

Understanding LDA in Source Code Analysis

David Binkley[†] Daniel Heinz^{*} Dawn Lawrie[†] Justin Overfelt[‡]

[†]Computer Science Department, Loyola University Maryland, Baltimore, MD, USA

^{*}Department of Mathematics and Statistics, Loyola University Maryland, Baltimore, MD, USA

[‡]Booz | Allen | Hamilton, Linthicum, MD, USA

ABSTRACT

Latent Dirichlet Allocation (LDA) has seen increasing use in the understanding of source code and its related artifacts in part because of its impressive modeling power. However, this expressive power comes at a cost: the technique includes several tuning parameters whose impact on the resulting LDA model must be carefully considered. An obvious example is the burn-in period; too short a burn-in period leaves excessive echoes of the initial uniform distribution. The aim of this work is to provide insights into the tuning parameter's impact. Doing so improves the comprehension of both, 1) researchers who look to exploit the power of LDA in their research and 2) those who interpret the output of LDA-using tools. It is important to recognize that the goal of this work is *not* to establish values for the tuning parameters because there is no universal *best setting*. Rather appropriate settings depend on the problem being solved, the input corpus (in this case, typically words from the source code and its supporting artifacts), and the needs of the engineer performing the analysis. This work's primary goal is to aid software engineers in their understanding of the LDA tuning parameters by demonstrating numerically and graphically the relationship between the tuning parameters and the LDA output. A secondary goal is to enable more informed setting of the parameters. Results obtained using both production source code and a synthetic corpus underscore the need for a solid understanding of how to configure LDA's tuning parameters.

Categories and Subject Descriptors: I.2.7 [Natural Language Processing]: Text analysis; I.5.4 [Applications]: Text processing.

General Terms: Algorithms, Experimentation, Measurement.

Keywords: Latent Dirichlet Allocation; hyper-parameters

1. INTRODUCTION

Latent Dirichlet Allocation (LDA) [2] is a generative model that is being applied to a growing number of software engineering (SE) problems [1, 3, 5, 6, 8, 9, 12, 13, 14]. By estimating the distributions of latent topics describing a text corpus constructed from source code and source-code related artifacts, LDA models can aid in program comprehension as they identify patterns both within the code and between the code and its related artifacts.

LDA is a statistical technique that expresses each document as a probability distribution of topics. Each topic is itself a probability distribution of words referred to as a topic model. Words can belong to multiple topics, and documents can contain multiple topics. To date, the LDA tools applied to SE problems have exclusively used Gibbs Sampling [10]. There are alternatives such as Variational Inference, which is faster than Gibbs Sampling with simpler diagnostics but approximates the true distribution with a simpler model [2]. As no alternative has yet to attract interest in SE, this paper focuses on LDA models built using Gibbs sampling.

Two significant challenges accompany the use of LDA. The first, which is often overlooked, is that Gibbs sampling produces random observations (random variates) from a distribution. It is the *distribution* that represents the topic space; *multiple observations* are needed to explore the distribution. By analogy, suppose one wishes to study the distribution of heights in a class of students (e.g., in order to estimate the average height). If the sample is a single random observation, the results could change drastically among samples. For example, if the one observation is a tall child, one would think that children in the class are taller than they actually are. An accurate picture of the *distribution* requires multiple observations.

It is important to recognize the difference between trying to get a point estimate (e.g., of a mean) versus trying to infer a posterior distribution. For example, Expectation-Maximization requires a single observation because it is designed to estimate parameters where the successive estimates converge to a point. By contrast, Gibbs sampling is designed to generate random samples from a complicated distribution. The Gibbs sampler does not converge to a single value; it converges “in distribution.” That is, after sufficient burn-in, the observations will be generated from the desired distribution, but they are still random. Thus, a large sample is required to produce a reliable picture of the posterior distribution.

This issue is illustrated in a software context through a replication of work by Grant and Cordy [3] in which they search for the ideal *topic count*, one of the tuning parameters. Figure 1 shows five observations (the five solid lines) and the average of 48 samples (the bold dashed line) where the highest score indicates the desired topic count. Based on any single observation (any solid line) the analysis concludes that the best topic count is 5, 10, 25, 50, or 75. From the average, it is clear that samples in which 75 is the best choice are quite rare (akin to selecting the particularly tall child).

Those familiar with the work of Grant and Cordy may mistakenly assume that Figure 1 is a replica of their work. However, there is a key difference: the similar figure shown by Grant and Cordy shows a single observation of multiple programs while Figure 1 shows multiple observations of a single program. Figure 1 illustrates that considering only a single observation is insufficient.

Unfortunately, taking multiple samples is rare when LDA is applied in SE, where most users consider only a single observation as

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ICPC '14, June 220133, 2014, Hyderabad, India

Copyright 2014 ACM 978-1-4503-2879-1/14/06 ...\$15.00.

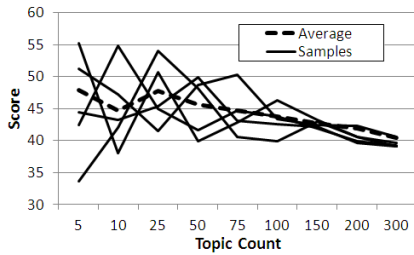


Figure 1: Variation of Samples

“the answer.” One challenge in aggregating multiple samples is the topic exchangeability problem: “there is no a-priori ordering on the topics that will make the topics identifiable between or even within runs of the algorithm. . . . Therefore, the different samples cannot be averaged at the level of topics.” [11].

The second challenge when applying LDA is the number and complexity of the tuning parameters employed. Providing an intuition for these parameters is this paper’s goal. An alternative to such understanding is to automate their setting based on some external oracle. For example, Panichella et al. [9] use clustering metrics as an oracle and a genetic algorithm to identify settings that lead to good clustering (high metric values). This approach relieves an engineer of the burden of setting the parameters and improves performance for applications where clustering is an appropriate oracle. In contrast, the goal of this paper is to provide an understanding of the tuning parameters’ influence and thus to enable their informed setting, which improves the understanding of engineers using LDA and those using LDA-based tools across a wide range of problems. Even those using automatic settings (such as provided by Panichella et al.) will benefit from a better understanding of the LDA parameters chosen by the automatic technique and their implication.

The tuning parameters can be divided into two groups: LDA hyper-parameters and Gibbs sampling settings. The LDA hyper-parameters are features of the model proper, while the Gibbs sampling settings are features of the inference method, used to fit the model. In other words, if a different method is selected, such as Variational Inference, only the LDA hyper-parameters would still be relevant. Hyper-parameters are so named because they are the input to the process whose output is two matrices, which are in turn the input parameters used by the generative LDA process to predict document words. The LDA hyper-parameters α , β , and tc , as well as Gibbs sampling settings b , n , and si are defined as follows [10]:

- α – a Dirichlet prior on the per-document topic distribution
- β – a Dirichlet prior on the per-topic word distribution
- tc – the topic count (number of topics) in the model
- b – the number of burn-in iterations
- n – the number of samples (random variates)
- si – the sampling interval

Without an understanding of the impact of each parameter, it is difficult to correctly interpret an LDA model and, more importantly, evaluate the output of an LDA-using tool. It is possible to gain the necessary insight by studying the underlying mathematics [11, 2]; however, this is intensive work. Visualization provides an effective alternative for illustrating the impact of the parameters. Fortunately, some parameter’s impact is straightforward. To better understand the tuning parameters five research questions are considered; for each of the five tuning parameters p excluding si , the research question “How does p visually impact LDA output?” is considered. Unfortunately, si , the sampling interval, cannot be adequately studied using current tools (see Section 3.5).

