What is Wrong with Topic Modeling?
(and How to Fix it Using Search-based Software Engineering)

Amritanshu Agrawal*, Wei Fu*, Tim Menzies*

Department of Computer Science, North Carolina State University, Raleigh, NC, USA

Abstract

Context: Topic modeling finds human-readable structures in unstructured textual data. A widely used topic modeling technique is Latent Dirichlet allocation. When running on different datasets, LDA suffers from "order effects", i.e., different topics are generated if the order of training data is shuffled. Such order effects introduce a systematic error for any study. This error can relate to misleading results; specifically, inaccurate topic descriptions and a reduction in the efficacy of text mining classification results.
Objective: To provide a method in which distributions generated by LDA are more stable and can be used for further analysis.
Method: We use LDADE, a search-based software engineering tool which uses Differential Evolution (DE) to tune the LDA's parameters. LDADE is evaluated on data from a programmer information exchange site (Stackoverflow), title and abstract text of thousands of Software Engineering (SE) papers, and software defect reports from NASA. Results were collected across different implementations of LDA (Python+Scikit-Learn, Scala+Spark) across Linux platform and for different kinds of LDAs (VEM, Gibbs sampling). Results were scored via topic stability and text mining classification accuracy.
Results: In all treatments: (i) standard LDA exhibits very large topic instability; (ii) LDADE's tunings dramatically reduce cluster instability; (iii) LDADE also leads to improved performances for supervised as well as unsupervised learning.
Conclusion: Due to topic instability, using standard LDA with its "off-the-shelf" settings should now be depreciated. Also, in future, we should require SE papers that use LDA to test and (if needed) mitigate LDA topic instability. Finally, LDADE is a candidate technology for effectively and efficiently reducing that instability.

Keywords: Topic modeling, Stability, LDA, tuning, differential evolution.

1. Introduction

The current great challenge in software analytics is understanding unstructured data. As shown in Figure 1, most of the planet's 1600 Exabytes of data does not appear in structured sources (databases, etc) [1]. Mostly the data is of *unstructured* form, often in free text, and found in word processing files, slide presentations, comments, etc.

Such unstructured data does not have a pre-defined data model and is typically text-heavy. Finding insights among unstructured text is difficult unless we can search, characterize, and classify the textual data in a meaningful way. One of the common techniques for finding related topics within unstructured text (an area called topic modeling) is Latent Dirichlet allocation (LDA) [2].

This paper explores systematic errors in LDA analysis. LDA is a non-deterministic algorithm since its internal weights are updated via a stochastic sampling process (described later in this paper). We show in this paper that this non-determinism means that the topics generated by LDA on SE data are subject to order effects, i.e., different input orderings can lead to different topics. Such instability can:

- Confuse users when they see different topics each time the algorithm is re-run.
- Reduce the efficacy of text mining classifiers that rely on LDA to generate their input training data.

To fix this problem, we propose LDADE: a combination of

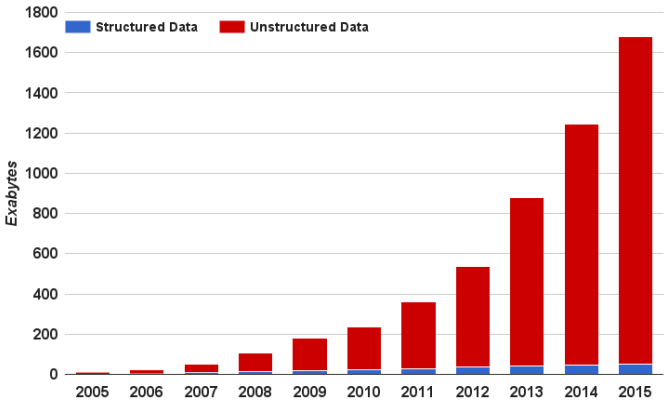

Figure 1: Data Growth 2005-2015. From [1].

*Corresponding author: Tel: +1-919-637-0412 (Amritanshu)
Email addresses: aagrawa8@ncsu.edu (Amritanshu Agrawal), wfu@ncsu.edu (Wei Fu), tim.menzies@gmail.com (Tim Menzies)
URL: https://amritag.wixsite.com/amrit (Amritanshu Agrawal)

LDA and a search-based optimizer (differential evolution, or DE) [3]) that automatically tunes LDA's $< k, \alpha, \beta >$ parameters. We prefer LDADE to other methods for two reasons, 1) LDADE is orders of magnitude faster and 2) other methods do not address the problem of order effects (for evidence on this second point, see our comparison against the LDA-GA method used by Panichella et al. [4], in the results section).

This paper tests LDADE by applying text mining to three data sets: 1) Data from a programmer information exchange site (Stackoverflow), 2) Title and abstract text of 15121 SE papers (Citemap) and 3) Software defect reports from NASA (Pits). Using these datasets, we explore these research questions:

- **RQ1**: **Are the default settings of LDA incorrect?** We will show that using the default settings of LDA for SE data can lead to systematic errors since stability scores start to drop after $n = 5$ terms per topic.
- **RQ2**: **Does LDADE improve the stability scores?** LDADE dramatically improves stability scores using the parameters found automatically by DE.
- **RQ3**: **Does LDADE improve text mining classification accuracy?** Our experiments shows that LDADE also improves classification accuracy.
- **RQ4**: **Do different data sets need different configurations to make LDA stable?** LDADE finds different "best" parameter settings for different data sets. Hence reusing tunings suggested by any other previous study for any dataset is *not* recommended. Instead, it is better to use automatic tuning methods to find the best tuning parameters for the current data set.
- **RQ5**: **Are our findings consistent when using different kinds of LDA or with different implementations?** Results were collected across different implementations of LDA across Linux platform and for different kinds of LDAs. Across all these implementations, the same effect holds: (a) standard LDA suffers from order effect and topic instability and (b) LDADE can reduce that instability.
- **RQ6**: **Is tuning easy?** We show that, measured in the terms of the internal search space of the optimizer, tuning LDA is much simpler than standard optimization methods.
- **RQ7**: **Is tuning extremely slow?** The advantages of LDADE come at some cost: tuning with DE makes LDA three to five times slower. While this is definitely more than not using LDA, this may not be an arduous increase given modern cloud computing environments.
- **RQ8**: **How better LDADE is compared against LDA-GA?** Our analysis confirms that LDADE is much faster and achieves stabler results than LDA-GA.
- **RQ9**: **Should topic modeling be used "off-the-shelf" with their default tunings?** Based on these findings, our answer to this question is an emphatic "no". We can see little reason to use "off-the-shelf" LDA for any kind of SE data mining applications.

The rest of this paper is structured as follows. Section 2 argues that stabilizing the topics generated by LDA is important for several reasons. Related work is reviewed in Section 3 and the methods of this paper are discussed in Section 4. We have answered above research questions in Section 5. This is followed by a discussion on the validity of our results and a section describing our conclusions. Note that the main conclusion of this paper is that, henceforth, we should require SE papers that use LDA to test and (if needed) mitigate LDA topic instability.

2. Motivation

2.1. LDA is Widely Used

We study LDA since this algorithm is a widely-used technique in recent research papers appearing in prominent SE venues. Tables 1 [28] and 2 show top SE venues that published SE results and a sample of those papers respectively. For details on how Table 2 is generated, see Section 3.4.

As to *how* LDA is used, it is important to understand the distinction between supervised and unsupervised data mining algorithms. In the general sense, most data mining is "supervised" when you have data samples associated with labels and then use machine learning tools to make predictions. For example, in the case of [16, 26], the authors used LDA as a feature extractor to build feature vectors which, subsequently, were passed to a learner to predict for a target class. Note that such supervised LDA processes can be fully automated and do not require human-in-the-loop insight to generate their final conclusions.

However, in case of, "unsupervised learning", the final conclusions are generated via a manual analysis and reflection over, e.g., the topics generated by LDA. Such cases represent [7, 21] who used a manual analysis of the topics generated from some SE tasks textual data by LDA as part of the reasoning within the text of that paper. It is possible to combine manual and automatic methods, please see the "Semi-supervised" LDA paper of Le et al. [24].

However, as shown in our sample, one observation could be made that out of the 28 studies in Table 2, 23 of them make

Table 1: Top SE venues that published on Topic Modeling from 2009 to 2016.

Venue	Full Name	Count
ICSE	International Conference on Software Engineering	4
CSMR-WCRE / SANER	International Conference on Software Maintenance, Reengineering, and Reverse Engineering / International Conference on Software Analysis, Evolution, and Reengineering	3
ICSM / ICSME	International Conference on Software Maintenance / International Conference on Software Maintenance and Evolution	3
ICPC	International Conference on Program Comprehension	4
ASE	International Conference on Automated Software Engineering	3
ISSRE	International Symposium on Software Reliability Engineering	2
MSR	International Working Conference on Mining Software Repositories	8
OOPSLA	International Conference on Object-Oriented Programming, Systems, Languages, and Applications	1
FSE/ESEC	International Symposium on the Foundations of Software Engineering / European Software Engineering Conference	1
TSE	IEEE Transaction on Software Engineering	1
IST	Information and Software Technology	3
SCP	Science of Computer Programming	2
ESE	Empirical Software Engineering	4

Table 2: A sample of the recent literature on using topic modeling in SE. Sorted by number of citations (in column3). For details on how this table was generated, see Section 3.4.

