label. If we believe that vanilla LDA lacks these correlations (because of a deficiency of the model), topics that have these correlations should better represent the collection. Intuitively, these words represent a user thinking of a concept they believe is in the collection (e.g., "*christian*") and then attempting to think of additional words they believe should be connected to that concept.

For the 20NEWS dataset, we rank the top 200 words for each class by IG, and delete words associated with multiple labels to prevent correlations from merging. The smallest class had 21 words remaining after removing duplicates (due to high overlaps of 125 overlapping words between "*religion.misc*" and "*christian*", and 110 overlapping words between "*religion.misc*" and "*alt.atheism*"), so the top 21 words for each class were the ingredients for our simulated correlations. For example, for the class "*christian*", the 21 correlated words include "catholic, scripture, resurrection, pope, sabbath, spiritual, pray, divine, doctrine, orthodox ". We simulate a user's ITM session by adding a word to each of the twenty positive correlations until each of the correlations has twenty-one words.

We evaluate the quality of the topic models through an extrinsic classification task. We represent a document's features as the topic vector (the multinomial distribution θ in Sect. 3) and learn a mapping to one of the twenty newsgroups using a supervised classifier (Hall et al. 2009). As the topics form a better low-dimensional representation of the corpus, the classification accuracy improves.

Our goal is to understand the phenomena of ITM, not classification, so the classification results are well below state of the art. However, adding interactively selected topics to state of the art features (tf-idf unigrams) gives a relative error reduction of 5.1 %, while adding topics from vanilla LDA gives a relative error reduction of 1.1 %. Both measurements were obtained without tuning or weighting features, so presumably better results are possible.

We set the number of topics to be the same as the number of categories with the goal of the final twenty topics capturing the "user's" desired topics and hope the topics can capture the categories as well as additional related information. While this is not a classification task, and it is not directly comparable with state of the art classifiers like SVM, we expect it performs better than the **Null** baseline, which is proved by Figs. 10 and 11.

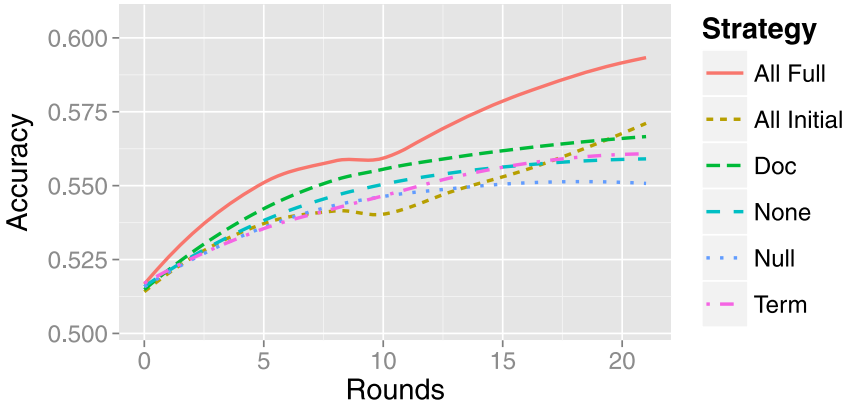

Fig. 10 Accuracy (*y-axis*) using different ablation strategies as additional correlations are added (*x-axis*). We start with 100 iterations, then for each round, add one more word for each of the 20 positive correlations, and run 10 additional iterations. **Null** represents standard LDA, as the lower baseline. **All Initial** and **All Full** are non-interactive baselines, and **All Full** is the upper baseline. The results of **None**, **Term**, **Doc** are more stable (as denoted by the bars), and the accuracy is increased gradually as more correlated words are added

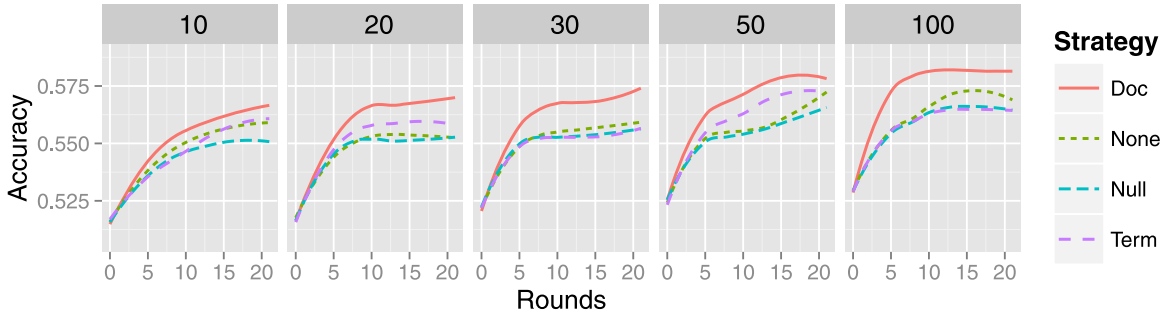

Fig. 11 Classification accuracy by strategy and number of iterations between rounds. We start with 100 iterations, then for each round, add one more word for each of the 20 positive correlations, and run additional 10 iterations. The **Doc** ablation strategy performs best, suggesting that the document context is important for ablation correlations. While more iterations are better, there is a tradeoff with interactivity

This experiment is structured as a series of rounds. Each round adds an additional correlation for each newsgroup (thus twenty words are added to the correlations per round, one per newsgroup). After a correlation is added to the model, we ablate topic assignments according to one of the strategies described in Sect. 5.1, run inference for some number of iterations, extract the new estimate of the per-document topic distribution, learn a classifier on the training data, and apply that classifier to the test data. We do 21 rounds in total, and the following sections investigate the choice of number of iterations and ablation strategy. The number of LDA topics is set to 20 to match the number of newsgroups. The hyperparameters for all experiments are $\alpha = 0.1$, $\beta = 0.01$ for uncorrelated words, $\beta = 100$ for positive correlations and $\beta = 10^{-6}$ for negative correlations.

We start the process after only 100 iterations of inference using a vanilla LDA model. At 100 iterations, the chain has not converged, but such small numbers of iterations is a common practice for impatient users initially investigating a dataset (Evans 2013; Carbone 2012).[12] After observing initial topics, the user then gradually updates the topics, allowing inference to continue.

Moreover, while the patterns shown in Fig. 11 were broadly consistent with larger numbers of iterations, such configurations sometimes had too much inertia to escape from local extrema. More iterations make it harder for the correlations to influence the topic assignment, another reason to start with smaller numbers of initial iterations.

Investigating ablation strategies First, we investigate which ablation strategy best incorporates correlations. Figure 10 shows the classification accuracy of six different ablation strategies for each of 21 rounds. Each result is averaged over five different chains using 10 additional iterations of Gibbs sampling per round (other numbers of iterations are discussed in Sect. 6.1). As the number of words per correlation increases, the accuracy increases as models gain more information about the classes.

To evaluate whether our model works better, we first compare our model against a baseline without any correlations. This is to test whether the correlations help or not. This baseline is called **Null**, and it runs inference for a comparable number of iterations for fair comparison. While **Null** sees no correlations, it serves as a lower baseline for the accuracy but shows the effect of additional inference. Figure 10 shows that the **Null** strategy has a lower accuracy than interactive versions, especially with more correlations.

[12] A machine learning purist would argue that such usage is incorrect, as you only want samples from a converged Markov chain. Without commenting on this debate, this experiment reflects the reality of how topic models are used for analyzing text.

We also compare our model with non-interactive baselines: **All Initial** and **All Full** with all correlations known *a priori*. **All Initial** runs the model for the only the initial number of iterations (100 iterations in this experiment), while **All Full** runs the model for the total number of iterations added for the interactive version. (That is, if there were 21 rounds and each round of interactive modeling added 10 iterations, **All Full** would have 210 iterations more than **All Initial**). **All Full** is an upper baseline for the accuracy since it both sees the correlations at the beginning and also runs for the maximum number of total iterations. **All Initial** sees the correlations before the other ablation techniques but it has fewer total iterations.

In Fig. 10, both **All Initial** and **All Full** show a larger variance (as denoted by bands around the average trends) than the interactive schemes. This can be viewed as akin to simulated annealing, as the interactive settings have more freedom to explore in early rounds. For topic models with **Doc** or **Term** ablation, this freedom is limited to only correlated words or words related with correlated words. Since the model is less free to explore the entire space, these ablation strategies result in much lower variance.

All Full has the highest accuracy; this is equivalent to where users know all correlations *a priori*. This strategy corresponds to an omniscient and infinitely patient user. Neither of these properties are realistic. First, it is hard for users to identify and fix all problems at once. Often smaller problems are not visible until larger problems have been corrected. This requires multiple iterations of inspection and correction. Second, this process requires a much longer waiting time, as all inference must be rerun from scratch after every iteration.

The accuracy of each interactive ablation strategy is (as expected) between the lower and upper baselines. Generally, the correlations will influence not only the topics of the correlated words, but also the topics of the correlated words' context in the same document. **Doc** ablation gives more freedom for the correlations to overcome the inertia of the old topic distribution and move towards a new one influenced by the correlations.

How many iterations do users have to wait? For a fixed corpus and computational environment, the number of iterations is the primary factor that determines how long a user has to wait. While more iterations can get closer to convergence, it also implies longer waiting time. So we need to balance convergence and waiting time.

Figure 11 shows the effect of using different numbers of Gibbs sampling iterations between rounds. For each of the ablation strategies, we run 10, 20, 30, 50, 100 additional Gibbs sampling iterations. As expected, more iterations increase accuracy, although improvements diminish beyond 100 iterations. With more correlations, additional iterations help less, as the model has more *a priori* knowledge to draw upon.