After describing LDA and the six tuning parameters in more detail in Section 2, Section 3 considers their impact on production

source code and a controlled vocabulary that makes patterns more evident and easier to describe. Finally, Section 4 takes a further look at the topic count including a replication of the Grant and Cordy study. This replication also hints at future work by considering an implicit parameter, the input corpus. After the empirical results, Section 5 considers related work, and Sections 6 and 7 provide directions for future work and a summary.

2. BACKGROUND

This section provides technical depth behind LDA, a generative model that describes how documents may be generated. Readers familiar with LDA should skim this section to acquaint themselves with the notation and terminology being used, especially regarding the hyper parameters α and β .

LDA associates documents with a distribution of topics where each topic is a distribution of words. Using an LDA model, the next word is generated by first selecting a random topic from the set of topics T , then choosing a random word from that topic’s distribution over the vocabulary W . Mathematically, let θ_{td} be the probability of topic t for document d and let ϕ_{tw} be the probability of word w in topic t . The probability of generating word w in document d , $p(w|d)$, is $\sum_{t \in T} \theta_{td} \phi_{tw}$.

Fitting an LDA model to a corpus of documents means inferring the hidden variables ϕ and θ . Given values for the Gibbs’ settings (b , n , si), the LDA hyper-parameters (α , β , and tc), and a word-by-document input matrix, M , a Gibbs sampler produces n random observations from the inferred posterior distribution of θ and ϕ .

To elucidate the roles of the various Gibbs settings and LDA hyper-parameters, this section describes the effect of each when the others are held constant. The roles of b and n are straightforward. Gibbs sampling works by generating two sequences of random variables, $(\theta^{(1)}, \theta^{(2)}, \dots)$ and $(\phi^{(1)}, \phi^{(2)}, \dots)$, whose *limiting* distribution is the desired posterior distribution. In other words, the Gibbs sequences produce θ ’s and ϕ ’s from the desired distribution, *but only after a large number of iterations*. For this reason it is necessary to discard (or burn) the initial observations. The setting b specifies how many iterations are discarded. If b is too low, then the Gibbs sampler will not have converged and the samples will be polluted by observations from an incorrect distribution.

The Gibbs setting n determines how many observations from the two Gibbs sequences are kept. Just as one flip of a coin is not enough to verify the probability that a weighted coin lands heads, the Gibbs sampler requires a sufficiently large number of samples, n , to provide an accurate picture of the topic space. If n is too small, the picture is incomplete. In particular, it is inappropriate to take a single observation ($n = 1$) as “the answer.” Any single observation is next-to-useless on its own.

The third Gibbs setting, si , controls how the resulting samples are *thinned*. A Gibbs sampler may be afflicted by autocorrelation, which is correlation between successive observations. In this case, two successive observations provide less information about the underlying distribution than independent observations. Fortunately, the strength of the autocorrelation dies away as the number of iterations increases. For example, the correlation between the 100th and 103rd values is weaker than the correlation between the 100th and 102nd values. The setting si specifies how many iterations the Gibbs sampler runs before returning the next useful observation. If si is large enough, the observations are practically independent. However, too small a value risks unwanted correlation. To summarize the effect of b , n , and si : if any of these settings are too low, then the Gibbs sampler will produce inaccurate or inadequate information; if any of these settings are too high, then the only penalty is wasted computational effort.

Unfortunately, as described in Section 6, support for extracting interval-separated observations is limited in existing LDA tools. For example, Mallet [7] provides this capability but appears to suffer from a local maxima problem¹. On the other hand, the LDA implementation in *R*, which is used to produce most of the presented data, does not support extracting samples at intervals. It does compute, under the option “document_expects” the aggregate sum of each θ but, interestingly, not ϕ , from each post burn-in iteration. However, this mode simply sums topic counts from adjacent iterations, which is problematic for two reasons. First, it does not take into account topic exchangeability [11] where a given column in the matrix need not represent the same topic during all iterations. Second, it does not provide access to models other than the final model; thus, there is no support for extracting samples at intervals.

An alternate approach, used in this paper, is to run the Gibbs sampler n independent times, recording the last observation of each run. This approach currently has broader tool support and guarantees independent observations, though it does require more cycles than sampling at intervals.

Turning to the LDA hyper-parameters, the value of α determines strength of prior belief that each document is a uniform mixture of the tc available topics. Thus, it influences the distribution of topics for each document. Mathematically, α can be interpreted as a prior observation count for the number of times a word from topic t occurs in a document, before having observed any actual data. When α is large, the prior counts drown out the empirical information in the data. As a result, the posterior distribution will resemble the prior distribution: a uniform mixture of all tc topics. Conversely, if α is small, then the posterior distribution will attribute most of the probability to relatively few topics. In summary when α is large, nearly every document will be composed of every topic in significant amounts. In contrast when α is small, each document will be composed of only a few topics in significant amounts.

In a similar way, the role of β influences the word distribution within each topic. A high value for β leads to a more uniform distribution of words per topic. Said another way, larger values of β favor a greater number of words per topic, while smaller values of β favor fewer words per topic.

Finally, the impact of the number of topics tc is more challenging to characterize but is important as it affects the interpretation of results [11]. Too few topics result in excessively broad topics that include “too much.” That is, topics will tend to capture some combination of multiple concepts. On the other hand, too many topics leads to diluted meaningless topics where a single concept is spread over multiple topics. These topics can become uninterpretable as they pick out idiosyncratic word combinations [11].

3. UNDERSTANDING LDA’S TUNING

Of the tuning parameters described in Section 2 gaining an appreciation for the impact of α and β is key. The central challenge here is to distinguish between topics that concentrate the majority of probability on a single word (or document) from topics that are more uniformly distributed across the entire vocabulary (within the documents). Mathematically, this distinction is captured by the Shannon Entropy of the probabilities in the LDA model. The entropy of a probability distribution measures the amount of randomness. It is defined as the expected number of bits needed to express a random observation from the distribution.

Entropy is maximized when the distribution is uniform and minimized as probability is concentrated on only a few points. In ϕ , the entropy H_t of a topic t is

¹An illustration can be found at www.cs.loyola.edu/~binkley/topic_models/additional-images/mallet-fixation

Table 1: Programs studied

Program	Functions	Size (LoC)	Vocabulary Size
JHotDraw 0.22	6,945	289K	8,180
JabRef 2.6	5,112	117K	12,879
cook 2.0	1,304	86K	7,743
snns 4.2	2,224	105K	13,002
gzip 4.2	46	9K	3,184
PostgreSQL 7.0.0	4535	294K	25,004

$$H_t = - \sum_{w \in W} \phi_{tw} \log_2(\phi_{tw}) \quad (1)$$

where W is the set of vocabulary words. The entropy of the topics in θ has the corresponding definition. When the topic distribution is uniform, the entropy reaches its maximum value of $\log_2(|W|)$. Thus, there is no way to predict the next word since any choice is equally likely. When the topic is concentrated on a few dominant words, one has a strong idea of which word will be next for a given topic. As a result, the entropy is small. In the extreme case, if $\phi_{tw} = 1$ for word w , then the next word is no longer random, and the entropy is zero. To summarize, the entropy of a topic in ϕ indicates how many high probability words it contains.

To examine the effect of α and β on the topic distributions, the average entropy over all topics across the entire sample of ϕ s is used. Likewise, the average entropy for all documents across the entire sample of θ s is used to understand the effect of α and β on θ , where (as with ϕ) large entropy indicates an emphasis on more uniform distributions. Henceforth, “entropy in ϕ ” refers to “the average entropy of all topics in a sample of ϕ s” and “entropy in θ ” refers to “the average entropy of all documents in a sample of θ s.”