REF	Year	Citations	Venues	Mentions instability in LDA?	Uses Default Parameters	Does tuning?	Conclusion	Tasks / Use cases	Unsupervised or Supervised
[5]	2011	112	WCRE	Y	Y	N	Explored Configurations without any explanation.	Bug Localisation	Unsupervised
[6]	2010	108	MSR	Y	Y	N	Explored Configurations without any explanation. Reported their results using multiple experiments.	Traceability Link recovery	Unsupervised
[7]	2014	96	ESE	Y	Y	N	Explored Configurations without any explanation. Choosing right set of parameters is a difficult task.	Stackoverflow Q&A data analysis	Unsupervised
[4]	2013	75	ICSE	Y	Y	Y	Uses GA to tune parameters. They determine the near-optimal configuration for LDA in the context of only some important SE tasks.	Finding near-optimal configurations	Semi-Supervised
[8]	2013	61	ICSE	Y	Y	N	Explored Configurations without any explanation.	Software Requirements Analysis	Unsupervised
[9]	2011	52	MSR	Y	Y	N	They validated the topic labelling techniques using multiple experiments.	Software Artifacts Analysis	Unsupervised
[10]	2014	44	RE	Y	Y	N	Explored Configurations without any explanation.	Requirements Engineering	Unsupervised
[11]	2011	44	ICSE	Y	Y	N	Open issue to choose optimal parameters.	A review on LDA Mining software repositories using topic models	Unsupervised
[12]	2014	35	SCP	Y	Y	N	Explored Configurations without any explanation.	Software Artifacts Analysis	Unsupervised
[13]	2012	35	MSR	Y	Y	N	Choosing the optimal number of topics is difficult.	Software Defects Prediction	Unsupervised
[14]	2014	31	ESE	Y	Y	N	Choosing right set of parameters is a difficult task.	Software Testing	Unsupervised
[15]	2009	29	MSR	Y	Y	N	Explored Configurations without any explanation and accepted to the fact their results were better because of the corpus they used.	Software History Comprehension	Unsupervised
[16]	2013	27	ESEC/FSE	Y	Y	Y	Explored Configurations using LDA-GA.	Traceability Link recovery	Supervised
[17]	2014	20	ICPC	Y	Y	N	Use heuristics to find right set of parameters.	Source Code Comprehension	Unsupervised
[18]	2013	20	MSR	Y	Y	N	In Future, they planned to use LDA-GA.	Stackoverflow Q&A data analysis	Unsupervised
[19]	2014	15	WebSci	Y	Y	N	Explored Configurations without any explanation.	Social Software Engineering	Unsupervised
[20]	2013	13	SCP	Y	Y	N	Their work focused on optimizing LDAs topic count parameter.	Source Code Comprehension	Unsupervised
[21]	2012	13	ICSM	Y	Y	N	Explored Configurations without any explanation.	Software Requirements Analysis	Unsupervised
[22]	2015	6	IST	Y	Y	N	Explored Configurations without any explanation. Choosing right set of parameters is a difficult task.	Software re-factoring	Supervised
[23]	2016	5	CS Review	Y	Y	N	Explored Configurations without any explanation.	Bibliometrics and citations analysis	Unsupervised
[24]	2014	5	ISSRE	N	Y	N	Explored Configurations without any explanation.	Bug Localisation	Semi-Supervised
[25]	2015	3	JIS	Y	Y	N	They improvised LDA into ISLDA which gave stability across different runs.	Social Software Engineering	Unsupervised
[26]	2015	2	IST	Y	Y	Y	Explored Configurations using LDA-GA.	Software Artifacts Analysis	Supervised
[27]	2016	0	JSS	N	Y	N	Explored Configurations without any explanation. Choosing right set of parameters is a difficult task.	Software Defects Prediction	Unsupervised

extensive use of LDA for the purposes of *unsupervised exploration*. At some point these papers, browsed the LDA topics to guide their subsequent analysis. For example: 1) Barua et al. [7] used the LDA topics to summarize the topics and trends in Stackoverflow, 2) Galvis et al. [8] used LDA to gain an understanding of the nature of user statements in requirements documents, and many more.

As witnessed by the central columns of Table 2, many prior papers [4, 16, 26] have commented that the results of a topic modeling analysis can be affected by tuning the control parameters of LDA. Yet as reported in Section 3.4, a repeated pattern in the literature is that, despite these stated concerns, researchers rarely take the next step to find ways to find better control tunings.

2.2. *Standard LDA Can Make Misleading Conclusions*

Standard practice in papers that use LDA is to present a table showing the top, say, 40 topics [7]. This section shows one example where, due to LDA instability, the contents of such tables can only be described as mostly illusionary. Later in this paper (in Section 5.1) we show that this example is actually representative of a general problem: changing the input ordering dramatically changes the topics reported by LDA.

Barua et al. [7] analysed the topics and trends in Stackoverflow in which they reported the top 40 topics found at Stackoverflow. Table 3 shows our attempt to reproduce their results. Note that we could not produce a verbatim reproduction since they used the data of Stackvoverflow dumped on 2012 while we used the current dump of 2016. The analysis in Table 3 is generated by running LDA twice with different randomly gen-

Table 3: LDA topic instability. Shows results from two runs. Column1 reports the percent of words from a topic seen in its nearest match in the second run.

% overlap with closest topic in run2	Topic name	top 9 words in topic
100	Xaml Binding	grid window bind valu wpf silverlight control xaml properti
100	Ruby on Rails	rubi rail gem lib user end rvm app requir
100	MVC	http com java apach bean org springframework sun servlet
88	Objective C	self cell nsstring iphon nil object anim alloc view
88	Function Return Types	function const char void includ amp return int std
77	Testing	chang tri work use problem like set code test
77	Socket Communications	Main send socket connect run Android Activity thread start task process time
77	Q and A	know need want way use question time like make
77	OO Programming	instanc void type new method class return object public
77	HTML Links	com http www url html site content page link
77	Display	height jpg imag png src size img color width
66	Windows/VS Tips	studio net dll web use mvc control asp visual
66	Website Design	left span style text class div color css width
66	Web Development	script jqueri function form input type ajax var javascript
66	Java Programming	java void new privat return null int public string
66	Date/Time Format	select day valu option year format month time date
66	Database	column order group select join tabl row queri null
66	Android User View	height content textview parent wrap android width view layout
66	.NET Framework	net form text checkbox control click label asp button
55	iOS App Development	applic user need iphon develop want app use like
55	MySQL	connect sql databas tabl row queri data mysql server
55	HTML Form	form login user usernam password page XML control action session
55	Git Operations	folder upload open git use path read file directori
55	Email Message	address form send field email messag error valid contact
55	Eclipse	maven jar target build eclips version depend plugin project
44	iOS App Development	applic user need iphon develop want app use like
44	Regular Expressions	function array replac regex match echo php post string
44	Media Player	object video play game player obj use data audio
44	Float number Manipulation	doubl valu count rang number data length float point
44	Compiling	librari compil lib includ usr command file error test
44	.NET Framework	net form text checkbox control click label asp button
33	Ruby Version Manager	end lib user rvm rubi rail app gem requir
33	Scripting Language	def line self modul print import templat django python
33	NodeJS	tag function node express parent use list like variabl
33	C# Programming	net web asp servic mvc use control applic wcf
33	Python Programming	line python file print command script curl output run
33	Android Debugging	com java lang debug method info android error androidruntim
22	Visual Studio	window project file error librari compil visual build dll
22	Information Systems	messag email log address contact send mail phone locat
22	Flex4 Development	flash function new var list click sub event item

erated input orderings (keeping all other parameter settings intact). After that, the topics generated from the first run where scored according to the overlap of their words from the second run.

We observed that while a very few topics appear verbatim in the two runs (see the top three lines of Table 3), but most do not. In fact, 25 of the 40 topics in that figure show instability (have an overlap of 55% or less).

In Table 3, we only examine the top $n = 9$ words in each topic of run1 and run2. We selected $n = 9$ since, later in this paper, we find that an analysis of $n > 9$ words per topics leads to near zero percent overlap of the topics generated in different runs (see Section 5.1). That is, the instabilities of Table 3 get even *worse* if we use *more* of the LDA output.

2.3. LDA Stabilization Means Better Inference

Inference with LDA can be assessed via topic similarities (as done in Table 3) and via the classification performance if the LDA topics are used as features to be fed into a classifier. As shown later in this paper, we can use LDADE to increase the similarities of the LDA topics generated by LDA (see Section 5.2).

As to the effects on classification accuracy, one way to score a classifier is via the F_β score that combines *precision* p and *recall* r as follows:

$$F_\beta = (1+\beta^2)\frac{pr}{p\beta^2 + r} \quad (1)$$

We compared the analysis with F_1 ($\beta = 1$) and F_2 ($\beta = 2$) metrics seen in text mining classification using standard and stable topics generated by LDADE. The F_2 score is useful when reporting classification results since it favors classifiers that do not waste the time on false positives. In Section 5.3, we report significant and large improvements in both the evaluation metrics. That is, LDADE not only improves topic stability, but also the efficacy of the inference that subsequently uses those topics.

3. Related Work

3.1. Topic Modeling

LDA is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. It learns the various distributions (the set of topics, their associated word probabilities, the topic of each word, and the particular topic mixture of

Table 4: Example (from [7]) of generating topics from Stackoverflow. For each topic, we show just the five most heavily weighted words.

Topic: String Manipulation	Topic: Function	Topic: OO Programming	Topic: UI Development	Topic: File Operation
string	function	class	control	file
charact	paramet	method	view	directori
encod	pass	object	event	path
format	return	call	button	folder
convert	argument	interfac	click	creat

each document). What makes topic modeling interesting is that these algorithms scale to very large text corpuses. For example, in this paper, we apply LDA to whole of Stackoverflow, as well as to two other large text corpuses in SE.

Table 4 illustrates topic generation from Stackoverflow. To find these topics, LDA explores two probability distributions:

- $\alpha = P(k|d)$, probability of topic k in document d;
- $\beta = P(w|k)$, probability of word w in topic k.

Initially, α and β may be set randomly as follows: each word in a document was generated by first randomly picking a topic (from the documents distribution of topics) and then randomly picking a word (from the topics distribution of words). Successive iterations of the algorithm count the implications of prior sampling which, in turn, incrementally updates α and β.

Binkley et al. [17] performed an extensive study and found that apart from α and β, the other parameters that define LDA are:

- k = number of topics
- b = number of burn-in iterations;
- si = the sampling interval.

Binkley et al.'s study of the LDA settings was a mostly manual process guided by their considerable background knowledge and expertise and program comprehension. In the field of program comprehension, the Binkley article is the state of the art in applications of LDA to software engineering.

To that work, this paper adds a few extra conclusions. Firstly, we explore LDA in fields other than program comprehension. Secondly, we ask the question "what if the analysts lacks extensive background knowledge of the domain?". In that circumstance, some automatic method is needed to support an informed selection of the LDA parameters.

3.2. About Order Effects

This paper uses tuning to fix "order effects" in topic modeling. Langley [29] defines such effects as follows:

> *A learner L exhibits an order effect on a training set T if there exist two or more orders of T for which L produces different knowledge structures.*

Many learners exhibit order effects, e.g., certain incremental clustering algorithms generate different clusters, depending on the order with which they explore the data [29]. Hence, some algorithms survey the space of possible models across numerous random divisions of the data (e.g., Random Forests [30]).