For all numbers of additional iterations, while the **Null** serves as the lower baseline for accuracy in all cases, the **Doc** ablation clearly outperforms the other ablation schemes, consistently yielding a higher accuracy. Thus, there is a benefit when the model has a chance to relearn the document context when correlations are added, and **Doc** provides the flexibility for topic models to overcome the inertia of the old topic distribution but does not throw away the old distribution entirely. The difference is greater with more iterations, suggesting **Doc** needs more iterations to "recover" from unassignment.

The number of additional iterations per round is directly related to users' waiting time. According to Fig. 11, more iterations for each round achieves higher accuracy, while increasing wait time. This is a tradeoff between latency and model quality, and may vary based on users, applications, and data.

However, the luxury of having hundreds or thousands of additional iterations for each correlation would be impractical. For even moderately sized datasets, even one iteration per

second can tax the patience of individuals who want to use the system interactively. Studies have shown that a long waiting time may affect cognitive load, making it harder for a user to recall what they were doing or the context of the initial task (Ceaparu et al. 2004). Based on these results and an *ad hoc* qualitative examination of the resulting topics, we found that 30 additional iterations of inference was acceptable; this is used in later experiments, though this number can vary in different settings.

6.2 Users in loop

To move beyond using simulated users adding the same words regardless of what topics were discovered by the model, we needed to expose the model to human users. We solicited approximately 200 judgments from Mechanical Turk, a popular crowd-sourcing platform that has been used to gather linguistic annotations (Snow et al. 2008), measure topic quality (Chang et al. 2009; Stevens et al. 2012), and supplement traditional inference techniques for topic models (Chang 2010). After presenting our interface for collecting judgments, we examine the results from these ITM sessions both quantitatively and qualitatively.

Figure 12 shows the interface used in the Mechanical Turk tests. The left side of the screen shows the current topics in a scrollable list, with the top 30 words displayed for each topic.

Users create correlations by clicking on words from the topic word lists. The word lists use a color-coding scheme to help the users keep track of which words are already in correlations. The right side of the screen displays the existing correlations. Users can click on

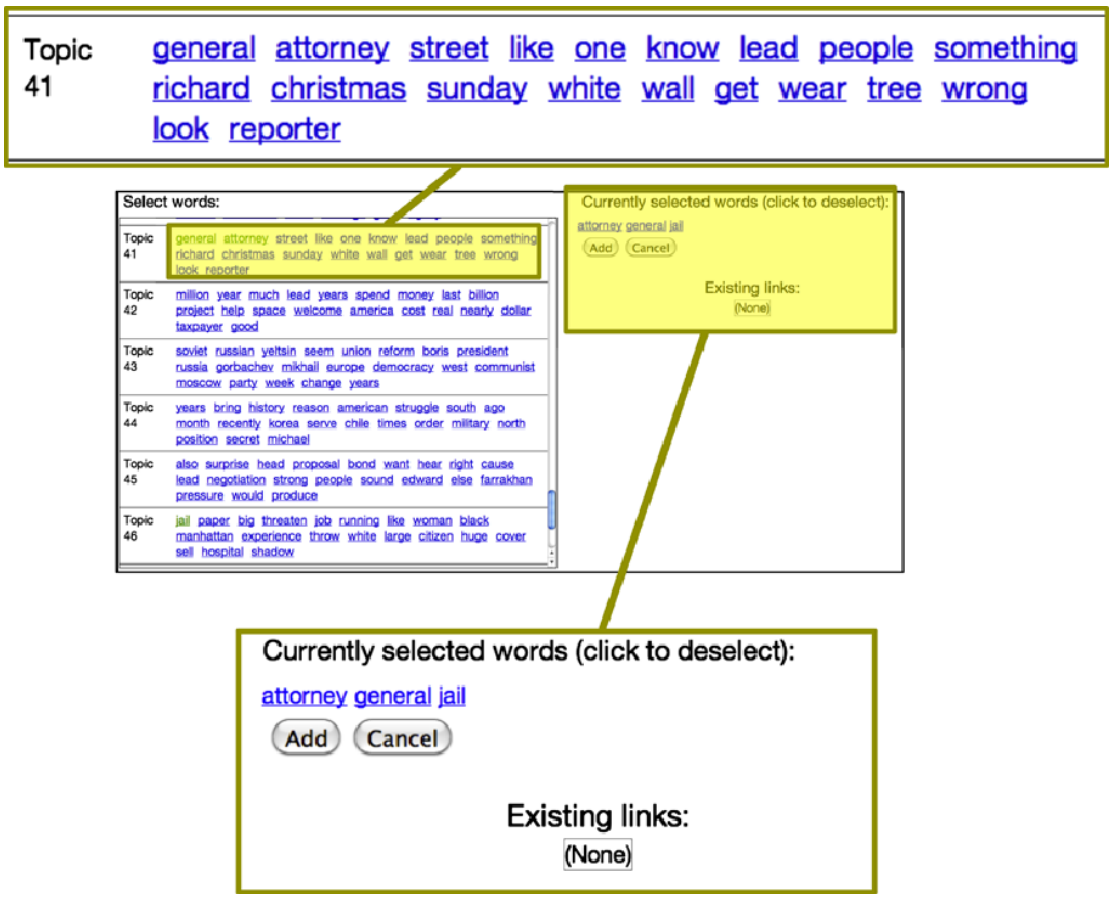

Fig. 12 Interface for Mechanical Turk experiments. Users see the topics discovered by the model and select words (by clicking on them) to build correlations to be added to the model

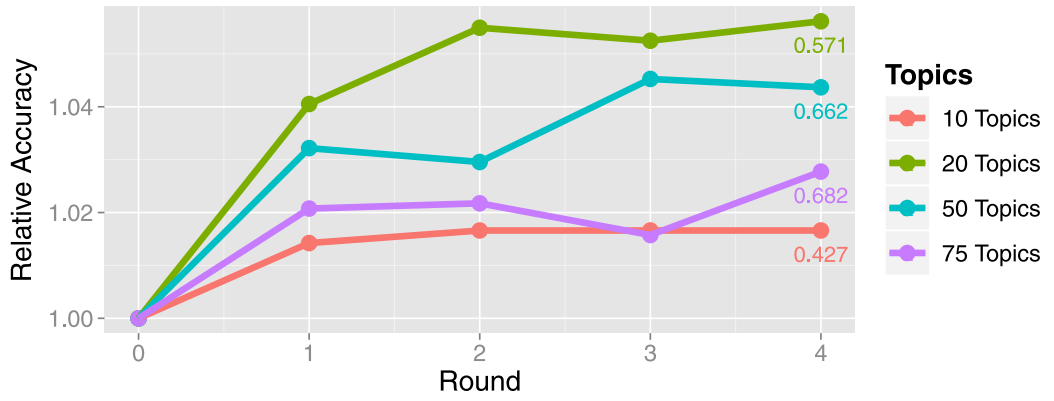

Fig. 13 The relative accuracy improvement (using round 0 as a baseline) of the best Mechanical Turk user session for each of the four numbers of topics, with the actual accuracy marked for the last round. While the 10-topic model does not provide enough flexibility to create good correlations, the best users could clearly improve classification with more topics

icons to edit or delete each one. The correlation being built is also shown in its own panel. Clicking on a word will remove that word from the current correlation.

Users were not given a specific goal; instead, they were instructed to add correlations between words so that the topics (we called them "word groups" in the instructions) made more sense. This was intentionally underspecified, as we wanted to see what would happen when ITM was placed in the hands of untrained users.

As in Sect. 6.1, we can compute the classification accuracy for users as they add words to correlations. The best users, who seemed to understand the task well, were able to increase the classification accuracy (Fig. 13). The median user, however, had an accuracy improvement indistinguishable from zero. Despite this, we can examine the users' behavior to better understand their goals and how they interact with the system.

The correlation sizes ranged from one word to over forty. The more words in the correlation, the more likely it was to noticeably affect the topic distribution. This observation makes sense given our updating method. A correlation with more words will probably cause the topic assignments to be reset for more documents.

Most of the large correlations (more than ten words) corresponded to the themes of the individual newsgroups. Some common themes for large correlations were:

- Themes that matched a single newsgroup: religion, space exploration, health, foreign countries, cars, motorcycles, graphics, encryption
- Themes that spanned multiple related newsgroups: sports, government, computers, cars/motorcycles
- Themes that probably matched a sub-topic of a single newsgroup: homosexuality, Israel, computer programming.

Some users created correlations with both "*baseball*" and "*hockey*" words, while others separated them. ("*baseball*" and "*hockey*" are in separate newsgroups.) The separate correlations often contained overlapping words. Even so, the choice of combined vs. separate correlations almost always determined whether baseball and hockey would be in the same topic in the model. A similar situation occurred with "*cars*" and "*motorcycles*", which are discussed in separate newsgroups.

Some users created inscrutable correlations, like {"better", "people", "right", "take", "things"} and {"fbi", "let", "says"}. They may have just clicked random words to finish the task quickly. While subsequent users could delete poor correlations, most chose not

to. Because we wanted to understand broader behavior we made no effort to squelch such responses.

The two-word correlations illustrate an interesting contrast. Some pairs are linked together in the corpus, like {"jesus", "christ", "solar", "sun"}. With others, like {"even", "number"} and {"book", "list"}, the users seem to be encouraging collocations to be in the same topic. However, the collocations may not be present in any document in this corpus.

Not all sensible correlations led to successful topic changes. Many users grouped "mac" and "windows" together, but they were almost never placed in the same topic. The corpus includes separate newsgroups for Macintosh and Windows hardware, and divergent contexts of "mac" and "windows" overpowered the prior distribution.

