RESEARCH

Speeding Up Calibration of Latent Dirichlet Allocation Model to Improve Topic Analysis in Software Engineering

Jorge López*, Denise Woit and Andriy Miranskyy

*Correspondence:
jlopez@ryerson.ca
Department of Computer Science, Ryerson University, Toronto, ON, CANADA
Full list of author information is available at the end of the article

Abstract
Extraction of topics from large text corpuses helps improve Software Engineering (SE) processes. Latent Dirichlet Allocation (LDA) represents one of the algorithmic tools to understand, search, exploit, and summarize a large corpus of data (documents), and it is often used to perform such analysis. However, calibration of the models is computationally expensive, especially if iterating over a large number of topics. Our goal is to create a simple formula allowing analysts to estimate the number of topics, so that the top X topics include the desired proportion of documents under study. We derived the formula from the empirical analysis of three SE-related text corpuses. We believe that practitioners can use our formula to expedite LDA analysis. The formula is also of interest to theoreticians, as it suggests that different SE text corpuses have similar underlying properties.

Keywords: Probabilistic Topic Models; Number of Topics; Information Retrieval; Bag-of-Words

1 Introduction

Latent Dirichlet Allocation (LDA) is a probabilistic model that can uncover common topics within a group of documents (such as support tickets[1] or defect descriptions), based on the analysis of words extracted from these documents. LDA is employed in a wide variety of areas, ranging from music [1] to robot learning [2]. LDA is gaining popularity in the SE community, where it is used in a number of areas including software maintenance [3], traceability [4, 5], a posteriori requirements identification [1], and program comprehension [1]. LDA is attractive because it is unsupervised, requiring no a priori annotations.

Analysis of various Software Engineering text corpora aids in software development. For example, analysis of artifacts created during development can improve software traceability [6], while mining text of defects and support tickets can help to understand which particular features are causing users the most grief [3]. The former improves code comprehension; the latter enhances business risk analysis, and prioritization of maintenance tasks. My work is motivated by my industrial experience in software maintenance, resolving product issues. A representative example is as follows. The software product was returning a cryptic error message if it

[1]Reports on specific problems in an issue tracking system, including their statuses and additional significant data that are logged in a call center or help desk.

could not connect to the central data repository (which happened frequently). This error message triggered more than a 1,000 calls from confused clients, overloading support personnel. This problem came to attention a month later, when an analyst read (mined) thousands of support tickets opened within the last month and identified a cluster of tickets associated with the error message. Fixing the problem (thereby satisfying customers and relieving support personnel) was simple: the message text was improved, and documentation associated with the message was enhanced. However, identifying the problem was laborious and time consuming. Other practitioners report similar challenges [3]. To diagnose such problems, the analyst often needs to identify a small set of topics containing a large portion of the documents, so that s/he can recognize "key" topics: e.g., the analyst from our example sifted through thousands of support tickets to find priority problem areas. Fortunately, automatic topic analysis techniques, such as LDA [7] can expedite the process.

Selecting the optimal number of topics to be identified in the LDA model determines greatly its performance and outcome expected [8]. Thus, the problem can be described as follows: an analyst needs to pick K, such that top X topics would contain a certain fraction F of the documents. For example, pick the value of K, such that top 5 topics would contain 80% of all the support tickets under study. This outcome can be achieved by iterating over the values of K until reaching the desired outcome. This process, however, is resource-intensive.

Probabilistic topic models, such as LDA, require to be provided with the value of number of topics K. Internally, the model also needs two additional parameters: α and β, that are calculated automatically for each parameter K [4]. It is known that the processing time, necessary to find the optimal values of K by iterating, grows exponentially [9]. Therefore, this thesis addresses the following **research question**: How can we quickly select the number of topics K so that the top X topics include a certain fraction F of the N documents under study?

We are proposing a formula to quickly estimate K in LDA by means of X, F, and N, which would be invaluable in the realm of software system development and maintenance. For example, practitioners can use the formula to determine a short-list of the most frequently reported product field failures, which, in turn, can inform their decisions regarding corrective product maintenance; businesses can identify the most frequently reported user issues in order to inform their decisions regarding product evolution, and risk analysis; management can obtain a more definitive answer to the question "how many problematic areas does the product contain?", rather than an answer such as "somewhere between 5 and 100 depending on the granularity".

TODO: add description of the following sections, once the paper is finalized

2 Dataset

2.1 Description

We will perform analysis of Questions and Answers (Q&A) forums hosted on stackoverflow.com. StackOverflow is a popular place, frequented by professional and enthusiast programmers, where they can post Q&A related to a particular product or group of products. We will focus on three particular forums, namely:

1 android (android.stackoverflow.com)[10]
2 database administration(dba.stackoverflow.com)[11]
3 salesforce (salesforce.stackoverflow.com)[12]

StackOverflow archives the Q&A and makes it publicly available for download at `https://archive.org/download/stackexchange` [13]. In Table 1 we show a summary statistics of these datasets.

Table 1 Datasets considered

Dataset	Selected data from-to (period)	Number of questions
android	2014-2015 (monthly)	15765
dba	2012-2014 (monthly and quarterly)	39174
salesforce	2014-2015 (monthly)	25965

As we plan to process several periods for these datasets, the expanded table of processed datasets is as Table 2 illustrates.

Table 2 Expanded list of number of subsets processed by dataset. Year refers to the yearly data that is available to be extracted from the corpus.

Dataset	Periodicity	Year	Num. of files	Num. of subsets
android	monthly	2014	12	588
android	monthly	2015	12	588
dba	monthly	2012	12	588
dba	monthly	2013	12	588
dba	monthly	2014	12	588
dba	quarterly	2012	4	196
dba	quarterly	2013	4	196
dba	quarterly	2014	4	196
salesforce	monthly	2014	12	588
salesforce	monthly	2015	12	588
		Totals	**96**	**4704**

2.2 Processing

All our datasets undergo a common processing comprised by the following steps (see Figure 1) which include:

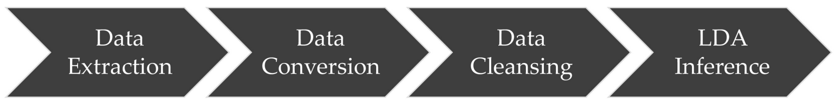

Figure 1 Process of data for performing a LDA inference

1 Data Extraction. The text corpus used was the dba forum of StackExchange (`https://dba.stackexchange.com`). The dataset was extracted from `https://archive.org/download/stackexchange` [13].
2 Data Conversion. The file from the repository was in XML format and for performing the LDA inference we needed the data in CSV format; therefore, a Perl script for doing the conversion was created. This script keeps the text for all the questions but eliminates those answers that have less than three votes and were not officially accepted as an answer (by the person who asked the question).

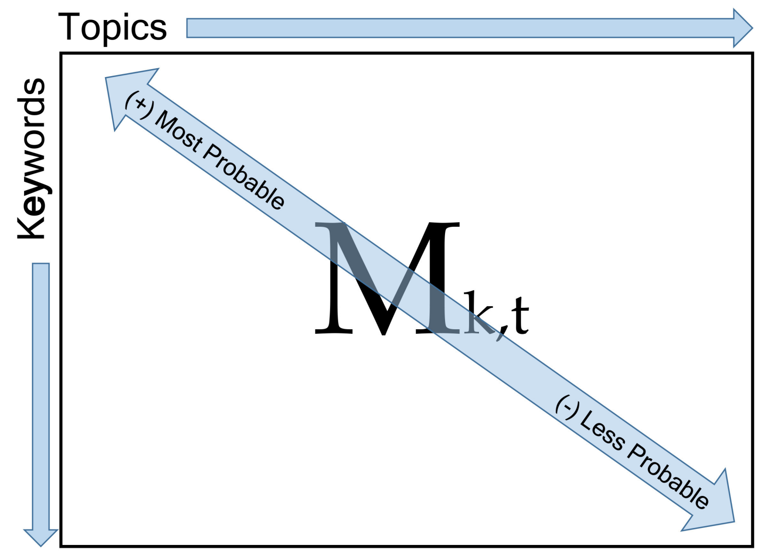

Figure 2 Output of an LDA inference

3 Data Cleansing. This step involved the following actions:
  (a) Converting to lowercase all words.
  (b) Removal of punctuation signs.
  (c) Removing stopwords (i.e., most common words in English, such as "and", "the", or "a").
  (d) Stemming applicable words (i.e., reducing words to their word stem, such as "computers" and "computing" to "comput").
  (e) White-space char elimination.
  (f) Removing terms which have tf-idf smaller than the median of all the tf-idf values. I.e., we eliminate the terms with low "importance".

4 LDA inference. Here is where the LDA is applied to the data. This processing is particular to what our objective is. In our case, we have two types of processing, the first one is to illustrate the usage of LDA by generating a matrix of topic-keywords (similar to Figure 2), and the second one is to generate term frequencies to perform analysis and deduce the formula for calibrating the model.

2.3 Example of topic extraction

The volume of QA in stackexchange.com forums varies from forum to forum. In the case of database-administrators-site (extracted from `https://dba.stackexchange.com`) [11], there were approximately 50,000 entries at the time we downloaded the data. The file that we extracted included several periods by year and month. This repository contains unstructured data for both questions and answers. One of the managers in charge of the product might be wondering what the users are talking about it. Then, LDA can be applied and its output provide the necessary analytics to help responding this question.