From the description offered above in Section 3.1, we can see how topic modeling might be susceptible to order effects and how such order effects might be tamed: 1) In the above description, k, α and β are initialized at random then updated via an incremental re-sampling process. Such incremental updates are prone to order effects, and 2) other way is to initialize, k, α and β to some useful value. As shown in Section 5.4, the key to applying this technique is that different data sets will require different initializations, i.e., the tuning process will have to be repeated for each new data set.

3.3. Tuning: Important and (Mostly) Ignored

The impact of tuning is well understood in the theoretical machine learning literature [31]. When we tune a data miner, what we are really doing is changing how a learner applies its heuristics. This means tuned data miners use different heuristics, which means they ignore different possible models, which means they return different models, i.e., *how* we learn changes *what* we learn.

Yet issues relating to tuning are poorly addressed in the software analytics literature. Fu et al. [32] surveyed hundreds of recent SE papers in the area of software defect prediction from static code attributes. They found that most SE authors do not take steps to explore tunings (rare exception: [33]). For example, Elish et al [34] compared support vector machines to other data miners for the purposes of defect prediction. That paper tested different "off-the-shelf" data miners on the same data set, without adjusting the parameters of each individual learner. Similar comparisons of data miners in SE, with no or minimal pre-tuning study, can be found in the work on Lessmann et al. [35] and, most recently, in Yang et al [36].

In summary, all our literature reviews of the general (non-LDA) software analytics literature show that the importance of tuning is often mentioned, but never directly addressed.

3.4. LDA, Instability and Tuning

Within the LDA literature, some researchers have explored LDA instability. We searched scholar.google.com for papers published before August 2016, for the conjunction of "lda" and "topics" or "stable" or "unstable" or "coherence". Since 2012, there are 189 such papers, 57 of which are related to software engineering results. Table 2 gives a broad discussion on these papers. In short, of those papers:

- 28/57 mention instability in LDA.
- Of those 28, despite mentioning stability problems, 10 papers still used LDA's "off-the-shelf" parameters;
- The other 28-10=18 papers used some combination of manual adjustment or some under-explained limited exploration of tunings based on "engineering judgment" (i.e., some settings guided by the insights of the researchers).
- Only 4 of the authors acknowledge that tuning might have a large impact on the results.

Apart from tuning, there are several other workarounds explored in the literature in order to handle LDA instability. Overall, there was little systematic exploration of tuning and LDA in the SE literature. Instead, researchers relied on other methods that are less suited to automatic reproduction of prior results.

In the literature, researchers [10, 37, 38] manually accessed the topics and then used for further experiments. Some made

use of Amazon Mechanical Turk to create gold-standard coherence judgements [39]. All these solutions are related to results stability rather than model stability. Note that this workaround takes extensive manual effort and time.

Another approach to tame LDA instability is to incorporate user knowledge into the corpus. For example, SC-LDA [40] can handle different kinds of knowledge such as word correlation, document correlation, document label and so on. Using such user knowledge, while certainly valuable, is somewhat subjective. Hence, for reasons of reproducibility, we prefer fully automated methods.

Some researchers used genetic algorithms to learn better settings for LDA [4, 16, 26]. Genetic algorithms are themselves a stochastic search process. That is, the changes to input orderings explored in this paper would introduce further conclusion instability from the genetic algorithms. In principle, that instability could be removed via extra runs of genetic algorithms over multiple sub-samples of that, where the GA goals are augmented to include "similar topics should be found in different runs". That said:

- None of the prior work using GAs to improve LDA have applied those sub-sampling stability test;
- If done naively, adding further goals and data subsampling to a GA runs the risk of dramatically increasing the runtimes. One reason to prefer LDADE is that it terminates very quickly.

Finally, other researchers explore some limited manual parameter tuning for LDA (e.g., experiment with one parameter: cluster size) [8, 41] to achieve higher stability by just increasing the number of cluster size. Note that the automatic tuning methods explored by this paper can explore multiple parameters. Further, our analysis is repeatable.

4. Methods

This section describes our evaluation methods for measuring instability as well as the optimization methods used to reduce that instability.

4.1. Data Sets

To answer our research questions, and to enable reproducibility of our results, we use three open source datasets summarized in Table 5 and described below. These 3 datasets are unrelated which solve different SE tasks. We wanted to make sure our LDADE is useful for these 3 tasks. This puts emphasis on the importance of stability in LDA.

PITS is a text mining data set generated from NASA software project and issue tracking system (PITS) reports [42, 43]. This text discusses bugs and changes found in big reports and review patches. Such issues are used to manage quality assurance, to support communication between developers. Topic modeling in PITS can be used to identify the top topics which can identify each severity separately. The dataset can be downloaded from the PROMISE repository [44]. Note that, this data comes from six different NASA projects, which we label as PitsA, PitsB, etc.

Table 5: Statistics on our datasets. PitsA, PitsB, etc refer to the issues from six different NASA projects.

	Size	
Data set	**Before Preprocessing**	**After Preprocessing**
PitsA	1.2 MB	292 KB
PitsB	704 KB	188 KB
PitsC	143 KB	37 KB
PitsD	107 KB	26 KB
PitsE	650 KB	216 KB
PitsF	549 KB	217 KB
Citemap	8.6 MB	3.7 MB
Stackoverflow	7 GB	589 MB

Stackoverflow is the flagship site of the Stack Exchange Network which features questions and answers on a wide range of topics in computer programming. There has been various studies done to find good topics on Stackoverflow for SE [7, 18, 45, 46]. Topic modeling on Stackoverflow is useful for finding patterns in programmer knowledge. This data can be downloaded online[1].

Citemap contains titles and abstracts of 15121 papers from a database of 11 senior software engineering conferences from 1992-2016. Most of this data was obtained in the form of an SQL dump from the work of Vasilescu et al. [47] and some are collected by Mathew et al [48]. People have studied healthiness of software engineering conferences [49]. This dataset is available online[2].

For this study, all datasets were preprocessed using the usual text mining filters [50]:

- Stop words removal using NLTK toolkit[3] [51] : ignore very common short words such as "and" or "the".
- Porter's stemming filter [52]: delete uninformative word endings, e.g., after performing stemming, all the following words would be rewritten to "connect": "connection", "connections", "connective", "connected", "connecting".
- Tf-idf feature selection: focus on the 5% of words that occur frequently, but only in small numbers of documents. If a word occurs w times and is found in d documents and there are W, D total number of words and documents respectively, then tf-idf is scored as follows:

$$tfidf(w,d) = \frac{w}{W} * \log\frac{D}{d}$$

Table 5 shows the sizes of our data before and after preprocessing. These datasets are of different sizes and so are processed using different tools: 1) PITS and Citemap is small enough to process on a single (four core) desktop machine using Scikit-Learn [53] and Python, and 2) Stackoverflow is so large (7GB) that its processing requires extra hardware support. This study used Spark and Mllib on a cluster of 45 nodes to reduce the runtime.

[1] http://tiny.cc/SOProcess
[2] https://github.com/ai-se/Pits_lda/blob/master/dataset/citemap.csv
[3] http://www.nltk.org/book/ch02.html

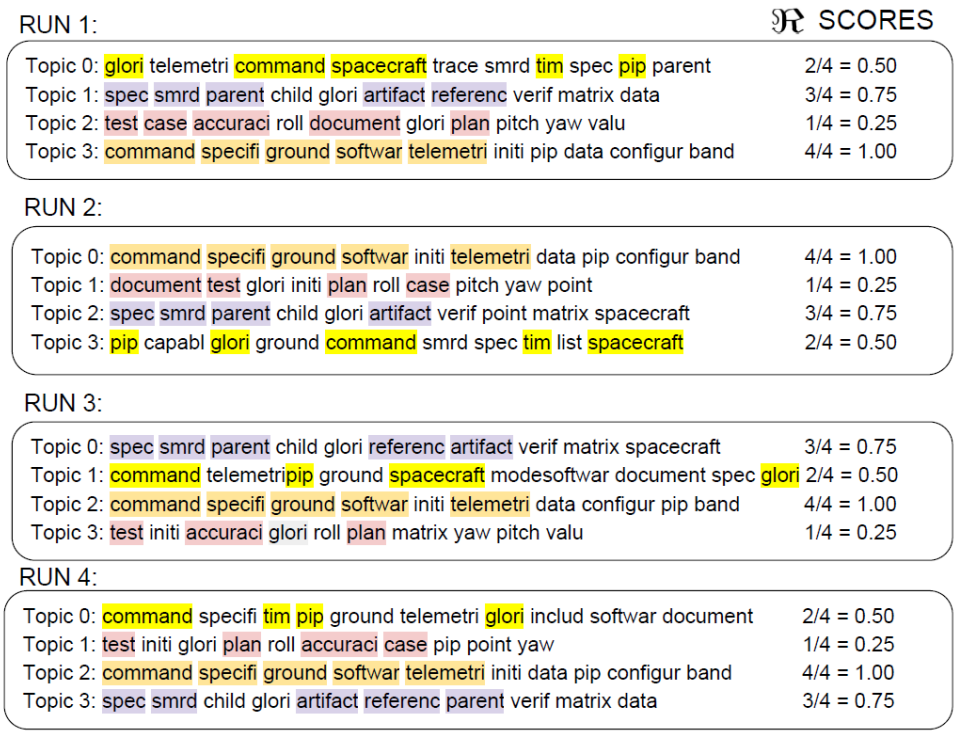

Figure 2: Example of topics overlap off size $n = 5$ across multiple runs.

4.2. Similarity Scoring

To evaluate topics coherence in LDA, there is a direct approach, by asking people about topics, and an indirect approach by evaluating *pointwise mutual information (PMI)* [39, 54] between the topic words. We could not use any of these criteria, as it requires experts to have domain knowledge. *Perplexity* is the inverse of the geometric mean per-word likelihood. The smaller the perplexity, the better (less uniform) is the LDA model. The usual trend is that as the value of perplexity drops, the number of topics should grow [19]. Researchers caution that the value of perplexity does not remain constant with different topic and corpus sizes [55]. Perplexity depend on its code implementation and the type of datasets used. Since we are using different implementations of LDA across different platforms on various datasets, we are not using perplexity as evaluation measure.

There is well known measure, called *Jaccard Similarity* [8, 54], for measuring similarity. But we modified the measure to do a cross-run similarity of topics. For this work, we assess topic model stability via the *median number overlaps of size n words* (*size_of_topic*), which is denoted as $\Re_n$.

