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

Amritanshu Agrawal[a,*], Wei Fu[a,*], Tim Menzies[a,*]

[a] *Department of Computer Science, North Carolina State University, Raleigh, NC, USA*

Abstract

Context: Topic Modeling finds human-readable structures in large sets of unstructured data. A widely used topic modeler is Latent Dirichlet Allocation. When run on different datasets, LDA suffers from "order effects" i.e. different distributions are generated if the training data was shuffled into a different order. Such order effects introduce a systematic error for any study.
Objective: To provide a method in which distributions generated by LDA are more stable and can be used for further analysis.
Method: We introduce LDADE, a Search-Based Software Engineering tool that tunes LDA's parameters using DE (Differential Evolution).
Evaluation: 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 different platforms (Linux, Macintosh) and for different kinds of LDAs (the traditional VEM method, or using Gibbs sampling).
Results: LDA suffers from order effects and our LDADE was able to stabilize the distribution. A case study against the Barua et al. [1] was performed and also found that there has been systematic error by directly using default settings of LDA.
Conclusion In all tests, the pattern was the same: LDADE's tunings dramatically reduces cluster instability. The implications 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?

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) [2]. Rather it is *unstructured* data, often in free text, and found in word processing files, slide presentations, comments, etc 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) [3]. Topic modeling is widely used in Software Engineering and many papers in recent years have reported topic modeling results at numerous SE venues (see Table 1 and 2). These two tables show how important LDA has been in various SE tasks and most of them are recently published.

Topic models built by LDA are non-deterministic in behaviour. Due to non-determinism, topics generated by LDA are not stable [4, 1]. It learns the various distributions which is a problem of Bayesian inference [3]. We are talking about model instability itself which further affects the results. Many industries, working on a product development in text analytics have been widely using it. They want better results in their product. Menzies et al. [5] showed how model instability can give rise to large effort estimations across various projects. And we do need to avoid these large effort estimations.

LDA takes certain set of parameters which try to take us

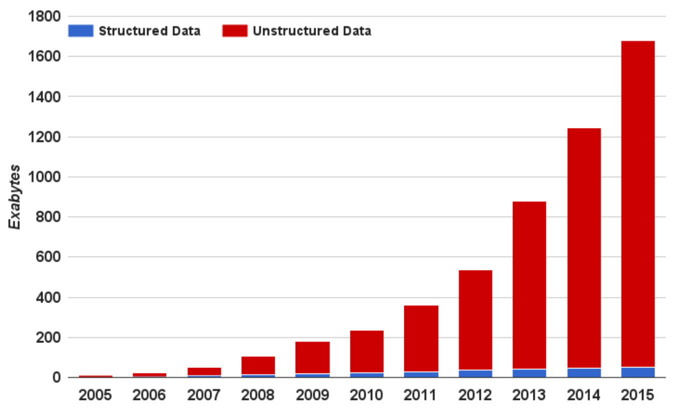


Figure 1: Data Growth 2005-2015. From [2].

*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: http://ai4se.net/ (Amritanshu Agrawal)

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 1: SE Venues that publish on Topic Modeling.

closer to good cluster distributions. But it is an art to find the right parameters setting that control the choices within a data miner. But it is very impractical to get the right parameters settings due to larger search space of configurations. Nevertheless, we rarely tuned LDA since we reasoned that a LDAs default tunings have been well-explored by the developers of those algorithms (in which case tuning would not lead to stable topic models). Due to larger search space of configurations, it is not even feasible to find such tunings. And that is why, LDA is usually used with off-the-shell default parameters. Many researchers have agreed to the idea of using different configurations in LDA. But, we have seen rare work done when it comes to tune the important parameters of LDA.

Researchers use topic models mostly in either of two ways. Firstly, topics may be used as *feature selector* that finds useful inputs which are then used by, say, a classifier to characterize different kinds of software (e.g. buggy or not [6]). In this mode, no human ever need to read the generated topics but the cluster distribution still suffers from instabilities and they pose problem in classification tasks.

Secondly, researchers may present and reflect on the generated topics in order to offer some insight into the structure of the data. In this second mode, researchers present and discuss the specifics of the generated topics in order to defend particular conclusions.
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 any of the previous study is affected due to instability in LDA?** We will show that the case study against the Barua et al. [1] was performed and also found that there has been systematic error by directly using default settings of LDA. We can say that all the previous studies done using the "off-the-shelf" default parameters need to be re-run to get rid of the systematic error.

- **RQ3**: **Does LDADE improve the stability scores?** In our work, we will show dramatic improvement in the stability scores with the parameters found automatically by DE. And we also found that the most important parameters which affect cluster stability are α and β.

- **RQ4**: **Does LDADE improve the classification F2 accuracy scores?** We will show improvement in the F2 scores with the parameters found automatically by DE. We found that for classification, k, number of topics matters the most.

- **RQ5**: **Do different data sets need different configurations to make LDA stable?** We will show that DE finds different "best" parameter settings for different data sets. Hence reusing tunings suggested by other researchers from other data sets are a bad idea. Instead, use automatic tuning methods to find the best tuning parameters for the current data set.

- **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**: **What is the runtime cost of tuning?** 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**: **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.

Before proceeding, we offer three caveats. Firstly, having mitigated for unstable topics in LDA, the reader may be asking "ok, but what useful conclusions can you draw from this new kind of stabler LDA?". In a companion paper submitted to IST journal [28] we apply LDADE to 15121 SE papers to find trends in topics between 1992 to 2016. That paper reports, what topics are becoming more/less prominent; what topics are most frequently accepted to what conferences; which conferences mostly overlap with other conferences; and what topics most distinguish different conferences.

Secondly, as witnessed by the central columns of Table 2, many prior papers have commented that the results of a topic modeling analysis can be affected by tuning the control parameters of LDA. Yet a repeated pattern in the literature is that, despite these stated concerns, researchers rarely take the next step to find ways to better control tunings. To the best of our knowledge, apart from this paper, there are no reports in the SE literature of simple reproducible automatic methods that can resolve the problem of unstable topics.