When looking at the entropy in ϕ , a higher value indicates a more uniform distribution, which is demonstrated by having many important words per topic. As β increases, the entropy approaches $\log_2(|W|)$, where the uniform prior drowns out the observed data. With small values of β , lower entropy accompanies a non-uniform distribution, which means fewer high-probability words are in each topic. Thus, entropy is expected to increase as β increases.

Considering the entropy of source code requires some source code. The code used in the empirical investigations, shown in Table 1, was chosen to facilitate its future comparison with related work [13, 3]. When analyzing source code, the *documents* used in the LDA model are typically the functions, methods, files, or classes of the source code. In this study C functions and Java methods are used as the documents.

Figure 2 shows the two entropy graphs for the program *cook*, which are representative of the graphs for the programs in Table 1. In the top plot, the non-bold lines show the average entropy in the observed topic-word distributions across the entire Gibbs sample of ϕ s plotted against the value of β on a log-scale. Each non-bold line shows a different α value, with the largest value at the bottom and the smallest value at the top. The bold line shows the average entropy across all values of α . Similarly, the bottom plot shows the average entropy in the document-topic distribution against α , where the bold line combines all choices of β for a particular choice in α . Each non-bold line shows a different β value, with the largest value at the bottom and the smallest value at the top.

The lines of both graphs have similar shapes, although the lower ones are smoother because more values of α are considered. The biggest difference between the patterns in the graphs is that the entropy for θ exhibits a ceiling effect that is not seen in the graph

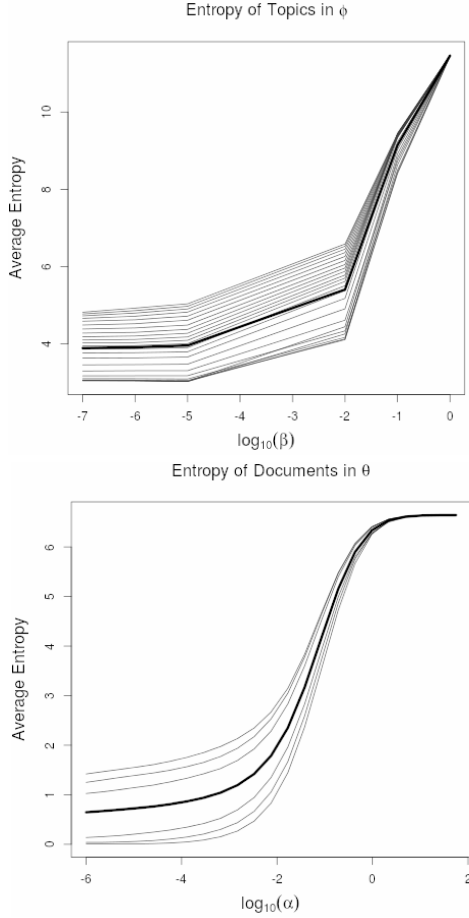


Figure 2: Entropy of ϕ and θ for the program cook (shown using a log scale for β and α on the x -axis of the upper and lower graph, and using a linear scale for entropy on the y -axis). Each non-bold line represents a fixed value for the parameter not represented by the x -axis where the bold line is the average.

for ϕ . In fact, both curves are S-shaped, but the top part of the S is outside the range of the graph for ϕ . This is because the maximum entropy for a document (in θ) is $\log_2(tc)$, but the maximum entropy for a topic (in ϕ) is $\log_2(|W|)$. In general, the number of words is much larger than the number of topics, so that the ceiling for the ϕ graph is much higher. When suitably large values of β are used, the entropy of ϕ would eventually reach its ceiling.

Starting with the upper graph for ϕ , as β grows larger, the entropy goes from low (topics of a less uniform distribution dominated by a few words) to high (topics of a more uniform distribution more likely to generate a wider range of words). Notice that β , which directly influences the prior distribution of ϕ , has a larger effect than α on the entropy in ϕ , as are shown in the top graph of Figure 2. Each line in this graph shows the effect of β for a fixed value of α . As β ranges from 10^{-7} to 10^0 , the entropy increases from the 3 to 5 range to a maximum around 11. The effect of α for a fixed value of β is seen by observing the width of the band of lines for a particular position on the x -axis. The widest range is only about 0.5 (from 2.5 to 3), which corresponds to the biggest effect α exhibits. From this analysis, it is clear that β has a much larger impact on the entropy of ϕ .

Turning to θ , the opposite hyper-parameter has the dominant effect. Entropy increases dramatically as α increases, while the impact of β is small and diminishes as α increases. Thus, as α increases, documents tend to include more topics, which corresponds

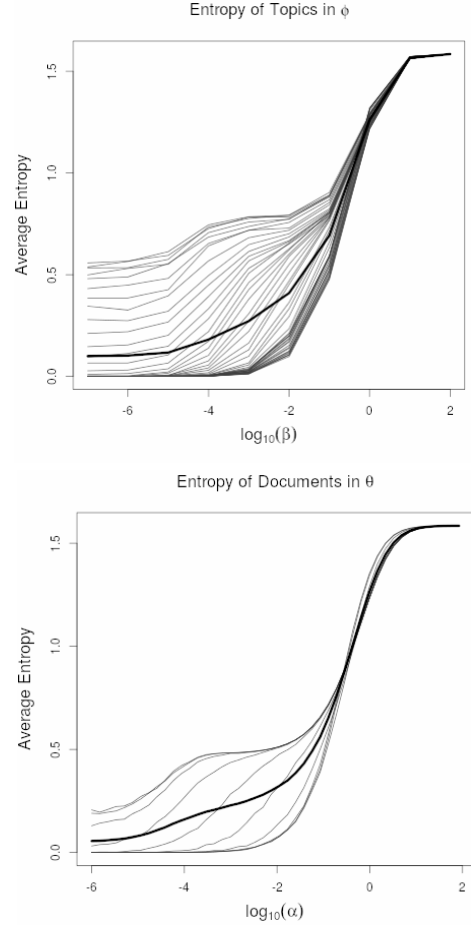


Figure 3: Entropy of ϕ and θ for the controlled vocabulary.

to higher entropy. This pattern is repeated for all values of β . Thus, for θ , higher values of α bring all documents closer to containing all topics. To illustrate the value of understanding these settings, consider a software engineer tasked with refactoring some code using an LDA-based tool where each topic captures a concept from the code. In this case the engineer should choose a smaller value for α ; thus causing documents (functions) to receive a dominant topic and consequently suggesting a refactoring.

Figure 2 also shows the interesting interaction between α and β . The hyper-parameter α has a much bigger impact on ϕ when the value of β is smaller. Conversely, β has a much bigger impact on θ when α is small. Reasons for this are discussed in Section 3.6.

3.1 Control Vocabulary Experiment

Following the advice of Steyvers and Griffiths, a deeper understanding of the hyper-parameters is illustrated by “generating artificial data from a known topic model and applying the algorithm to check whether it is able to infer the original generative structure” [11]. The rest of this section uses a small fixed vocabulary to study in greater detail the impact of n , b , si , tc , α , and β on the distributions in θ and ϕ . Figure 3 shows the entropy graphs for the controlled vocabulary. The key observation is that the graphs show the exact same trend as seen in Figure 2. Hence, the intuition gained from the controlled vocabulary can be transferred to models for source code. The slight variation in the shape of the curves of Figure 3 results from the small size of the controlled vocabulary.

Document	Name	Contents
Doc 1	A	aa aa aa aa
Doc 2	B	bb bb bb bb bb
Doc 3	AB	aa aa aa bb bb bb
Doc 4	AC	aa aa aa aa cc cc

Figure 4: Vocabulary for the controlled vocabulary experiment.