For this measurement, we first determine the maximum size of topics we will study. For that purpose, we will study the case of $n \leq 9$ (we use 9 as our maximum size since the cognitive science literature tells us that 7 ± 2 is a useful upper size for artifacts to be browsed by humans [56]).

Next, for $1 \leq n \leq 9$, we will calculate the median size of the overlap, computed as follows:

- Let one *run* of our rig shuffle the order of the training data, then build topic models using the data;
- m runs of our rig execute m copies of one run, each time using a different random number seed,
- Topics are said to be stable, when there are x occurrences of n terms appearing in all the topics seen in the m runs.

For example, consider the topics shown in Figure 2. These are generated via four *runs* of our system. In this hypothetical example, we will assume that the runs of Figure 2 were generated by an LDA suffering from topic instability. For $n = 5$, we note that Topic 0 of run1 scores $\frac{2}{4} = 0.5$ since it shares 5 words with topics in only two out of four runs. Repeating that calculation for the other run1 topics shows that:

- Topic 1 of run1 scores $\frac{3}{4} = 0.75$;
- Topic 2 or run1 scores $\frac{1}{4} = 0.25$;
- Topic 3 of run1 scores $\frac{4}{4} = 1$.

From this information, we can calculate $\Re_5$ (the *median number overlaps of size* $n = 5$ *words*) as:

$$median(0.5, 0.75, 0.25, 1) = 0.625$$

Figure 3 shows the $\Re_n$ scores of Figure 2 for $1 \leq n \leq 9$. From this figure, we can see LDA topic instability since any report of the contents of a topic that uses more than three words per topic would be unreliable.

For the following analysis, we distinguish between the ***Raw score*** and the ***Delta score***:

- The two ***Raw scores*** are the $\Re_n$ median similarity scores seen *before* and *after* tuning LDA;
- The ***Delta score*** is the difference between the two ***Raw scores*** (after tuning - before tuning).

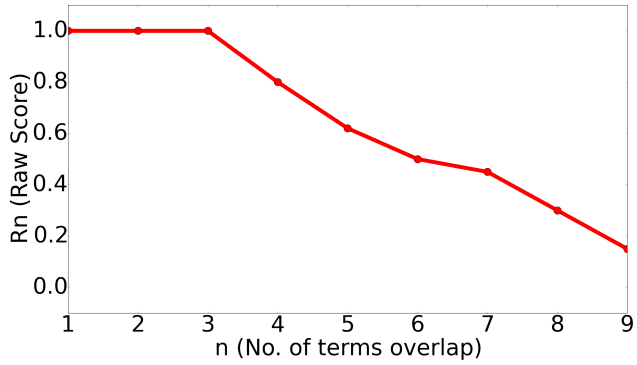

Figure 3: $\Re_n$ scores of Figure 2 for $1 \leq n \leq 9$

The pseudocode for these calculations is shown in Figure 4 (Algorithm 1) with the default set of parameters. In the following description, superscript numbers denote lines in the pseudocode. The data ordering is shuffled every time LDA is ran[7]. Data is in the form of *tfidf* scores of each word per document. Shuffling is done in order to induce maximum variance among the ordering of data with different runs of LDA. Function *lda*[8] returns k topics. Topics[8] are a list of lists which contains topics from all the different runs. A stability score is evaluated on every 10 runs (Fixed) of LDA, and this process is continued 10 (Fixed) times to avoid any sampling bias. At the end, the median score is selected as the untuned raw score ($\Re_n$)[4−12]. Hence, the runtimes comes from 10 evaluations of untuned experiment.

```
# Algorithm 1                                                    | 1
def LDASCORE(n, k, α, β, Data):                                  | 2
  Score = emptySet                                               | 3
  for j = 0 to 10 do                                             | 4
      Topics = emptySet                                          | 5
      for i = 0 to 10 do                                         | 6
          data = shuffle(Data)                                   | 7
          Topics.add(lda(k, α, β, data))                         | 8
      end for                                                    | 9
      Score.add(Overlap(Topics, n, k))                           | 10
  end for                                                        | 11
  Raw_Score = median(Score)                                      | 12
  return Raw_Score                                               | 13
```

Figure 4: Pseudocode for untuned LDA with Default Parameters

4.3. Tuning Topic Modeling with LDADE

LDADE is a combination of topic modeling (with LDA) and an optimizer (differential evolution, or DE) that adjusts the parameters of LDA in order to optimize (i.e., maximize) similarity scores.

We choose to use DE after a literature search on search-based SE methods. The literature mentions many optimizers: simulated annealing [57, 58]; various genetic algorithms [59] augmented by techniques such as DE (differential evolution [3]), tabu search and scatter search [60–63]; particle swarm optimization [64]; numerous decomposition approaches that use heuristics to decompose the total space into small problems, then apply a response surface methods [65, 66]. Of these, we use DE for two reasons. Firstly, it has been proven useful in prior SE tuning studies [32]. Secondly, our reading of the current literature is that there are many advocates for differential evolution like Vesterstrom et al. [67] showed DE to be competitive with particle swarm optimization and other GAs.

LDADE adjusts the parameters of Table 6. Most of these parameters were explained above. Apart from them, there are 2 different kinds of LDA implementations as well and they are: 1) VEM is the deterministic *variational EM* method that computes α and β via expectation maximization [68], and 2) Gibbs sampling [69, 70] is a Markov Chain Monte Carlo algorithm, which is an approximate stochastic process for computing and updating α and β. Topic modeling researchers in SE have argued that Gibbs leads to stabler models [71, 72] (a claim which we test, below).

We manually run these other inference techniques according to different implementations. We need to make sure that these instabilities do not hold for just 1 inference technique, or 1 implementation.

Table 6: List of parameters tuned by this paper

Parameters	Defaults	Tuning Range	Description
k	10	[10,100]	Number of topics or cluster size
α	None	[0,1]	Prior of document topic distribution. This is called alpha
β	None	[0,1]	Prior of topic word distribution. This is called beta

Figure 5 (Algorithm 2) shows the pseudocode of LDADE. DE evolves from the *NewGen* of candidates from a current *Pop*. Each candidate solution in the *Pop*[10] is a set of parameters (Tunings). The values of this set is selected randomly from Table 6 in the *InitializePopulation*[10] function. *Pop* variable is now a matrix of 10×3 since $np = 10$.

Cur_gen[9] and *NewGen*[16] variables are the list of Tunings, and $\Re_n$ score which comes similarly from Algorithm 1[4−12] (Figure 4). But for LDADE, everytime Algorithm 1 uses different values of parameters found by DE. The runtimes comes from 1 DE run which does about $iter * np$ evaluations of tuned experiment. DE is driven by a goal (called quality/fitness funtion), which in this case is maximizing the $\Re_n$ (raw) score calculated using *ldascore*[19] function.

The main loop of DE[15] runs over the *Pop*, replacing old items with new Candidates (if new candidate is better). DE generates *new Candidates* via *Extrapolate*[18] function between current solutions of the *Cur_gen* variable. Three solutions a, b and c [32] are selected at random from the *pop*. Each of these solution is nothing but a set of (k, α and β). For each tuning parameter i[34], at some crossover probability (cr [35]), we replace the *old* tuning with *newf*. We mutate a solution with the equation $y_i = a_i + f \times (b_i - c_i)$ where f is a parameter controlling crossover. The trim function[38] limits the new value to the legal range min..max of that parameter.

The loop invariant of DE is that, after the zero-th iteration[15], the *Pop* contains examples that are better than at least one other

```
# Algorithm 2
def LDADE(np = 10, # size of frontier
          f = 0.7, # differential weight
          cr = 0.3, # crossover probability
          iter = 3, # number of generations
          n, # words per topic
          Data, # tf*idf scores
          Goal ∈ Maximizing ℜn (Raw) score)
  Cur_Gen = emptySet
  Pop = InitializePopulation(np) # A matrix of 10 by 3
  for i = 0 to np − 1 do
      temp = ldascore(n, Pop[i], Data)
      Cur_Gen.add([Pop[i], temp])
  end for
  for i = 0 to iter do
      NewGen = emptySet
      for j = 0 to np − 1 do
          S_i = Extrapolate(Pop[j], Pop, n, cr, f, np)
          temp = ldascore(n, S_i, Data)
          if temp ≥ Cur_Gen[j][1] then
              NewGen.add([S_i, temp])
          else
              NewGen.add([Cur_Gen[j][0], Cur_Gen[j][1]])
          end if
      end for
      Cur_Gen = NewGen
  end for
  Raw_Score, best_set = GetBestSolution(Cur_Gen)
  return Raw_Score, best_set

def Extrapolate(old, pop, cr, f, np)
      a, b, c = threeOthers(pop) # select any 3 items
      newf = emptySet
      for i = 0 to np − 1 do
          if cr ≤ random() then
              newf.add(old[i])
          else
              newf.add(trim(i, (a[i]+f*(b[i] − c[i]))))
          end if
      end for
      return newf
```

Figure 5: Pseudocode for DE with a constant number of iterations

candidate[20]. As the looping progresses, the *Pop* is full of increasingly more valuable solutions which, in turn, also improve the candidates, which are Extrapolated from the Population. LDADE finds the optimal configuration and the $\Re_n$ (Raw) score[28] using *GetBestSolution* function for a particular dataset to be used with LDA for further SE task. One may argue why we have been encoding $\Re_n$ score in the solution. This is shown in pseudocode just to keep track of the values but it is not used to drive our quality/fitness (*ldascore*) function.

Table 7 provides an overview of LDADE algorithm. Also, note that DEs have been applied before for parameter tuning (e.g., see [32, 73, 74]) but this is the first time they have been applied to tune LDA to increase stability.

5. Results

In this section, any result from the smaller data sets (Pits and Citemap) come from Python implementation based on Scikit-Learn running on a 4 GB ram machine (Linux). Also, any results from the larger data (Stackoverflow) comes from a Scala implementation based on Mllib [75] running on a 45 node Spark system (8 cores per node).

Table 7: Overview of Algorithm LDADE

Keywords	Description
Differential weight ($f = 0.7$)	Extent of mutation to be performed on the candidates
Crossover probability ($cr = 0.3$)	Representation of the survival of the candidate in the next generation
Population Size ($np = 10$)	Frontier size in a generation
No. of Generations ($iter = 3$)	How many generations need to be performed
Fitness Function (*ldascore*)	Driving factor of DE
GetBestSolution Function	Returns optimal configuration and it's corresponding $\Re_n$ score
Extrapolate Function	Selects between the current candidate and next candidate
Output	Optimal Configurations to use with LDA for further SE Task

Note that, for the RQ3, there are some intricate details with classification results. After tuning (Goal is still to maximize the $\Re_n$ score) and finding the optimal "k", we trained a Linear Kernel SVM classifier using document topic distributions as features just like used by Blei et al [2].