Thirdly, while the specifics of this paper are about taming instability in topic modeling, we also think that this work makes a more general point. This is the second time, DE has been

Reference ID	Year	Citations	Venues	Mentions instability in LDA?	Talks about using-off-the shell parameters?	Does tuning?	Conclusion	Tasks / Use cases
[7]	2011	112	WCRE	Y	Y	N	Explored Configurations without any explanation.	Bug Localisation
[4]	2010	108	MSR	Y	Y	N	Explored Configurations without any explanation. Reported their results using multiple experiments.	Traceability Link recovery
[1]	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
[8]	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
[9]	2013	61	ICSE	Y	Y	N	Explored Configurations without any explanation.	Software Requirements Analysis
[10]	2011	45	MSR	Y	Y	N	They validated the topic labelling techniques using multiple experiments.	Software Artifacts Analysis
[11]	2014	44	RE	Y	Y	N	Explored Configurations without any explanation.	Requirements Engineering
[12]	2011	44	ICSE	Y	Y	N	Open issue to choose optimal parameters.	A review on LDA
[13]	2014	35	SCP	Y	Y	N	Explored Configurations without any explanation.	Software Artifacts Analysis
[14]	2012	35	MSR	Y	Y	N	Choosing the optimal number of topics is difficult.	Software Defects Prediction
[15]	2014	31	ESE	Y	Y	N	Choosing right set of parameters is a difficult task.	Software Testing
[16]	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
[17]	2013	27	ESEC/FSE	Y	Y	Y	Explored Configurations using LDA-GA.	Traceability Link recovery
[18]	2014	20	ICPC	Y	Y	N	Use heuristics to find right set of parameters.	Source Code Comprehension
[19]	2013	20	MSR	Y	Y	N	In Future, they planned to use LDA-GA.	StackOverflow Q&A data analysis
[20]	2014	15	WebSci	Y	Y	N	Explored Configurations without any explanation.	Social Software Engineering
[21]	2013	13	SCP	Y	Y	N	Their work focused on optimizing LDAs topic count parameter.	Source Code Comprehension
[22]	2012	13	ICSM	Y	Y	N	Explored Configurations without any explanation (Just with no. of topics).	Software Requirements Analysis
[23]	2015	6	IST	Y	Y	N	Explored Configurations without any explanation. Choosing right set of parameters is a difficult task.	Software re-factoring
[24]	2016	5	CS Review	Y	Y	N	Explored Configurations without any explanation.	Bibliometrics and citations analysis
[25]	2014	5	ISSRE	N	Y	N	Explored Configurations without any explanation.	Bug Localisation
[26]	2015	3	JIS	Y	Y	N	They improvised LDA into ISLDA which gave stability across different runs.	Social Software Engineering
[27]	2015	2	IST	Y	Y	Y	Explored Configurations using LDA-GA.	Software Artifacts Analysis
[6]	2016	0	JSS	N	Y	N	Explored Configurations without any explanation. Choosing right set of parameters is a difficult task.	Software Defects Prediction

Table 2: A sample of the recent literature on using topic modeling in SE. Note that some of these papers are widely-cited.

applied to automatically adjust the control parameters of an algorithm used in software analytics. Previously, Fu et al. [29] showed that tuning software defect predictors can lead to large improvements of the performance of those predictors and that those tunings are different in different data sets. So, it is important to apply automatic tuning as part of the analysis of any new data set. Note that those old conclusions are the same as for this paper. These two results suggests that it is time to revisit past results to check:

- In the past, just how poorly have we been using software analytics in the past?
- And how much better can we improve future results using methods taken from search-based software engineering?

2. Related Work

2.1. Tuning: Important and Ignored

The impact of tuning are well understood in the theoretical machine learning literature [30]. 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. [29] 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: [31]). For example, Elish et al. [32] 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. Yet Fu et al. showed that (a) finding useful tunings for software defect prediction is remarkably easy using DE; (b) different data sets require different tunings; hence, for software defect prediction, (c) tuning is essential for all new data sets.

Accordingly, we are now engaged in an on-going research project that explores tuning for software analytics.

- Find some data mining widely used in SE;
- Check the conclusions of that data miner under different parameter tunings;
- See if those different tunings significantly change the results of that learner;
- Look for ways to better manage the exploration of those tunings.

This paper explores these four steps in the context of topic modeling.

2.2. Topic Modeling

Latent Dirichlet Allocation (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 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.

Figure 2 illustrates topic generation from StackOverflow. To find these words, LDA explores two probability distributions:

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

Figure 2: Example (from [1]) of generating topics from StackOverflow. For each topic, we show just the five most heavily weighted words.

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

Initially, α, 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, β.

Binkley et al. [18] performed an extensive study and found that apart from α, β, the other parameters that define LDA are:

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

But finally Binkley et al., with all the many researchers found the only important parameters that matters the most are k, α, β. Even our results found the same.

2.3. About Order Effects

This paper uses tuning to fix "order effects" in topic modeling. Langley [33] 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 [33]. Hence, some algorithms survey the space of possible models across numerous random divisions of the data (e.g. Random Forests [34]).

From the description offered above in §2.2, we can see (a) how topic modeling might be susceptible to order effects and (b) how such order effects might be tamed:

- In the above description, k, α, β are initialized at random then updated via an incremental re-sampling process. Such incremental updates are prone to order effects.
- One technique to reduce the effect of different data orderings is to initialize, k, α, β to some large value. The trick when applying this technique is that different data sets will requiring different initializations; i.e. the tuning process will have to be repeated for each new data set.

2.4. WorkArounds for LDA Instability Problem

In order to understand how other researchers have explored LDA instability, in April 2016, we searched scholar.google.com 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:

- 29 mention instability in LDA.
- Of those 29, despite mentioning stability problems, 10 papers used LDA's "off-the-shelf" parameters;
- The other 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 [35, 36, 11] manually accessed the topics and then used for further experiments. Some made use of Amazon Mechanical Turk to create gold-standard coherence judgements [37]. All these solutions are related to results stability rather than model stability. Note that this workaround takes extensive manual effort and time.