Other correlations led to topic changes that were not necessarily meaningful. For example, one user created a correlation consisting of male first names. A topic did emerge with these words, but the rest of the words in that topic seemed random. This suggests that the set of male first names aren't associated with each other in the corpus. Preliminary experiments on newspaper articles had similar correlations that created a more meaningful topic associated with obituaries and social announcements.

Finally, many correlations depend on a user's background and perspective, showing the flexibility of this approach. Some users grouped "israeli", "jewish", "arab", and "muslim" with international politics words, and others with religion words. On the other hand, "christian" was always grouped with religion words. The word "msg" appears to have two different interpretations. Some users grouped it with computer words (reading it as a message), while others grouped it with health words (reading it as a food additive).

As mentioned in Sect. 3, topic models with a tree-based prior can represent situations where words have multiple meanings. In previous work, the paths in the tree—provided by WordNet—correspond to the distinct meanings of a word (Boyd-Graber et al. 2007). Users found the formalism intuitive enough to build their own small WordNets to distinguish the different meanings of "msg".

6.3 User study

New systems for information access are typically investigated through task-based user studies to determine whether the new approach allows users to complete specific tasks as well as with current systems. Wacholder and Liu (2008), for example, compared traditional paper-based book indices with full-text search for answering questions in large text collections. Following their lead, we compare the information-seeking effectiveness using both interactive and non-interactive topic modeling.

We asked users to fill the role of the running example a political scientist attempting to find legislation relevant to "*immigration and refugee issues*" (among other topics). Using full-text search aided by either vanilla topic models or interactive topic models (itm), users were asked to answer questions based content in a collection of legislative debates.

We found that users were able to answer the questions equally well in both groups: with itm (experimental group) and without itm (control group). However, users in the group using itm had radically different strategies for how they found information in the corpus. Rather than relying on full-text search, users used topic models to find relevant information.

6.3.1 Legislative corpus

In the process of becoming a law, potential US legislation is sponsored by a congressperson and introduced for debate by a committee in either the US House of Representatives (lower

chamber) or the US Senate (upper chamber). Once introduced, the bill is debated within the chamber it was introduced. Our corpus contains transcripts of these debates for the 109th congress, which served during the 2005 and 2006 calendar years.

The corpus is available online from GovTrack.[13] Each page is associated with a bill and a vote. Uninteresting procedural bills, with less than 20 % "Yea" votes or less than 20 % "Nay" votes, are removed. We selected a subset of this congressional debate dataset that includes ten bills and their associated debates. Each debate has multiple turns (a single uninterrupted speech by a unique congressperson), and we use each turn as a document for topic modeling. This yields 2,550 documents in total; we ignore all temporal, speaker-related, or legislative organization. While this is somewhat unrealistic for a real-world study of legislative information, we will use some of this discarded information to aid evaluation. The subset includes bills on immigration, the estate (death) tax, stem cell research, and others. Detailed information can be found in Appendix A.

6.3.2 ITM *interface for exploring text corpora*

The ITM interface is a web-based application.[14] In contrast to the interface discussed in Sect. 6.2, it provides a comprehensive interface for navigating source documents, searching, viewing topics, and modifying topics. It provides a workflow for users to select model parameters (corpus and number of topics), create an initial topic model, name the topics, and refine the topics using ITM. The interface also provides multiple ways for a user to explore the corpus: a full-text search over all documents, a full-text search within a single topic, a listing of documents associated with each topic, and links to access specific documents. We walk through this workflow in detail below.

From the initial screen (Fig. 14), users specify the session information, such as user name, corpus, number of topics, etc. Once users click "start", the interface loads the initial set of topics, including the top topic words and related documents, as shown in Fig. 15. The top topic words are displayed such that the size of a word is proportional to the probability of this word appearing in the topic.

After clicking on the topic, users can view additional information and, most importantly, edit the topic (editing is disabled for the control group). After clicking on a topic, three "bins" are visible: all, ignore, important. Initially, all of the topic words are in the "all" bin. As shown in Fig. 16, users can drag words to different bins based on their importance to the topic: words that are important to the topic to the "important" bin, words that should be ignored in this topic to the "ignored" bin, and words that should be stopwords in the whole corpus to "trash". Users can also add new words to this topic by typing the word and clicking "add".[15]

Once the user has finished editing a topic, changes are committed by pressing the "Save" button. The backend then receives the users' feedback. The model adds a positive correlation between all words in the "important" bin, a negative correlation between words in "ignored" bin and words in "important" bin, and removes words in the "trash" from the model. With these changes to the model, the ITM relearns the topics and updates the topics. While in principle users may update the topics as many times as they wish, our study limited a user's

[13] http://www.govtrack.us/data/us/109/.

[14] This ITM interface is with a HTML and jQuery (http://jquery.com/) front end, connected via Ajax and JSON.

[15] Only words present in the model's vocabulary can be added; this constraint is enforced via an autocomplete function.

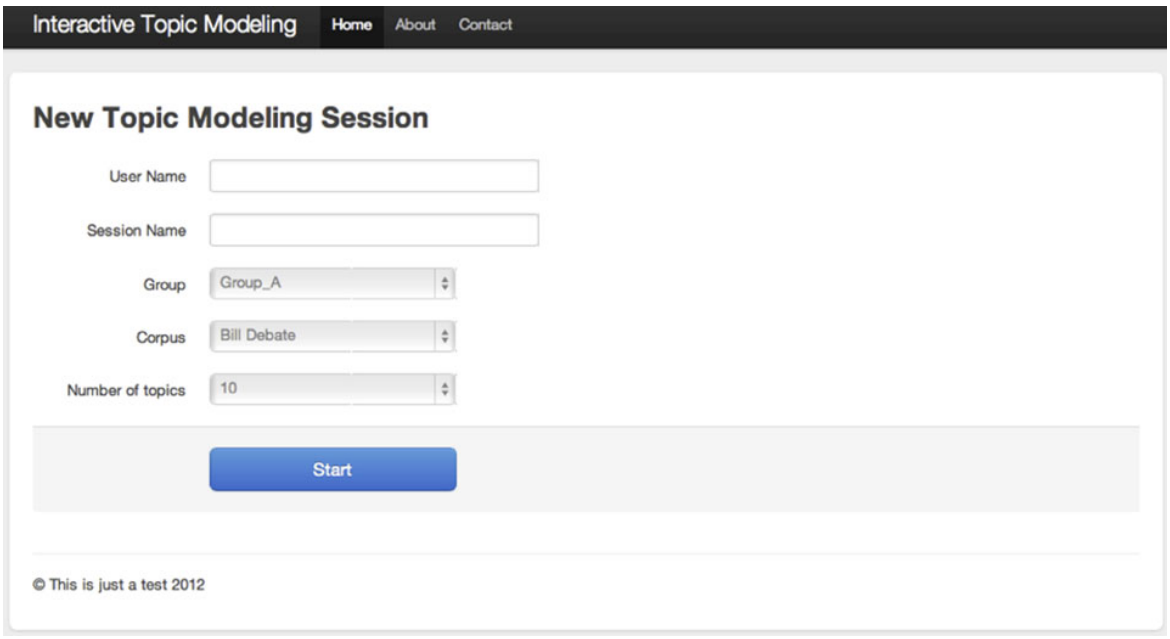

Fig. 14 The start page of ITM interface: users specify the user name, session name, corpus, number of topics, and experimental group (Group A: control group (LDA only); Group B: experimental group (ITM))

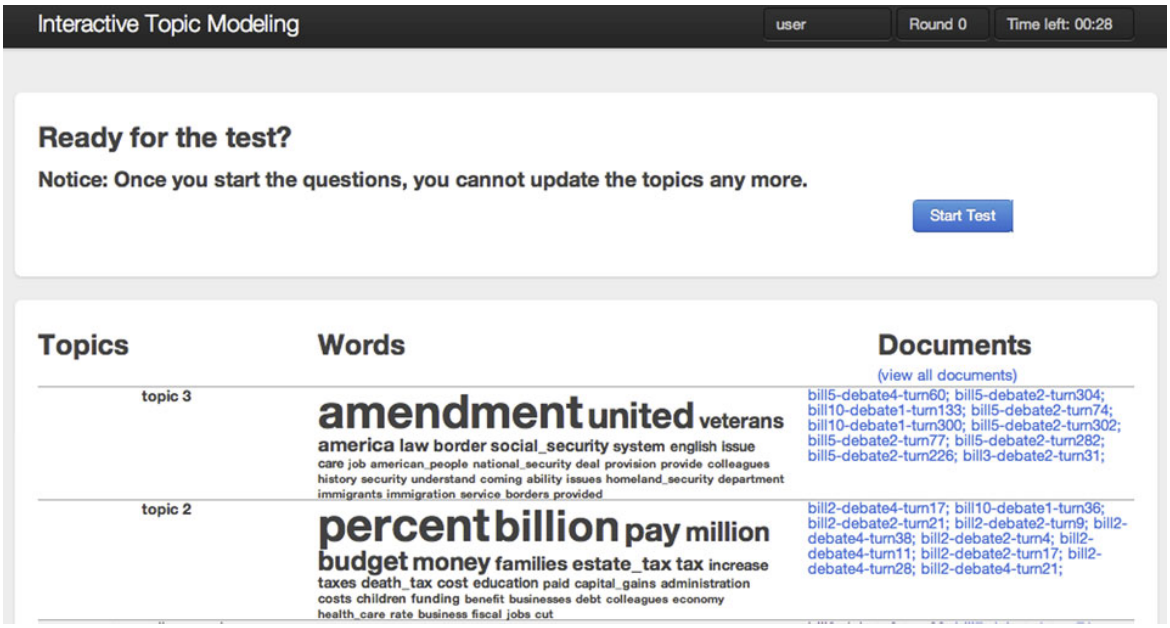

Fig. 15 Two topics displayed in the ITM interface. The most probable words in each topic are displayed with the size proportional to the probability of a word appearing in this topic. The documents most associated with each topic are shown in each topic's panel. The user can view all documents by selecting "view all documents"

exploration and modification of topics to fifteen minutes. Then, the users entered the next phase of the study, answering questions about the corpus.