5.1. *RQ1: Are the default settings of LDA incorrect?*

This research question checks the core premise of this work, that changes in the order of training data dramatically affects the topics learned via LDA. Note that if this is *not true*, then there would be no value added to this paper.

Figure 6 plots n vs $\Re_n$ for untuned LDA. Note that the stability collapses the most after $n = 5$ words. This means that any report of LDA topics that uses more than five words per topic will be changed, just by changing the order of the inputs. This is a significant result since the standard advice in the LDA papers [4, 76] is to report the top 10 words per topic. As shown in Figure 7a, it would be rare that any such 10 word topic would be found across multiple runs.

Result 1
Using the default settings of LDA for these SE data can lead to systematic errors due to topic modeling instability.

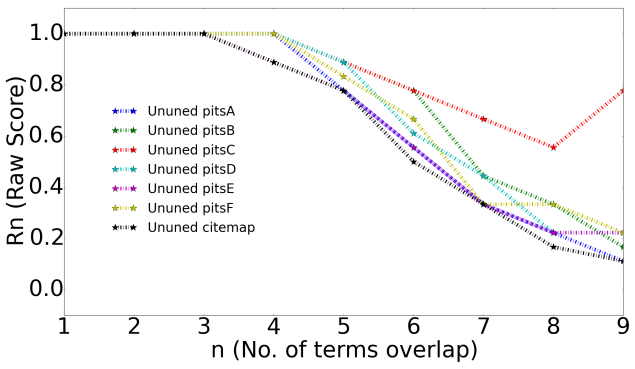

Figure 6: *Before* tuning: uses LDA's default parameters

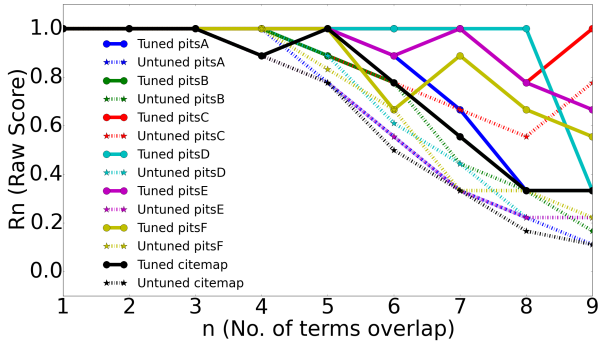

Figure 7a: *After* tuning: uses parameters learned by DE.

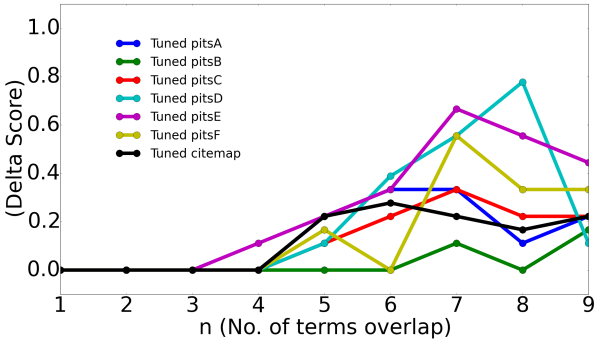

Figure 7b: *Delta = After - Before*.

Figure 7: **RQ1, RQ2** stability results over ten repeated runs. In these figures, *larger* numbers are *better*.

5.2. *RQ2: Does LDADE improve the stability scores?*

Figure 7a and Figure 7b show the stability improvement generated by tuning. Tuning never has any negative effect (reduces stability) and often has a large positive effect, particular after 5 terms overlap. The largest improvement we saw in PitsD dataset which for up to 8 terms overlap was 100% (i.e., was always found in all runs). Overall, after reporting topics of up to 7 words, in the majority case (66%), those topics can be found in models generated using different input orderings.

Continuing on the result of PitsD which achieves the highest improvement. We observed that about 92% of the data in that sample has the severity of level 3. All the other Pits Datasets have mixed samples of severity level. When data is less skewed LDADE achieves the highest improvement which was not achieved when just LDA was being used. So, this makes it highly unimaginable to use just LDA for highly skewed data. This also emphasizes the use of LDADE more.

Accordingly, our answer to **RQ2** is:

Result 2
For stable clusters, tuning is strongly recommended for future LDA studies. α and β matter the most for getting good clusters.

5.3. *RQ3: Does LDADE improve text mining classification accuracy?*

We studied some other StackExchange websites data dump for classification results which were generated by Krishna et al [77]. These datasets are categorized into binary labels indicating which documents are relevant and non relevant. The goal of our DE was still to maximize the $\Re_n$ score. We did not change the goal of our DE based on classification task. After finding the optimal k, α and β, we trained a Linear Kernel SVM classifier using document topic distributions just like used by Blei et al [2]. We used a 5-fold stratified cross validation to remove any sampling bias from the data. 80% was used as training and 20% as testing, and for the tuning phase, out of 80%, 65% was used for training and 15% as validation set.

In Figures 8 and 9, the x-axis represents different datasets as generated. Y-axis represents the F1 score in Figure 8 and in Figure 9, Y-axis represents F2 score (from Equation 1) which weights recall higher than precision [78]. In the figures, "untuned_10" indicates LDA used with default parameters of $k = 10$, "tuned_k" represents tuning of parameters selected from Table 6 with value of k written beside it, and "tuned 10" shows $k = 10$ with α and β tuned by our LDADE method. At the bottom of the figure, we show the variance which is inter-quartile (75th-25th percentile, i.e., IQR) range.

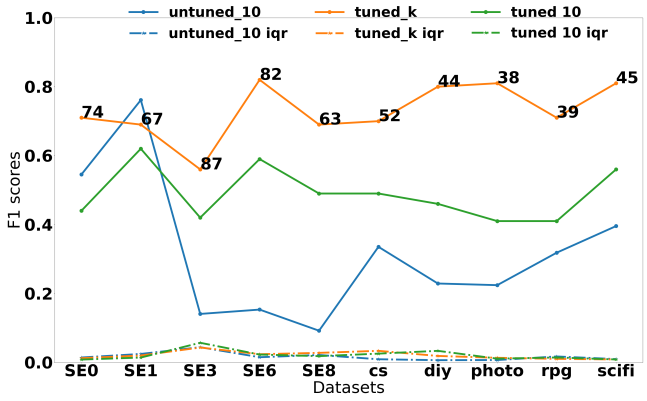

Figure 8: Tuning and Untuned F1 results for Classification SE Task. The numbers in black show the k values learned by LDADE.

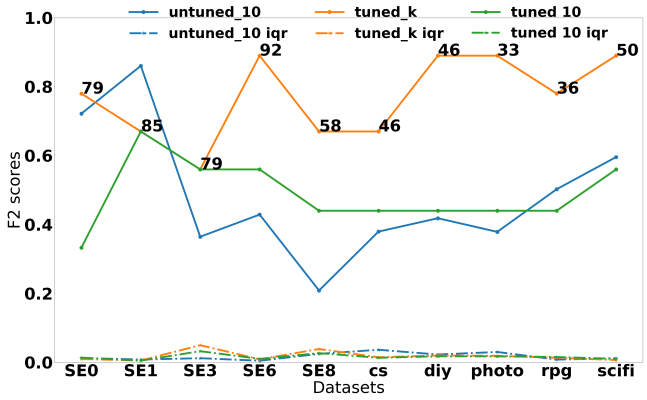

Figure 9: Tuning and Untuned F2 results for Classification SE Task. The numbers in black show the k values learned by LDADE.

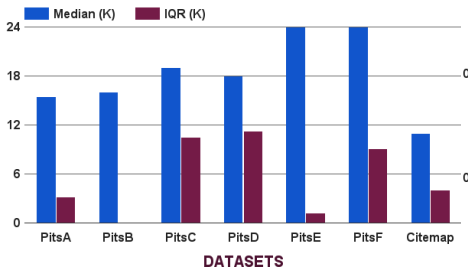

Figure 10: Datasets vs Parameter (k) variation

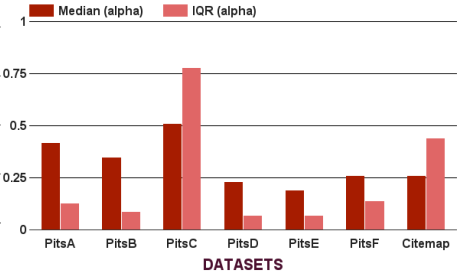

Figure 11 Datasets vs Parameter (α) variation

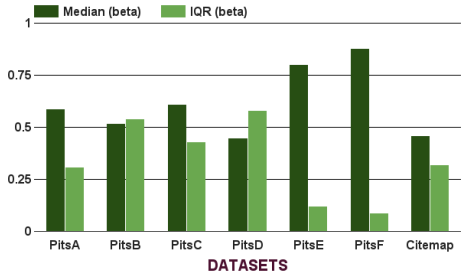

Figure 12 Datasets vs Parameter (β) variation

There is about 100% minimum improvement over the untuned ("untuned_10") with LDADE ("tuned_k"). Even when LDADE is tuned just with α and β, keeping k constant we see minimum improvement of about 25%. These results also validate the study of Blei et al. [2] in which they stated that for a classification task, value of k matters the most. Since the variance seen are so small, that this makes the results highly stable.

Just for completeness, we ran the Scott-Knot [79] procedure for statistical differences using the A12 effect size [80] test and bootstrapping [81] which have a confidence levels of 1%. Based on those tests, we assert that the tuned results are different and better to the untuned results.

Hence, we say:

Result 3
For any SE classification task, tuning is again highly recommended. And k matters the most for a good classification accuracy.

5.4. *RQ4: Do different data sets need different configurations to make LDA stable?*

Figures 10, 11, and 12 show the results of tuning for word overlap of $n = 5$. Median values across 10 tunings and IQR (a non-parametric measure of variation around the median value) are shown in these plots. Note that in Figure 10, IQR=0 for PitsB dataset which shows tuning always converged on the same final value.