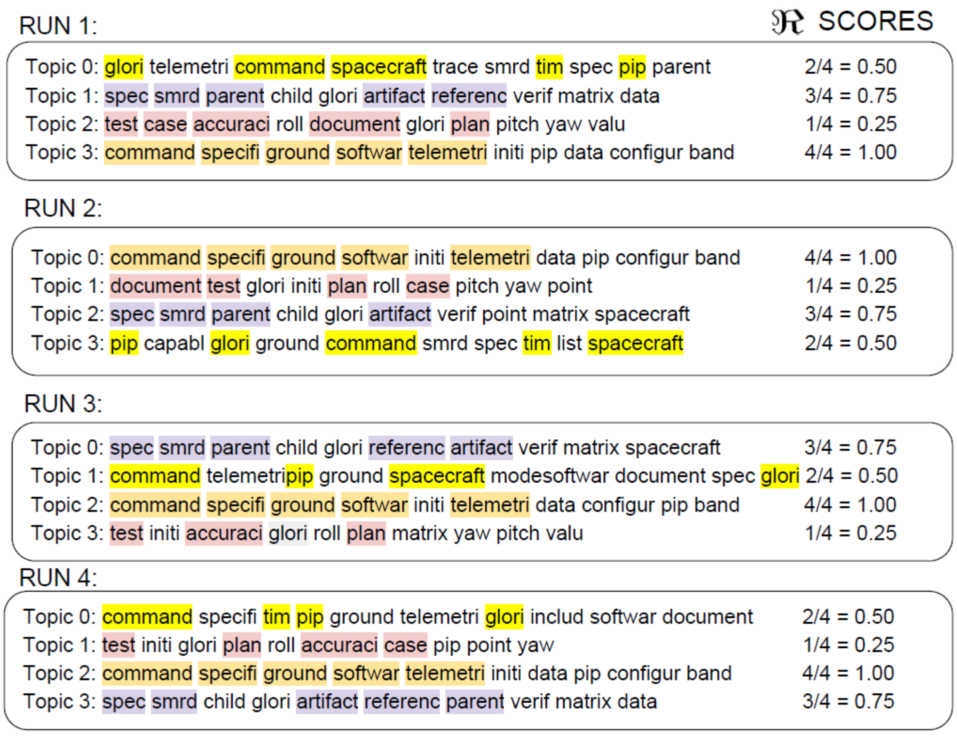
Another approach to tame LDA instability is to incorporate user knowledge into the corpus. For example, SC-LDA [38] can handle different kinds of knowledge such as word correlation, document correlation, document label and so on. Using such user knowledge, while certainly valuable, is somewhat subjective. Hence, for reasons of reproducibility, we prefer fully automated methods.

Some researchers [8, 17, 27] used Genetic Algorithm (GA) to tune the parameters– but for different purposes than LDADE. Panichella et al. [8] used GAs to explore parameters. This method is quite time consuming than using a simple multiobjective optimizer like differential evolution [39].

Finally, other researchers explore some limited manual parameter tuning for LDA (e.g. experiment with one parameter: cluster size) [9, 40] achieved 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.

3. Methods

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



RUN 1: ℜ SCORES

Topic 0: glori telemetri command spacecraft trace smrd tim spec pip parent 2/4 = 0.50
Topic 1: spec smrd parent child glori artifact referenc verif matrix data 3/4 = 0.75
Topic 2: test case accuraci roll document glori plan pitch yaw valu 1/4 = 0.25
Topic 3: command specifi ground softwar telemetri initi pip data configur band 4/4 = 1.00

RUN 2:

Topic 0: command specifi ground softwar initi telemetri data pip configur band 4/4 = 1.00
Topic 1: document test glori initi plan roll case pitch yaw point 1/4 = 0.25
Topic 2: spec smrd parent child glori artifact verif point matrix spacecraft 3/4 = 0.75
Topic 3: pip capabl glori ground command smrd spec tim list spacecraft 2/4 = 0.50

RUN 3:

Topic 0: spec smrd parent child glori referenc artifact verif matrix spacecraft 3/4 = 0.75
Topic 1: command telemetripip ground spacecraft modesoftwar document spec glori 2/4 = 0.50
Topic 2: command specifi ground softwar initi telemetri data configur pip band 4/4 = 1.00
Topic 3: test initi accuraci glori roll plan matrix yaw pitch valu 1/4 = 0.25

RUN 4:

Topic 0: command specifi tim pip ground telemetri glori includ softwar document 2/4 = 0.50
Topic 1: test initi glori plan roll accuraci case pip point yaw 1/4 = 0.25
Topic 2: command specifi ground softwar telemetri initi data pip configur band 4/4 = 1.00
Topic 3: spec smrd child glori artifact referenc parent verif matrix data 3/4 = 0.75

Figure 3: Example of topics overlap off size $n = 5$ across multiple runs.

	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

Table 3: Statistics on our datasets. PitsA, PitsB, etc refer to the issues from six different NASA projects.

3.1. Data Sets

To answer our research questions, and to enable reproducibility of our results, we use three open source datasets summarized in Table 3 and described below.

PITS is a text mining data set generated from NASA software project and issue tracking system (PITS) reports [41, 42]. 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 [43]. Note that, this data comes from six different NASA projects, which we label as PitsA, PitsB, etc.

StackOverflow is the flagship site of the Stack Exchange Network which features questions and answers on a wide range of topics in computer programming. 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. This data was obtained in the form of an SQL dump from the work of Vasilescu et al. [44]. This dataset is available on-line[2].

For this study, all datasets were preprocessed using the usual text mining filters [45]:

- Stop words removal using NLTK toolkit[3] [46] : ignore very common short words such as "and" or "the".
- Porter's stemming filter [47]: delete uninformative word endings; e.g. after 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

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

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 3 shows the sizes of our data before and after pre-processing. These datasets are of different sizes and so are processed using different tools:

- PITS and Citemap is small enough to process on a single (four core) desktop machine using Scikit-Learn [48] and Python.
- StackOverflow is so large (7GB) that its processing requires extra hardware support. This study used a Spark and Mllib on a cluster of 45 nodes to reduce the runtime.

