Taming Topic Instability For Reasoning Over Unstructured Data in Software Engineering

Amritanshu Agrawal
Computer Science, NCSU
Raleigh, North Carolina
aagrawa8@ncsu.edu

Tim Menzies
Computer Science, NCSU
Raleigh, North Carolina
tim.menzies@gmail.com

Wei Fu
Computer Science, NCSU
Raleigh, North Carolina
wfu@ncsu.edu

***Abstract*—Topic Modeling has been widely used to identify patterns in a corpus across various fields. The Latent Dirichlet Allocation (LDA) algorithm is a widely used topic modeling algorithm which in practice is called using its defaults. We report that these defaults can lead to highly misleading results in Software Engineering. It is very important to generate stable topics using the Non-Deterministic LDA method. But one of the curse is setting the right parameters that control the LDA. We ran differential evolution (DE) as an optimizer to explore the tuning space (as a first step) then tested it against the default parameter settings. We found that these tunings were remarkably simple, and took hundreds, not thousands of evaluations to obtain very good results. Since the improvements are so large, and (2) the tuning is so simple, that many industries, who are using LDA, should change the standard methods for better throughput and accuracy of their products. The implication for other kinds of topic modeling algorithms is now an open and pressing issue. Even we can improvise the actual clusters to get better results.**

Keywords*—*Topic modeling, Stability, LDA, tuning, differential evolution.

I. Introduction

In the 21st century, we have now access a mount of data about software engineering. International Data Corporation (IDC), figure 1, shows the number to be 1600 Exabytes. Most of these data is in an unstructured format [40]. Unstructured data does not have a pre-defined data model and is typically text-heavy. Finding insights among unstructured text is extremely difficult unless we can search, characterize, and classify their text data in a meaningful way.

One of the common algorithm for finding related topics within unstructured text (an area called topic modeling) is Latent Dirichlet Allocation (LDA) as shown in the next section. It is usually used using off-the-shell default parameters. But what we found is due to the algorithms non-deterministic behaviour, the topics generated are not stable. In the next section, we will be exploring this problem more and the workaround given by other researchers.

There are various libraries which provide LDA implementation and each library comes with a set of parameters. It is very impractical to get the right combination of parameters due to larger search space. To tackle the above problem, we observed that the parameters taken by LDA can be tuned. In the next coming sections, we will show that many researchers have agreed to the idea of using different configurations for generating topics using LDA. But, we have seen rare work done when it comes to tune the important parameters of LDA for getting stable topics. Most people have went ahead with LDA's default parameters and have not searched the whole configurations space.

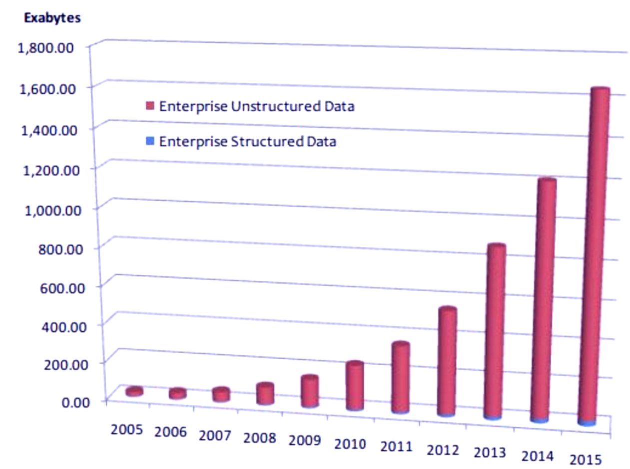

Figure 1: Data Growth 2005-2015

To tackle the problem of how to tune, we have seen that one of the simplest multiobjective optimizers (differential evolution [51]) works very well for tuning in certain kinds of software analytics [15]. There has been success stories where DE has helped in tuning the parameters [15], [6], [11]. Prior to this work, our intuition was that tuning would change the behavior of LDA, to some degree. Also, we suspected that tuning would take so long time and be so CPU intensive that the benefits gained would not be worth the effort.

The results of this paper show that the above points are false since:

1) Tuning LDA is remarkably simple;
2) And can dramatically improve stability in the topic generation.

We can say that now, we can mitigate stability of non-deterministic LDA. Many industries, working on a product development in text analytics have been widely using LDA. This gives them a chance to incorporate better throughput and accuracy in their product.

Those results were found by exploring six research questions:

- **RQ1**: *Do the default settings lead to misleading results?* We will show that topics generated are more stabled with tuned parameters rather with the default settings.
- **RQ2**: *Does tuning improve the stability scores of LDA?* In our work, we will show dramatic improvement in the stability scores with the tuned parameters rather than going for default parameters.
- **RQ3**: *Does different data need different configurations to make LDA stable? Does it change some predefined parameter values of lda?* We will show that different sets of configurations are found out by DE. For one of the results, we showed that we need topic size of less than 30 rather than 67 [17].
- **RQ4**: *Is tuning easy?* We have seen that one of the simpler multiobjective optimizers (differential evolution [51]) works very well for tuning in major software analytics.
- **RQ5**: *Is tuning impractically slow?* We achieved dramatic improvements in the stability scores of LDA in less than 300 evaluations (!); i.e., very quickly.
- **RQ6**: *Should data miners be used "off-the-shelf" with their default tunings?* Wei et al. [15] showed that for defect prediction from static code measures, their answer is an emphatic "no". And we also say that for LDA, it is a "no".