These figures show that the IQR ranges are being varied over median by about 50% in most cases of datasets. . Some of the above numbers are far from the standard values, e.g., Garousi et al. [23] recommend using $k = 67$ topics yet in our data sets, best results were seen using $k \leq 24$. These results suggest that how tuning selects the different ranges of parameters. Clearly:

Result 4
Do not reuse tunings suggested by other researchers from other data sets. Instead, always re-tune for all new data.

5.5. *RQ5: Are our findings consistent when using different kinds of LDA or with different implementations?*

To validate this research question, it was insightful to compare our results with: the Pits and Citemap results, executed in Scikit-Learn and Python running on a desktop machine as well as the Stackoverflow data set executed in Scala using Mllib running on a Spark cluster.

Figure 13 shows tuning results for Stackoverflow, Citemap, and PitsA using Scala/Spark cluster (for results on other data sets, see https://goo.gl/UVaql1).

Another useful comparison is to change the internal of the LDA, sometimes using VEM sampling and other times using Gibbs sampling.

Figure 14 compares the VEM vs Gibbs sampling (for results on other datasets, see https://goo.gl/faYAcg). When compared with the Python/desktop results of Figure 7 we see the same patterns that tuning never makes stability worse and sometimes,

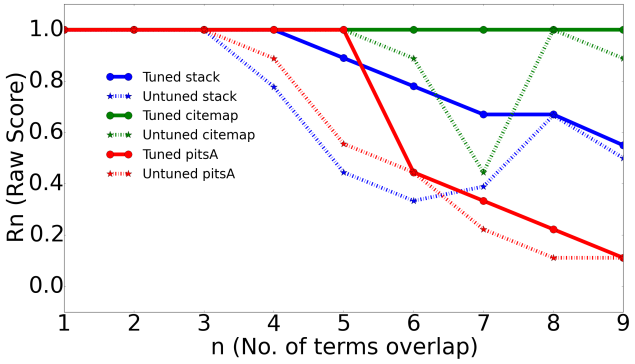

Figure 13: Spark Results

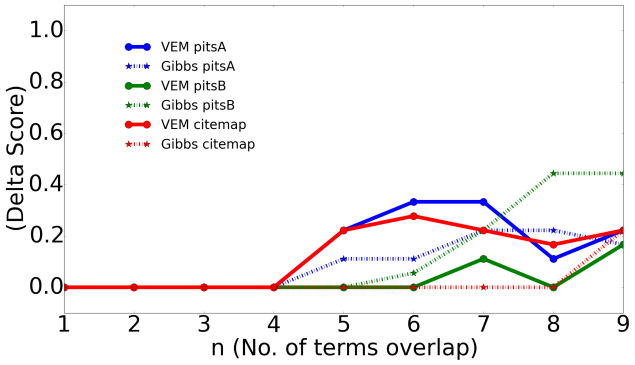

Figure 14: GIBBS vs VEM

tuning dramatically improves it (in particular, see the Citemap results of Figure 13).

That said, there are some deltas between VEM and Gibbs where it seems tuning is more important for VEM than Gibbs (evidence: the improvements seen after tuning are largest for the VEM results of Figure 14 and at https://goo.gl/faYAcg). The reason being, VEM only gets to a local optimum that depends on initialization and other factors in the optimization problem [82]. On the other hand a good sampling algorithm like Gibbs sampling can achieve the global optimum. In practice because of finite number of samples that are used, sampling can give a sub-optimal solution. LDADE works as another kind of sampler which selects good priors at the initialization and can achieve better sub-optimal solutions, which is why LDADE has better benefits on VEM.

Result 5
Instability is not due to any quirk in the implementation of LDA. Instability is consistent and LDADE can stabilize.

5.6. *RQ6: Is tuning easy?*

The DE literature recommends using a population size np that is ten times larger than the number of parameters being optimized [3]. For example, when tuning k, α and β, the DE literature is recommending $np = 30$. Figure 15 explores $np = 30$ vs the $np = 10$ we use in Algorithm 2 (Figure 5) (as well as some other variants of DE's F and CR parameters). The figure shows results just for Citemap and, for space reasons, results relating to other data sets are shown at https://goo.gl/HQNASF. After reviewing the results from all the datasets, we can say that there is not much of an improvement by using different F, CR, and Population size. So our all other experiments used $F = 0.7$, $CR = 0.3$ and $np = 10$. Also:

Result 6
Finding stable parameters for topic models is easier than standard optimization tasks.

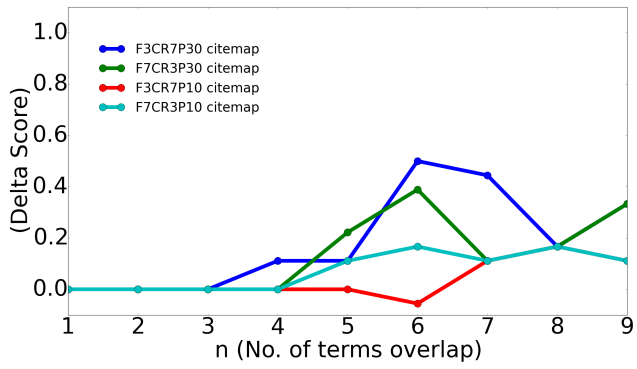

Figure 15: Terms vs Delta Improvement using Different settings of DE

5.7. *RQ7: Is tuning extremely slow?*

Search-based SE methods can be very slow. Wang et al. [83] once needed 15 years of CPU time to find and verify the tunings required for software clone detectors. Sayyad et al. [84] routinely used 10^6 evaluations (or more) of their models in order to extract products from highly constrained product lines. Hence, before recommending any search-based method, it is wise to consider the runtime cost of that recommendation.

To understand our timing results, recall that untuned and LDADE use Algorithm 1 and Algorithm 2 respectively. Based on the psuedocode shown above, our pre-experimental theory is that tuning will be three times slower than not tuning (since DE is taking 3 generations to terminate).

Figures 16 and 17 check if this theory holds true in practice. Shown in blue and red are the runtimes required to run LDA untuned and LDADE (respectively). The longer runtimes (in red) include the times required for DE to find the tunings. Overall, tuning slows down LDA by a factor of up to five (which is very close to our theoretical prediction). Hence, we say:

Result 7
Theoretically and empirically, LDADE costs three to five times more runtime as much as using untuned LDA.

While this is definitely more than not using DE, but this may not be an arduous increase given modern cloud computing environments.

5.8. *RQ8: How better LDADE is compared against LDA-GA?*

Panichella et al. [4] did not consider order effects, which could result in instability of LDA. Another issue could be with their use of Genetic Algorithms to find optimal configurations. Such GAs can be very time consuming to run. Fu et al. [85] argues for faster software analytics, saying that the saved CPU time could be put to better uses.

We ran LDA-GA with their same settings but on our datasets and report the delta of $\Re_n$ of LDADE against $\Re_n$ of LDA-GA. In Figure 18, the positive value shows LDADE performed better than LDA-GA and negative shows LDA-GA performed better. In only about 2 cases (PitsB, PitsC), it is seen that LDA-GA performed better. The reason being, LDA-GA takes a larger number of evaluations which was able to find a better sub-optimal configuration, but it took somewhere between 24-70 times more time than LDADE (See Figure 19). In Figure 19, we are seeing variations in the difference between the runtimes of LDADE and LDA-GA in these datasets. This is due to the settings of GA. LDA-GA has early termination criteria which depends on the quality of dataset. LDA-GA should take somewhere from 1000 to 10,000 evaluations where as our DE would take about 30 evaluations. If a particular dataset is less skewed, it will be terminated early in LDA-GA. These results verify that our initial assumptions hold true, i.e., we can get much faster and more stabler results with LDADE than LDA-GA.

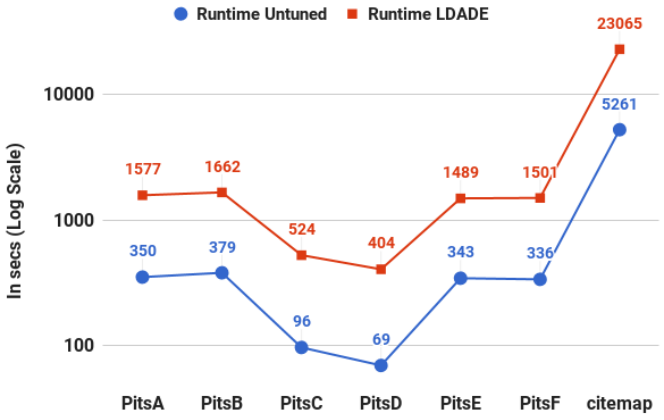

Figure 16 VEM: Datasets vs Runtimes

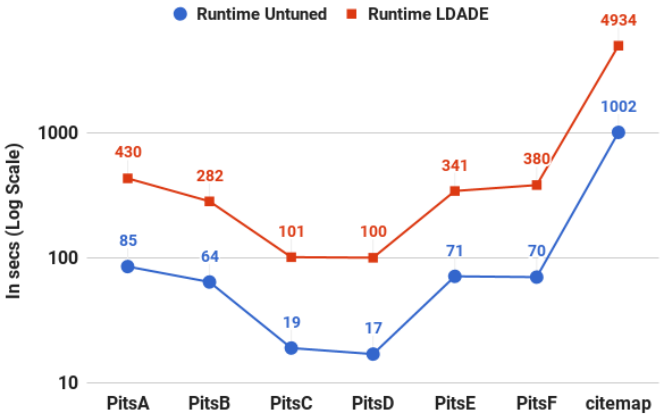

Figure 17: Gibbs: Datasets vs Runtimes

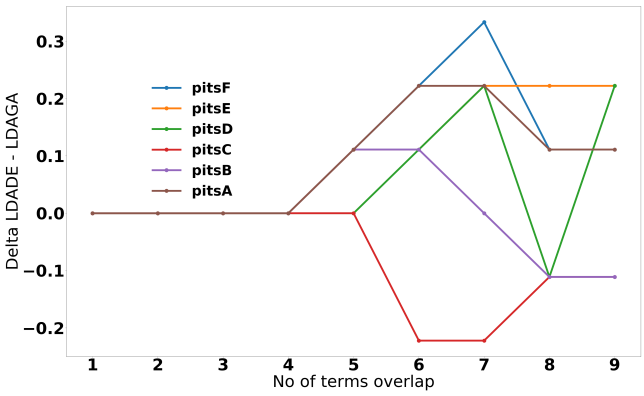

Figure 18: Terms vs Delta comparison of LDADE against LDA-GA

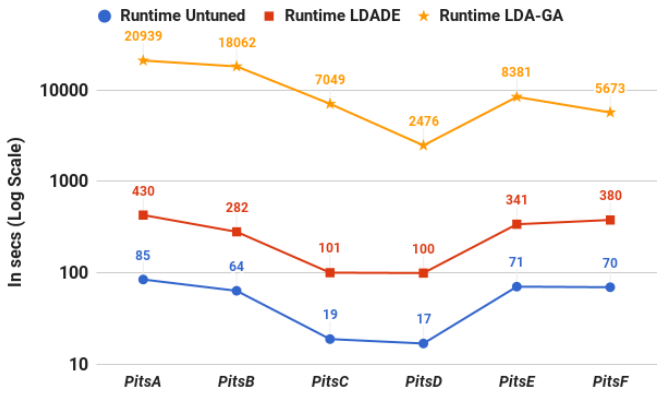

Figure 19: Gibbs: Runtime comparisons between LDADE and LDA-GA

Result 8
LDA-GA is affected by order effects and it is slower than LDADE. Also, LDADE achieves stabler results.