3.2. 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)* [37, 49] 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 [20]. Researchers caution that the value of perplexity doesn't remain constant with different topic size and with dictionary sizes [20, 50]. A lot depend on the code implementation of perplexity 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.

Another approach researchers have used is *Jaccard Similarity* [49, 9]. We define our measure similar to it. For this work, we assess topic model stability via the *median number overlaps of size n words* (*size_of_topic*), which we denote $\Re_n$.

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 [51]).

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 builds topic models using the data;
- m runs of our rig executes m copies of one run, each time using a different random number seed,
- We say topics are 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 3. These are generated via four *runs* of our system. In this hypothetical example, we will assume that the runs of Figure 3 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 4 shows the $\Re_n$ scores of Figure 3 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.

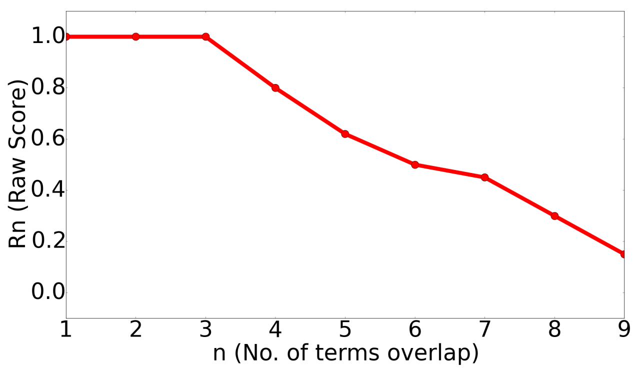


Figure 4: $\Re_n$ scores of Figure 3 for $1 \leq n \leq 9$

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).

The pseudocode for these calculations is shown in 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 run[5]. Data is in the form of term frequency 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. Topics[6] are a list of lists which contains topics from all the different runs. A stability score is evaluated on every 10 runs (Fixed), and this process is continued 10 (Fixed) times. At the end, the median score is selected as the untuned raw score ($\Re_n$)$^{3-11}$. Hence, the runtimes comes from $10 * 10$ evaluations of untuned experiment.

Algorithm 1 Pseudocode for untuned LDA with Default Parameters

```
Input: Data, n, k, α, β (Defaults)
Output: Raw_Score
 1: function LDASCORE( n, Data)
 2:     Score ← ∅
 3:     for j = 0 to 10 do
 4:         for i = 0 to 10 do
 5:             data ← shuffle(Data)
 6:             Topics.add(lda(k, α, β ))
 7:         end for
 8:         Score.add(Overlap(Topics, n, l[0]))
 9:     end for
10:     Raw_Score ← median(Score)
11:     return Raw_Score
12: end function
```

3.3. 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 [52, 53]; various genetic algorithms [54] augmented by techniques such as DE (differential evolution [39]), tabu search and scatter search [55, 56, 57, 58]; particle swarm optimization [59]; numerous decomposition approaches that use heuristics to decompose the total space into small problems, then apply a response surface methods [60, 61]. Of these, we use DE for two reasons. Firstly, it has proved useful in prior SE tuning studies [62]. Secondly, our reading of the current literature is that there are many advocates for differential evolution.

LDADE adjusts the parameters of Table 4. Most of these parameters were explained above. Apart from them, there are 2 different kinds of LDA implementations and they are:

- VEM is the deterministic *variational EM* method that computes α, β via expectation maximization [63].
- Gibbs sampling [62, 64] is a Markov Chain Monte Carlo algorithm, which is an approximate stochastic process for computing and updating α, β. Topic modeling researchers in SE have argued that Gibbs leads to stabler models [65, 66] (a claim which we test, below).

We manually adjust these inference techniques according to different implementations across different platforms.

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

Table 4: List of parameters tuned by this paper

Algorithm 2 shows LDADE's version of DE. DE evolves a *NewGeneration* of candidates from a current Population. Each candidate solution in the Population is a pair of (Tunings, Scores). Tunings are selected from Table 4 and Scores come similarly from Algorithm 1[3−11]. Note that there is not any outer loop[3]: in Algorithm 2, LDA is only run as 1 rig[11&12]. Here, the runtimes comes from $iter * np * 10$ evaluations of tuned experiment.

Algorithm 2 Pseudocode for DE with a constant number of iterations

```
Input: np= 10, f= 0.7, cr= 0.3, iter= 3, Goal ∈ Finding maximum score
Output: Raw_Score, final_generation
 1: function DE(np, f, cr, iter, Goal)
 2:     Cur_Gen ← ∅
 3:     Population ←InitializePopulation(np)
 4:     for i = 0 to np − 1 do
 5:         Cur_Gen.add(Population[i],ldascore(Population[i],n,Data)
 6:     end for
 7:     for i = 0 to iter do
 8:         NewGeneration ← ∅
 9:         for j = 0 to np − 1 do
10:             S_i ←Extrapolate(Population[j],Population,cr,f,np)
11:             if ldascore(S_i) ≥ Cur_Gen[j][1] then
12:                 NewGeneration.add(S_i,ldascore(S_i,n, Data))
13:             else
14:                 NewGeneration.add(Cur_Gen[j])
15:             end if
16:         end for
17:         Cur_Gen ← NewGeneration
18:     end for
19:     Raw_Score ← GetBestSolution(Cur_Gen)
20:     final_generation ← Cur_Gen
21:     return Raw_Score, final_generation
22: end function
23: function EXTRAPOLATE(old, pop, cr, f, np)
24:     a, b, c ← threeOthers(pop, old)
25:     newf ← ∅
26:     for i = 0 to np − 1 do
27:         if cr ≤ random() then
28:             newf.add(old[i])
29:         else
30:             if typeof(old[i])== bool then then
31:                 newf.add(not old[i])
32:             else
33:                 newf.add(trim(i,(a[i]+f∗(b[i] − c[i]))))
34:             end if
35:         end if
36:     end for
37:     return newf
38: end function
```

The main loop of DE[9] runs over the *Population*, replacing old items with new Candidates (if new candidate is better). DE generates *new Candidates* via extrapolating[23] between current solutions in the frontier. Three solutions a, b, c are selected at random. For each tuning parameter i, at some probability cr, we replace the old tuning x_i with y_i. For booleans, we use $y_i = x_i$ (see line 31). For numerics, $y_i = a_i + f \times (b_i - c_i)$ where f is a parameter controlling crossover. The trim function[33] 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[7], the *Population* contains examples that are better than at least one other candidate. As the looping progresses, the *Population* is full of increasingly more valuable solutions which, in turn, also improves the candidates, which are Extrapolated from the Population. Hence, Vesterstrom et al. [67] found DE to be competitive with particle swarm optimization and other GAs.