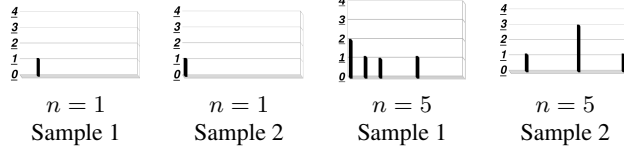


Figure 5: The need for multiple samples

The fixed vocabulary, shown in Figure 4, includes three words aa, bb, and cc organized in four documents (A, B, AB, and AC) whose names reflect their contents (e.g., Document A contains only word aa). To better understand the influence of α and β , it is necessary to first consider the Gibbs settings. While important, their impact is more straightforward. The example was designed so that when using $tc = 3$, a topic could be dedicated to each word.

3.2 Sampling Study

It is important to emphasize that the goal of this study is not to determine the “correct” value for n , the number of observations, but rather to illustrate its impact. When considering the impact of n , fixed values for α , β , b , si , and tc are used. The specific values were chosen to illustrate various patterns. Other choices show the same patterns, although sometimes not as pronounced.

One of the most common errors made when applying LDA in SE is to set $n = 1$, taking a single *observation* as the *topic distribution*. The Gibbs sampling process is doing exactly what its name implies, *sampling*. Thus, a *sample* of observations is **required** to gain an understanding of the posterior distribution. To illustrate, Figure 5 shows four charts that all use excessively small values of n . In all four charts, the y -axis shows the sample count where more common samples have larger counts. The x -axis shows possible outcomes for ϕ (the same pattern occurs with θ). The x -axis is shown unlabeled since particular values are unimportant.

The left two charts show samples containing a single observation ($n = 1$), and the right two charts show samples containing five observations ($n = 5$). From these four charts several conclusions can be drawn. First, using only one observation gives *two completely different answers*: the bars in the left two graphs are in different columns (i.e., different topic models.) Second, while there is some overlap with $n = 5$, it is not evident that the two charts are samples from the same distribution. For example, the tallest bar in each of the $n = 5$ charts is in a different column. The important conclusion is that when n is too small, an insufficient picture of the distribution is attained. Thus choosing too small a value for n leads to potentially misleading (effectively random) results.

While the goal of this paper is to understand the impact of the parameters, rather than establish appropriate values for them, it is clear from the data that too small a value of n leads to an incomplete picture. One method to make sure that n is sufficiently large is to compare several independent samples of size n . If the downstream results of the analysis are suitably consistent, then one knows that the samples are large enough. At this point, one final sample should be generated using the determined sample size. (The reason for drawing one more sample is that it is statistically inappropriate to use observations both for validating settings and for inference.) As

Table 2: JHotdraw’s color gradient topic across multiple samples

Topic Number	Top 15 Words
22	color gradient space index compon rgb model system min pixel math max red paint arrai
7	color gradient space index model rgb pixel src paint data param arrai alpha awt focu
34	color gradient space index compon rgb model system min pixel paint red math max focu

a conservative choice, the charts generated in the remainder of the paper use a minimum value of 1000 for n , which is likely excessive.

3.3 Canonical Labels for LDA Topics

Before considering the impact of the next parameter, the burn-in b , a heat-map visualization of the data is introduced. One challenge in aggregating, let alone graphing topic models, is that there is no a-priori ordering of the topics that makes them identifiable between samples. As Steyvers and Griffiths observe, one cannot simply match the topic id numbers over different samples [11]. An illustration of the *topic exchangeability problem* is seen in the topic evolution work of Thomas et al. [13]. A replication of their work (shown in Table 2) tracks the topic containing the words color gradient. This data shows that this topic is assigned varying topic numbers across different observations.

Fortunately, for the controlled vocabulary, it is possible to establish topic identity using dominant words and dominant documents. For the topic-word distribution ϕ , topics are labeled by their high-probability words. High probability is established using a cut-off value. For example, consider a topic with the three words “goalie”, “ball”, and “player” that are assigned the probabilities 0.50, 0.45, 0.05, respectively. Using the cut-of of 0.10 for “high probability,” this topic is named ball-goalie. To provide a canonical name, the words of the name are sorted alphabetically. In subsequent observations, any topic that has only probabilities for “goalie” and “ball” greater than the cut-of will be given the same name. These names lead to identifiable topics across observations.

Extending this notion, a set of topics ϕ can be represented by a collection of names. Since the topics have no prescribed order, a canonical order is produced by sorting the names of each topic. Thus the topics of a particular observation ϕ are described by a sorted list of (sorted) topic names. For example, the tall bar in the center of the rightmost chart shown in Figure 5 has the name “aa-bb-cc aa bb,” which represents three topics named aa-bb-cc, aa, and bb, respectively. This naming scheme makes it possible to visualize the topic-word distribution by counting how often each name-list occurs in the n observations.

A similar approach is used for the topic-document distribution, θ , where a topic is named using documents for which it is assigned high probability. This may initially seem backwards, as it is more common to speak of which topics make up a document. However, the documents are fixed while the topic ids suffer from the topic exchangeability problem. Thus, a topic in θ is named using the sorted list of those documents that place a “high” probability on that topic. For example, suppose Documents 1-4 assign probabilities of 0.44, 0.55, 0.05, and 0.24 to a topic. Using a cut-off of 0.10, this topic would be named A-B-AC by combining the names of Documents 1, 2, and 4. Similar to ϕ , θ is labeled by a sorted list of topic names.

As shown in Figure 5, a sample of observations can be visualized using a bar chart where the x -axis shows topic names (either lists

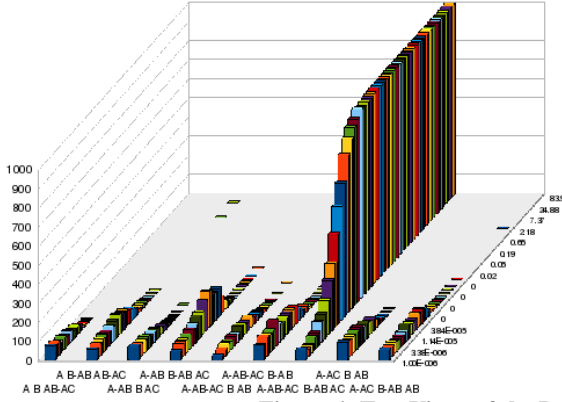


Figure 6: Two Views of the Data where α is varied

of lists of words for ϕ or lists of lists of documents for θ), and the y -axis is a count of the number of observations that produce a given name. An example is shown in Figure 6 using the documents from Figure 4. This figure displays the results from 46 different α values which are displayed on the z -axis. Because α ranges over eight orders of magnitude the non-linear power series $10^{-6} * (9/4)^z$ for $z = 0$ to 45 is used. Each sample was produced from $n = 1000$ observations. In total, the 3D chart shown in Figure 6 summarizes 46,000 observations. For clarity, topic models observed less than 1% of the time are not shown, which is fewer than 460 samples.

The chart clearly shows that choosing different values for α has a dramatic influence on the topic models. Using larger values of α , the dominant name list is A-AB-AC B-AB AC, corresponding to three topics: a topic dominant in the three documents A, AB, and AC (capturing the word aa), a topic dominant to B and AB (capturing the word bb), and a topic dominant to AC (capturing the word cc). This validates using controlled vocabulary as suggested Steyvers and Griffiths [11].

The same patterns can be seen using heat maps where gray scale replaces the z -axis (e.g., as seen in the right of Figure 6). Although the 3D view better highlights contour differences, it can obscure values hidden behind tall bars. For example, the encircled light grey bar to the left of the top of the black bar in the heatmap of Figure 6 is hidden in the 3D chart of the same data. Heatmaps are used in the remainder of the data presentation.