Topic Modeling have been used in various spheres of Software Engineering. Sun et al. reported a survey of LDA's usage to support various SE tasks between 2003 and 2015 [52]. The number of papers published across top major conferences and journals are listed in figure 2. The survey in figure 3 show that there is an increasing concern in this area. Topic modeling is applied in various SE tasks, including source code comprehension, feature location, software defects prediction, developer recommendation, traceability link recovery, re-factoring, software history comprehension, software testing and social software engineering.

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	3
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
ESEC/FSE	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

Figure 2: Survey Venue and Statistics

There has been various other work done in Requirements Engineering where it was necessary to analyze the text and come up with the important topics [5], [53], [34]. People have used topic modeling in prioritizing test cases, and identifying the importance of test cases [22], [62], [59]. Increasingly, it has also become very important to have automated tools to do

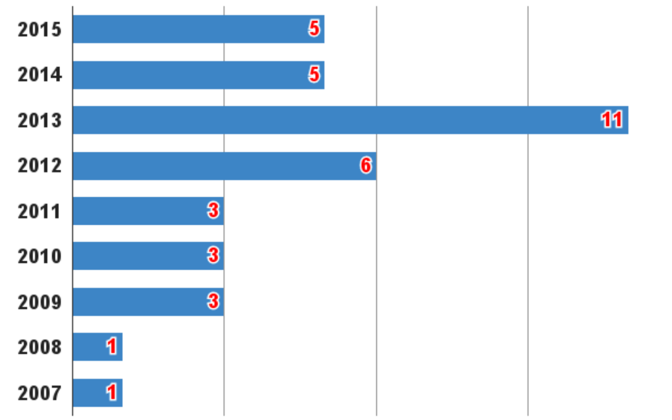

Figure 3: The distribution of the papers published in different years

a Systematic Literature Review (SLR) [55]. We found these papers [50], [2], [31] who have used clustering algorithms (topic modeling) to do SLR.

This gives us the motivation of generating stable topics in software engineering which will make product development faster, cheaper and accurate.

II. Previous Work

Are the impacts of instability in LDA addressed in the text analytics? To answer that question, in April 2016, we searched scholar.google.com for the conjunction of "lda" and "topics" or "stable" or "unstable" or "coherence". We got about 475 results. We selected for papers since 2012, we got about 189 results. After discarding the non-SE papers, we read over this sample of 57 highly-cited SE text analytics papers. Only about 21 papers talk about instability in LDA (more details can be found at *https://goo.gl/Bpc6Vb*). Also, we found in that sample was that only one author acknowledged the impact of tunings. About 60% papers in our sample did not adjust the "off-the-shelf" configuration of the LDA. Even if they did, then they only played with number of topics (k). Of the remaining papers the authors have acknowledged the fact of topics instabilities due to LDA. The literature review for rest of the papers are categorized in the following subsections.

A. Problems with LDA

In LDA, learning the various distributions is a problem of Bayesian inference. It also use the alternative inference techniques like Gibbs sampling [57] and expectation propagation (VEM) [38]. These techniques gave rise to non deterministic ways which are responsible for the instability in the topic modeling. Gibbs sampling inference method for LDA runs too slow for large dataset with many topics. The topics learned by LDA sometimes are difficult to interpret by end users. LDA suffers from instability problem [60], [47]. Online learning [24] for LDA uses constant memory requirements and empirically converges faster than batch collapsed Gibbs sampling. However, it still requires a full pass through the entire corpus each iteration. It can therefore be slow to apply to very large datasets, and is not naturally suited to settings where new data is constantly arriving.

Rationality about the topics instability is important if an industry is using topic modeling in all their use cases [28], [45]. They defined other terminologies which go hand in hand when it comes about stability point of view. Topic coherence, the semantic interpretability of the top terms usually used to describe discovered topics. Intruder word are those, which has low probability in the topic of interest, but high probability in other topics. The word intrusion measures topic interpretability differently to observed coherence.

Different inference techniques in LDA try to maximize the resulting lower bound on the log likelihood [9]. The expected complete log likelihood of the data can have many local maximas, which leads to different distributions and in turn leads to instability in the output. The distributions which are found with the help of LDA, result from the same dataset with the same vocabulary and model parameters, any differences between them are entirely due to the randomness in inference techniques [26]. This randomness affects perplexity variations, word and document ratios. There is a problem of finding the optimal number of clusters, but different configuration parameters may lead us to the stable topics.

We saw that this is just not happening with only particular programming language or particular library. This instability is happening irrespective of the tools in which LDA has been implemented. Some of the papers mentioned using GibbsLDA++ written in C++ and they observed the same instability [30], [54], [21]. Some papers mentioned about using Python implementation of LDA and observed the same instability [21]. LDA implemented in JAVA language produced the same incoherent topics [32], [23]. And, we observed this problem is in the Scikit-Learn version of LDA, implemented in Python, as well as Spark Mllib library.

All these recently [60], [47], [28], [45], [26], [30], [54], [21], [32], [23], [43], [33], [49], [29] papers have identified the problems of instability in LDA clusters or topics and have tried to give a work around to solve these issues. This tells us that our proposal of finding a solution to make clusters or topics generated more stable is quite valid.

B. Tuning: Important and Ignored

This section argues that tuning is an under-explored software analytics. The impact of tuning is well understood [8]. Yet issues of tuning are rarely or poorly addressed in the defect prediction literature. 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.