Note that DEs have been applied before for parameter tuning (e.g. see [8, 17, 27, 68, 69, 29]) but this is the first time they have been applied to tune LDA to increase stability.

4. Experimentation

In this section, any result from the smaller data sets (Pits and Citemap) come from Python implementation based on Scikit-Learn running on a single 4-core machine. Also, any results from the larger data (StackOverflow) comes from a Scala implementation based on Mllib [70] running on a 45 node Spark system (8 cores per node).

Note that, for the most part, we defer the StackOverflow results to the §5 *Threats to Validity* where we will show that (a) topic modeling instability exists across multiple platforms and implementation languages; (b) tuning can improve stability for different platforms and implementations for LDA.

4.1. RQ1: Are the default settings of LDA correct?

This first research question checks the core premise of this article– 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 5 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 [8, 71] is to report the top 10 words per topic. As shown in Figure 6a, 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 SE data can lead to systematic errors due to topic modeling instability.

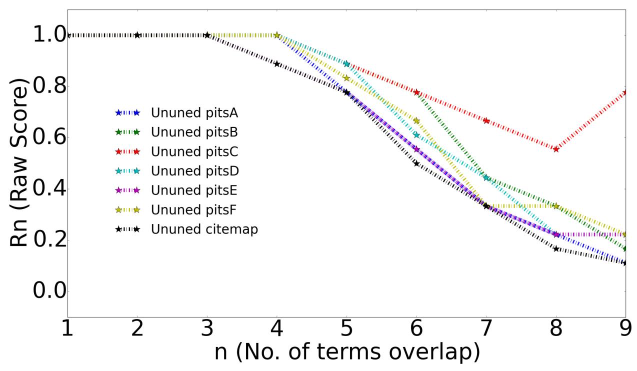


Figure 5: *Before* tuning: uses LDA's default parameters

4.2. RQ2: Does any of the previous study is affected due to instability in LDA?

Barua et al. [1] analysed topics and trends in StackOverflow. And they presented 40 topics with 9 words in each topic We couldn't replicate the exact topics since they performed LDA on the data dump of StackOverflow available before 2012. We ran LDA twice with different input orderings and recorded 40 topics. The idea is since they are making conclusions based on their 40 topics with 1 input ordering. Their results could be greatly affected. Our recorded topics can be seen in Table 5 and 6. From both these tables, it is easily seen that they we are getting different topics with different words.

Top LDA words	
Topic 1:	string new public int null return privat void java
Topic 2:	thread process run connect start time socket send task
Topic 3:	android activ intent new public view context void import
Topic 4:	request servic http facebook respons client api use server
Topic 5:	item event list var function new click flash sub
Topic 6:	class public method object new type void instanc return
Topic 7:	page http html com url link content www site
Topic 8:	error test compil librari command includ file lib usr
Topic 9:	object video play game player obj use data audio
Topic 10:	use like way need question want know make time
Topic 11:	app use applic want like develop iphon need user
Topic 12:	user session password login usernam action control form page
Topic 13:	xml encod string data pars element json attribut use
Topic 14:	android layout content width view parent height wrap textview
Topic 15:	python self print django modul line import def templat
Topic 16:	model set public entiti class product new object properti
Topic 17:	java org apach com bean servlet springframework http sun
Topic 18:	tabl select column queri row null group order join
Topic 19:	work use test tri code problem chang like set
Topic 20:	imag img png jpg height width src size color
Topic 21:	file path upload folder read use directori git open
Topic 22:	bind grid properti wpf window valu control silverlight xaml
Topic 23:	function jqueri javascript var script type input form ajax
Topic 24:	div class css style width left text color span
Topic 25:	amp valu key true fals nbsp return case param
Topic 26:	error java androidruntim android info com debug method lang
Topic 27:	option date valu select time day year month format
Topic 28:	email messag valid form field address contact error send
Topic 29:	number data point valu float doubl count rang length
Topic 30:	int char amp const std return void includ function
Topic 31:	php array echo post string match replac regex function
Topic 32:	asp text button label form net click control checkbox
Topic 33:	map search locat googl report document vim keyword new
Topic 34:	server window instal run use log local applic set
Topic 35:	net asp mvc web visual studio dll use control
Topic 36:	rubi rail gem lib end user rvm app requir
Topic 37:	list function node variabl parent tag use like express
Topic 38:	self view iphon nsstring alloc object nil cell anim
Topic 39:	project eclips plugin build version target jar maven depend
Topic 40:	databas sql mysql data queri row connect tabl server

Table 5: Run 1 showing the 40 topics discovered by LDA

Result 2
Using the "off-the-shelf" default parameters we can say that all the previous studies need to be re-run to get rid of any possible errors due to instability in LDA clusters.

4.3. RQ3: Does tuning improve the stability scores of LDA?

Figure 6a and Figure 6b shows 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 was 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

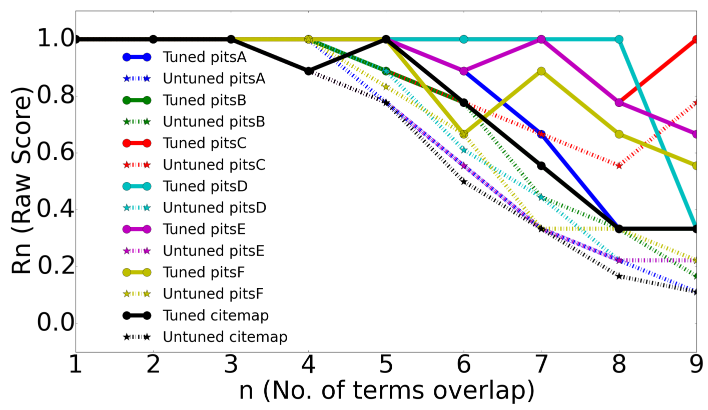


Figure 6a: *After* tuning: uses parameters learned by DE.