All LDA-related computations are performed using R-package 'topicmodels' [14] (which in turn uses the C code implementation of LDA by Blei et al. [7]).

For performing the analysis to illustrate the usage of LDA, we follow the processing of the dataset as depicted in Figure 1.

The output of this LDA inference for each of the five topics would produce a list of keywords, their probability of belonging to a given topic, number of documents, and idf. Example of an output for topic 1 is given in Table 3.

Table 3 Topic 1 of the LDA inference of 5 topics and 20 keywords

KEYWORD	PROBABILITY	#doc	idf
server	0.022	1582	0.214
databas	0.021	1497	0.238
mysql	0.018	710	0.562
user	0.013	948	0.436
use	0.013	1631	0.200
sql	0.012	1151	0.352
connect	0.012	874	0.471
log	0.011	714	0.559
error	0.011	959	0.431
can	0.010	1582	0.214
file	0.009	829	0.494
master	0.009	407	0.803
replic	0.008	481	0.731
set	0.008	921	0.449
run	0.007	1036	0.397
creat	0.007	902	0.458
oracl	0.007	393	0.818
slave	0.006	288	0.953
tri	0.006	1018	0.405
data	0.006	847	0.485

For better appreciation of the data, let us render the data in this table in the graphical format. Figure 3 shows the probability of occurrence in descending order for each of the keywords in topic 1 (first of the K number of topics selected). Figure 4 shows the number of documents[2], where each of the keywords appears. The last plot in this sequence is Figure 5; it shows that the closer the value of idf to 0 is for a term, the more widespread the term is, and vice versa.

Now, let us plot the keywords of topic 1 to show its idf and probability as Figure 6 indicates. The figure shows that the most important and widespread words are 'server' and 'databas' and the the least widespread ones are 'master', 'oracl', 'slave'. Eyeballing the overall set of keywords, we may conjecture that this topic aggregates Q&A related to issues associated with setting up a replicated database engine and establishing connectivity between the replicas, as these the are terms commonly used in this field.

We would like to mention that a manual validation of the subsets of documents has taken place.

3 Calibration

In this section we will develop our proposed model for calculating in a straightforward way the number of topics for implementing LDA. Calibration of LDA can consume a large amount of computing resources. Steyvers and Griffits [15], agree on the idea that selecting the number of topics influences greatly the understandability of the results produced by LDA. A K which is too small can produce very broad topics, and one which is too large may affect the interpretability of the topics.

[2]The overall number of documents for topic 1 is 2587 (N).

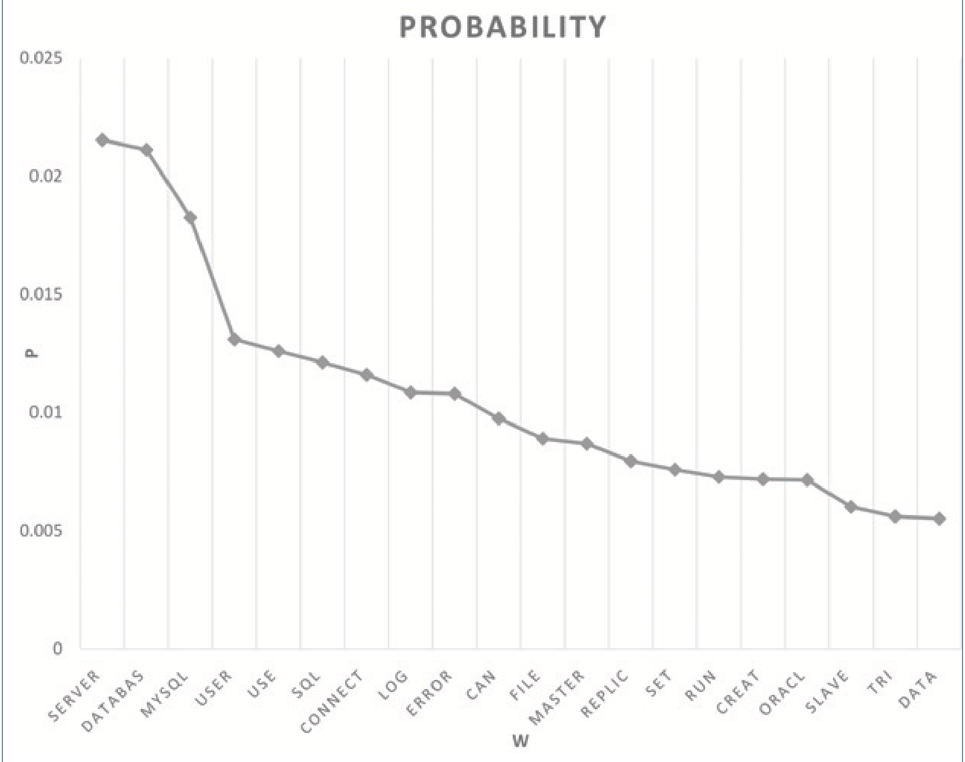

Figure 3 Probability per term for Topic 1

Our aim is to create a simple formula for readily calculating the number of topics, therefore addressing our research question:

How can we quickly select the number of topics K so that the top X topics include a certain fraction F of the N documents under study?

Note that we do not speed up computation of a particular LDA model. Rather, we drastically reduce the number of LDA models that have to be computed and calibrated. In the worst-case scenario, one will need to compute up to N models, while usage of our formula reduces the number of models to, at best, one. In practice, given that our approximation is not perfect, one may have to compute a couple of LDA models to obtain the desired value of F.

3.1 Datasets

3.2 Processing of datasets for model calibration

In order to generate the information to perform our analysis for deducing the formula to calibrate the model, we will execute the processing of our datasets. The specific step corresponds to the LDA inference consists in processing each considered dataset (android, dba, and salesforce) for several periods: years-months/quarters, comprising 96 data files. Moreover we consider each of these yearly files to be divided into temporal subsets (i.e., 12 for monthly and 4 for quarterly subsets) times the number of top X we computed (49, i.e. $X = 2, 3, \ldots, 50$). We will produce for each dataset/period the topic frequency for the words contained in them. For each dataset, we will calculate the distribution of the number of posts per topic,

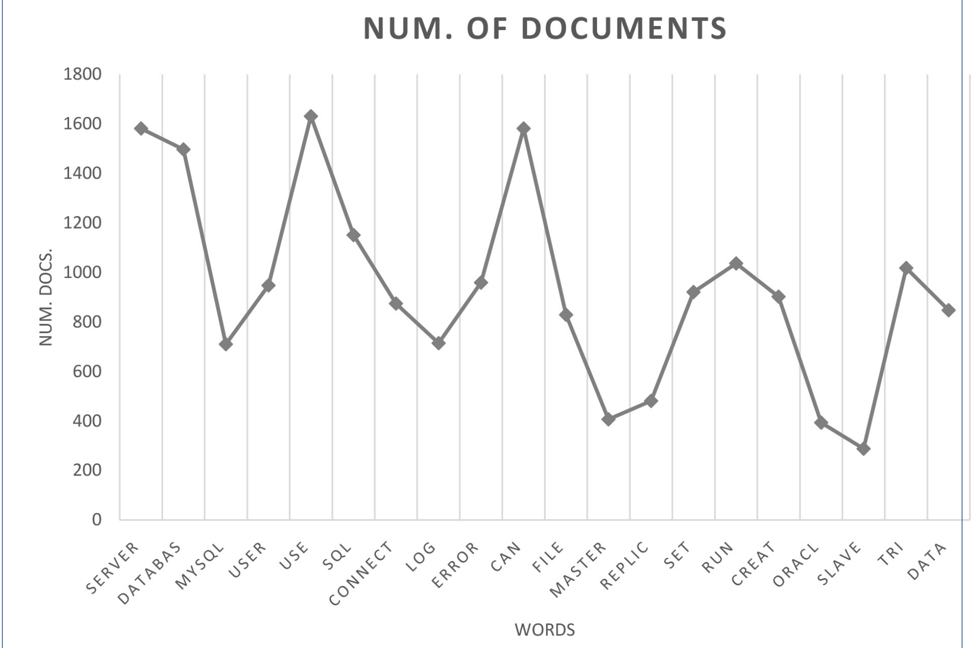

Figure 4 Document frequency per term for topic 1

with $K = 2, 3, \dots, N$. For each K and for each top topics $X = 2, 3, \dots, 50$, we will calculate the empirical value of F as follows:

$$F = P/N, \tag{1}$$

where P is the overall number of documents in top X topics.

Once we have generated the frequency files for each dataset-year-period (monthly/quarterly), we proceed to bind all of these files in one file.

After doing this we commence our analysis for inferring our formula. The developed scripts for supporting this analysis are: "verifyFit_k200.R" (See Appendix **??**) and – "verifyFit_k200.R_analysis.R" (See Appendix **??**) for generating the performance plots.