In the question answering phase (Fig. 17), users have three options to explore the data to answer the questions: by reading through related documents associated with a topic, searching through all of the documents through full-text search, or via a text search *restricted to a single topic*. The full-text search is important because it is a commonly used means of finding data within large corpora (Shneiderman et al. 1997) and because it has been used in

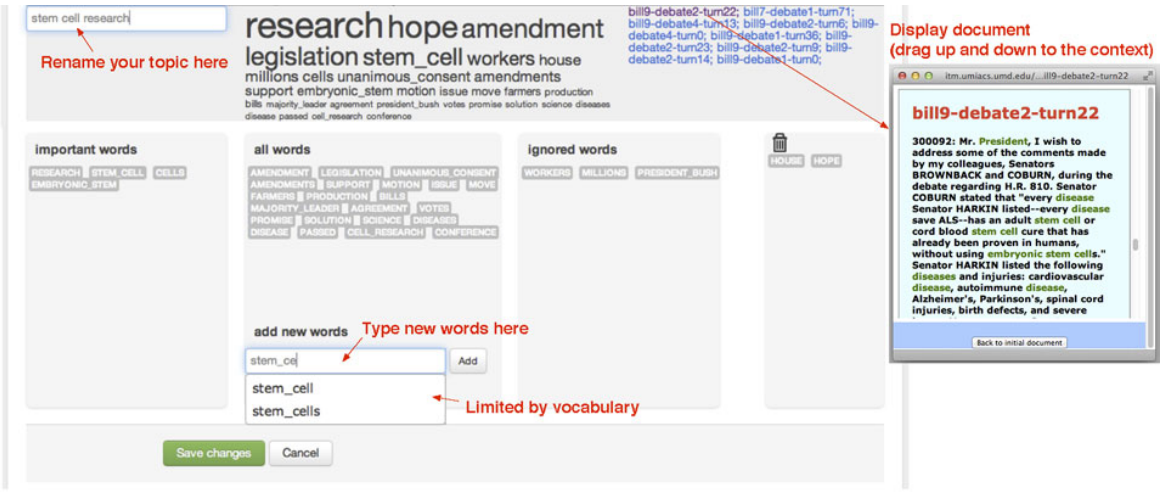

Fig. 16 ITM interface for refining a topic. Users can put words into different "bins", name topics, and add new words to the topic

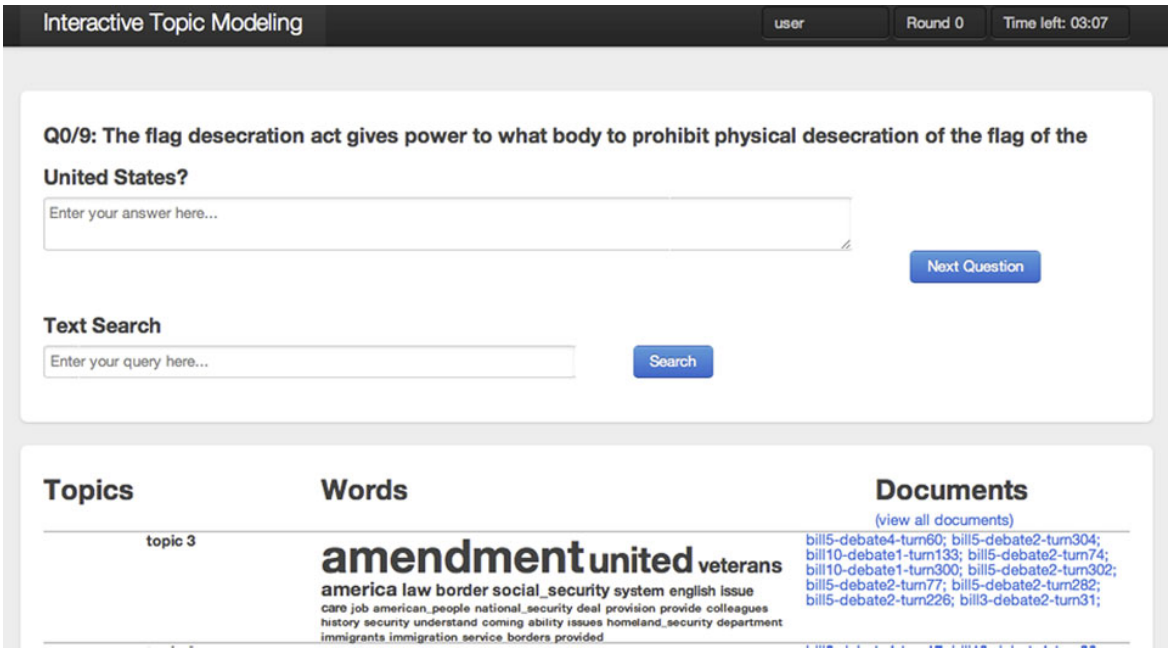

Fig. 17 The ITM interface along with a test to measure how well users were able to extract information from the underlying dataset. Users see one question at a time. They have multiple possible strategies for answering the question: searching keywords globally, checking the related topics or topic documents, or doing a full-text search within a topic, i.e., only seeing results from documents associated with a specific topic. After users click "Next question", they proceed to the next question, and users are not allowed to go back to previous questions

previous information-seeking studies (Wacholder and Liu 2008). Initial studies, where access to the data was restricted to only topic model information, were too difficult. We expect users to use topics when they are useful and use full-text search when topics are less useful in answering a question. After each question, users click "Next question" to proceed; users cannot return to previous questions.

6.3.3 User population

To evaluate the effectiveness of ITM for information-seeking tasks, we compare the performance of users in two groups: and experimental group (ITM) and a control group (vanilla LDA).

For the experimental group, users start with an initial set of topics and can refine the topics using ITM for up to fifteen minutes. They then start the test phase for thirty minutes. They are provided with the refined topics for use during the test.

The control group also has access to the initial topics, but they cannot refine the topics. They are given up to fifteen minutes to check the topics, rename the topics, and review documents associated with the topics. This is to avoid experimental differences caused by the experimental group benefiting from *exploring* the corpus rather than from *interactive topic modeling*. After spending up to fifteen minutes exploring the corpus, the control group also has thirty minutes to answer the test questions.

The study participants are randomly assigned to a group. Each participant views a video explaining how to use the interface and do the test. During the study, the system logs the related information of each user. After the study, participants complete a survey on their educational/technical background and familiarity with legislation or topic models.

The study had twenty participants (ten for each group). All of the users are fluent in English. Participants are either students pursuing a degree in computer science, information science, linguistics, or working in a related field. A post-test user survey revealed that most users have little or no knowledge about congressional debates and that users have varied experience with topic models.

We designed ten free response questions by exploring this legislation corpus, including questions regarding legislation which deals with taxes, the US-Mexico border, and other issues. The full text of the questions appears in Appendix B.

6.3.4 User study analysis

We examined two aspects of the experiment: how well the experimental group's final topics replicated ground-truth annotations (below, we refer to this metric as **refine**) and how well both the groups answered the questions (**test**).

Our experiment views the corpus as an unstructured text collection (a typical use case of topic models); however, each turn in the dataset is associated with a single bill. We can view this association as the true clustering of the dataset. We compare this clustering against the clustering produced by assigning each document to a cluster corresponding to its highest-probability topic.

We compare these reference clusters to the clusters produced by ITM using **variation of information** (Meilă 2007). This score has a range from zero to infinity and represents the information-theoretic "distance" between two partitions (lower is better). Using this information, we compute the variation of information (Meilă 2007) between the true labels and the topic modeling clusters. While we have a good initial set of topics (the initial variation of information score is low), users in the experimental group—who claimed to have little knowledge about the legislative process—still can reduce this score by refining the topics. To avoid bias from users, users do not know that their topics will be evaluated by variation of information.

As shown in Fig. 18, ten users in the experimental group started with the same initial topics and refined refine the topics for multiple rounds. In the given fifteen minutes, some users played with ITM for up to eight rounds while one user only tried two rounds. Although

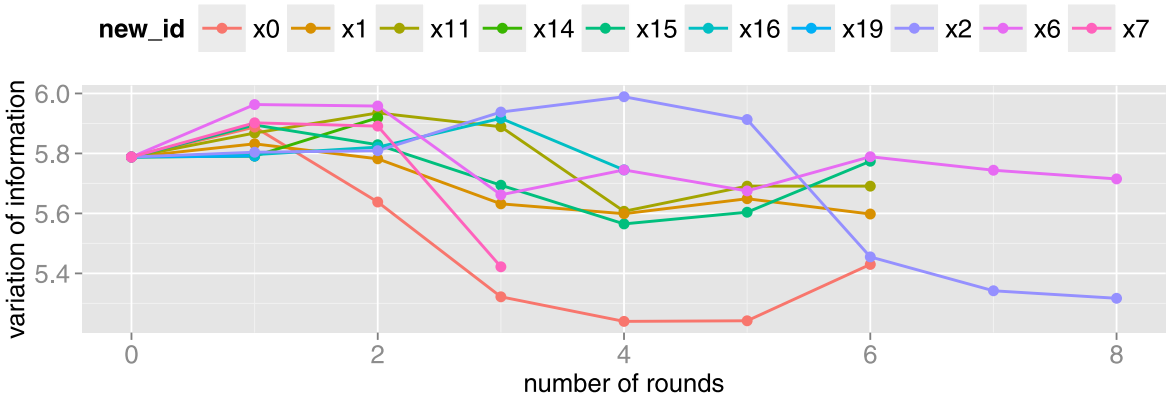

Fig. 18 Variation of information of users in experimental group at each round. Users are labeled from x0 to x19. Ten of them are randomly assigned to experimental group. All ten users start with the same initial topics and are able to refine the topics for the extent of the refinement phase. Most users in the experimental group successfully reduced the variation of information (where lower is better)

users occasionally increased the variation of information, by the end of the refinement phase a majority of users successfully reduced the variation of information of the topics.