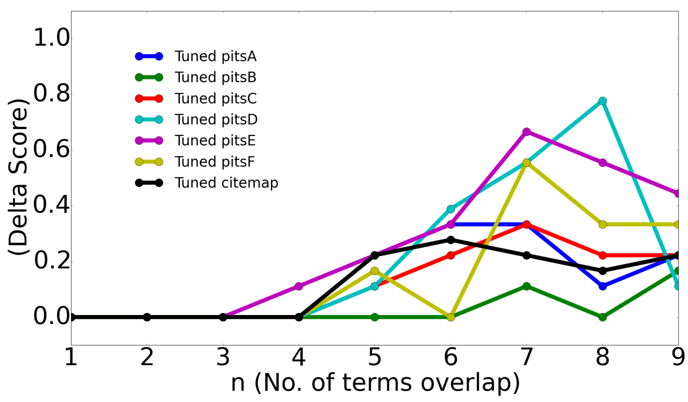


Figure 6b: *Delta = After - Before.*

Figure 6: **RQ1, RQ2** stability results over ten repeated runs. In these figures, *larger* numbers are *better*.

Top LDA words
Topic 1: bind properti grid wpf window valu control silverlight xaml
Topic 2: android layout activ view button content textview intent parent
Topic 3: debug info memori git svn error branch stack commit
Topic 4: function javascript var jqueri script ajax document data load
Topic 5: div class css width html span left jqueri href
Topic 6: model facebook json django api user app comment profil
Topic 7: tabl row queri sql select column key null insert
Topic 8: amp text color html font style nbsp titl bodi
Topic 9: date time day datetim month year format compani hour
Topic 10: entiti sql databas data use queri string new connect
Topic 11: class public object string return new method set type
Topic 12: work problem tri use chang code set doesn issu
Topic 13: event asp true click button net control text server
Topic 14: number data point valu use hash algorithm calcul random
Topic 15: valu option select level citi countri param state dropdown
Topic 16: java new public error import void string int androidruntim
Topic 17: user use data want databas applic creat store need
Topic 18: use like way need question know look make code
Topic 19: form input type valu field text valid label button
Topic 20: string search text match charact word replac like regex
Topic 21: app iphon view tab applic video use screen window
Topic 22: http com url googl map www html link page
Topic 23: item list product group categori order price filter sort
Topic 24: page php content view post index html control titl
Topic 25: rubi rail gem lib user end rvm app requir
Topic 26: data string read byte new write stream encod use
Topic 27: self function variabl foo tag def templat class attribut
Topic 28: messag email log address contact send mail phone locat
Topic 29: imag img height jpg width png size draw canva
Topic 30: request user session password login server http respons usernam
Topic 31: window project file error librari compil visual build dll
Topic 32: line python file print command script curl output run
Topic 33: thread server connect run client process socket start task
Topic 34: test version eclips plugin instal run use project maven
Topic 35: java org apach com http bean servlet springframework sun
Topic 36: int amp return char includ const std void error
Topic 37: self nsstring alloc object nil iphon view cell void
Topic 38: net web asp servic mvc use control applic wcf
Topic 39: php array mysql echo queri post result error row
Topic 40: file xml path node upload folder directori document filenam

Table 6: Run 2 showing the 40 topics discovered by LDA

models generated using different input orderings. Accordingly, our answer to **RQ2** is:

Result 3
For stable clusters, tuning is strongly recommended for future LDA studies. α and β matters the most for getting good clusters.

4.4. *RQ4:Does LDADE improve the classification F2 accuracy scores?*

We studied some other StackExchange websites data dump for classification results which were generated by Krishna et al. [72]. These datasets are categorized into binary labels saying which documents are relevant and non relevant. Here goal of our DE was still to maximize the $\Re_n$ score. It shouldn't be confused with maximizing some other accuracy goals.

In Figure 7, x-axis represents different datasets as generated. Y-axis represents the F2 scores which weigh recall higher than precision [73]. One legend called as "untuned" basically uses default parameters with $k = 10$. Other legends in the graph shows tuning of k for 20, 40, 80, 200 by keeping α and β fixed. There are about 20% improvement than the untuned results.

Result 4
For any SE classification task, tuning is again strongly recommended. And k matters the most for a good classification accuracy..

4.5. *RQ5: Do different data sets need different configurations to make LDA stable?*

Figures 8, 9, and 10 show the results of tuning for word overlap of $n = 5$. On display in each set of vertical bars are the median values generated across 10 tunings. Also, shown are the inter-quartile range (IQR) of those tunings (the IQR is the 75th-25th percentile values and is a non-parametric measure of variation around the median value). Note that in Figure 8, IQR=0

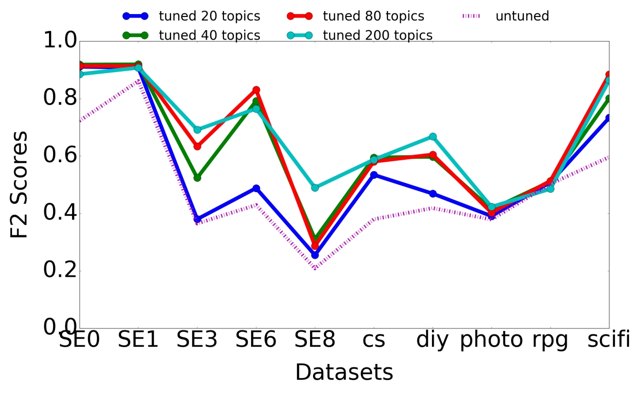


Figure 7: Tuning and Untuned results for Classification SE Task

for PitsB dataset where tuning always converged on the same final value.