3.3 Time consumption to iterate over K

Figure 7 illustrates the processing time to iterate over K in order to determine the number of topics for the model. Here, we have plotted the times needed to compute LDA model for a given K, while $K = 2, 3, \dots, N$. We arbitrarily chose the Android dataset, for January of 2014. This data subset has $N = 614$. Notice that time to compute the LDA model grows (non-monotonically) with the increase of K. The maximum time for computing the model (when $K = 602$) is ≈ 6690 seconds; the overall time to compute all 613 models for this subset is ≈ 1276576 seconds (or ≈ 15 days). The times shown here are for one subset; however, the behaviour is similar for all the subsets under study. Thus, it is beneficial to reduce the number of calibrations of LDA models.

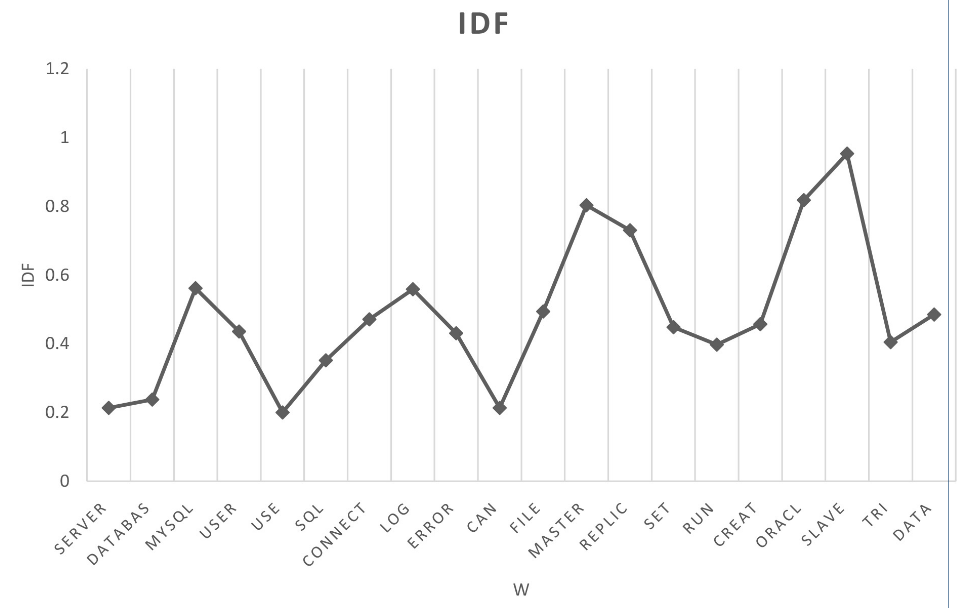

Figure 5 Keywords vs idf for topic 1.

3.4 Creating fitted values

We notice that the graph of $\log(F)$ against $\log(K)$ for a given X forms a straight line (e.g., see Figure 8), suggesting Power Law relationship between F and K [16], [17]. The law takes form

$$F = aK^b, \tag{2}$$

where a and b are some constants. Log transformation of Equation 2 yields:

$$\log(F) = \log(a) + b\log(K). \tag{3}$$

Eyeballing Figure 8 suggests that the values of $\log(a)$ and b vary (at least with the change in X). The question then becomes: can we estimate the values of a and b for a given dataset based on some attributes of the dataset?

Given that N and X are (almost) readily available to us, let us assume that a and b are governed by N and X. In order to find $a(X, N)$ and $b(X, N)$, we will compute empirical values of a and b, denoted by $\hat{a}$ and $\hat{b}$, respectively, by fitting linear regression model (Equation 3) to the data points for each dataset and for each value of X (we set $X = 2, 3, \ldots, 50$).

We will consider two models: 1) a more flexible one, where a and b are independent of each other and 2) a more constrained one, where $a = X^{-b}$. The second model is obtained via ansatz (an educated guess).

Further exploratory analysis shows that if $K > 0.75N$ or if $K > 200$, then the linear relation between $\log(F)$ against $\log(K)$ breaks (example is shown in Figure 9). From practical perspective, an analyst would rarely be interested in a large number of topics K. Thus, if we filter out data for large values of K, then we may improve

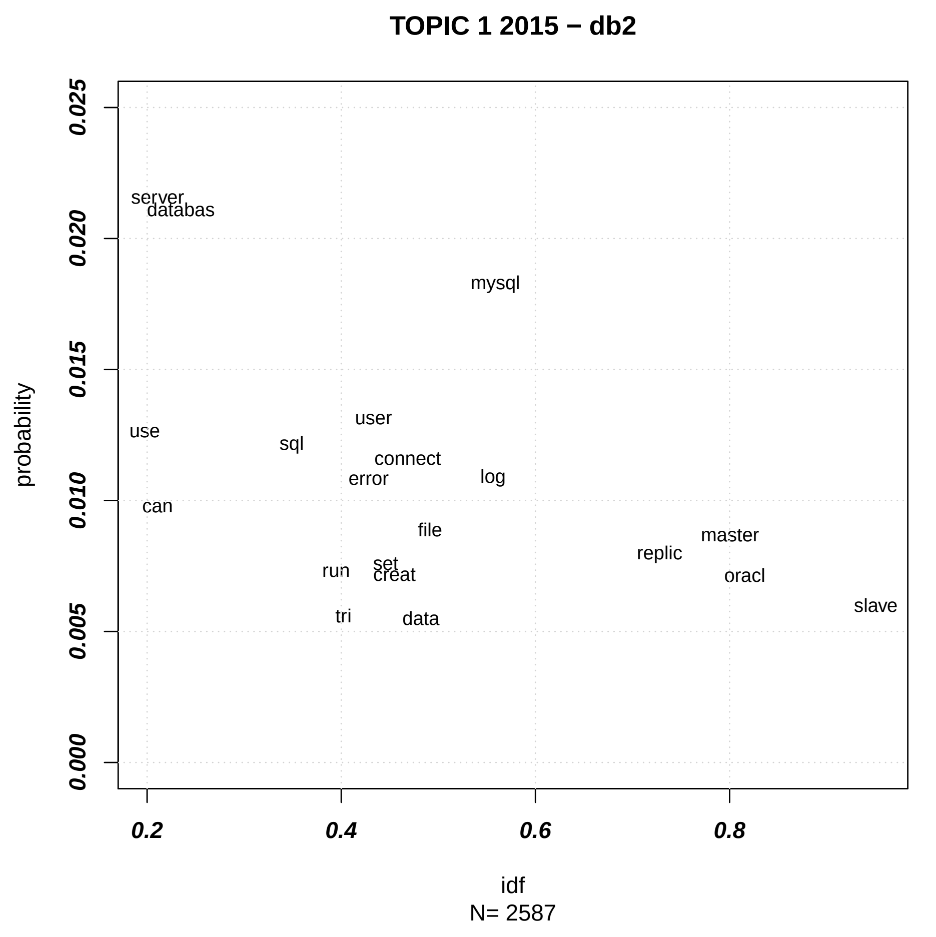

Figure 6 LDA inference for topic 1 showing terms plotted against its idf and probability.

the fit of linear models while preserving the models' practicality. We will try two different filters, hence the two datasets:

1 Dataset 1: retain the data if $K \leq 0.75N$;

2 Dataset 2: retain the data if $K \leq 0.75N$ and $K \leq 200$.

We discuss in the following sections the fit of the flexible and the constraint models. Comparison of the models is given in Section 4.

3.4.1 Flexible Model

In the flexible case, Equation 2 becomes

$$F = a(X,N)K^{b(X,N)}. \tag{4}$$

Solving it for K yields:

$$K = \left[\frac{F}{a(X,N)}\right]^{1/b(X,N)}. \tag{5}$$

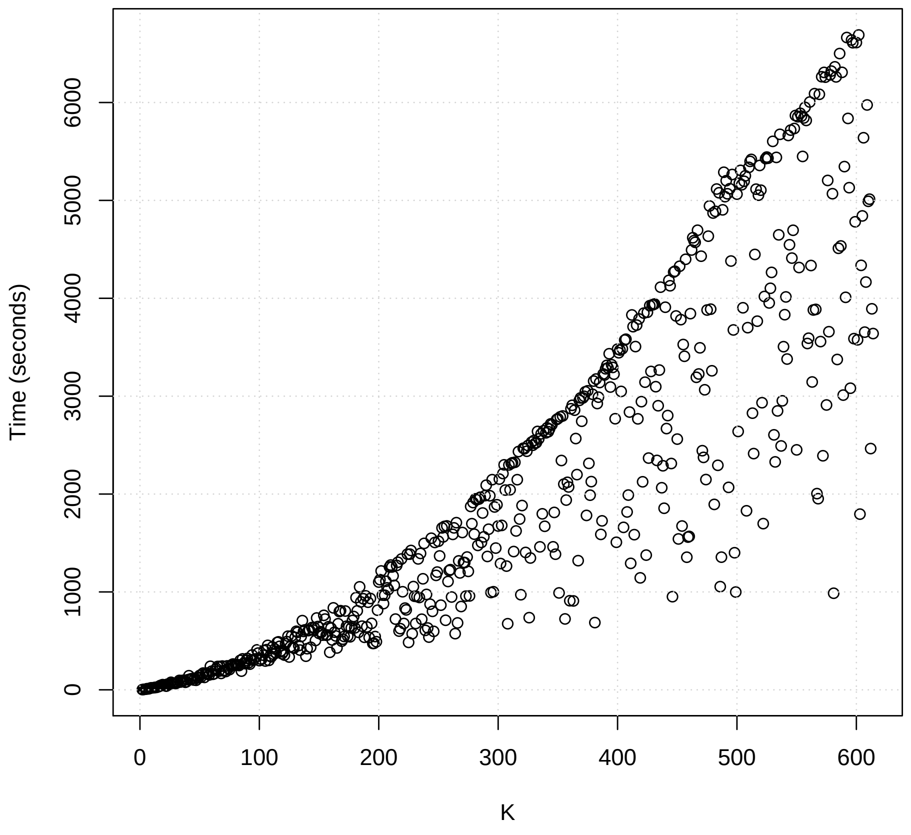

Figure 7 Time needed to compute (calibrate) LDA model for a given number of topics K (the input data are Android subset, January 2014).