5.9. *RQ9: Should topic modeling be used "off-the-shelf" with their default tunings?*

Figure 7 shows that there is much benefit in tuning. Figures 10, 11, and 12 show that the range of "best" tunings is very dataset specific. Hence, for a new dataset, the off-the-shelf tunings may often fall far from the useful range. Figures 16 and 17 show that tuning is definitely slower than otherwise, but the overall cost is not prohibitive. Hence:

Result 9
Whatever the goal is, whether using the learned topics, or cluster distribution for classification we cannot recommend using "off-the-shelf" LDA.

6. Threats to Validity

As with any empirical study, biases can affect the final results. Therefore, any conclusions made from this work must be considered with the following issues in mind:

Sampling bias threatens any experiment, i.e., what matters there may not be true here. For example, the data sets used here come after various pre-processing steps and could change if pre-processed differently. And that is why, all our datasets can be downloaded from the footnotes of this paper and researchers can explore further. Even though we used so many data sets, there could be other datasets for which our results could be wrong or have lesser improvement.

Learner bias: For running LDA, we selected other parameters as default which are of not much importance. But there could be some datasets where by tuning them there could be much larger improvement. And for RQ2, we only experimented with linear kernel SVM. There could be other classifiers which can change our conclusions. Data Mining is a large and active field and any single study can only use a small subset of the known data miners.

Evaluation bias: This paper uses topic similarity ($\Re_n$) and F2 measures of evaluation but there are other measures which

are used in software engineering which includes perplexity, performance, accuracy, etc. Assessing the performance of stable LDA is a clear direction for future work.

Order bias: With each dataset, how data samples are picked and put into LDA is completely random. Since this paper also consider input order effects, though there could be times when the input order could be with lesser variance. To mitigate this order bias, we ran the experiment 10 times by randomly changing the order of the data samples each time.

Another threat to validity of this work is that it is a quirk of the control parameters used within our DE optimizer. We have some evidence that this is not the case. Figure 15 and other results[4] explored a range of DE tunings and found little difference across that range. Also, Table V explores another choice within DE – how many evaluations to execute before terminating DEs. All the results in this paper use an evaluation budget of 30 evaluations. Table V compares results across different numbers of evaluations. While clearly, the more evaluations the better, there is little improvement after the 30 evaluations used in this paper.

Table 8: Evaluations vs Stability Scores

Datasets\Evaluations	10	20	30	50
PitsA	0.9	0.9	1.0	1.0
PitsB	0.9	0.9	0.9	1.0
PitsC	0.9	1.0	1.0	1.0
PitsD	0.9	1.0	1.0	1.0
PitsE	0.9	0.9	1.0	1.0
PitsF	0.9	0.9	0.9	0.9
Citemap	0.67	0.67	0.77	0.77
Stackoverflow	0.6	0.7	0.8	0.8

The conclusions of this paper are based on a finite number of data sets and it is possible that other data might invalidate our conclusions. As with all analytics papers, any researcher can do is to make their conclusions and materials public, then encourage other researchers to repeat/refute/improve their conclusions.

7. Conclusion

Based on the above, we offer some general recommendations. Any study that shows the topics learned from LDA, and uses them to make a particular conclusion, needs to first tune LDA. We say this since the topics learned from untuned LDA are unstable, i.e., different input orderings will lead to different conclusions. However, after tuning, stability can be greatly increased.

Unlike the advise of Lukins et al. [76], LDA topics should not be reported as the top ten words. Due to order effects, such a report can be highly incorrect. Our results show that up to eight words can be reliably reported, but only after tuning for stability using tools like LDADE.

[4]https://goo.gl/HQNASF

Any other studies which are making use of these topic distributions need to be tuned first before using them in their further tasks. We do not recommend to use someone else's pre-tuned LDA since, as shown in this study, the best LDA tunings vary from data set to data set. These results also ask us to revisit the previous case studies which have used LDA using "off-the-shelf" parameters to rerun by tuning these parameters using automated tools like LDADE. Our experience is that this recommendation is not an arduous demand since tuning adds less than a factor of five to the total run times of an LDA study.

More generally, we comment that the field of software analytics needs to make far more use of search-based software engineering in order to tune their learners. In other work, Fu et al. have shown that tuning significantly helps defect prediction [32] and an improvement also shown for LDA [4]. In this work, we have shown that tuning significantly helps topic modeling by mitigating a systematic error in LDA (order effects that lead to unstable topics). The implication of this study for other software analytics tasks is now an open and pressing issue. In how many domains can search-based SE dramatically improve software analytics?

Acknowledgements

The work is partially funded by NSF award #1506586.

References

[1] A. Nadkarni, N. Yezhkova, Structured versus unstructured data: The balance of power continues to shift, IDC (Industry Development and Models) Mar.

[2] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, in: the Journal of machine Learning research, Vol. 3, JMLR. org, 2003, pp. 993–1022.

[3] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (4) (1997) 341–359.

[4] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, A. De Lucia, How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms, in: Proceedings of the 2013 International Conference on Software Engineering, IEEE Press, 2013, pp. 522–531.

[5] S. Rao, A. Kak, Retrieval from software libraries for bug localization: a comparative study of generic and composite text models, in: Proceedings of the 8th Working Conference on Mining Software Repositories, ACM, 2011, pp. 43–52.

[6] R. Oliveto, M. Gethers, D. Poshyvanyk, 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, IEEE, 2010, pp. 68–71.

[7] A. Barua, S. W. Thomas, A. E. Hassan, What are developers talking about? an analysis of topics and trends in stack overflow, Empirical Software Engineering 19 (3) (2014) 619–654.

[8] L. V. Galvis Carreño, K. Winbladh, Analysis of user comments: an approach for software requirements evolution, in: Proceedings of the 2013 International Conference on Software Engineering, IEEE Press, 2013, pp. 582–591.

[9] A. Hindle, N. A. Ernst, M. W. Godfrey, J. Mylopoulos, Automated topic naming to support cross-project analysis of software maintenance activities, in: Proceedings of the 8th Working Conference on Mining Software Repositories, ACM, 2011, pp. 163–172.

[10] E. Guzman, W. Maalej, How do users like this feature? a fine grained sentiment analysis of app reviews, in: Requirements Engineering Conference (RE), 2014 IEEE 22nd International, IEEE, 2014, pp. 153–162.

[11] S. W. Thomas, Mining software repositories using topic models, in: Proceedings of the 33rd International Conference on Software Engineering, ACM, 2011, pp. 1138–1139.

[12] S. W. Thomas, B. Adams, A. E. Hassan, D. Blostein, Studying software evolution using topic models, Science of Computer Programming 80 (2014) 457–479.

[13] T.-H. Chen, S. W. Thomas, M. Nagappan, A. E. Hassan, Explaining software defects using topic models, in: Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on, IEEE, 2012, pp. 189–198.

[14] S. W. Thomas, H. Hemmati, A. E. Hassan, D. Blostein, Static test case prioritization using topic models, Empirical Software Engineering 19 (1) (2014) 182–212.

[15] S. K. Bajracharya, C. V. Lopes, Mining search topics from a code search engine usage log., in: MSR, Citeseer, 2009, pp. 111–120.

[16] S. Lohar, S. Amornborvornwong, A. Zisman, J. Cleland-Huang, Improving trace accuracy through data-driven configuration and composition of tracing features, in: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ACM, 2013, pp. 378–388.

[17] D. Binkley, D. Heinz, D. Lawrie, J. Overfelt, Understanding lda in source code analysis, in: Proceedings of the 22nd International Conference on Program Comprehension, ACM, 2014, pp. 26–36.

[18] M. Linares-Vásquez, B. Dit, D. Poshyvanyk, An exploratory analysis of mobile development issues using stack overflow, in: Proceedings of the 10th Working Conference on Mining Software Repositories, IEEE Press, 2013, pp. 93–96.

[19] S. Koltcov, O. Koltsova, S. Nikolenko, Latent dirichlet allocation: stability and applications to studies of user-generated content, in: Proceedings of the 2014 ACM conference on Web science, ACM, 2014, pp. 161–165.

[20] S. Grant, J. R. Cordy, D. B. Skillicorn, Using heuristics to estimate an appropriate number of latent topics in source code analysis, Science of Computer Programming 78 (9) (2013) 1663–1678.

[21] A. Hindle, C. Bird, T. Zimmermann, N. Nagappan, Relating requirements to implementation via topic analysis: Do topics extracted from requirements make sense to managers and developers?, in: Software Maintenance (ICSM), 2012 28th IEEE International Conference on, IEEE, 2012, pp. 243–252.

[22] Y. Fu, M. Yan, X. Zhang, L. Xu, D. Yang, J. D. Kymer, Automated classification of software change messages by semi-supervised latent dirichlet allocation, Information and Software Technology 57 (2015) 369–377.

[23] V. Garousi, M. V. Mäntylä, Citations, research topics and active countries in software engineering: A bibliometrics study, Computer Science Review 19 (2016) 56–77.

[24] T.-D. B. Le, F. Thung, D. Lo, Predicting effectiveness of ir-based bug localization techniques, in: 2014 IEEE 25th International Symposium on Software Reliability Engineering, IEEE, 2014, pp. 335–345.

[25] S. I. Nikolenko, S. Koltcov, O. Koltsova, Topic modelling for qualitative studies, Journal of Information Science (2015) 0165551515617393.