These figures show how tuning selects the different ranges of parameters. Some of the above numbers are far from the standard values; e.g. Garousi et al. [24] recommend using $k = 67$ topics yet in our data sets, best results were seen using $k \leq 24$. Clearly:

Result 5
Do not reuse tunings suggested by other researchers from other data sets. Instead, always retune for all new data.

4.6. *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 [39]. For example, when tuning k, α, β, the DE literature is recommending $np = 30$. Figure 11 explores $np = 30$ vs the $np = 10$ we use in Algorithm 2 (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 isn't 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.

4.7. *RQ7: What is the runtime cost of tuning?*

Search-based SE methods can be very slow. Wang et al. [74] once needed 15 years of CPU time to find and verify the tunings required for software clone detectors. Sayyad et al. [75] 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, tuned LDA use Algorithm 1, Algorithm 2 respectively. Based on the psuedocode shown above, our pre-experimental expectation is that tuning will be three times slower than not tuning.

Figures 12 and 13 check if that theoretical holds true in practice. Shown in blue and red are the runtimes required to run LDA untuned, tuned (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, tuning LDA 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.

4.8. *RQ8: Should data miners be used "off-the-shelf" with their default tunings?*

Figure 6 shows that there is much benefit in tuning. Figures 8, 9, and 10 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 12 and 13 show that tuning is definitely slower than otherwise, but the overall cost is not prohibitive. Hence:

Result 8
Whatever the goal is, whether using the learned topics, or cluster distribution for classification we cannot recommend using "off-the-shelf" LDA.

5. Threats to Validity

The usual software analytics threats to validity hold for this paper:

- Our conclusions assume that the goal of topic modeling is to produce stable topics that humans will browse and reflect on as well as for any classification task. Most of our SE applications can be bucketed into these 2 cases. AS mentioned in the introduction, there are some class of papers that do *not* do that. Rather, they use LDA as part of some internal process where the topics are intermediaries used to generate some other goal [6]. Our results might change for such scenarios.
- 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. For example, the footnotes of this paper contain all the urls where other researchers can download our materials and explore the conclusions of this paper.

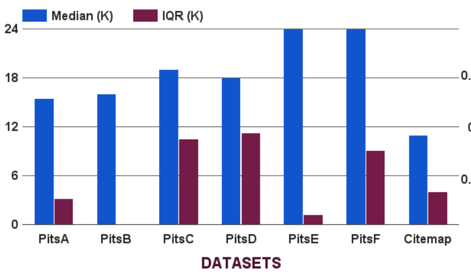


Figure 8: Datasets vs Parameter (k) variation

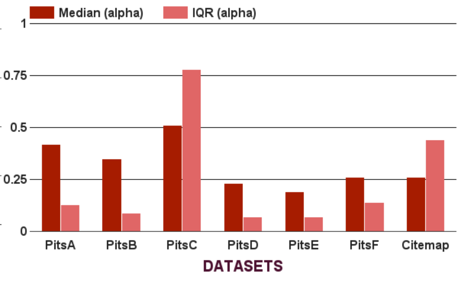


Figure 9 Datasets vs Parameter (α) variation

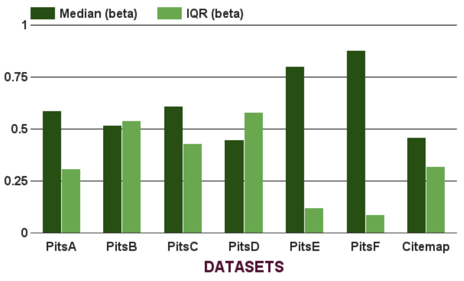


Figure 10 Datasets vs Parameter (β) variation

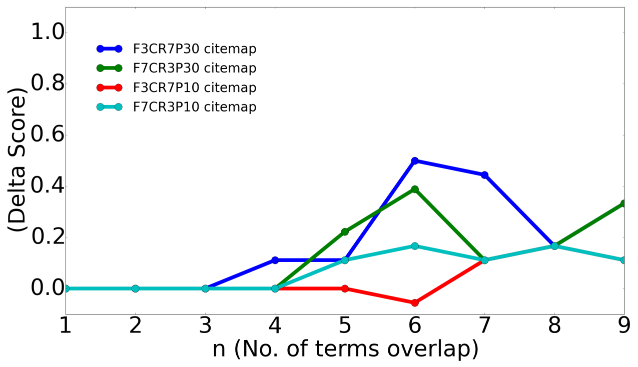


Figure 11: Terms vs Delta Improvement using Different settings of DE

As to more specific threats to validity, one issue might be that our conclusions on "LDA" are really quirks of a specific implementation. To check this, it is insightful to compare our results with:

- The Pits and Citemap results, executed in Scikit-Learn and Python running on a desktop machine.
- The StackOverflow data set executed in Scala using Mllib running on a Spark cluster.

Another useful comparison is to change the internal of the LDA:

- Sometimes use VEM sampling;
- Sometimes use Gibbs sampling;

Figure 14 compares the VEM vs Gibbs sampling. and Figure 15 shows tuning results for StackOverflow, Citemap, and PitsA using Scala/Spark cluster (for results on other data sets, see https://goo.gl/faYAcg and https://goo.gl/UVaql1) When compared with the Python/desktop results of Figure 6 we see the same patterns:

- Tuning never makes stability worse.
- Sometimes, it dramatically improves it (in particular, see the Citemap results of Figure 15).

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).

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 11 and https://goo.gl/HQNASF 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.

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

Table 7: Evaluations vs Stability Scores

6. Conclusion and Future Work

Based on the above, we offer a specific and general conclusion. Most specifically, we recommend

- 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. [71], 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.