The value of N can be easily extracted from the dataset. The values of F and X are provided by an analyst. However, we still need to define the functional form of $a(X, N)$ and $b(X, N)$ to compute Equation 5.

To compute $a(X, N)$ we examined a number of relations between $\hat{a}$ and X, N. A regression of the form

$$\hat{a} = \alpha_1 + \alpha_2 \ln(X) + \alpha_3 \ln(N) \tag{6}$$

yields good results. For both datasets, the regression model is valid and statistically significant. In the case of Datasets 1 and 2 it explains most of the variability ($R^2 \approx 0.87$ and $R^2 \approx 0.96$, respectively). As expected, the fit to Dataset 2 is better than to Dataset 1, as we filter larger number of "anomalous" data points. Details of validity and the values of coefficients are given in Table 4.

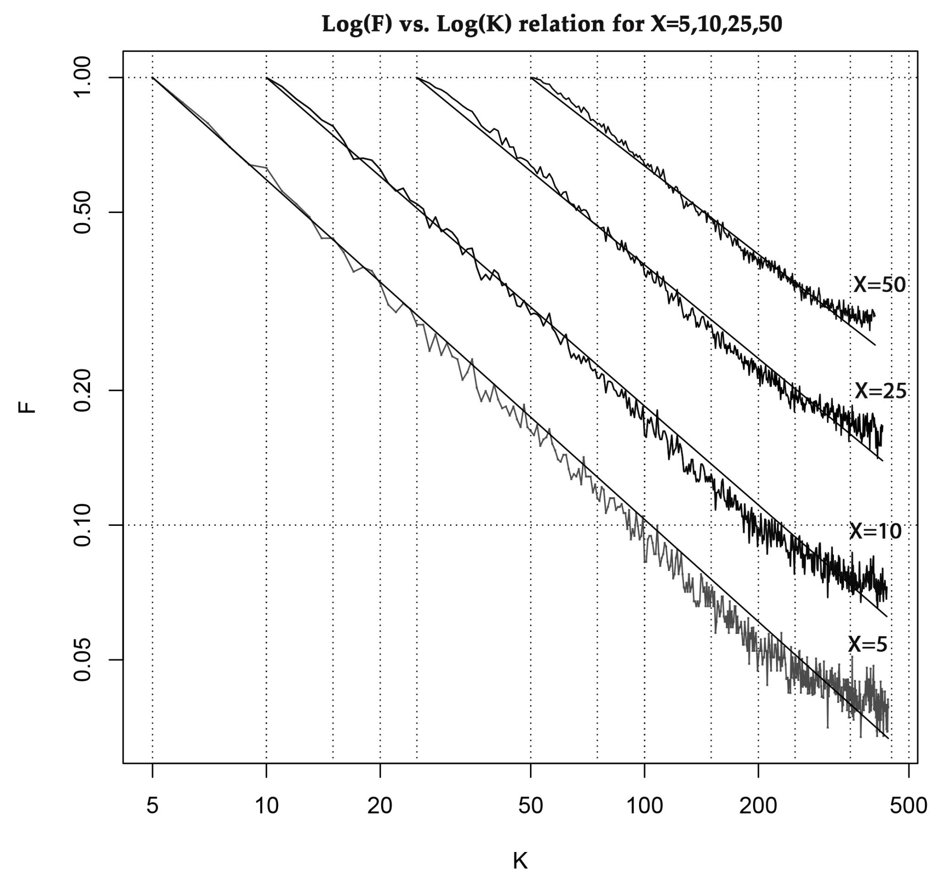

Figure 8 Fitted empirical data for $X = 5, 10, 25, 50$

We also defined a more complex (yet statistically significant relation between $\hat{a}$ and X, N, namely

$$\hat{a} = \alpha_1 + \alpha_2 \ln(X) + \alpha_3 \ln(N) + \alpha_4 \ln(X)^2 + \alpha_5 \ln(N)^2. \tag{7}$$

The model is valid and statistically significant as shown in Table 4. In the case of Datasets 1 and 2 the model explains more variability than Equation 6 ($R^2 \approx 0.91$ and $R^2 \approx 0.98$, respectively). However, the improvement is not dramatic.

Similarly, we explored relations between $\hat{b}$ and X, N to compute $b(X, N)$. A regression of the form

$$\hat{b} = \alpha_1 + \alpha_2 X + \alpha_3 N \tag{8}$$

yields adequate results. As in the above case (Eq. 6), the model is valid and statistically significant, but it cannot explain as much variability: $R^2 \approx 0.54$ for Datasets 1 and $R^2 \approx 0.78$ Dataset 2. Details of validity and the values of coefficients are given in Table 5.

A more complex relation between $\hat{b}$ and X, N is given by

$$\hat{b} = \alpha_1 + \alpha_2 X + \alpha_3 N + \alpha_4 N^2 + \alpha_5 \ln(N) + \alpha_6 \ln(X). \tag{9}$$

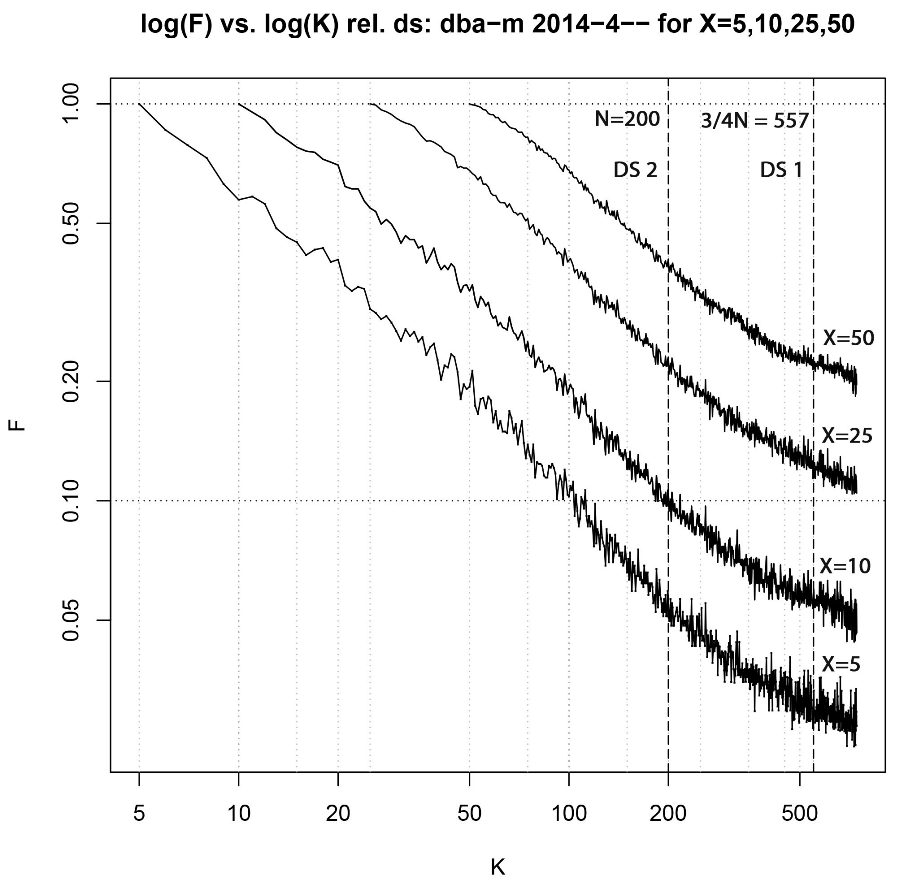

Figure 9 Linear relation between $log(F)$ and $log(K)$ breaks approximately at $K > 200$, example for X=5,10,25, and 50

Table 5 shows that the model is valid and statistically significant[3]. In the case of the Datasets 1 and 2, the model explains more variability than Equation 8 ($R^2 \approx 0.60$ and $R^2 \approx 0.83$, respectively). However, it still cannot explain as much variability as the models for the intercept term (defined in Equations 6 and 7).

3.4.2 Constrained Model

In the constrained case, Equation 2 becomes

$$F = a(X, N)K^{b(X,N)} = X^{-b(X,N)}K^{b(X,N)}. \tag{10}$$

[3]In the case of Dataset 1, p-value of the N^2 term is > 0.05. We keep it for consistency – addition of the term does not degrade the R^2 value. To make sure that inclusion of the N^2 term does not bias the final result, we also executed the final validation (discussed in Section 4) while using the formula with and without the N^2 term. For Dataset 1, we obtained the same result for both formulas; for Dataset 2, the result was marginally better for the formula without the N^2 term. However, the models calibrated on Dataset 1 prevailed over the models calibrated on Dataset 2. Thus, from practical perspective, it is acceptable to keep the N^2 term (assuming that one will favor better performing models).