3.4 Burn-In

When considering the burn-in interval b , the following values are used: $n = 1000$, $tc = 3$, $si = \infty$ (see Section 3.5), and $\beta = 10^{-2}$, (these values aid in the presentation as they produce sufficiently interesting structure). Other values show the same pattern, although less pronounced.

The Gibbs sampling process starts by assigning each topic a probability of $1/tc$ under each document, and each of the words in the input corpus a probability of $1/|W|$ under each topic where W is the set of unique words in the corpus; thus, the early samples are not useful as too much of this initial uniform assignment remains. For ϕ , the heat map for this initial assignment appears as a single black line with the name list aa-bb-cc aa-bb-cc aa-bb-cc. The initial value for θ also produces a single black line.

The burn-in study considers values of b ranging from 1 to 65,536 in powers of two. A subset of these are shown in Figure 7.² As is evident in the chart for $b = 1$, after a single iteration the initial con-

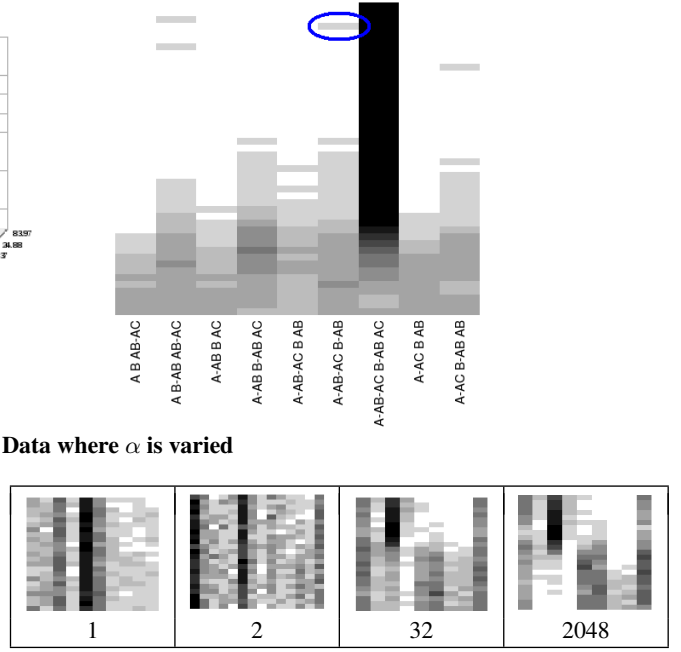


Figure 7: ϕ Burn-in Study (θ shows the same pattern)

figuration (the single black line) is still dominant. However, there are also nine other columns in the graph indicating some influences from the data. Early on this variation increases with more possibilities appearing on the x -axis (more columns). By iteration 32 the final distribution is becoming evident (e.g., there are fewer columns in this graph than the one for 2 iterations). This is because some of the topics fall below the 1% threshold. By iteration 2048 the *distribution* of ϕ has stabilized and remains unchanged through the rest of the 65,536 iterations. This experiment was replicated several times. While there are minor variations in the early iterations, the same general pattern emerged: a differing initial explosion and then arrival at the same final distribution.

Based on this data subsequent experiments were designed to have a minimum burn in of $b = 2000$. A smaller value is likely acceptable; thus making $b = 2000$ a conservative choice. (As noted in Section 3.5, the effective burn in turned out to be $b = 22,000$.)

3.5 Sampling Interval

The final Gibbs setting is the sampling interval, si . Given an implementation capable of correctly extracting multiple variates, the algorithm would iterate b times and then start sampling, taking n samples at intervals of si . These samples would be taken at iterations $b + si, b + 2 * si, \dots, b + n * si$. However, because sampling proved problematic with three LDA tools available (see Section 6), samples were collected by constructing n separate models, using the final iteration as the observation (the random variate). This leaves the empirical study of si to future work after tools better support sampling. Because the original design called for each execution to yield $n = 1000$ samples with a sample interval up to $si = 20$, each execution generates one data point with an effective burn-in of $b = 22,000$.

3.6 Impact of α and β

This section considers the impacts first of β , then of α . The other parameters are fixed with $b = 22,000$, $n = 1000$, and $tc = 3$. The initial discussion focuses on the length and frequency of the name lists that label the x -axis of the heat maps for ϕ (top row) and θ

²The remaining graphs can be found at www.cs.loyola.edu/~binkley/topic_models/additional-images/burn-in-study.

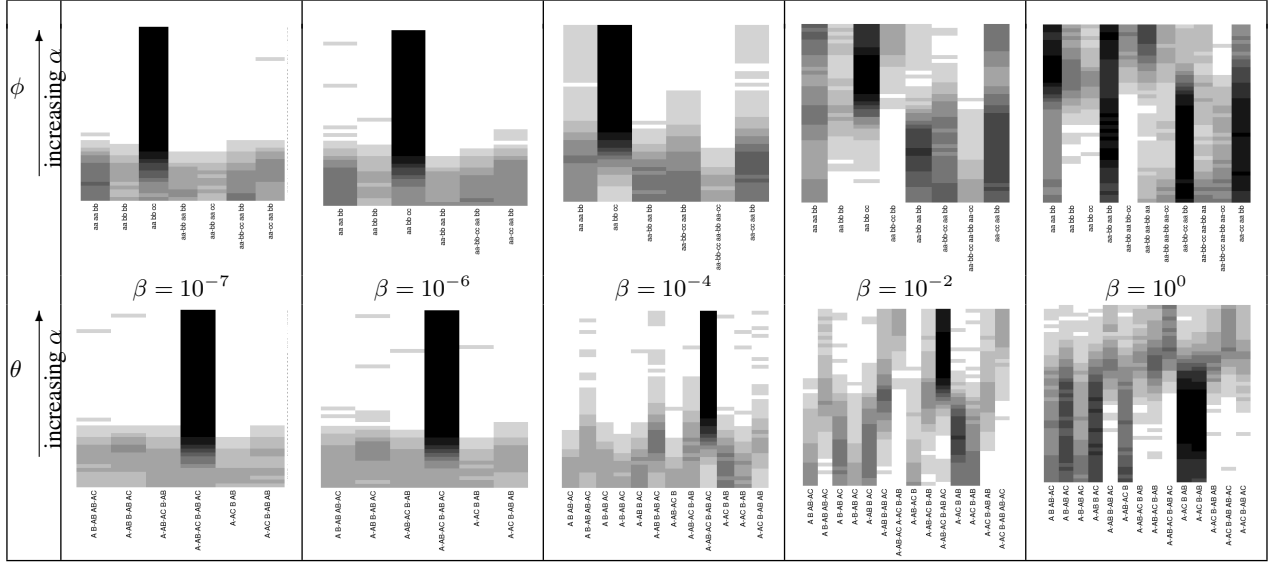


Figure 8: Impact of α and β (Top: ϕ ; Bottom: θ)

(bottom row) of Figure 8. The y -axis on each heat map shows α , which is the non-linear power series, $10^{-6} * (9/4)^z$ for $z = 0$ to 45. Each vertical pair of heat maps (those in the same column) show a given value of β .

To begin with, a smaller value of β favors fewer words per topic. In the top row of Figure 8, the heat maps concentrate on $\phi = \{aa\ bb\ cc\}$, for which each topic is dominated by a single word. For very low values of β , only seven distinct name lists appear (recall that the heat maps are filtered to only show word-topic matrices that represent at least 1% of the observations). Three of these consist solely of single word topics. In the remaining four, only a single topic shows multiple words. Values of ϕ containing more than one multi-word topic do not appear until $\beta = 10^{-4}$. As β increases, multi-word topics appear more commonly. At $\beta = 10^2$ (not shown), each value of ϕ contains at least one multi-word topic.