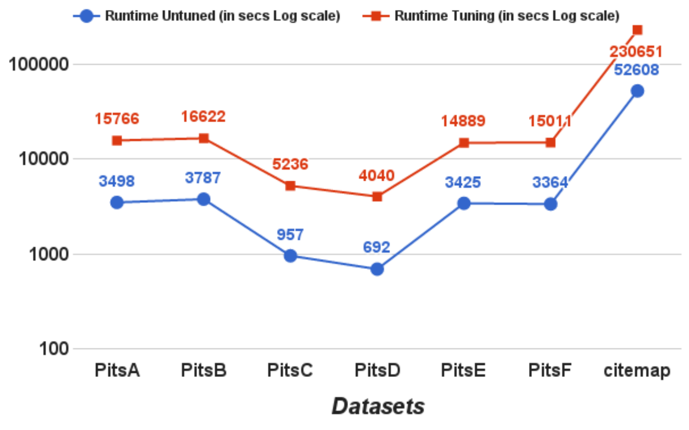


Figure 12 VEM: Datasets vs Runtimes

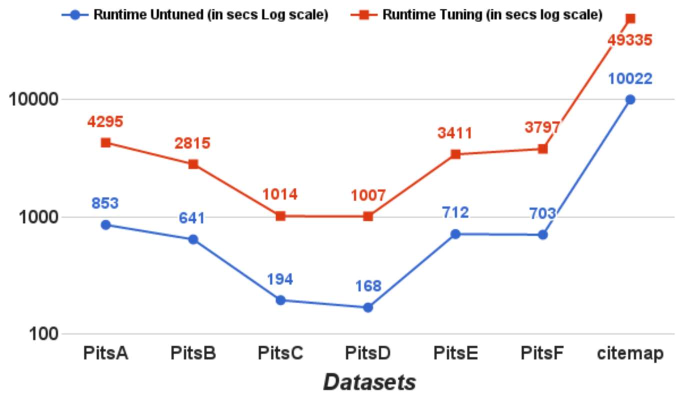


Figure 13: Gibbs: Datasets vs Runtimes

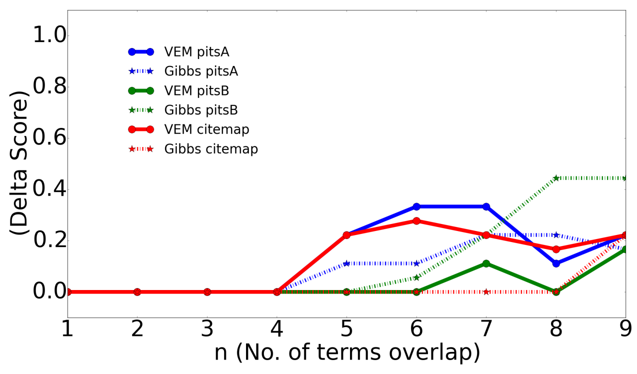


Figure 14: GIBBS vs VEM

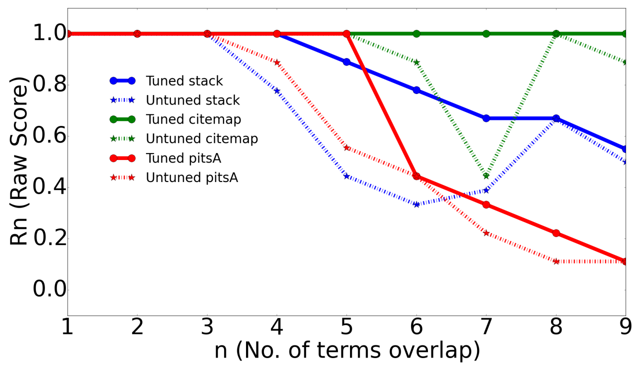


Figure 15: Spark Results

- Any other studies which are making use of these topic distribution for classification results need to be tuned first before using them in their further tasks.
- Do not download someone else's pre-tuned LDA since, as shown here, the best LDA tunings vary from data set to data set.
- These results demand us to visit 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 [29] (a result that has also been reported by other researchers [8]). 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 implications 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. 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.

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

[3] 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.

[4] 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.

[5] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, T. Zimmermann, Local vs. global lessons for defect prediction and effort estimation, ieee trans. software eng., preprint, published online dec. 2012; http://goo. gl/k6qno. tim menzies is a full professor in computer science at the lane, in: Department of Computer Science and Electrical Engineering, West Virginia University. Contact, Citeseer.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

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

[27] 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.

[28] G. Mathew, A. Agrawal, T. Menzies, Trends in topics at SE conferences (1992-2016), Submitted to IST Journal. Available from `tiny.cc/trendsSE`.

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

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

[31] 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.

[32] 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.

[33] J. H. Gennari, P. Langley, D. Fisher, Models of incremental concept formation, Artificial Intelligence 40 (1) (1989) 11 – 61. doi:http://dx.doi.org/10.1016/0004-3702(89)90046-5. URL `http://www.sciencedirect.com/science/article/pii/0004370289900465`

[34] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32. arXiv:/dx.doi.org/10.1023%2FA%3A1010933404324, doi:10.1023/A:1010933404324.

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

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

[42] 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.

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

[44] 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.

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

[46] 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.

[47] M. Porter, The Porter Stemming Algorithm (1980).

URL http://tartarus.org/martin/PorterStemmer/

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] A. T. Goldberg, On the complexity of the satisfiability problem, Ph.D. thesis, New York University (1979).

[55] F. Glover, C. McMillan, The general employee scheduling problem. an integration of ms and ai, Computers & operations research 13 (5) (1986) 563–573.

[56] R. P. Beausoleil, MOSS multiobjective scatter search applied to non-linear multiple criteria optimization, European Journal of Operational Research 169 (2) (2006) 426–449.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] 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.

[64] T. L. Griffiths, M. Steyvers, Finding scientific topics, Proceedings of the National academy of Sciences 101 (suppl 1) (2004) 5228–5235.

[65] L. Layman (2016).

[66] 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.

[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] 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.

[69] 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.

[70] 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.

[71] S. K. Lukins, N. A. Kraft, L. H. Etzkorn, Bug localization using latent dirichlet allocation, Information and Software Technology 52 (9) (2010) 972–990.

[72] 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.

[73] D. M. Powers, Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

[74] 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.

[75] 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.