Table 4 Linear regression models for intercept term $\hat{a}$ (flexible model: Equations 6 and 7). The values in brackets represent 90% confidence interval of the coefficients.

	Dataset 1		Dataset 2	
Coefficient at	Eq. 6	Eq. 7	Eq. 6	Eq. 7
$\ln(X)$	0.613***	1.057***	0.649***	1.081***
	(0.607, 0.618)	(1.029, 1.085)	(0.646, 0.652)	(1.067, 1.095)
$\ln(N)$	0.053***	−4.386***	−0.116***	2.694***
	(0.043, 0.063)	(−4.628, −4.143)	(−0.122, −0.110)	(2.569, 2.818)
$\ln(X)^2$		−0.086***		−0.083***
		(−0.091, −0.080)		(−0.086, −0.081)
$\ln(N)^2$		0.319***		−0.202***
		(0.302, 0.337)		(−0.211, −0.193)
Constant	−0.144***	14.716***	1.171***	−9.047***
	(−0.215, −0.073)	(13.875, 15.557)	(1.128, 1.214)	(−9.480, −8.615)
Observations	4,704	4,704	4,704	4,704
R^2	0.874	0.906	0.956	0.976
Adjusted R^2	0.874	0.906	0.956	0.976
Residual Std. Error	0.182 (df = 4701)	0.157 (df = 4699)	0.109 (df = 4701)	0.081 (df = 4699)
F Statistic	16,277.930*** (df = 2; 4701)	11,341.730*** (df = 4; 4699)	51,006.570*** (df = 2; 4701)	47,647.760*** (df = 4; 4699)
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 5 Linear regression models for slope term $\hat{b}$ (flexible model:Equations 8 and 9). The values in brackets represent 90% confidence interval of the coefficients.

	Dataset 1		Dataset 2	
Coefficient at	Eq. 8	Eq. 9	Eq. 8	Eq. 9
X	0.002***	0.002***	0.002***	0.002***
	(0.002, 0.002)	(0.002, 0.002)	(0.002, 0.002)	(0.002, 0.002)
N	−0.00001***	−0.0001***	0.00003***	0.0003***
	(−0.00002, −0.00001)	(−0.0002, −0.0001)	(0.00003, 0.00003)	(0.0003, 0.0004)
N^2		5.21e−09		−4.83e−08***
		(−2.44e−09, 1.29e−08)		(−5.25e−08, −4.41e−08)
$\ln(X)$		0.014***		0.007***
		(0.011, 0.016)		(0.006, 0.009)
$\ln(N)$		0.116***		−0.202***
		(0.095, 0.137)		(−0.213, −0.190)
Constant	−0.744***	−1.450***	−0.841***	0.276***
	(−0.746, −0.741)	(−1.567, −1.334)	(−0.842, −0.840)	(0.212, 0.340)
Observations	4,704	4,704	4,704	4,704
R^2	0.538	0.599	0.782	0.825
Adjusted R^2	0.537	0.598	0.782	0.824
Residual Std. Error	0.032 (df = 4701)	0.030 (df = 4698)	0.018 (df = 4701)	0.016 (df = 4698)
F Statistic	2,732.760*** (df = 2; 4701)	1,401.956*** (df = 5; 4698)	8,443.119*** (df = 2; 4701)	4,419.474*** (df = 5; 4698)
Note:				*p<0.1; **p<0.05; ***p<0.01

Solving it for K yields:

$$K = \left[\frac{F}{X^{b(X,N)}} \right]^{1/b(X,N)}. \tag{11}$$

To find a relation between $\hat{b}$ and X, N we explore two regression models. A simple regression model is given by:

$$b(X,N) = \alpha_1 + \alpha_2 X + \alpha_3 N, \tag{12}$$

and a more complex one by:

$$b(X,N) = \alpha_1 + \alpha_2 X + \alpha_3 N + \alpha_4 \ln(X). \tag{13}$$

Both models are valid and statistically significant. Model's details are given in Table 6. Simple model (Eq. 12) explains 79% of variability ($R^2 \approx 0.79$), while a more complex model, given by Eq. 13, explains 83% of variability ($R^2 \approx 0.83$).

Table 6 Linear regression models for $\hat{b}$ (constrained model: Equations 12 and 13). The values in brackets represent 90% confidence interval of the coefficients.

	Dataset 1		Dataset 2	
Coefficient at	Eq. 12	Eq. 13	Eq. 12	Eq. 13
X	0.002*** (0.002, 0.002)	0.001*** (0.001, 0.001)	0.003*** (0.003, 0.003)	0.001*** (0.001, 0.001)
N	0.00002*** (0.00001, 0.00002)	0.00002*** (0.00001, 0.00002)	0.00003*** (0.00003, 0.00003)	0.00003*** (0.00003, 0.00003)
$\ln(X)$		0.031*** (0.030, 0.032)		0.035*** (0.033, 0.037)
Constant	−0.767*** (−0.768, −0.766)	−0.819*** (−0.821, −0.817)	−0.777*** (−0.778, −0.775)	−0.836*** (−0.839, −0.833)
Observations	4,704	4,704	4,704	4,704
R^2	0.817	0.873	0.787	0.834
Adjusted R^2	0.817	0.873	0.787	0.834
Residual Std. Error	0.016 (df = 4701)	0.013 (df = 4700)	0.021 (df = 4701)	0.019 (df = 4700)
F Statistic	10,510.100*** (df = 2; 4701)	10,802.240*** (df = 3; 4700)	8,695.582*** (df = 2; 4701)	7,864.944*** (df = 3; 4700)
Note:				*p<0.1; **p<0.05; ***p<0.01

4 Analysis of results

In order to estimate goodness of fit of our approximation of a and b, we compute the RMSE (defined by Equation 14) between the actual value of F and the ones given to us by Equation 4 for each dataset and for each value of X (where $X = 2, 3, \ldots, 50$). Summary statistics for various models is given in Tables 7 and 8 and Figures 10 and 11.

$$RMSE = \sqrt{\frac{\sum (y - \hat{y})^2}{n}}, \tag{14}$$

where y is an actual value of F, $\hat{y}$ is the predicted value of F, and n is the number of values to predict.

By looking at the Tables 7 and 8 we can appreciate that the performance of the model varies based on various factors, discussed in the following paragraphs.

Table 7 Summary statistics for the RMSE: Dataset 1

Statistic	n	Mean	St. Dev.	Min	Max
Flex-simple	4,704	0.028	0.013	0.006	0.072
Flex-complex	4,704	0.019	0.005	0.006	0.042
Constrained-simple	4,704	0.018	0.006	0.006	0.040
Constrained-complex	4,704	0.018	0.006	0.004	0.039
Individual fit-flex	4,704	0.018	0.005	0.005	0.032
Individual fit-constrained	4,704	0.017	0.006	0.004	0.038

Table 8 Summary statistics for the RMSE: Dataset 2

Statistic	n	Mean	St. Dev.	Min	Max
Flex-simple	4,704	0.034	0.020	0.010	0.226
Flex-complex	4,704	0.022	0.007	0.009	0.071
Constrained-simple	4,704	0.023	0.006	0.009	0.046
Constrained-complex	4,704	0.023	0.006	0.007	0.044
Individual fit-flex	4,704	0.018	0.005	0.006	0.038
Individual fit-constrained	4,704	0.021	0.006	0.005	0.041

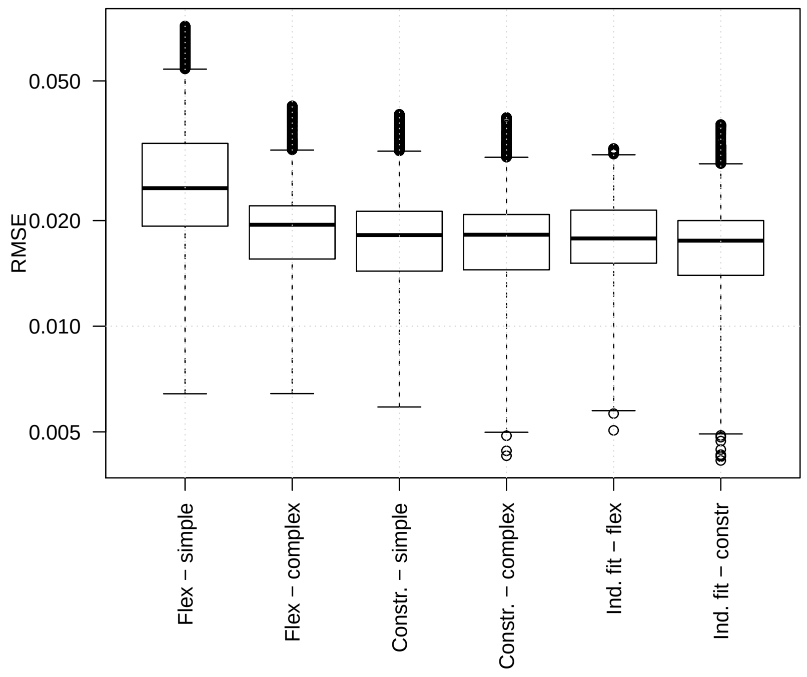

Figure 10 Summary statistics for the RMSE: Dataset 1. 'Flex - simple' uses Equations 6 and 8; 'Flex - complex' uses Equations 7 and 9. 'Constr - simple' uses Equation 12; 'Constr - complex' uses Equation 13. 'Ind. fit - flex' shows RMSE for flexible linear model (Eq. 4 fitted individually to every data subset), while 'Ind. fit - constr' depicts RMSE for constraint linear model (Eq. 10 fitted individually to every data subset).