Wei et al. [15] shows that only few authors acknowledged the impact of tunings. A few other papers did acknowledge that one data miner may not be appropriate for all data sets. Those papers tested different "off-the-shelf" data miners on the same data set. For example, Elish et al. [13] compared support vector machines to other data miners for the purposes of defect prediction. Those papers tested different "off-the-shelf" data miners on the same data set. Wei et al. [15] showed that finding useful tunings is very easy. This result is both novel and unexpected. Learners are very amenable to tuning. Hence, they are also very amenable to significant performance improvements. Given the low number of evaluations required, they asserted that tuning should be standard practice for anyone using data miners.

C. WorkAround to LDA Instability Problem

1) *Unsupervised:*

Solution 1: To evaluate the stability of topics, most papers [33], [32], [21] manually accessed the topics and then used for further experiments. Some made use of Amazon Mechanical Turk to create gold-standard coherence judgements [28].

This workaround will take a lot of manual effort and time which is not feasible for many. In the age of automation, we need an efficient way of making LDA stable.

Solution 2: Users have external knowledge regarding word correlation, which can be taken into account to improve the semantic coherence of topic modeling methods. SCLDA [60] can handle different kinds of knowledge such as word correlation, document correlation, document label and so on. One advantage of SCLDA over existing methods is that it is very fast to converge.

We do not have enough resources to have a user knowledge for the type of datasets we used, and that's the reason we are not using this solution to improve stability.

Solution 3: The most common parameters in the LDA are number of clusters (k), number of iterations (n), document topic prior (α), and word topic prior (β). The other solution is tuning these parameters of LDA. Tuning the parameters and using different configurations were used by [47], [29], [43]. They [16], [54] achieved higher stability by just increasing the number of cluster size.

We found this to have better advantage than the other solutions based on the tuning work done by Wei [15]. And in our work, we have used this solution.

2) *Supervised:*

Other Solutions: *Labeled LDA* [48], a topic model that constrains LDA by defining a one-to-one correspondence between LDA's latent topics and user tags. This allows Labeled LDA to directly learn word-tag correspondences. Labeled LDA's improved expressiveness over traditional LDA with visualizations of a corpus made it to outperform SVMs by more than 3 to 1. By inncorporating domain knowledge, a *Dirichlet Forest* prior, in a Latent Dirichlet Allocation framework [3] gave improved performance. The prior is a mixture of Dirichlet tree distributions with special structures. Fold-all model, which allows the user to specify general domain knowledge in First-Order Logic (FOL). *Logic LDA* [4] is a scalable inference technique using stochastic gradient descent. *Quad-LDA* [42] is a framework in order to improve the coherence of the keywords per topic learned by LDA.

NMF-LDA [58] build a *Markov Random Field* (MRF) regularized LDA model, which defines a MRF on the latent topic layer of LDA to encourage words labeled as similar to share the same topic label. *Interactive Topic Modeling* (ITM) [25], allows untrained users to encode their feedback easily and iteratively into the topic models.

Labeled LDA can only handle document label knowledge. *Dirichlet Forest LDA, Quad-LDA, NMF-LDA and ITM* can only handle word correlation knowledge. *MRTF* [12] can only handle document correlation knowledge. *Logic LDA* can handle word correlation, document label knowledge and other kinds of knowledge. However, each knowledge has to be encoded as First-Order Logic. **All these subsequent frameworks helped in generating better stable topics**.

We will not be able to use any of these supervised solutions as our datasets are not labeled. And these all frameworks work best with the classified datasets. This is going to be our future work which we will be addressing next.

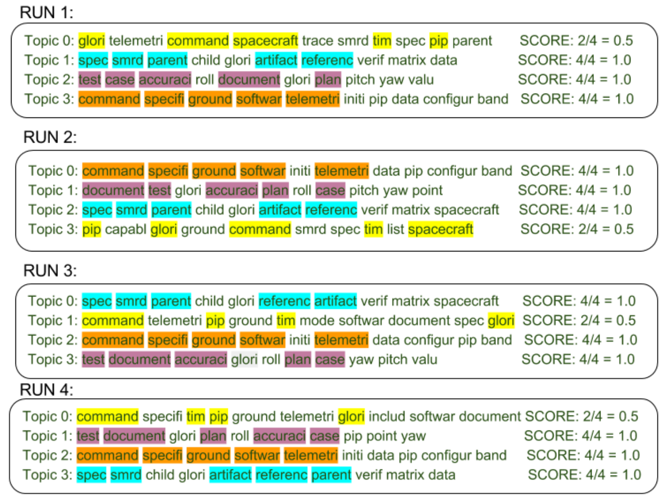

Figure 4: Example of 5 terms overlap across 4 runs and its stability score

D. Evaluation

To evaluate the topics coherence or the cluster stability using LDA, there has been number of evaluation measures proposed. There is a direct approach, by asking people about topics, and an indirect approach by evaluating pointwise mutual information (PMI) [28], [45]. PMI is an automatic method for estimating topic coherence based on pairwise information between the topic words. PMI being an automatic method, we are not sure of exact details of it which made us to not use this measure. We could not use direct approach, due to resource limitation to ask an expert for the type of datasets we used, and that's the reason we are not using this criteria to evaluate.