User "x2" provides an example of a successful ITM round. The user saw a topic mixing "*energy*"-related words with other words. To make a coherent topic about "*energy*", they put "oil", "natural gas", "gas", "production" and "resources" in the important bin, and put "patriot_act", "federal_government", "tex_cuts", "stem_cell" into the ignored bin. After updating, this topic became a coherent topic about "*energy*". After refining topics for eight rounds, they successfully made other topics more coherent; he named these topics "*homeland security*", "*immigration*", "*abortion*", "*energy*", "*flag burning*", etc., which match well with the corpus's true clusters. Thus this user successfully reduced the variation of information as shown in Fig. 18.

In addition to evaluating the variation of information for the experimental group, we also evaluated the users' answers to content-specific questions. While the difference between the groups' performance was not statistically significant, ITM changed the *usage pattern* to favor topic models over full text search.

To evaluate the **test**, we graded the answers and compared the scores of users in two groups. Of the 20 participants, two didn't use their session name correctly, meaning the interface didn't store their answers properly, and one user encountered an issue and wasn't able to finish the questions. Thus we have complete answers for 17 participants. Each question was graded by two graders with Scott's π agreement 0.839 (Artstein and Poesio 2005). While there is no significant difference between the two groups' test scores, the scores for experimental group had a much smaller variance compared to the control group.

To better understand how users answer the questions, the ITM system logs the number of full-text searches that include words from any of the topics (queried-topic-words) and the number of times that users used topics to filter query results (query-in-topic).

The process of modifying topics inspired users in the experimental group to use queries that included words from the topics (Fig. 19); this may be because users learned more key terms while exploring and refining the topics. These topic words are helpful to answer questions: users in experimental group queried topic words an average of 27.8 times, while the control group queried topic words 18.2 times on average. Users in the experimental group also used "query-in-topic" (restricting a full text search *within* a topic) more than the users in control group. This is probably because those users working with refined topics that are better aligned with the underlying bills (several questions were about specific bills).

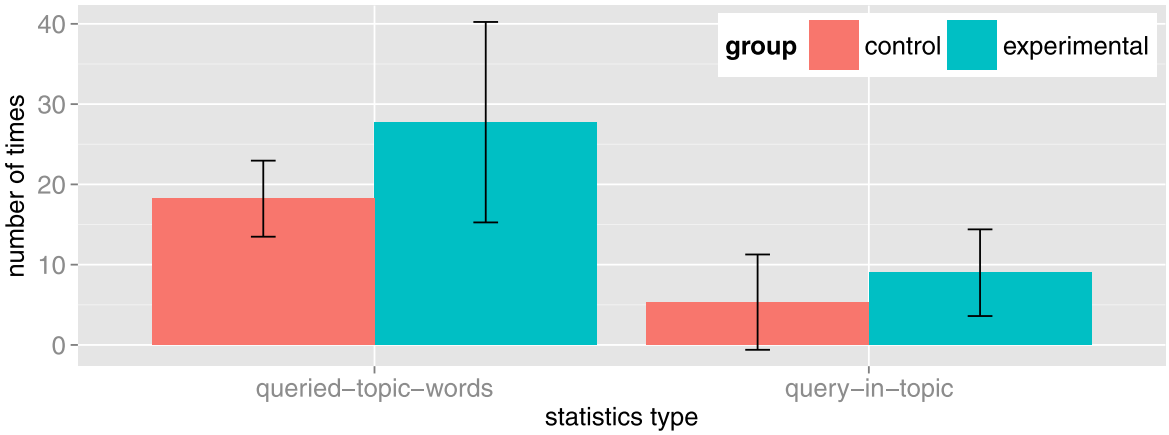

Fig. 19 Statistics show that users' search strategies during the user study used topics more than the control group

We also found that users in both groups click topics much more when the question is about the general understanding of the data set, for example, "Name 5 of the debated legislation in this data set.". For more detailed questions like "The Gulf of Energy Security act will provide revenue streams for which fund?", users in both groups prefer using text query directly.

However, Fig. 19 shows a large variance, so we should not overstate these results. In the conclusion, we discuss additional studies that can untangle the usefulness of topic models for evaluating information-seeking from other effects such as how familiar users are to topic models, whether they understand the task clearly, and whether they are effective consumers of information.

Some users in the control group also performed very well. For example, user "x5" in the control group obtained a high score. During the initial fifteen minute exploration phase, this user clicked on topics to review documents 71 times, substantially more than any user in either the control group or the experimental group. Users such as "x5", who are topic model-savvy have better intuitions about how topic models work and how they can be used to help explore a corpus. In the post-session survey, the user reported that the interface, designed to facilitate ITM (but disabled for the control group) helped them understand the corpus and answer the questions.

Not all users in the experimental group performed well on the task. One user only refined two topics, and some users failed to improve the topics (failed to reduce the variation of information). Some users complained that they weren't given enough time to update the topics.

In general, most reported liking the interface. Users from both the experimental group and the control group commented that the topics helped them answer some of the questions. Some users also commented that some of the questions were too detailed, suggesting that perhaps additional methods to search the corpus may be helpful.

This study provides evidence that the ITM interface assists users in exploring a large corpus and that topic modeling is helpful for users attempting to understand legislative documents. Users used ITM to improve the initial clusters; this is especially promising, as these users had little background knowledge of congressional debates and few had familiarity with topic models.

7 Automatically suggesting correlations

While we have demonstrated that ITM can encode correlations into topic models interactively, our pilot with users showed that it is often difficult, particularly for untrained users, to decide how to guide interactive topic models. This is because there are many possible choices: if the vocabulary size is V, there are about V^2 possible pair correlations (let alone higher order correlations). In this section, we build on heuristics proposed for topic coherence to suggest correlations automatically.

7.1 Generating new correlations

Newman et al. (2010) argues that topics whose words often appear close together in a reference corpus make more sense to users. They measure this through pointwise mutual information (PMI) averaged over all word pairs present in the top words of a topic (sorted in order of decreasing probability, as is usually done to show topics to a user). As a result, for a topic's top words T,

$$\text{PMI}_{\mathcal{C}}(T) \equiv \frac{\sum_{(w_i,w_j):w_i \neq w_j} \text{PMI}_{\mathcal{C}}(w_i, w_j)}{|T|(|T|-1)}, \tag{13}$$

where $\mathcal{C}$ is the corpus to compute PMI, and PMI is computed within a small local window. Topics that have a high score tend to be judged as "making sense" to users, and those that have lower scores tend to be judged as not making sense to users.

Based on this strategy, ITM could seek to improve this score by suggesting positive correlations for pairs with high PMI score and negative correlations for pairs with low PMI score. To ensure that we only suggest meaningful correlations, we weight suggestions by the tf-idf (Salton 1968) for each word (by taking the max over all documents). This focuses correlations toward pairs that are highly relevant to at least some subset of the documents (this prevents correlations capturing syntactic or other dependencies). Combining the PMI and tf-idf score, we rank word pairs by

$$\text{PC}(X, Y) = \max_d\big(\text{tf-idf}(X, d)\big) \cdot \max_d\big(\text{tf-idf}(Y, d)\big) \cdot \text{PMI}_{\mathcal{C}}(X, Y) \tag{14}$$

for positive correlations and by

$$\text{NC}(X, Y) = \frac{\max_d(\text{tf-idf}(X, d)) \cdot \max_d(\text{tf-idf}(Y, d))}{\text{PMI}_{\mathcal{C}}(X, Y)} \tag{15}$$

for negative correlations (NC). The pairs with the highest scores for these metrics become the suggestions. Since the number of word pairs is very large, we only consider the word pairs from different topics for PC and the word pairs from the same topic for NC.

Although we have an automatic technique for selecting correlations, this does not replace a fully interactive system. This is because PMI ignores words with multiple meanings (e.g., "msg" in Sect. 6.2) and user-specific information needs (e.g., the examples in Sect. 2).

7.2 Human evaluation over automatically generated correlations

We use the 20 Newsgroups corpus (20NEWS, described in Sect. 4.5) in this experiment. The topic number is set to be 20, and 100 iterations produce initial topics. Four rounds of interaction are performed with 50 iterations each round.