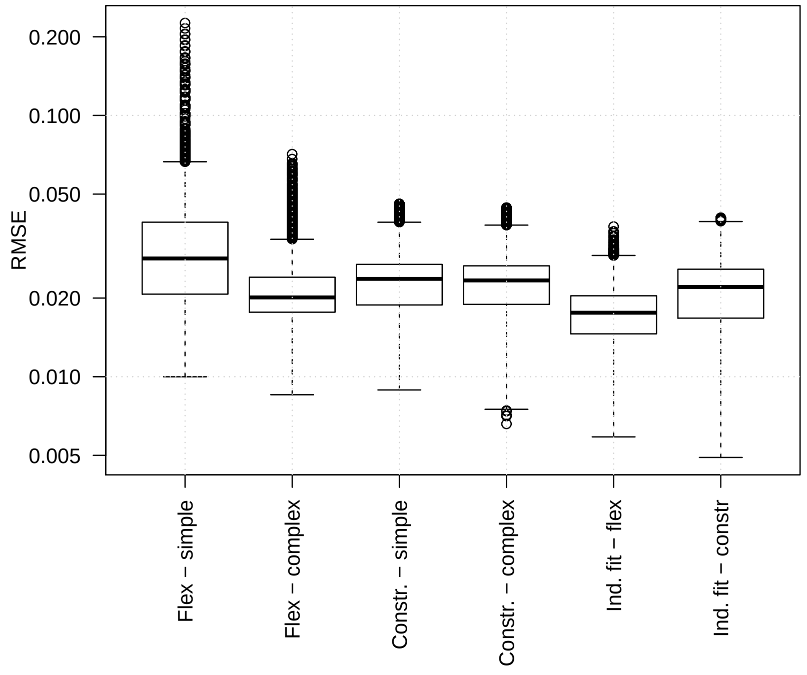

Figure 11 Summary statistics for the RMSE: Dataset 2. Flex - simple' uses Equations 6 and 8; 'Flex - complex' uses Equations 7 and 9. 'Constr - simple' uses Equation 12; 'Constr - complex' uses Equation 13. 'Ind. fit - flex' shows RMSE for flexible linear model (Eq. 4 fitted individually to every data subset), while 'Ind. fit - constr' depicts RMSE for constraint linear model (Eq. 10 fitted individually to every data subset).

4.1 Selection of the best performing model

In order to select the best performance model, we focus on the RMSE stats (provided in Tables 7 and 8) and select the model that minimizes the mean RMSE.

For Dataset 1 (DS1), among four models, constrained-complex model yields the lowest mean, min, and max RMSE values, comparable with the performance of individually-fitted lines. Thus, constrained-complex model is the winner for DS1.

In the case of Dataset 2 (DS2), RMSE of the flex-complex model is slightly lower than that of the constrained-complex model (0.022 vs. 0.032). However, the min and max values of the flex-complex are worse than that of the constrained-complex model (0.009 vs. 0.007 and 0.071 vs. 0.044, respectively). Thus, the constrained-complex can be considered a winner for DS2 too.

4.2 Analysis of the best performing model

To verify robustness of the results of the the best performing (constrained-complex) model and make sure that we are not overfitting, we performed 10-fold cross validation[4]. We split 96 data subsets into ten partitions. We then fit Equation 13 to nine out of ten partitions. We then fit the constrained model (Equation 10) to the raw data of the tenth partition and compute the RMSE value. The process is repeated nine more times, alternating partitions of the test and train sets. The resulting ten values of the RMSE are shown in Figure 12. As we can see, for both Dataset 1 (mean ≈ 0.018) and Dataset 2 (mean ≈ 0.024) the RMSE values are comparable to those reported in Tables 7 and 8. Thus, we are not overfitting.

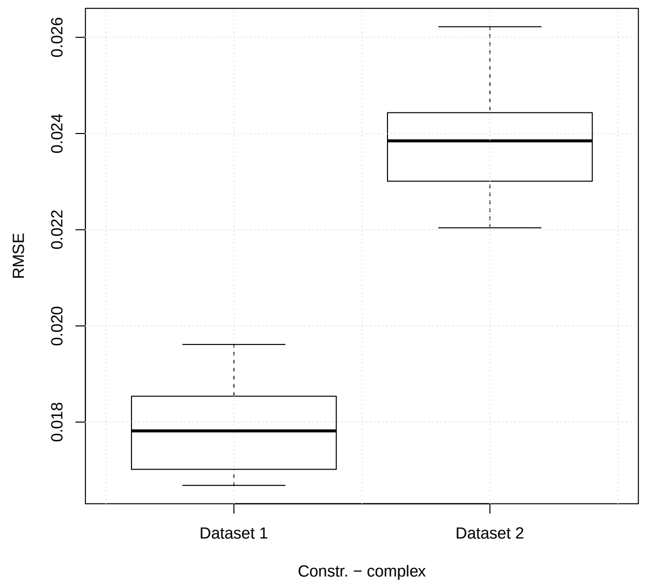

Figure 12 10-fold cross validation of the fit of the 'Constrained - complex' model (Equation 13).

In Figures 13 and 14 we can observe that for small values of X the RMSE is smaller and it increases as X gets larger. The X grows fast until approximately $X = 20$ and then it starts to slow down in growth.

In the other hand, in Figures 15 and 16 we can see that the relation between N and the RMSE varies in a similar manner to X for small values of N and large values of N. Therefore, we can say that the quality of RMSE is not influenced significantly by N, but it is influenced by X in the sense that the smaller the value of X the better the predictions are.

In the case of the datasets analyzed, for the Dataset 1 the fitting from higher to lower is android, salesforce, dba monthly, and dba quarterly (see Figure 17) and for the Dataset 2 is android, dba quarterly, salesforce monthly and dba monthly (see Figure 18).

Finally, Figure 19 shows one plot that demonstrates the fitting achieved with the constrained-complex model for Dataset 1. By eyeballing this plot we can appreciate that starting at the left corner the fit is very good, then it goes above the data and

[4]The listing for performing the validation is given in Appendix ??

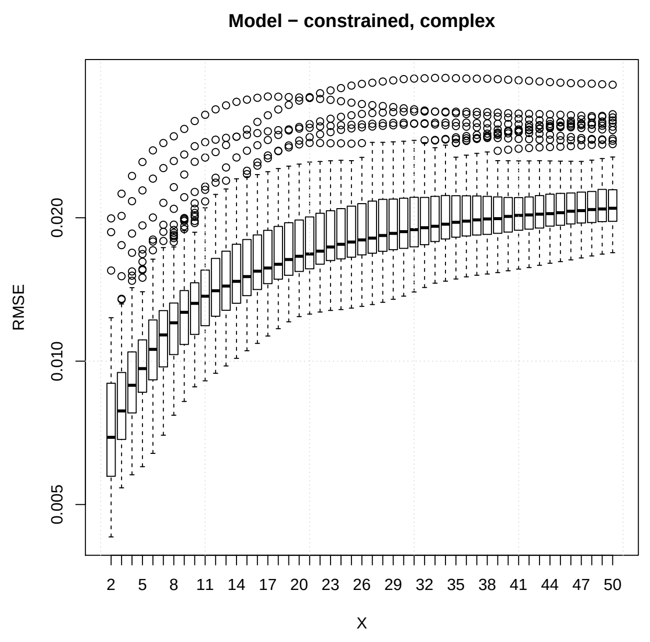

Figure 13 Performance of the constrained complex model for different values of X: Dataset 1.

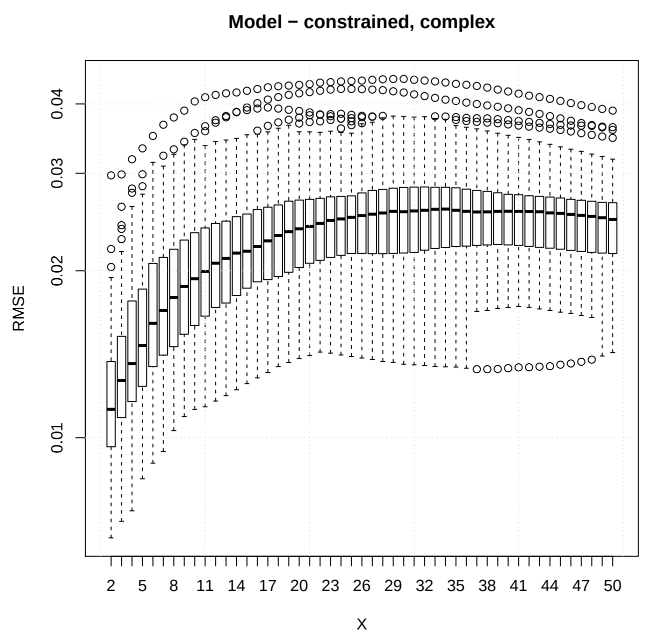

Figure 14 Performance of the constrained complex model model for different values of X: Dataset 2.

when it gets the right side, it starts to deteriorate. The same behaviour applies to all X's shown ($X = 5, 10, 25$, and 50, these values of X were selected at random).