Perplexity is monotonically decreasing in the likelihood of the data, and is algebraically equivalent to the inverse of the geometric mean per-word likelihood. It shows how well topic-word and word-document distributions predict new test samples. The smaller the perplexity, the better (less uniform) is the LDA model. Problem with perplexity is that the value of perplexity drops as the number of topics grows and perplexity depends on the dictionary size [26].

Many researchers uses Jaccard Similarity to check for the overlap of terms across multiple runs. We define our measure similar to it. We say topics are stable, when there are x times n% of terms overlap in a topic across m runs. We can see an example of 5 terms overlap in figure 4. We take median of all the scores generated by evaluating topic. In this example, figure 4, median score is 1.0 for each run. So overall median score comes out to be 1.0. We always select top 10 terms within a topic suggested by some researchers. It makes sense to us that to have a successful topic modeling, to define each topic you need top few terms which will completely define that topic. If we get different words then it will mean different topics which will put any product based on topic modeling at risk.

E. Notes on LDA

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) and is a problem of Bayesian inference. The original method used is a variational Bayes approximation of the posterior distribution [9], alternative inference techniques use Gibbs sampling [20] and Variational Expectation Maximisation (VEM) [38].

Regardless of the methods, all these attempts to fit the model to data using maximum likelihood. Lesser the likelihood, more the convergence of the model and expected to achieve better results. The perplexity, used by convention in topic modeling, is monotonically decreasing in the likelihood of the data.

The pseudocode for untuned LDA is shown in Algorithm 1 with the default set of parameters. In the following description, superscript numbers denote lines in the pseudocode. The data is shuffled everytime LDA is run^{L5}. Data is in the form of term frequency scores of each word per document. Shuffling is done in order to induce maximum variance among different runs of LDA. Topics^{L6} is a list of list which contains topics from all the different runs. A stability score is evaluated on every 10 runs, and this process is continued 10 times. At the end, median score is selected as the untuned final score^{L3-L11}.

F. Tuning Algorithms

The other solution which we discussed was tuning the parameters of LDA. Here is a list of optimizers used widely in research: simulated annealing [14], [36]; various genetic algorithms [19] augmented by techniques such as differential evolution [51], tabu search and scatter search [18], [7], [39], [41]; particle swarm optimization [46]; numerous decomposition approaches that use heuristics to decompose the total space into small problems, then apply a response surface methods [27], [63]. Of these, the simplest are simulated annealing (SA) and differential evolution (DE), each of which can be coded in less than a page of some highlevel scripting language. Our reading of the current literature is that there are more

Algorithm 1 Pseudocode for untuned LDA with Default Parameters

```
Input: terms, Data
Output: Final_Score
function LDASCORE( term, Data)
    Score ← ∅
    for j = 0 to 10 do
        for i = 0 to 10 do
            data ← shuffle(Data)
            Topics.append(lda(k = 10,α = none,β = none ))
        end for
        Score.append(Overlap(Topics, term,l[0] ))
    end for
    Final_Score ← median(Score)
    return Final_Score
end function
```

advocates for differential evolution than SA. For example, Vesterstrom and Thomsen [56] found DE to be competitive with particle swarm optimization and other GAs. DEs have been applied before for parameter tuning (e.g. see [44], [11], [15]) but this is the first time they have been applied to tune the configurations of LDA.

We will tune hyperparameters of LDA using DE. DE also comes up with parameters like number of iterations, mutation, crossover rate, population size, and different selection strategies. The objective of LDA is to maximize the stability scores. Based on the literature review, we found that we don't need to have topics more than 10 terms. We also found the important parameters that need to be tuned. See Table I. We also used online variational Bayesian approximation as well as gibbs sampling for LDA, with random input row order of data to make sure that in each run the learning is different. This gave us more concrete results as few tuned set of parameters which we found performed good across each run.

The pseudocode for differential evolution is shown in Algorithm 2. In the following description, superscript numbers denote lines in the pseudocode. DE evolves a *NewGeneration* of candidates from a current Population. Our DE will terminate after a certain number of evaluations. We ran initial experiments to define this constant. We will talk about this in section V. Each candidate solution in the Population is a pair of (Tunings, Scores). Tunings are selected from Table I and Scores come from the stability score after running our LDA experiment$^{L23-L30}$.

The premise of DE is that the best way to mutate the existing tunings is to Extrapolate^{L31} between current solutions. 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 39). For numerics, $y_i = a_i + f \times (b_i - c_i)$ where f is a parameter controlling crossover. The trim function^{L41} limits the new value to the legal range min..max of that parameter.

The main loop of DE^{L9} runs over the Population, replacing old items with new Candidates (if new candidate is better). This means that, as the loop progresses, the Population is full of increasingly more valuable solutions. This, in turn, also improves the candidates, which are Extrapolated from the Population.