Another way of expressing the impact of high β is that it encourages greater uniformity (i.e., greater influence from the prior). Greater uniformity leads to topics that include multiple significant words. This pattern is evident in the $\beta = 10^0$ heat map by the existence of very long topic names. In this heat map, the second column from the right has the name list `aa-bb-cc aa-bb aa-cc`, which is two names shy of the maximum `aa-bb-cc aa-bb-cc aa-bb-cc`.

Similar patterns are evident in the topic-document heat maps (bottom row of Figure 8). In this case, topics with more dominant words can be combined in a greater number of ways. As an extreme example, suppose that the topics are $t_1 = aa$, $t_2 = bb$, and $t_3 = cc$. Since Document 3 contains both `aa` and `bb`, it will almost certainly place significant weight on topics t_1 and t_2 . On the other hand, if every topic has a high probability of each word (i.e., `aa-bb-cc`), it is possible for Document 3 to consist of only a single topic. Thus, the number of entries on the x -axis increases as β increases, because each topic is potentially relevant to more documents. The lesson here is that, while β directly controls the prior for ϕ and α directly controls the prior for θ , both tuning parameters influence both matrices due to the dependence between the per-topic word distributions and the per-document topic distributions.

Turning to α , when α is small the LDA model emphasizes fewer topics per document. This creates pressure to have complex topics or, in other words, topics with multiple words that can individually generate the words of a given document. Conversely, as α

increases the LDA model moves toward encouraging more topics per document. This enables (but does not require) simpler topics with a smaller number of dominant words. These patterns are clear in the upper $\beta = 10^{-2}$ heat map for ϕ . For example, the longest name list, `aa-bb-cc aa-bb aa-cc`, labels the 7th column while one of the shortest `aa bb cc`, labels the 3rd column. From the heat map, models that include the long name list are more common at the bottom of the heat map where α is small and grow less common (a lighter shade of gray) as α increases. In contrast, column three grows darker (more models have these topics) as α increases. This pattern is quite striking with smaller values of β . In particular, when $\beta = 10^{-7}$ and α is large, the only value of ϕ that appears is `aa bb cc`. A software engineer using an LDA-based tool to, for example, summarize methods might prefer this setting as each topic is dominated by a small number of key words.

In the topic-document heat map (θ , bottom) for $\beta = 10^{-2}$, the pressure toward multiple-topic documents when α is large is evident in the column with the black top. This column captures models with the name list `A-AB-AC`, `B-AB`, and `AC` where two of the topics have multiple significant documents. These topics have the single significant words `aa`, `bb`, and `cc`, respectively. The fourth column from the right has the name list `A-AC B AB`. This list is the most frequent of the short name lists and is clearly more common where α is small because the pressure to have fewer (but more complex) topics for each document leads to topics being significant to fewer documents (i.e., yielding shorter topic name lists).

The heatmaps provide an intuition for the parameters, but are only effective for the controlled vocabulary. They do not scale well to larger data sets. For example, even the smallest vocabulary of `gzip` includes 3,184 unique words, resulting in $2^{3184^{tc}}$ possible names for ϕ . Recall that the motivation behind the topic names was to identify patterns in the dominant words of the topics of ϕ and the dominant documents for a given topic of θ . Fortunately, this distinction is also captured in the Shannon Entropy of the probabilities in the LDA model. For example, topics with long names have many high probability words and therefore a larger Shannon Entropy, but topics with short names have few high probability words and a smaller Shannon Entropy.

The entropy graphs for the controlled vocabulary, shown in Figure 3 mirror the patterns seen in the heat maps. For example, in the

upper graph for ϕ , as β grows larger, the entropy goes from low (short topic names corresponding to a less uniform distribution) to high (more multi-word topic names). As expected β , which directly specifies the prior distribution of ϕ , has a larger effect than α on the entropy in ϕ . That is, a larger change in ϕ 's entropy is observed when fixing α and varying β , than when the reverse is done. This is particularly true for larger values of β .

Encouragingly, the same pattern is seen in Figure 8. For example, in the heat map for ϕ when $\beta = 10^{-7}$, more of the observations have longer names (signifying higher entropy) at the bottom when α is small than at the top where α is large (where the three topics are each dominated by a single word). Far more dramatic is the increase in topic-name length as β increases. The prevalence of observations with the longest names by the time $\beta = 10^0$ corresponds to more uncertainty regarding the word chosen for a given topic, which in turn corresponds to higher entropy. Finally, with larger values of β , the effect of α diminishes. Here the prior drowns out the data and entropy increases to the maximum possible value of $\log_2(|W|)$.

Turning to θ , the opposite hyper-parameter has the dominant effect: entropy increases dramatically as α increases, while the impact of β is small and diminishes as α increases. This can be seen in the heat map for θ with $\beta = 10^{-7}$ in Figure 8 where the longest name A-AB-AC, B-AB, AC grows more common (its bar gets darker) as α increases. Thus as α increases longer names are more common, which corresponds to higher entropy. This pattern is repeated for all values of β . For example, when $\beta = 10^0$, the topics with longer names become more common (their bars get darker) toward the top of the chart where α is larger. Thus, for θ , higher values of α bring all documents closer to containing all topics (i.e., long topic names are more common). When α is small, documents tend to focus on a few particular topics.

In summary, the variety seen in Figure 8 clearly illustrates that care is needed when choosing values for α and β . One goal of this paper is to visually provide an intuition for their impact on LDA models (rather than an attempt to establish “best” values). Clearly, the “best” values for α and β are problem- and goal-dependent. For example, when trying to generate concept labels, a smaller value of β would lead to topics that included fewer words in the hope of providing more focused concept labels. In contrast, when performing feature location a larger value for α would encourage more topics per document and thus increase recall as a topic of interest would be more likely to be found in the code units that truly implement the feature. In summary, both the problem being solved and the objectives of the software engineer using the solution impact the choice of α and β . This underscores the values of the goal of this paper: to improve engineer and researcher comprehension regarding their impact in order to improve their use of LDA models in software engineering.

3.7 Topic Count

The final LDA parameter considered is the topic count. The models considered above with the controlled vocabulary use three topics, which is large enough for each word to have its own topic depending on the influences of α and β . This flexibility helps to illustrate the impact of α and β .

Steyvers and Griffiths observe that “the choice of the number of topics can affect the interpretability of the results [11].” To summarize their conclusions, too few topics results in very broad topics that provide limited discrimination ability. On the other hand, having too many topics leads to topics that tend to capture idiosyncratic word combinations. They go on to note that one way to determine the number of topics is to “choose the number of topics that leads

to best generalization(sic) performance to new tasks [11].” This is in essence the approach used by Panichella et al. [9] and by Grant and Cordy [3] who tune topic count to a specific task. The next section considers a source-code example.

At one extreme, with a single topic the model degenerates to a unigram model, which posits that all documents are drawn from a single distribution captured by ϕ . At the other extreme an infinite number of topics is possible because two topics need only differ in the weights they assignment to the words. The controlled vocabulary is too small to effectively study the topic count. A more meaningful empirical consideration is given in the next section using production source code.

4. TOPIC COUNT IN SOFTWARE

To further study the impact of topic count, a replication of a Grant and Cordy experiment is presented. This is followed by a look at topic count’s impact on entropy. Grant and Cordy’s goal is to determine “a general rule for predicting and tuning the appropriate topic count for representing a source code” [3]. This is done using heuristics to evaluate the ability of an LDA model to identify related source code elements. In its strictest application, the heuristic considers two functions *related* if they are in the same source-code file. This can be relaxed to consider two functions as related if they are in the same directory, have the same parent directory, etc. In this replication, *related* is defined as “in the same directory.”