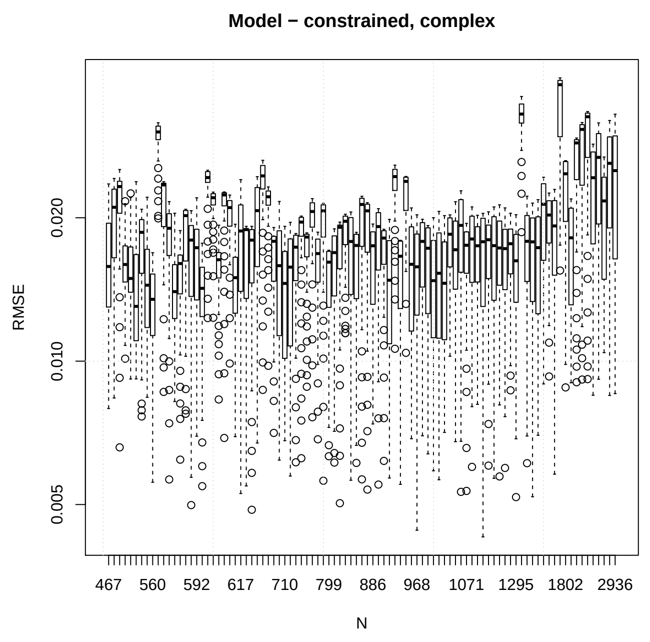

Figure 15 Performance of the constrained complex model for different values of N: Dataset 1.

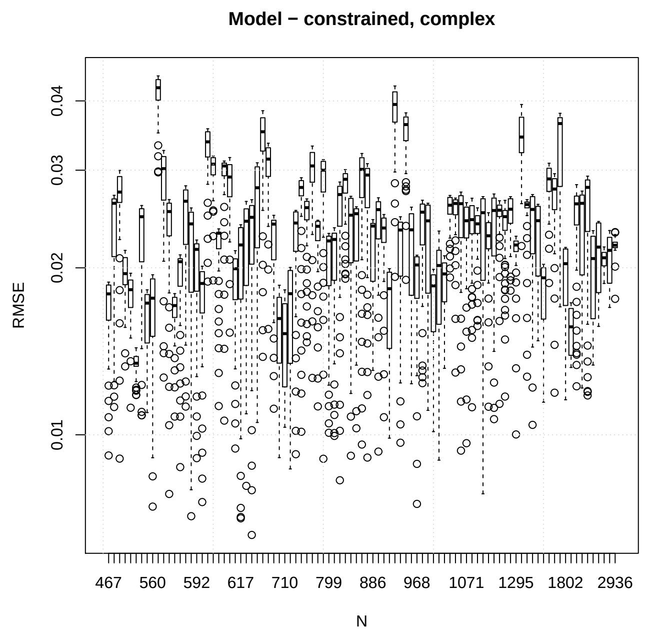

Figure 16 Performance of the constrained complex model model for different values of N: Dataset 2.

In the case of Dataset 1, the model yields better results (i.e. lower RMSE) for data splits per-month in comparison with the data splits per-quarter, as can be seen

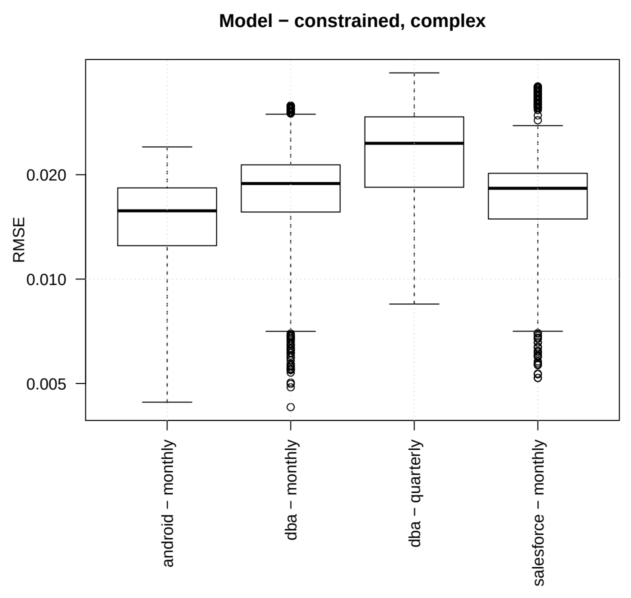

Figure 17 Performance of the constrained complex mode per dataset: Dataset 1.

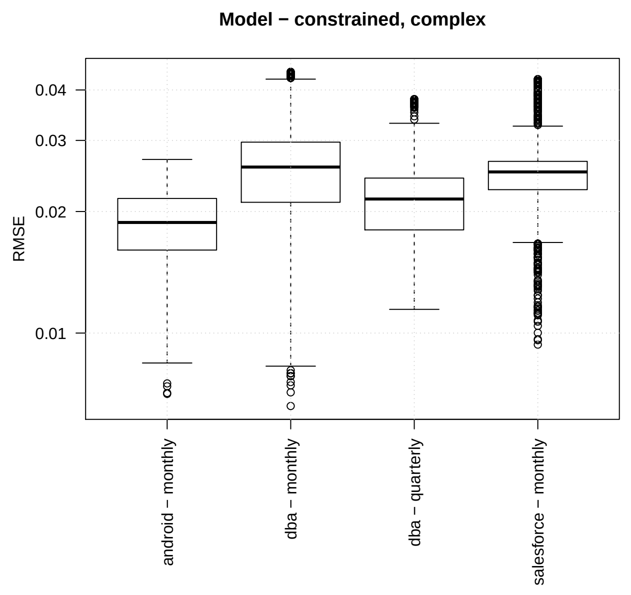

Figure 18 Performance of the constrained complex model per dataset: Dataset 2.

in Figure 17. We conjecture that the larger number of documents in the quarterly subsets plays a role here: Figure 15 shows that RMSE increases slightly for large values of N.

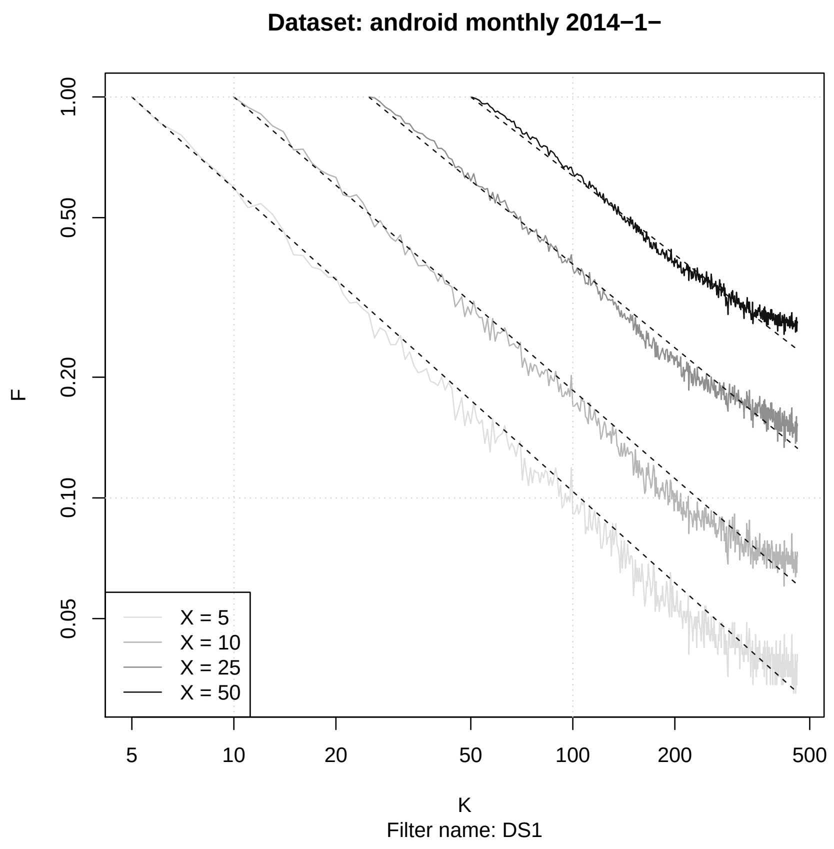

Figure 19 Fitting for a selection of top X using the constrained-complex model for Dataset 1

However, in the case of Dataset 2 the RMSE for per-quarter data splits is on par or better than in the case of per-month splits (as can be seen in Figure 18). We also see that for small and large values of N the performance is comparable as shown in Figure 16. We conjecture that this can be explained by the fact that we truncate our K values at 200 and get rid of the error factor contributed by the larger values of K.

4.2.1 The case when $K > 0.75N$

Based on our experiments, Equation 2 holds for $K \leq 0.75N$. As $K \to N$, the Power Law approximation deteriorates (see Figure 8 for an example). Likely, this is because the average number of documents per topic becomes too small for LDA to process. It seems, empirically, that the average number of posts per topic should be 4 or greater, hence the $K \leq 0.75N$ constraint.