Algorithm 2 Pseudocode for DE with a constant number of evaluations

```
Input: np = 10, f = 0.7, cr = 0.3, iter = 3, Goal ∈ {Data, term, ...}
Output: S_best, 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.append(Population[i],ldascore(Population[i],
        term, 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.append(S_i,ldascore(S_i, term, Data))
13:             else
14:                 NewGeneration.append(Cur_Gen[j])
15:             end if
16:             Cur_Gen ← NewGeneration
17:         end for
18:     end for
19:     S_best ← GetBestSolution(Cur_Gen)
20:     final_generation ← Cur_Gen
21:     return S_best, final_generation
22: end function
23: function LDASCORE(l, term, Data)
24:     Topics ← ∅
25:     for i = 0 to 10 do
26:         data ← shuffle(Data)
27:         Topics.append(lda(k = l[0],α = l[1],β = l[2]))
28:     end for
29:     return Overlap(Topics, term,l[0] )
30: end function
31: function EXTRAPOLATE(old, pop, cr, f, np)
32:     a, b, c ← threeOthers(pop, old)
33:     newf ← ∅
34:     for i = 0 to np − 1 do
35:         if cr ≤ random() then
36:             newf.append(old[i])
37:         else
38:             if typeof(old[i])== bool then then
39:                 newf.append(not old[i])
40:             else
41:                 newf.append(trim(i,(a[i]+f∗(b[i] − c[i]))))
42:             end if
43:         end if
44:     end for
45:     return newf
46: end function
```

The data is shuffled everytime LDA is run^{L26}. Data is in the form of term frequency scores of each word per document. Shuffling is done in order to induce maximum variance among different runs of LDA. Topics^{L27} is a list of list which contains topics from all the different runs. A stability score is evaluated on every 10 runs. At the end, median score is selected as the final score$^{L23-L30}$.

III. EXPERIMENTATION

A. Data Sets

To answer our research questions, we will be working on couple of open source datasets to verify that tuning does help the stability scores. All these datasets have been preprocessed using the normal steps of stopwords removal, stemming, and then considering the top 5% of tf-idf scores. More details about the statistics of datasets can be found in the table II.

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

Table I: List of parameters tuned by this paper

The datasets used are:-

- **PITS**: This is a text mining data set generated from project and issue tracking system (PITS) reports [35], [37]. This data is about the bugs and changes in the code, to submit and review patches, to manage quality assurance, to support communication between developers, etc. The aim is to identify the top topics which can identify each severity separately. The dataset can be downloaded from this repository [1] [1].
- **StackOverflow**: StackOverflow is a privately held website, the flagship site of the Stack Exchange Network. It features questions and answers on a wide range of topics in computer programming. The data can be downloaded from various sources [2] [3]. This dataset is so big that it can only be run on Spark with a cluser of nodes to reduce the runtime. Results of this dataset is only coming from Spark mllib implementation.
- **Citemap**: A citation, also called a reference, uniquely identifies a source of information. Citemap can be used to create citations for the map. Dataset can be downloaded from [4].

Datasets	Size before preprocessing	Size after preprocessing
PitsA	1.2MB	292 KB
PitsB	704KB	188KB
PitsC	143KB	37KB
PitsD	107KB	26KB
PitsE	650KB	216KB
PitsF	549KB	217KB
Ciemap	8.6MB	3.7MB
Stackoverflow	7GB	527MB

Table II: Statistics on the datasets

IV. Experimental Results

A. *RQ1: Do the default settings lead to misleading results?*

Figure 5 results are coming from Variational Expectation Maximisation (VEM) method of LDA implemented in Python. X-axis represents number of terms overlap. Y-axis represents the raw stability score. Different lines are for the different datasets. We can see that tuning results are either always above the default settings (untuned) or stayed the same. With default settings, it can surely lead to misleading results.

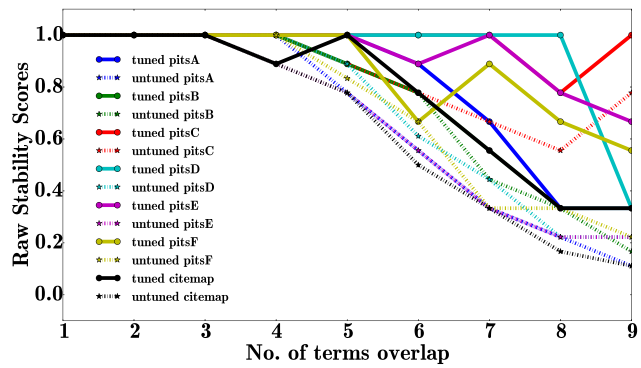

Figure 5: Terms vs Raw Scores between tuning and default parameters

B. *RQ2: Does tuning improve the stability scores of LDA?*

In figure 6, x-axis represents number of terms overlap. Y-axis represents the delta improvement which is $(tuning - untuned)$ results. If the lines are above x-axis, that means tuning performed better than the untuned. Different lines are for the different datasets. It clearly shows that the answer to RQ2 is "yes" - tuning has a positive effect on stability scores. Tuning either helped it or remained the same but it never had any negative effect. The dramatic improvement started showing after 5 terms overlap. The most improvement which we observed was in PitsD dataset for 8 terms overlap of about 80%. That's why we recommend that we shouldn't list more than 6 terms per topic, as stability starts dropping.