For a given function f , an LDA model is used to identify the top N functions related to f . This identification is done using cosine similarity between each function’s topic vector and is described in greater detail by Grant and Cordy [3]. The quality of the LDA model is measured by counting the number of the top N functions that are related according to the *related* heuristic. The count ranges from one (a function is always related to itself) to N . The average count over all functions is the *score* assignment to the LDA model.

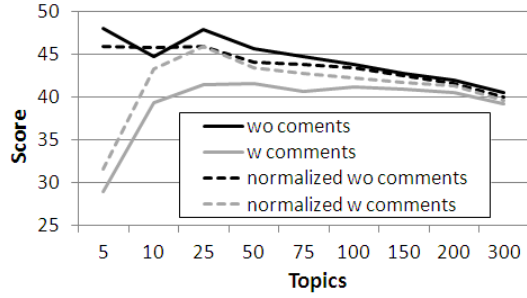
To assess the impact of topic count, Grant and Cordy compare LDA model scores for a range of topic counts, using several programs including PostgreSQL (<http://www.postgresql.org>), which is used in this replication. While the degree of sampling in the original is unclear (unless otherwise noted) each data point in the figures of this section represents the average of $n = 48$ observations.

In the original experiment, comments were stripped from the source code. To enable consideration of their impact, the replication considers models with and without comments. In addition, it considers the impact of vocabulary normalization on each corpus [4]. Normalization separates compound-word identifiers and expands abbreviations and acronyms so that the words from identifiers better match the natural language found elsewhere (such as the requirements or comments).

The resulting scores for each of the four corpora are shown in Figure 9 for both $N = 10$ and $N = 50$. The impact of having too-large a topic count is evident, especially with $N = 50$. Past $tc = 50$ topics, all four corpora show a steady decline. The impact of too small a topic count is more pronounced in the upper graph, where including comments tends to increase the size of the vocabulary. Comparing the solid black line (without comments) and the solid gray line (with comments), early on there are too few topics in the model, which yields lower scores for the larger vocabulary. The lack of discrimination ability is evident in these broad topics. Consistent with Grant and Cordy’s results, these data illustrate the consequences of choosing too few or too many topics.

Finally, normalization’s impact can be seen by comparing corresponding solid and dashed lines. For the comment-free corpora (black lines) normalization has little impact. This is in essence

Replication with $N = 10$ (min score = 10, max score = 10^2)



Replication with $N = 50$ (min score = 50, max score = 50^2)

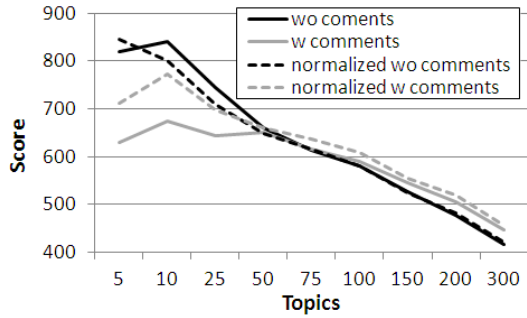


Figure 9: Grant and Cordy Replication

because universally replacing an abbreviation such as `cnt` with its expansion, `count`, does not change the word frequencies used to build the LDA model. Rather, it simply swaps one word for another. However, when comments are included, they tend to include “full words” and thus replacing `cnt` with `count` brings the vocabulary of the source code more in line with that of the comments. This change does impact the counts used to build the LDA model. In essence there is a reduction and a focusing of the vocabulary leading to a higher score. For example, rather than requiring both `cnt` and `count`, after normalization `count` suffices.

Another way of considering topic count returns to the entropy graphs. Figure 10 shows the entropy graphs for three of the six programs from Table 1 using an LDA model with either 25 or 100 topics. The charts for 50, 200, and 300 topics show the same trends. An additional program, `cook`, is shown in Figure 2. The graphs for the two remaining programs show the same pattern.

The graphs in Figure 10 show interesting interactions among tc , β , and α . Comparing a specific program’s two entropy graphs for ϕ , reveals that the higher topic count intensifies the effect of β . Notice that the y -axis ranges from a lower number when $tc = 100$ compared to $tc = 25$. Intuitively, when words are split among $tc = 100$ topics, each topic may concentrate on relatively fewer words, compared to when there are only 25 topics. As explained in Section 3, this results in lower entropy for $tc = 100$, as the topic is more predictable.

While tc affects the minimum observed entropy, it does not change the maximum. For a model with $|W|$ words, the entropy for a topic has an upper limit of $\log_2(|W|)$; it is defined by the number of words, but not the number of topics. When β is set very high, it pushes each topic toward a uniform distribution of words and the entropy toward the upper limit. Interestingly, the topic count affects how *quickly* the entropy reaches this limit. In other words, when $tc = 100$, the entropy graphs for ϕ start lower, but approach the maximum more quickly. For example, looking at the entropy curves for PostgreSQL, the $tc = 100$ graph exhibits a steeper slope in the middle piece of the S-curve, which is where the increase in

entropy is most prominent. As a result, when considering a large value of β , the entropy tends to be higher when $tc = 100$. Specifically, the entropy at $\beta = 10^{-1}$ is higher when $tc = 100$ for all three programs in Figure 10. As a minor note, the interaction between β and tc causes a small quirk in the entropy graphs for `gzip`. Namely, every entropy curve has a ceiling effect as the entropy approaches its maximum possible value. At some threshold, increasing β or α cannot have a large effect on the entropy because it is almost maximized already. This results in the prominent S-pattern in the graphs for θ . However, most of the graphs for ϕ do not show this pattern because β does not reach the threshold. Essentially, only part of the S-pattern is inside the range of the graph. The ceiling effect only appears for `gzip` when $tc = 100$ because the small vocabulary size results in a low ceiling. The large number of topics causes the entropy to approach this ceiling rapidly.

The interaction between topic count and α is straightforward. For a given value of α , the entropy is larger when tc increases from 25 to 100. This is most obvious for large values of α , when the entropy is near its maximum, $\log_2(tc)$. Thus, the graphs with 100 topics have a higher ceiling. Nevertheless, the effect of tc is similar for small values of α . A more subtle facet of the interaction is that the effect of tc for large values of α seems to persist even when taking into account the different maximum values, where the model is closer to the uniform distribution when tc is larger.

5. RELATED WORK

Two recent studies have considered determining various subsets of the tuning parameters. The first is the work replicated in Section 4 where Grant and Cordy determine the optimal number of topics for 26 programs [3]. Section 4 considered just one of these. They conclude by noting that “the number of topics used when modeling source code in a latent topic model must be carefully considered.” The impact of topic count on model quality is visually evident in Figure 9. One question Grant and Cordy raise is the value of retaining comments. This question was considered in the replication described in the previous section.

Second, the work of Panichella et al. [9] employs a genetic algorithm to identify “good” hyper-parameter values. The challenge here is (automatically) judging “goodness.” They do so by mapping the LDA output to a graph and then applying a clustering algorithm where existing clustering metrics are used as a proxy for “good”. Here the use of clustering drives the LDA models to favor single thought topics because such topics produce more cohesive clusters. The impact on the hyper-parameters is to favor β being small and α being large. The approach is validated by showing that it improves performance on three software engineering tasks, traceability link recovery, feature location, and software artifact labeling. In light of the data presented in Section 3, it would be interesting to explore the interplay between the sampling of the posterior distribution, which goes un-described, and the genetic algorithm, which searches over multiple models.

6. FUTURE WORK

Several extensions to this work are possible. As described in Section 2 commonly used implementations of Gibbs sampling provide little support for extracting multiple interval-separated samples. To ensure independence, the data presented in this paper used n independent observations (n independent executions of R’s Gibbs sampling process).