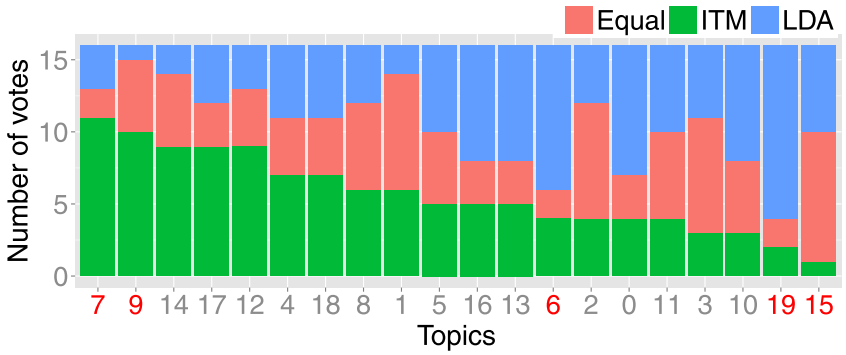

Fig. 20 The total number of votes across rounds for each topic by users on Mechanical Turk (16 votes for each topic). *x-axis* is in a decreasing order of the number of votes for ITM. The *red colored topics* are significantly different from a uniform vote distribution, while the others are not (tested by χ^2-test). In general, there is no clear preference from users between models with correlations (ITM) and models without (LDA), which is the result in imbalanced attention being focused on some topics more than others (Color figure online)

In this experiment, we have two different topic models: one uses automatically generated correlations based on PMI_{20news} (five positive and five negative correlations each round) and the other group runs for same number of iterations without any correlations. We name the two models as correlation group (ITM) and non-correlation group (LDA), respectively.

For evaluation, we showed the resulting topics to users on Mechanical Turk and asked whether they preferred the correlated topics (ITM), the control topics (LDA), or they looked equally coherent (Equal). Four users compared each pair. The positioning (i.e., left vs. right) and order of words within a topic were shuffled to avoid bias.

We first compared the votes for each group in each topic, as shown in Fig. 20, in decreasing order of the number of votes for the correlated group (ITM). Users have a significant preference in five of the total 20 topics (colored in red, by χ^2-test). Users did not have a clear preference overall; this was counter-intuitive, as the correlations were changing the topics dramatically. There were three reasons that the correlations did not always favor the ITM group:

– first, topics that are most confusing (as measured by (14) and (15)) get the correlations to improve coherence; thus the correlations affect some topics more than others
– second, because some topics have more correlations, other topics have fewer correlations; thus those topics are similar to the uncorrelated case
– finally, some topics are ignored by correlations **and** have side-effects of correlations in the other topics; these are not always good for the coherence of the ignored topics.

We describe each of these scenarios with examples, below.

The confusing topic gets the correlation Some topics did show substantial improvement, however. For example, Fig. 21 shows how Topic 7 changes as correlations are added (we only show the correlations related with Topic 7). Initially, the topic mixes "*transportation*" and "*health*", in the first round, a negative correlation between "car" and "cancer" pushed "car" away. Though "cars" remained, additional relevant terms—"medicine", "pain", "aids" and "effects" appeared. In the second round, when a negative correlation between "cancer" and "msg" was added, "cars" disappeared and most words in the correlated group (ITM) were related with "*health*". The non-correlated group (LDA) also stabilizes but much more slowly than the correlated group (ITM); after the fourth round, however, users do not have a clear preference.

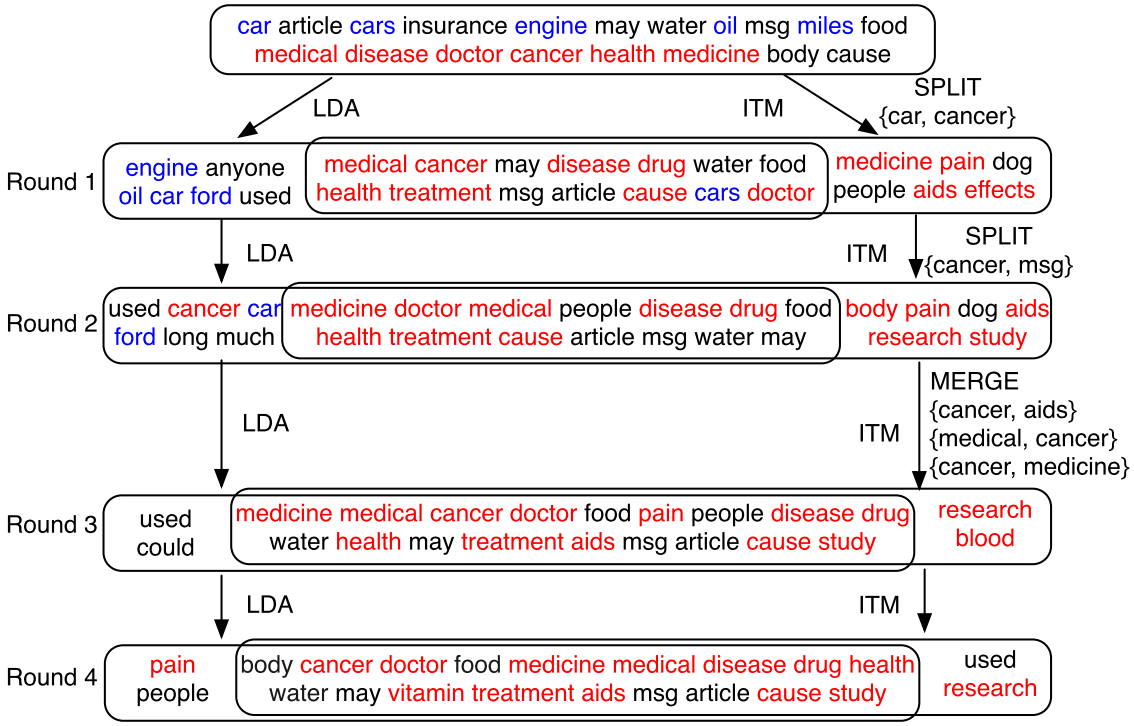

Fig. 21 Compare the "evolution" of Topic 7 between LDA and ITM for four rounds: the correlations related with this topic is shown and we compared the topic words between LDA and ITM. At the beginning, Topic 7 is a mixed topic between "*transportation*" (*blue*) and "*health*" (*red*). While adding more correlations and running for additional iterations, this topic becomes a pure "*health*" topic. While after four rounds, LDA and ITM got similar result, ITM successfully made it the topic converge to a good coherent topic much faster (Color figure online)

Table 4 The round votes for Topic 5 and Topic 19. For Topic 5, while users have no preference in R1 and R4, they prefer LDA in R2: one correlation "MERGE {window, widget}" was added in ITM, but LDA also had the two correlated words in its topic, so there was no clear improvement; in R3, two related correlations ("SPLIT {max, font}" and "SPLIT {max, window}") were added to ITM, and users prefer the improved topic in ITM. For Topic 19, no related correlations were added to ITM in the first three rounds; there was one unimportant correlation "SPLIT {max, model}" added in R4, which resulted in no clear improvement. So users clearly prefer LDA for Topic 19

Round	Topic 5			Topic 19		
	ITM	Equal	LDA	ITM	Equal	LDA
R1	2	1	1	0	2	2
R2	0	0	4	0	0	4
R3	3	0	1	1	0	3
R4	0	4	0	1	0	3

Ignored topics stay the same While Topic 7 was improved, Topic 5, initially including words "software, data, set, graphics, code, used, sun, user, following, text" etc., seemed the same for users' observing the LDA and ITM versions of the topic, as shown in Table 4. In the second round of Topic 5, one positive correlation {"window", "widget"} was added in ITM, but LDA already had these two correlated words in its topic, so the correlation had little effect on the topic. In next round, two related negative correlations {"max", "font"} and {"max", "window"} were added to ITM, which improved the topic, so users prefer the ITM. In first and last round, users have no preference between the two groups.

Ignored topics suffer side-effects Another case is that users sometimes decisively prefer the non-correlated group (LDA). For example, Topic 19, started with words "system, good, power, much, research, first, large, data, systems, work" etc. It only had one relevant correlation, a negative correlation {"max", "model"} added in the last round. However, the result of correlations in previous rounds negatively impacted Topic 19, which had words "stolen" from it by other topics. This shows that improving some topics (e.g., Topic 7) sometimes comes at the expense of others; Table 4 shows that users preferred the version of the topic left untouched by correlations.

Figure 21 shows that for LDA and ITM, Topic 7 converged to the same topic clear "health" topic, but ITM helped it converge the faster. We discuss the distinction between improvements and impatience in Sect. 2.3, which also gives an example of a topic that remains problematic even in a fully converged topic model.

8 Conclusion

Interactive topic modeling is a new framework for using and understanding statistical models that empowers users of topic models to evaluate and refine topics based on their unique expertise and perspective. These models offer new opportunities for wider and richer use of topic models, suggest new probabilistic models and methods for machine learning, and can serve as an exemplar for allowing users to improve statistical models.

Wider and richer uses of topic models Topic models have attracted interest from not only the machine learning experts, but also non-machine-learning experts in various fields including computational social science (Lazer et al. 2009; Grimmer 2010; Yang et al. 2011), history (Hall et al. 2008; Mimno et al. 2012), psychology (Landauer et al. 2006; Griffiths et al. 2007), biology (Ahmed et al. 2009; Ye et al. 2011), and music (Hu and Saul 2009). These researchers have been able to understand, explain, and illustrate data that were, by virtue of their size, inaccessible for a "deep reading". Study of these data are only possible using technology such as topic models.

But these investigations could be so much richer if this process were not a one-way experience. The adoption of topic models by broader communities represents an exciting opportunity for topic models to *learn from* researchers (usually, it's the other way around). This would have manifold benefits: models would reflect expert knowledge, models would be less frustrating and more malleable, and different models could reflect disagreements or different perspectives of model curators.

In addition to facilitating a richer experience for experts, interactive topic models offer an opportunity to better engage novices. Even though topic models have generated substantial interest from researchers outside machine learning, they remain less than user-friendly. Interactive models do not require users to be machine learning experts to improve and adjust models; this can help users slowly build statistical insights and facilitate greater uptake of computational methods in the social sciences.