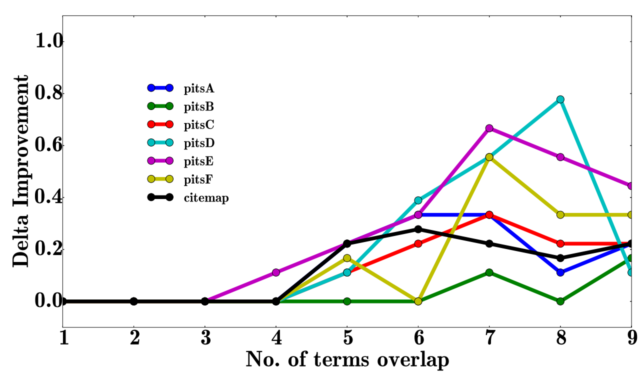

Figure 6: Terms vs Delta Improvement between tuning and default parameters

[1] http://openscience.us/repo/issues
[2] http://data.stackexchange.com
[3] http://blog.stackoverflow.com/tags/cc-wiki-dump/
[4] https://github.com/ai-se/citemap/blob/master/citemap.csv

C. *RQ3: Does different data need different configurations to make LDA stable? Does it change some predefined parameter values of lda?*

We can see these results at *https://goo.gl/Nin5pV*. From the spreadsheet, we can see that for different datasets, with different terms overlap, we are getting very high interquartile range (IQR). This means that tuning helped us to arrive at different sets of configurations making the topic generation more constant. And if looked at spreadsheet Row - 9 and 10, we are seeing IQR to be 0, that means tuning helped us to find only 1 set of configurations which got us to that Stable score. We will find many such similar number throughout the results.

The other results which we found is in the citemap datasets. If we look from Row 56-64, we see that we are getting good stable scores within 30 topic size. But in the [17] paper, they found that we will need a topic size of 67. Definitely, it can change predefined parameters of LDA.

D. *RQ4: Is tuning easy?*

In terms of the search space explored via tuning, is much smaller. To see this, recall from Algorithm 2 that DE explores a Population of size np = 10. This is a very small population size since Rainer Storn (one of the inventors of DE) recommends setting np to be ten times larger than the number of parameters being optimized. We also ran experiments with different sets of F, CR, and population size (*np*), and we see that it didn't change much.

We have shown results only for Citemap dataset using VEM implemented in Python which can be referred in Figure 7. For other datsets, we can refer at *https://goo.gl/HQNASF*. In Figure 7, different lines show different combinations of F, CR and Population size. F is selected either 0.3 or 0.7, and similarly CR is selected 0.3 or 0.7, and population size is selected either 10 or 30 (which is 10 times the number of parameters being optimized). These numbers are according to the original Rainer Storn [51] recommended settings.

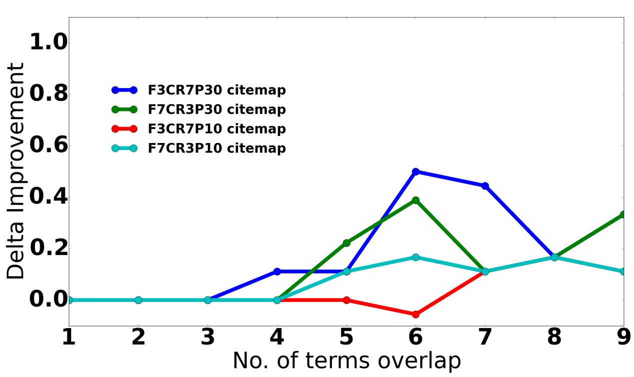

Figure 7: Terms vs Delta Improvement using Different settings of DE

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 $Popsize = 10$

E. *RQ5: Is tuning impractically slow?*

Figure 8 and 9, x-axis represents different datasets and y-axis represents the runtimes in seconds (Log_{10}). From the table III, we will show that we just need 300 evaluations to do tuning. Using this criteria, we can see that tuning runtimes is only about 5 times the runtimes without tuning. Figure 8 is with gibbs implemented in Python, and figure 9 is with VEM implemented in python.

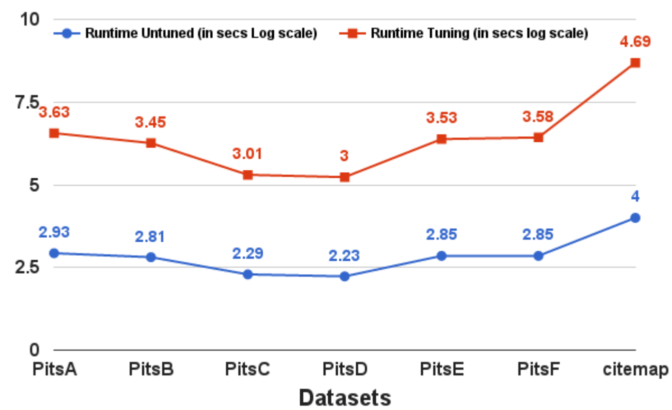

Figure 8: Gibbs: Datasets vs Runtimes

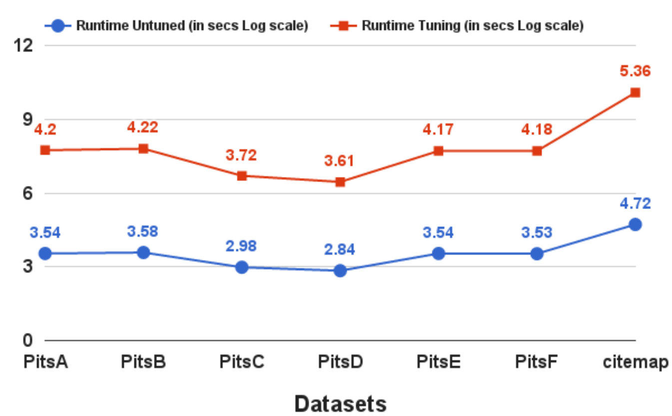

Figure 9: VEM: Datasets vs Runtimes

F. *RQ6: Should data miners be used "off-the-shelf" with their default tunings?*

Figure 10, x-axis represents different datasets and y-axis represents values of median and IQR. This is specific for parameter, number of topics (k). For results of parameters alpha and beta, this link *https://goo.gl/feKii8* can be referred. From the figure 10, we show how tuning selects the different ranges of values of parameters. These results are for 5 terms overlap, with VEM implementaion in Python. We observed the similar trend for other term overlaps and other sampling methods

(Refer *https://goo.gl/Nin5pV*). We can easily see that for different datasets, we need different parameters. Such large IQRs show that we get quite varied ranges of parameters. Hence, we answer RQ6 as "no" since, to achieve the improvements seen in this paper, tuning has to be repeated whenever the goals or data sets are changed. Given this requirement to repeatedly run tuning, it is fortunate that (as shown above) tuning is so easy and so fast.