The Gibbs’ implementation in Mallet [7] provides an option to output every si^{th} model. However, when configured to do so it is quite clear that the samples are correlated even with si set as high

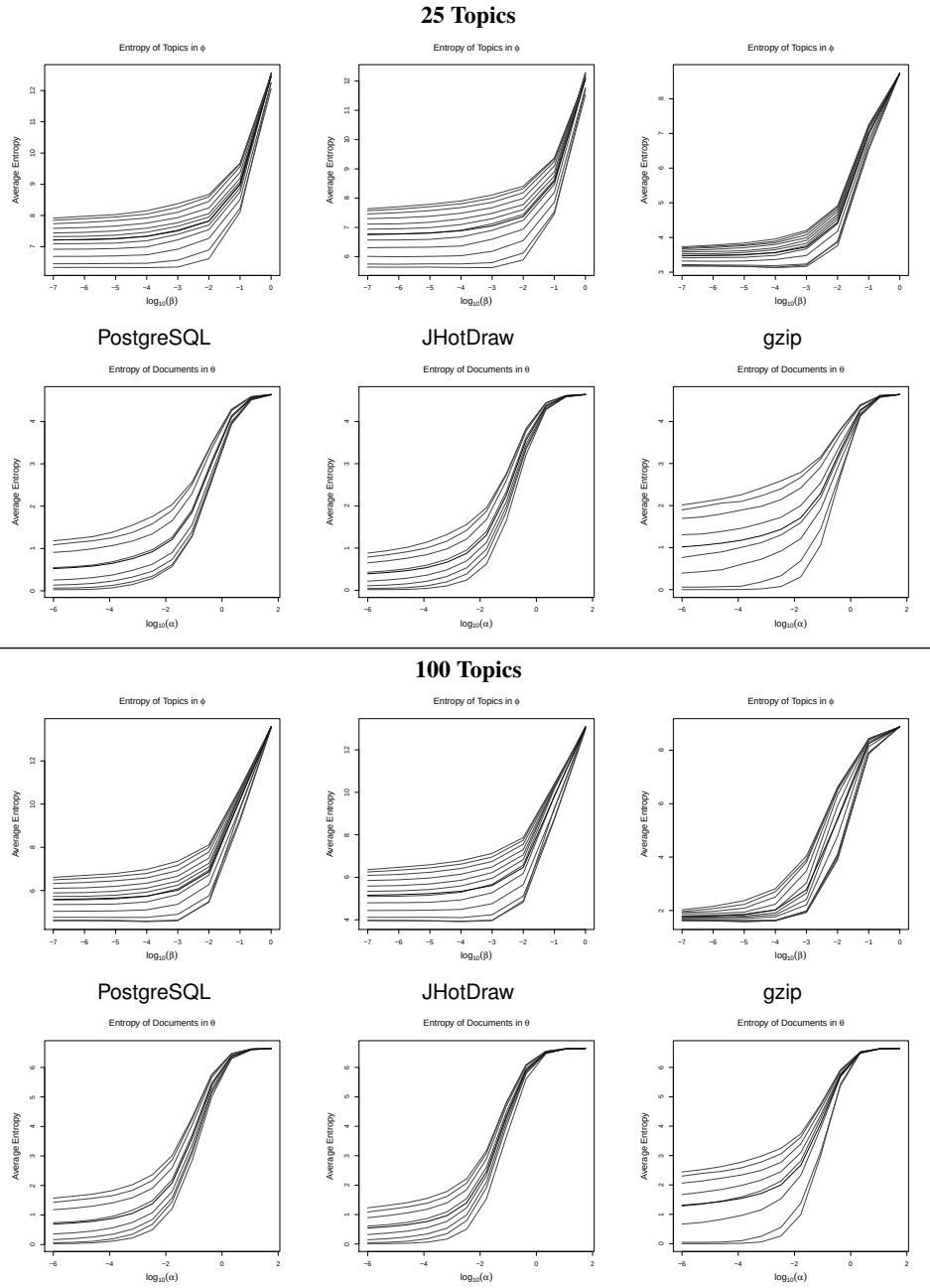


Figure 10: Entropy of ϕ and θ for the programs PostgreSQL, JHotDraw, and gzip.

as 997. The degree of auto-correlation is related to β . An empirical investigation indicates that Mallet gets “stuck” in different local maxima. Separate executions of the tool successfully sample the posterior distribution, but further investigation is warranted.

Finally, the last study considered four variations of the input corpus, and thus the future work concludes by considering the way in which the source code is mapped into an LDA input corpus. There are a range of other preprocessing options possible such as stemming and stopping.

7. SUMMARY

This paper aims to improve the understanding of the LDA models used by software engineers by raising their awareness of the effects of LDA tuning parameters. The empirical study visually il-

lustrates the impact of these parameters on the generated models. The better understanding of LDA made possible through visualization should help researchers consider how best to apply LDA in their work. In particular, sampling must play a bigger role in the discussion of tools that depend on LDA models. Such discussion should include how samples are aggregated since this is dependent on the task at hand.

8. ACKNOWLEDGEMENTS

Support for this work was provided by NSF grant CCF 0916081. Thanks to Malcom Gethers for his advice working with LDA and R, Scott Grant for his assistance in the replication of the topic count study, and Steve Thomas and Jeffrey Matthias for their assistance in the replication of the topic evolution work.

9. REFERENCES

- [1] H. Asuncion, A. Asuncion, and R. Taylor. Software traceability with topic modeling. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 95–104. ACM, 2010.
- [2] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.
- [3] S. Grant, J. Cordy, and D. Skillicorn. Using heuristics to estimate an appropriate number of latent topics in source code analysis. *Science of Computer Programming*, <http://dx.doi.org/10.1016/j.scico.2013.03.015>, 4 2013.
- [4] D. Lawrie and D. Binkley. Expanding identifiers to normalizing source code vocabulary. In *ICSM '11: Proceedings of the 27th IEEE International Conference on Software Maintenance*, 2011.
- [5] Erik Linstead, Paul Rigor, Sushil Bajracharya, Cristina Lopes, and Pierre Baldi. Mining concepts from code with probabilistic topic models. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 461–464. ACM, 2007.
- [6] Stacy K Lukins, Nicholas A Kraft, and Letha H Etzkorn. Source code retrieval for bug localization using latent dirichlet allocation. In *Reverse Engineering, 2008. WCRE'08. 15th Working Conference on*, pages 155–164. IEEE, 2008.
- [7] A. McCallum. Mallet: A machine learning for language toolkit, 2002.
- [8] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *Program Comprehension (ICPC), 2010 IEEE 18th International Conference on*, pages 68–71, 2010.
- [9] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshynanyk, and A. De Lucia. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *ACM/IEEE International Conference on Software Engineering (ICSE'13)*, May 2013.
- [10] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. *14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 3, 2008.
- [11] M. Steyvers and T. Griffiths. *Probabilistic Topic Models in Latent Semantic Analysis: A Road to Meaning*, Landauer, T. and Mc Namara, D. and Dennis, S. and Kintsch, W., eds. Lawrence Erlbaum, 2007.
- [12] Stephen W Thomas. Mining software repositories using topic models. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 1138–1139. IEEE, 2011.
- [13] Stephen W. Thomas, Bram Adams, Ahmed E. Hassan, and Dorothea Blostein. Modeling the evolution of topics in source code histories. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, MSR '11, 2011.
- [14] Kai Tian, Meghan Revelle, and Denys Poshyvanyk. Using latent dirichlet allocation for automatic categorization of software. In *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*, pages 163–166. IEEE, 2009.