Our future goal is to turn our running hypothetical political scientist interested in understanding "*immigration*" into a reality. We are currently working with political scientists to enable them to use topic models to discover interesting instances of where politicians have used loss or gain framing (Tversky and Kahneman 1992) to "spin" a topic to appeal to their core constituencies. Our user study showed that even novice users can use ITM to effectively explore political corpora, and we expect better results from motivated researchers willing to invest time understanding the ITM system.

However, the question of whether topic models assist users in information seeking requires more experimentation. We showed that ITM encouraged users to use topics to help them find information, but our population was too diverse and too small to be able to demonstrate that these techniques helped them to *better* or *more quickly* access the information.

Broadening the number of users for the user study would allow us to draw stronger conclusions about how interactive topic modeling changes or helps the way users seek out information from large corpora. In addition, with larger populations, a mixed-effects model could potentially untangle the effects of how familiar that users are to topic models, whether they understand the task clearly, their background knowledge of the subject, and whether they understand how to use the interface. Explicitly modeling and measuring these effects would effectively reduce the variance and help explain the interaction between these nuanced facets of user behavior.

New probabilistic models and inference techniques In our attempt to minimize the latency a user could experience during the process of interactive topic modeling, we developed new techniques for probabilistic inference that take advantage of sparsity in probabilistic models. While our approach was more complicated than first method designed specifically for Latent Dirichlet Allocation (Yao et al. 2009), it still offered substantial computational speedup. This suggests that taking advantage of sparsity could also be used in other probabilistic models such as context-free grammars (Johnson et al. 2007; Johnson 2010) and feature-based models (Monroe et al. 2008; Wu et al. 2010).

While this work focused on text in a single language, tree-based topic models have been used to explore multilingual corpora (Boyd-Graber and Resnik 2010; Jagarlamudi and Daumé 2010), and from there it is a small leap to multimodal data. Interactivity could be even more helpful in such scenarios, as interaction in a user's native language or in the most natural modality could help shape topics in the other language or modality. For example, an analyst trying to understand newspapers written in English and Arabic might be more comfortable massaging the topics in English, but those correlations would also improve the clustering for Arabic documents. Similarly, a topic-model clustering of image data with captions (Li Fei-Fei Perona 2005) might be difficult to interact with visually, but suggesting correlations based on captions might lead to useful image clustering.

Toward more interactive statistical models While interactive topic modeling can obviate or replace some of the newer topic models, some models seem apt for interactive topic modeling. For example, combining interactivity with dynamic topic modeling (Blei and Lafferty 2006; Wang et al. 2008) could help to improve historians or social scientists working with datasets over long time periods; supervised topic models could help researchers understand how documents interact with external socioeconomic indicators such as the sentiment (Pang and Lee 2008; Sayeed et al. 2012), consumer price index (Shoemaker 2011), stock price (Kogan et al. 2009), or geography (Eisenstein et al. 2010); and topic models that go beyond the bag of words (Wallach 2006; Boyd-Graber and Blei 2008) could help understand syntactic patterns and linguistic structure.

Finally, the interactions that we observe from users could help us understand how humans organize their mental vocabulary (Feldman et al. 2009). Instead of just accepting user feedback as a given, we can explicitly model it using techniques such as Bayesian coalescents (Teh et al. 2008; Görür and Teh 2009). A joint model of both the tree structure and the topic model could learn users' desires and preferences from text; this would be a more statistically-driven alternative to our approach of suggesting correlations and could help us learn more from our users.

Learning from users is not just a benefit, but it is an essential goal for machine learning algorithms to be accepted by researchers who are not computer scientists and eventually the broader public. Interactive topic models are an example of tools that can learn from and help users interact with large datasets, an essential tool for modern text-based research.

Acknowledgements We would like to thank Edmund Talley, Jonathan Chang, Jason Chuang, Philip Resnik and Leo Claudino for their helpful comments. This work was supported by National Science Foundation grant #0705832, Army Research Laboratory Cooperative Agreement W911NF-09-2-0072, and by National Science Foundation grant #1018625. Any opinions, findings, conclusions, or recommendations expressed are the authors' and do not necessarily reflect those of the sponsors.

Appendix A: Titles of the ten bills used in the user study (Sect. 6.3)

- H.R. 6061: Secure Fence Act of 2006
- H.R. 8: Death Tax Repeal Permanency Act of 2005
- S. 2271: USA PATRIOT Act Additional Reauthorizing Amendments Act of 2006
- S. 3711: Gulf of Mexico Energy Security Act of 2006
- S. 3711: Gulf of Mexico Energy Security Act of 2006
- S. 403: Child Custody Protection Act
- H.R. 4297: Tax Increase Prevention and Reconciliation Act of 2005
- S.J. Res. 12: Flag Desecration resolution
- H.R. 810: Stem Cell Research Enhancement Act of 2005
- H.R. 810: Stem Cell Research Enhancement Act of 2005.

Appendix B: Questions list in the user study (Sect. 6.3)

- The flag desecration act gives power to what body to prohibit physical desecration of the flag of the United States?
- The child custody protection act makes it what type of crime to take minors across state lines to circumvent laws requiring involvement of parents in abortion decisions?
- According to the debate for the secure fence act, what is the length (in miles) of the border that the U.S. shares with Mexico?
- The Gulf of Energy Security act will provide revenue streams for which fund?
- A senator compares the immigration debate to the release of what film?
- Name 5 of the debated legislation in this data set. Give either the full name or the number assigned to the debate.
- Name 2 of the debated legislation in this data set deals with taxes or the budget?
- What is the name of the debated legislation which proposes an amendment to the constitution?
- Name the 2 debated legislation in this data set that discusses illegal immigrants?

References

Abney, S., & Light, M. (1999). Hiding a semantic hierarchy in a Markov model. In *Proceedings of the Workshop on Unsupervised Learning in Natural Language Processing* (pp. 1–8).

Ahmed, A., Xing, E. P., Cohen, W. W., & Murphy, R. F. (2009). Structured correspondence topic models for mining captioned figures in biological literature. In *International conference on knowledge discovery and data mining* (pp. 39–48).

Andrzejewski, D., Zhu, X., & Craven, M. (2009). Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of the International Conference of Machine Learning*.

Artstein, R., & Poesio, M. (2005). *Kappa3 = alpha (or beta)* (Technical Report). University of Essex Department of Computer Science.

Astrachan, O. (2003). Bubble sort: an archaeological algorithmic analysis. In *Proceedings of the 34th SIGCSE technical symposium on computer science education*.

Bendapudi, N., & Leone, R. P. (2003). Psychological implications of customer participation in co-production. *Journal of Marketing*, *67*(1), 14–28.

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. Sebastopol: O'Reilly Media.

Blei, D. M., & Lafferty, J. D. (2005). Correlated topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Blei, D. M., & Lafferty, J. D. (2006). Dynamic topic models. In *Proceedings of the International Conference of Machine Learning*.

Blei, D. M., Ng, A., & Jordan, M. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Blei, D. M., Griffiths, T. L., & Jordan, M. I. (2010). *The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies*.

Boyd-Graber, J., & Blei, D. M. (2008). Syntactic topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Boyd-Graber, J., & Resnik, P. (2010). Holistic sentiment analysis across languages: Multilingual supervised latent Dirichlet allocation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Boyd-Graber, J., Blei, D. M., & Zhu, X. (2007). A topic model for word sense disambiguation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Boydstun, A. E., Glazier, R. A., & Phillips, C. (2013). Agenda control in the 2008 presidential debates. *American Politics Research*.

Bron, C., & Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, *16*(9), 575–577.

Carbone, K. (2012). Topic modeling: Confusion and excitement. http://dh201.humanities.ucla.edu/?p=502.

Ceaparu, I., Lazar, J., Bessiere, K., Robinson, J., & Shneiderman, B. (2004). Determining causes and severity of end-user frustration. *International journal of human-computer interaction*, *17*(3), 333–356.

Chang, J. (2010). Not-so-latent Dirichlet allocation: Collapsed Gibbs sampling using human judgments. In *NAACL Workshop: Creating Speech and Language Data With Amazon'ss Mechanical Turk*.

Chang, J., Boyd-Graber, J., Wang, C., Gerrish, S., & Blei, D. M. (2009). Reading tea leaves: How humans interpret topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Daumé, H. III. (2009). Markov random topic fields. In *Proceedings of Artificial Intelligence and Statistics*.

Dietz, L., Bickel, S., & Scheffer, T. (2007). Unsupervised prediction of citation influences. In *Proceedings of the International Conference of Machine Learning*.

Drouin, J. (2011). Foray into topic modeling. *Ecclesiastical Proust Archive*.

Eisenstein, J., O'Connor, B., Smith, N. A., & Xing, E. P. (2010). A latent variable model for geographic lexical variation. In *Proceedings of Empirical Methods in Natural Language Processing* (pp. 1277–1287).

Evans, P. (2013). *More fun with topic modeling*. http://mith.umd.edu/engl668k/?p=1595.

Feldman, N. H., Griffiths, T. L., & Morgan, J. L. (2009). Learning phonetic categories by learning a lexicon. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*.

Görür, D., & Teh, Y. W. (2009). An efficient sequential Monte Carlo algorithm for coalescent clustering. In *Proceedings of Advances in Neural Information Processing Systems*.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, *101*(1), 5228–5235.

Griffiths, T. L., Canini, K. R., Sanborn, A. N., & Navarro, D. J. (2007). Unifying rational models of categorization via the hierarchical Dirichlet process. In *Proceedings of the 29th Annual Conference of the Cognitive Science Society*.

Grimmer, J. (2010). A Bayesian hierarchical topic model for political texts: Measuring expressed agendas in senate press. *Political Analysis*.