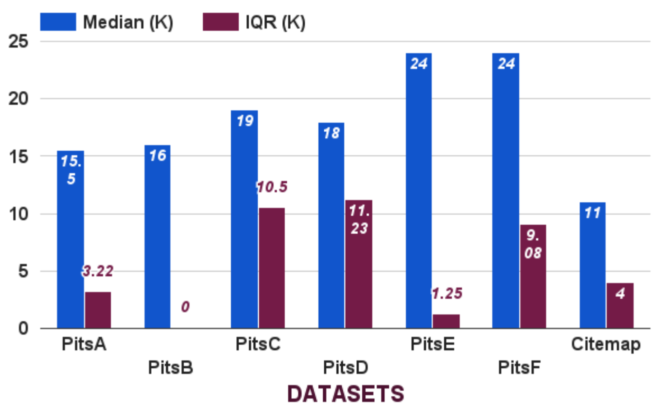

Figure 10: Datasets vs Parameter (k) variation

V. Threats to Validity

Is it a quirk of the implementation? This instability problem remained the same across any implementation of LDA. In figure 11, dashed lines represents the Python implementation and solid lines represents the Spark implementation. The y-axis represents the delta improvement between tuning and untuned results. We can see the same improvement across both the implementations. Here results of only 3 datasets are shown. For other datasets, results can be referred at *https://goo.gl/UVaql1*. So it is not a quirk of implementation.

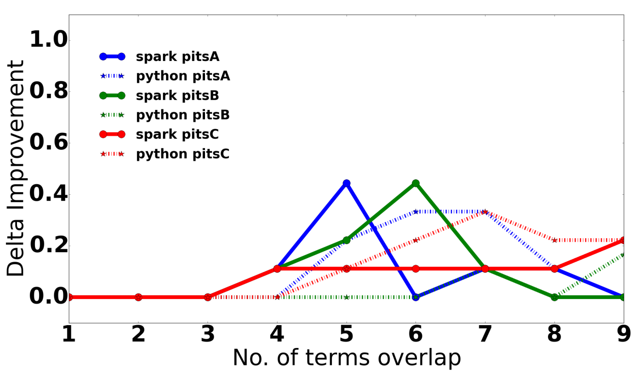

Figure 11: GIBBS vs VEM

Is it a quirk of the sampling method used? In Figure 12, dashed lines represents the Gibbs implementation and solid line represents the tradition VEM method. The y-axis represents the delta improvement between tuning and untuned results. We can see that even after 5 terms, the stability score went down in both Gibbs Sampling and VEM. We also saw the same magnitude of improvements. So it is not a quirk of a particular sampling method. Here results of only 3 datasets are shown. For other datasets, results can be referred at *https://goo.gl/faYAcg*. So it is not a quirk of implementation.

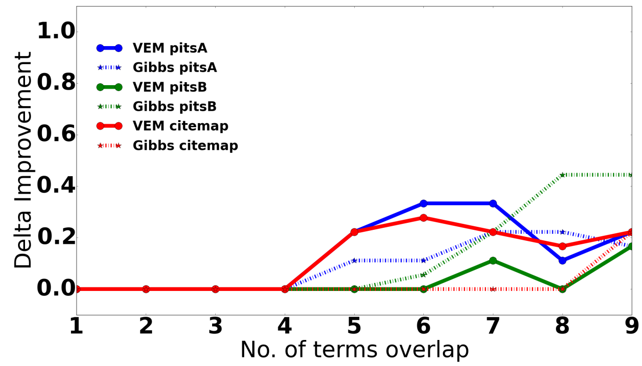

Figure 12: GIBBS vs VEM

Terminating criteria for DE From table III, we can see that after 300 evaluations, there was no improvement in the stability scores across all the datasets and it stayed the same.

Datasets\Evaluations	100	200	300	400
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
Ciemap	0.67	0.67	0.77	0.77
Stackoverflow	0.6	0.7	0.8	0.8

Table III: Evaluations vs Stability Scores

VI. Conclusion and Future Work

Our exploration of the six research questions listed in the introduction show that when doing topic modeling, analytics without parameter tuning are considered harmful and misleading. As more data getting generated day by day, it is really necessary to do accurate topic modeling. Now, with the help of tuning, we can generate stable topics to quite a good extent.

This paper showed that tuning improved the stability scores of LDA, sometimes the improvement is quite dramatic (about 80%). This paper also highlighted that we can now select the right set of parameters for different datasets to get stable topics. This paper combined with Wei et al. [15] suggests that data miners should not be used off-the-shell with their default tunings. Now, we can use the more stable LDA to find features and fed into a classifer to get improved precision and recall [10], [50]

As to future work, it is now important to explore the implications of these stable topics generated by LDA in a product development. We can now actually work with other workarounds mentioned in the section II-C with classified datasets. To improve stability, we can even try to improvise the actual clusters to get better results.