4.2.2 Summary

In conclusion, we can appreciate graphically (in Figure 20) the "path" that leads to the best-performing model $F = X^{-b(X,N)}K^{b(X,N)}$ with the more complex prediction variable $b(X,N) = \alpha_1 + \alpha_2 X + \alpha_3 N + \alpha_4 \ln(X)$. The model performs well with the $K \leq 0.75N$ (Dataset 1) filter.

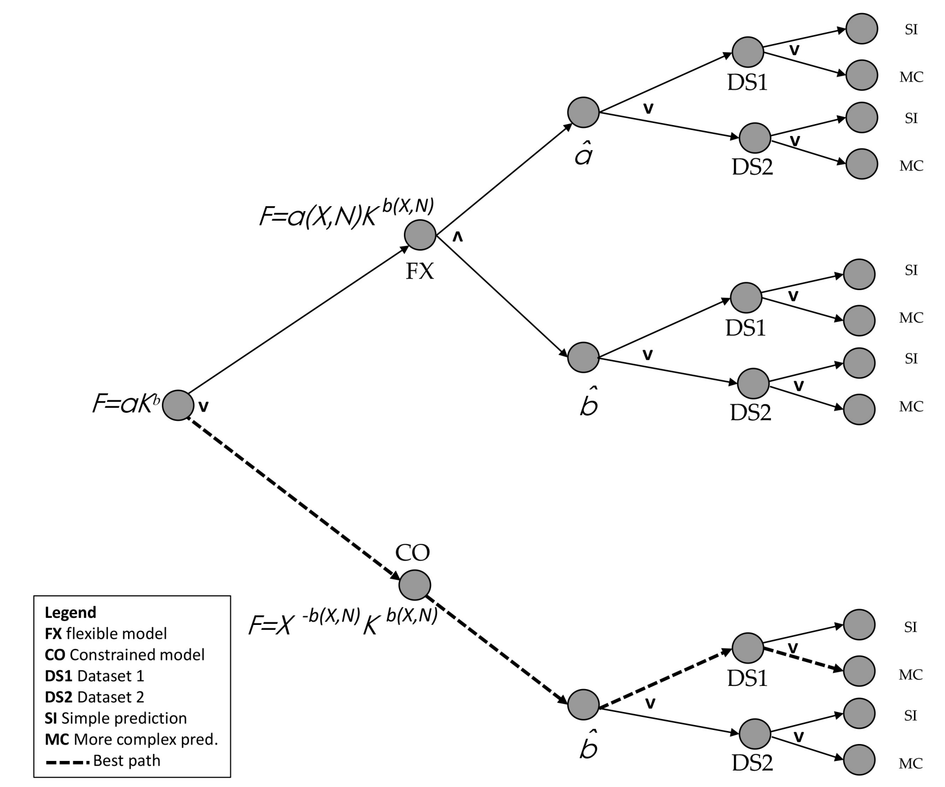

Figure 20 Decision tree for selecting the best fitting model. The path to the best-performing model is given by the dashed line.

5 Threats to Validity

In the following paragraphs we outline some of the limitations of our study and their impacts as per [18, 19].

Threats to conclusion validity are about the degree to which conclusions attained about relationships in our data are reasonable. To verify the absence of overfitting, we performed 10-fold cross validation analysis of our best model.

Threats to construct validity involves the relationship among the concepts and theories that backs the experiment and what is measured and affected. In this regard, the experiments that we have carried out were designed based on formal knowledge. We have also used the data from three different sources, partitioning the data either monthly or quarterly.

Threats to internal validity [20] comprise potential errors in our execution of the study process, these errors may influence the accuracy of our results and the conclusions we deduce from them. In order to avoid introducing bias we used an automated procedure for data extraction and processing, that included the creation of scripts in Perl and R languages.

Threats to external validity [21] include the degree to which we can generalize our results. The subjects of our study contain three datasets for several periods comprising thousands of data points. The results cannot be generalized to other datasets, rather the design of this study is based on the concept of the critical case [22]. If the model does not work "out-of-the-box", one can re-calibrate it by following our methodology.

6 Summary

LDA is a widespread information retrieval probabilistic topic model technique for performing analysis of text corpora. We have reviewed its roots, its basic functioning, its applications in the Software Engineering (SE) arena, exemplified its usage, and derived a formula from SE-related text corpora to calibrate the model.

The problem that we have addressed in this thesis, was to find in an expedited manner the number of topics (K), which the LDA model requires to be implemented.

To answer the research question "How can we quickly select the number of topics K so that the top X topics include a certain fraction F of the N documents under study?", we created a simple, closed-form Power Law expression (11), estimating K with X, F, and N as input. Although Power Law occurs frequently in SE [16], to the best of our knowledge, this is the first appearance of the power law in LDA parameter calibration. Moreover, we established that LDA models and (11) become unstable if $K > 0.75N$.

Practitioners can accelerate LDA analysis by using (11) with (13) to suggest the number of topics required to answer a particular question (e.g., to identify customers' pain-points and to prioritize maintainers' tasks). They can forego the often computationally-prohibitive current practice of iterating over all values of K to identify the optimal value. Formula (11) is also of interest to theoreticians as it suggests that different SE-related text corpora (in our case a technical Q&A forum and an issue-tracking system) might have similar underlying properties.

The formula is validated on three datasets and, as discussed in Section 5, cannot be generalized to other datasets. However the results do hint the existence of underlying structural similarities in the datasets.

In the future, we plan to analyze additional SE-related text corpora and extend our work to other domains. Also, we would like to find ways for improving the formula for K.

References

1. Grant, S., Cordy, J.R.: Estimating the optimal number of latent concepts in source code analysis. In: 2010 10th IEEE Working Conference on Source Code Analysis and Manipulation, pp. 65–74 (2010)
2. Endres, F., Plagemann, C., Stachniss, C., Burgard, W.: Unsupervised discovery of object classes from range data using latent dirichlet allocation. In: Robotics: Science and Systems, vol. 2, pp. 113–120. pdfs.semanticscholar.org, ??? (2009)
3. Aggarwal, A., Waghmare, G., Sureka, A.: Mining issue tracking systems using topic models for trend analysis, corpus exploration, and understanding evolution. In: Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering. RAISE 2014, pp. 52–58. ACM, New York, NY, USA (2014)
4. Asuncion, A., Welling, M., Smyth, P., Teh, Y.W.: On smoothing and inference for topic models. Uncertainty in Articial Intelligence (2009)
5. Jain, R., Ghaisas, S., Sureka, A.: SANAYOJAN: A framework for traceability link recovery between use-cases in software requirement specification and regulatory documents. In: Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering. RAISE 2014, pp. 12–18. ACM, New York, NY, USA (2014)
6. Asuncion, H.U., Asuncion, A.U., Taylor, R.N.: Software traceability with topic modeling. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, pp. 95–104. ACM, ??? (2010)
7. Blei, D.M.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
8. Grant, S., Cordy, J.R., Skillicorn, D.B.: Using heuristics to estimate an appropriate number of latent topics in source code analysis. Science of Computer Programming, 1673–1678 (2013)
9. Wang, B., Liu, Y., Liu, Z., Li, M.: Topic selection in latent dirichlet allocation. In: 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 756–760. ieeexplore.ieee.org, ??? (2014)

10. stackexchange.com: Android Enthusiasts Questions and Answers. http://android.stackexchange.com. Accessed: 2015-9-NA
11. stackexchange.com: Database Administrators Questions and Answers. http://dba.stackexchange.com. Accessed: 2015-9-NA
12. stackexchange.com: Salesforce Questions and Answers. https://salesforce.stackexchange.com. Accessed: 2015-9-NA
13. archive.org: Internet Archive. https://archive.org/download/stackexchange. Accessed: 2015-9-NA
14. Hornik, K., Grün, B.: topicmodels: An R package for fitting topic models. J. Stat. Softw. **40**(13), 1–30 (2011)
15. Steyvers, M., Griffiths, T.: Handbook of Latent Semantic Analysis, chapter 21: Probabilistic Topic Models. Lawrence Erbaum Associates, Mahwah, New Jersey, London (2007)
16. Louridas, P., Spinellis, D., Vlachos, V.: Power laws in software. ACM Trans. Softw. Eng. Methodol. **18**(1), 2–1226 (2008)
17. Newman, M.: Power laws, pareto distributions and zipf's law. Contemporary Physics **46**(5), 323–351 (2005)
18. Conradi, R., Wang, A.I.: Empirical Methods and Studies in Software Engineering: Experiences from ESERNET vol. 2765. Springer, ??? (2003)
19. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering **14**(2), 131–164 (2009)
20. Dieste, O., Grim, A., Juristo, N., Saxena, H., *et al.*: Quantitative determination of the relationship between internal validity and bias in software engineering experiments: Consequences for systematic literature reviews. In: Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium On, pp. 285–294 (2011). IEEE
21. Siegmund, J., Siegmund, N., Apel, S.: Views on internal and external validity in empirical software engineering. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 1, pp. 9–19. ieeexplore.ieee.org, ??? (2015)
22. Yin, R.K.: Case Study Research: Design and Methods, 5th edn. Sage publications, ??? (2013)