Gruber, A., Rosen-Zvi, M., & Weiss, Y. (2007). Hidden topic Markov models. In *Artificial Intelligence and Statistics*.

Hall, D., Jurafsky, D., & Manning, C. D. (2008). Studying the history of ideas using topic models. In *Proceedings of Empirical Methods in Natural Language Processing*.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *SIGKDD Explorations 11*.

Harary, F. (1969). *Graph Theory*. Reading: Addison-Wesley.

Heinrich, G. (2004). *Parameter estimation for text analysis* (Technical Report). http://www.arbylon.net/publications/text-est.pdf.
Hoffman, M., Blei, D. M., & Bach, F. (2010). Online learning for latent Dirichlet allocation. In *NIPS*.
Hopcroft, H., & Tarjan, R. (1973). Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, *16*(6), 372–378.
Hopkins, D. J. (2012). The exaggerated life of death panels: The limits of framing effects in the 2009–2012 health care debate.
Hu, D., & Saul, L. K. (2009). A probabilistic model of unsupervised learning for musical-key profiles. In *International Society for Music Information Retrieval Conference*.
Hu, Y., & Boyd-Graber, J. (2012a). Efficient tree-based topic modeling. In *Association for Computational Linguistics*.
Hu, Y., & Boyd-Graber, J. (2012b). Suggesting constraints for interactive topic modeling. In *ICML Workshop on Machine Learning in Human Computation and Crowdsourcing*.
Hu, Y., Boyd-Graber, J., & Satinoff, B. (2011). Interactive topic modeling. In *Proceedings of the Association for Computational Linguistics*.
Jagarlamudi, J., & Daumé, H. III. (2010). Extracting multilingual topics from unaligned corpora. In *Ecir, Milton Keynes, United Kingdom*.
Johnson, M. (2010). PCFGs, topic models, adaptor grammars and learning topical collocations and the structure of proper names. In *Proceedings of the Association for Computational Linguistics*.
Johnson, M., Griffiths, T. L., & Goldwater, S. (2007). Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Conference of the North American Chapter of the Association for Computational Linguistics*.
Kogan, S., Levin, D., Routledge, B. R., Sagi, J. S., & Smith, N. A. (2009). Predicting risk from financial reports with regression. In *Conference of the North American Chapter of the Association for Computational Linguistics*.
Landauer, T. K., McNamara, D. S., Marynick, D. S., & Kintsch, W. (Eds.) (2006). *Probabilistic Topic Models*. Hillsdale: Erlbaum.
Lau, J. H., Grieser, K., Newman, D., & Baldwin, T. (2011). Automatic labelling of topic models. In *Proceedings of the Association for Computational Linguistics* (pp. 1536–1545).
Lavine, M. (1992). Some aspects of Pólya tree distributions for statistical modeling. *The Annals of Statistics*, *20*(3), 1222–1235.
Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabasi, A. L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D., & Alstyne, M. V. (2009). Social science: Computational social science. *Science*, *323*(5915), 721–723.
Li, W., & McCallum, A. (2006). Pachinko allocation: Dag-structured mixture models of topic correlations. In *International Conference on Machine Learning* (pp. 577–584).
Li Fei-Fei Perona, P. (2005). A Bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition* (pp. 524–531).
Lin, W. H., Wilson, T., Wiebe, J., & Hauptmann, A. (2006). Which side are you on? identifying perspectives at the document and sentence levels. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.
Meeks, E. (2011). Comprehending the digital humanities. *Digital Humanities Specialist*.
Meilă, M. (2007). Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, *98*(5), 873–895.
Miller, G. A. (1990). Nouns in WordNet: A lexical inheritance system. *International Journal of Lexicography*, *3*(4), 245–264.
Mimno, D., Wallach, H., & McCallum, A. (2008). Gibbs sampling for logistic normal topic models with graph-based priors. In *NIPS 2008 Workshop on Analyzing Graphs: Theory and Applications*.
Mimno, D., Wallach, H., Talley, E., Leenders, M., & McCallum, A. (2011). Optimizing semantic coherence in topic models. In *Proceedings of Empirical Methods in Natural Language Processing*.
Mimno, D., Hoffman, M., & Blei, D. (2012). Sparse stochastic inference for latent Dirichlet allocation. In *Proceedings of the International Conference of Machine Learning*.
Monroe, B. L., Colaresi, M. P., & Quinn, K. M. (2008). Fightin' Words: Lexical Feature Selection and Evaluation for Identifying the Content of Political Conflict. *Political Analysis*, *16*(4), 372–403. 2008.
Nah, F. F. H. (2004). A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, *23*(3), 153–163.
Neal, R. M. (1993). *Probabilistic inference using Markov chain Monte Carlo methods* (Technical Report CRG-TR-93-1). University of Toronto.
Nelson, R. K. (2010). Mining the dispatch. http://dsl.richmond.edu/dispatch/.
Newman, D., Karimi, S., & Cavedon, L. (2009). External evaluation of topic models. In *Proceedings of the Aurstralasian Document Computing Symposium*.

Newman, D., Lau, J. H., Grieser, K., & Baldwin, T. (2010). Automatic evaluation of topic coherence. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Norman, D. A. (Ed.) (1993). *Things That Make Us Smart: Defending Human Attributes In The Age Of The Machine*, Reading: Addison-Wesley.

Norman, D. A. (2002). The Design of Everyday Things. Reprint paperback edn. Basic Books.

Séaghdha, D.Ó., & Korhonen, A. (2012). Modelling selectional preferences in a lexical hierarchy. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics*.

Pang, B., & Lee, L. (2008). *Opinion Mining and Sentiment Analysis*. Hanover: Now Publishers.

Paul, M., & Girju, R. (2010). A two-dimensional topic-aspect model for discovering multi-faceted topics. In *Association for the Advancement of Artificial Intelligence*.

Petterson, J., Alex, S., Caetano, T., Buntine, W., & Shravan, N. (2010). Word features for latent Dirichlet allocation. In *Neural Information Processing Systems*.

Ramage, D., Hall, D., Nallapati, R., & Manning, C. (2009). Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of Empirical Methods in Natural Language Processing*.

Resnik, P., & Hardisty, E. (2010). *Gibbs sampling for the uninitiated* (Technical Report UMIACS-TR-2010-04). University of Maryland.

Rosen-Zvi, M., Griffiths, T. L., Steyvers, M., & Smyth, P. (2004). The author-topic model for authors and documents. In *Proceedings of Uncertainty in Artificial Intelligence*.

Salton, G. (1968). *Automatic Information Organization and Retrieval*. New York: McGraw-Hill.

Sayeed, A. B., Boyd-Graber, J., Rusk, B., & Weinberg, A. (2012). Grammatical structures for word-level sentiment detection. In *North American Association of Computational Linguistics*.

Shneiderman, B., Byrd, D., & Croft, W. B. (1997). Clarifying search: A user-interface framework for text searches. *D-Lib Magazine*, *3*(1).

Shoemaker, O. J. (2011). Variance estimates for price changes in the consumer price index. *Bureau of Labor Statistics Report*.

Shringarpure, S., & Xing, E. P. (2008). mStruct: a new admixture model for inference of population structure in light of both genetic admixing and allele mutations. In *Proceedings of the International Conference of Machine Learning*.

Snow, R., O'Connor, B., Jurafsky, D., & Ng, A. (2008). Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of Empirical Methods in Natural Language Processing*.

Stevens, K., Kegelmeyer, P., Andrzejewski, D., & Buttler, D. (2012). Exploring topic coherence over many models and many topics. In *Empirical Methods in Natural Language Processing* (Vol. 20).

Talley, E. M., Newman, D., Mimno, D., Herr, B. W., Wallach, H. M., Burns, G. A. P. C., Leenders, A. G. M., & McCallum, A. (2011). Database of NIH grants using machine-learned categories and graphical clustering. *Nature Methods*, *8*(6), 443–444.

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, *101*(476), 1566–1581.

Teh, Y. W., Daumé, H. III., & Roy, D. M. (2008). Bayesian agglomerative clustering with coalescents. In *Proceedings of Advances in Neural Information Processing Systems*.

Templeton, C. (2011). Topic modeling in the humanities: An overview. *Maryland Institute for Technology in the Humanities Blog*.

Thomas, J. J., & Cook, K. A. (2005). *Illuminating the path: The research and development agenda for visual analytics*. Los Alamitos: IEEE Comput. Soc.

Tversky, A., & Kahneman, D. (1992). Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and Uncertainty*, *5*(4), 297–323.

Wacholder, N., & Liu, L. (2008). Assessing term effectiveness in the interactive information access process. *Information Processing and Management*, *44*(3), 1022–1031.

Wallach, H. M. (2006). Topic modeling: Beyond bag-of-words. In *Proceedings of the International Conference of Machine Learning*.

Wang, C., Blei, D. M., & Heckerman, D. (2008). Continuous time dynamic topic models. In *Proceedings of Uncertainty in Artificial Intelligence*.

Wei, X., & Croft, B. (2006). LDA-based document models for ad-hoc retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

Wu, X., Yu, K., Wang, H., & Ding, W. (2010). Online streaming feature selection. In *International Conference on Machine Learning* (pp. 1159–1166).

Yang, T. I., Torget, A., & Mihalcea, R. (2011). Topic modeling on historical newspapers. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*.

Yao, L., Mimno, D., & McCallum, A. (2009). Efficient methods for topic model inference on streaming document collections. In *Knowledge Discovery and Data Mining*.

Ye, X., Yu, Y. K., & Altschul, S. F. (2011). On the inference of Dirichlet mixture priors for protein sequence comparison. *Journal of Computational Biology*, *18*, 941–954.