[26] X. Sun, B. Li, H. Leung, B. Li, Y. Li, Msr4sm: Using topic models to effectively mining software repositories for software maintenance tasks, Information and Software Technology 66 (2015) 1–12.

[27] T.-H. Chen, W. Shang, M. Nagappan, A. E. Hassan, S. W. Thomas, Topic-based software defect explanation, Journal of Systems and Software.

[28] X. Sun, X. Liu, B. Li, Y. Duan, H. Yang, J. Hu, Exploring topic models in software engineering data analysis: A survey, in: 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), IEEE, 2016, pp. 357–362.

[29] J. H. Gennari, P. Langley, D. Fisher, Models of incremental concept formation, Artificial intelligence 40 (1-3) (1989) 11–61.

[30] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[31] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research 13 (Feb) (2012) 281–305.

[32] W. Fu, T. Menzies, X. Shen, Tuning for software analytics: Is it really necessary?, Information and Software Technology 76 (2016) 135–146.

[33] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, K. Matsumoto, Automated Parameter Optimization of Classification techniques for Defect Prediction Models, in: Proc. of the International Conference on Software Engineering (ICSE), 2016, pp. 321–332.

[34] K. O. Elish, M. O. Elish, Predicting defect-prone software modules using support vector machines, Journal of Systems and Software 81 (5) (2008) 649–660.

[35] S. Lessmann, B. Baesens, C. Mues, S. Pietsch, Benchmarking classification models for software defect prediction: A proposed framework and novel findings, IEEE Transactions on Software Engineering 34 (4) (2008) 485–496.

[36] Y. Yang, Y. Zhou, J. Liu, Y. Zhao, H. Lu, L. Xu, B. Xu, H. Leung, Effort-aware just-in-time defect prediction: simple unsupervised models could be better than supervised models, in: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM, 2016, pp. 157–168.

[37] G. Maskeri, S. Sarkar, K. Heafield, Mining business topics in source code using latent dirichlet allocation, in: Proceedings of the 1st India software engineering conference, ACM, 2008, pp. 113–120.

[38] W. Martin, M. Harman, Y. Jia, F. Sarro, Y. Zhang, The app sampling problem for app store mining, in: Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on, IEEE, 2015, pp. 123–133.

[39] J. H. Lau, D. Newman, T. Baldwin, Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality., in: EACL, 2014, pp. 530–539.

[40] Y. Yang, Improving the usability of topic models, Ph.D. thesis, NORTHWESTERN UNIVERSITY (2015).

[41] K. Tian, M. Revelle, D. Poshyvanyk, Using latent dirichlet allocation for automatic categorization of software, in: Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on, IEEE, 2009, pp. 163–166.

[42] T. Menzies, Improving iv&v techniques through the analysis of project anomalies: Text mining pits issue reports-final report, Citeseer.

[43] T. Menzies, A. Marcus, Automated severity assessment of software defect reports, in: Software Maintenance, 2008. ICSM 2008. IEEE International Conference on, IEEE, 2008, pp. 346–355.

[44] T. Menzies, R. Krishna, D. Pryor, The Promise Repository of Empirical Software Engineering Data, `http://openscience.us/repo` (2015).

[45] M. Allamanis, C. Sutton, Why, when, and what: analyzing stack overflow questions by topic, type, and code, in: Proceedings of the 10th Working Conference on Mining Software Repositories, IEEE Press, 2013, pp. 53–56.

[46] C. Rosen, E. Shihab, What are mobile developers asking about? a large scale study using stack overflow, Empirical Software Engineering 21 (3) (2016) 1192–1223.

[47] B. Vasilescu, A. Serebrenik, T. Mens, A historical dataset of software engineering conferences, in: Proceedings of the 10th Working Conference on Mining Software Repositories, IEEE Press, 2013, pp. 373–376.

[48] G. Mathew, A. Agrawal, T. Menzies, Trends in topics at se conferences (1993–2013), in: Proceedings of the 39th International Conference on Software Engineering Companion, IEEE Press, 2017, pp. 397–398.

[49] B. Vasilescu, A. Serebrenik, T. Mens, M. G. van den Brand, E. Pek, How healthy are software engineering conferences?, Science of Computer Programming 89 (2014) 251–272.

[50] R.-S. Feldman, J, The Text Mining Handbook, New York: Cambridge University Press, 2006.

[51] S. Bird, Nltk: the natural language toolkit, in: Proceedings of the COLING/ACL on Interactive presentation sessions, Association for Computational Linguistics, 2006, pp. 69–72.

[52] M. Porter, The Porter Stemming Algorithm (1980).
URL `http://tartarus.org/martin/PorterStemmer/`

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, Journal of Machine Learning Research 12 (Oct) (2011) 2825–2830.

[54] D. OCallaghan, D. Greene, J. Carthy, P. Cunningham, An analysis of the coherence of descriptors in topic modeling, Expert Systems with Applications 42 (13) (2015) 5645–5657.

[55] W. Zhao, J. J. Chen, R. Perkins, Z. Liu, W. Ge, Y. Ding, W. Zou, A heuristic approach to determine an appropriate number of topics in topic modeling, BMC bioinformatics 16 (13) (2015) 1.

[56] G. A. Miller, The magical number seven, plus or minus two: Some limits on our capacity for processing information., Psychological review 63 (2) (1956) 81.

[57] M. S. Feather, T. Menzies, Converging on the optimal attainment of requirements, in: Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on, IEEE, 2002, pp. 263–270.

[58] T. Menzies, O. Elrawas, J. Hihn, M. Feather, R. Madachy, B. Boehm, The business case for automated software engineering, in: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, ACM, 2007, pp. 303–312.
[59] A. T. Goldberg, On the complexity of the satisfiability problem, Ph.D. thesis, New York University (1979).
[60] F. Glover, C. McMillan, The general employee scheduling problem. an integration of ms and ai, Computers & operations research 13 (5) (1986) 563–573.
[61] R. P. Beausoleil, MOSS multiobjective scatter search applied to non-linear multiple criteria optimization, European Journal of Operational Research 169 (2) (2006) 426–449.
[62] J. Molina, M. Laguna, R. Martí, R. Caballero, Sspmo: A scatter tabu search procedure for non-linear multiobjective optimization, INFORMS Journal on Computing 19 (1) (2007) 91–100.
[63] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, A. Beham, Abyss: Adapting scatter search to multiobjective optimization, Evolutionary Computation, IEEE Transactions on 12 (4) (2008) 439–457.
[64] H. Pan, M. Zheng, X. Han, Particle swarm-simulated annealing fusion algorithm and its application in function optimization, in: Computer Science and Software Engineering, 2008 International Conference on, Vol. 1, IEEE, 2008, pp. 78–81.
[65] J. Krall, T. Menzies, M. Davies, Gale: Geometric active learning for search-based software engineering, Software Engineering, IEEE Transactions on 41 (10) (2015) 1001–1018.
[66] M. Zuluaga, G. Sergent, A. Krause, M. Püschel, Active learning for multi-objective optimization, in: Proceedings of the 30th International Conference on Machine Learning, 2013, pp. 462–470.
[67] J. Vesterstrøm, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, in: Evolutionary Computation, 2004. CEC2004. Congress on, Vol. 2, IEEE, 2004, pp. 1980–1987.
[68] T. Minka, J. Lafferty, Expectation-propagation for the generative aspect model, in: Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 2002, pp. 352–359.
[69] X. Wei, W. B. Croft, Lda-based document models for ad-hoc retrieval, in: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2006, pp. 178–185.
[70] T. L. Griffiths, M. Steyvers, Finding scientific topics, Proceedings of the National academy of Sciences 101 (suppl 1) (2004) 5228–5235.
[71] L. Layman, Personnel communication (2016).
[72] L. Layman, A. P. Nikora, J. Meek, T. Menzies, Topic modeling of nasa space system problem reports: research in practice, in: Proceedings of the 13th International Workshop on Mining Software Repositories, ACM, 2016, pp. 303–314.
[73] M. G. Omran, A. P. Engelbrecht, A. Salman, Differential evolution methods for unsupervised image classification, in: Evolutionary Computation, 2005. The 2005 IEEE Congress on, Vol. 2, IEEE, 2005, pp. 966–973.
[74] I. Chiha, J. Ghabi, N. Liouane, Tuning pid controller with multi-objective differential evolution, in: Communications Control and Signal Processing (ISCCSP), 2012 5th International Symposium on, IEEE, 2012, pp. 1–4.
[75] X. Meng, J. Bradley, B. Yuvaz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al., Mllib: Machine learning in apache spark, JMLR 17 (34) (2016) 1–7.
[76] S. K. Lukins, N. A. Kraft, L. H. Etzkorn, Bug localization using latent dirichlet allocation, Information and Software Technology 52 (9) (2010) 972–990.
[77] R. Krishna, Z. Yu, A. Agrawal, M. Dominguez, D. Wolf, The bigse project: lessons learned from validating industrial text mining, in: Proceedings of the 2nd International Workshop on BIG Data Software Engineering, ACM, 2016, pp. 65–71.
[78] D. M. Powers, Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.
[79] E. G. Jelihovschi, J. C. Faria, I. B. Allaman, Scottknott: a package for performing the scott-knott clustering algorithm in r, TEMA (São Carlos) 15 (1) (2014) 3–17.
[80] A. Vargha, H. D. Delaney, A critique and improvement of the cl common language effect size statistics of mcgraw and wong, Journal of Educational and Behavioral Statistics 25 (2) (2000) 101–132.
[81] B. Efron, R. J. Tibshirani, An introduction to the bootstrap, CRC press, 1994.
[82] A. Asuncion, M. Welling, P. Smyth, Y. W. Teh, On smoothing and inference for topic models, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2009, pp. 27–34.
[83] T. Wang, M. Harman, Y. Jia, J. Krinke, Searching for better configurations: a rigorous approach to clone evaluation, in: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ACM, 2013, pp. 455–465.
[84] A. S. Sayyad, J. Ingram, T. Menzies, H. Ammar, Scalable product line configuration: A straw to break the camel's back, in: Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on, IEEE, 2013, pp. 465–474.
[85] W. Fu, T. Menzies, Easy over hard: A case study on deep learning, arXiv preprint arXiv:1703.00133.
[86] M. Harman, The current state and future of search based software engineering, in: 2007 Future of Software Engineering, IEEE Computer Society, 2007, pp. 342–357.