This paper just investigated on stability of LDA using one optimizer. Hence, we can make no claim that DE is the best optimizer for all data miners. There are various other unstable data miners like Decision Tree Learning, Neural Networks, and Bayesian Learning [61] where optimizer can mitigate stability into all these miners.

REFERENCES

[1] The promise repository of empirical software engineering data, 2015.

[2] A. Al-Zubidy and J. C. Carver. Review of systematic literature review tools.

[3] D. Andrzejewski, X. Zhu, and M. Craven. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 25–32. ACM, 2009.

[4] D. Andrzejewski, X. Zhu, M. Craven, and B. Recht. A framework for incorporating general domain knowledge into latent dirichlet allocation using first-order logic. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1171, 2011.

[5] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor. Software traceability with topic modeling. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 95–104. ACM, 2010.

[6] Y. Bazi, N. Alajlan, F. Melgani, H. AlHichri, S. Malek, and R. R. Yager. Differential evolution extreme learning machine for the classification of hyperspectral images. *Geoscience and Remote Sensing Letters, IEEE*, 11(6):1066–1070, 2014.

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

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

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. volume 3, pages 993–1022. JMLR. org, 2003.

[10] T.-H. Chen, W. Shang, M. Nagappan, A. E. Hassan, and S. W. Thomas. Topic-based software defect explanation. *Journal of Systems and Software*, 2016.

[11] I. Chiha, J. Ghabi, and N. Liouane. Tuning pid controller with multi-objective differential evolution. In *Communications Control and Signal Processing (ISCCSP), 2012 5th International Symposium on*, pages 1–4. IEEE, 2012.

[12] H. Daumé III. Markov random topic fields. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 293–296. Association for Computational Linguistics, 2009.

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

[14] M. S. Feather and T. Menzies. Converging on the optimal attainment of requirements. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pages 263–270. IEEE, 2002.

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

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

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

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

[19] A. T. Goldberg. On the complexity of the satisfiability problem. 1979.

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

[21] E. Guzman and 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*, pages 153–162. IEEE, 2014.

[22] H. Hemmati, Z. Fang, and M. V. Mantyla. Prioritizing manual test cases in traditional and rapid release environments. In *Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on*, pages 1–10. IEEE, 2015.

[23] A. Hindle, N. A. Ernst, M. W. Godfrey, and 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*, pages 163–172. ACM, 2011.

[24] M. Hoffman, F. R. Bach, and D. M. Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

[25] Y. Hu, J. Boyd-Graber, B. Satinoff, and A. Smith. Interactive topic modeling. *Machine learning*, 95(3):423–469, 2014.

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

[27] J. Krall, T. Menzies, and M. Davies. Gale: Geometric active learning for search-based software engineering. *Software Engineering, IEEE Transactions on*, 41(10):1001–1018, 2015.

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

[29] S. Lohar, S. Amornborvornwong, A. Zisman, and 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*, pages 378–388. ACM, 2013.

[30] S. K. Lukins, N. A. Kraft, and L. H. Etzkorn. Source code retrieval for bug localization using latent dirichlet allocation. In *Reverse Engineering, 2008. WCRE'08. 15th Working Conference on*, pages 155–164. IEEE, 2008.

[31] C. Marshall and P. Brereton. Tools to support systematic literature reviews in software engineering: A mapping study. In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 296–299. IEEE, 2013.

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

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

[34] A. K. Massey, J. Eisenstein, A. I. Antón, and P. P. Swire. Automated text mining for requirements analysis of policy documents. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 4–13. IEEE, 2013.

[35] T. Menzies. Improving iv&v techniques through the analysis of project anomalies: Text mining pits issue reports-final report. 2008.

[36] T. Menzies, O. Elrawas, J. Hihn, M. Feather, R. Madachy, and B. Boehm. The business case for automated software engineering. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 303–312. ACM, 2007.

[37] T. Menzies and A. Marcus. Automated severity assessment of software defect reports. In *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*, pages 346–355. IEEE, 2008.

[38] T. Minka and J. Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc., 2002.

[39] J. Molina, M. Laguna, R. Martí, and R. Caballero. Sspmo: A scatter tabu search procedure for non-linear multiobjective optimization. *INFORMS Journal on Computing*, 19(1):91–100, 2007.

[40] A. Nadkarni and N. Yezhkova. Structured versus unstructured data: The balance of power continues to shift. *IDC (Industry Development and Models) Mar*, 2014.

[41] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, and A. Beham. Abyss: Adapting scatter search to multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 12(4):439–457, 2008.

[42] D. Newman, E. V. Bonilla, and W. Buntine. Improving topic coherence with regularized topic models. In *Advances in neural information processing systems*, pages 496–504, 2011.

[43] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *Program Comprehension (ICPC), 2010 IEEE 18th International Conference on*, pages 68–71. IEEE, 2010.

[44] M. G. Omran, A. P. Engelbrecht, and A. Salman. Differential evolution methods for unsupervised image classification. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 966–973. IEEE, 2005.

[45] D. O'Callaghan, D. Greene, J. Carthy, and P. Cunningham. An analysis of the coherence of descriptors in topic modeling. *Expert Systems with Applications*, 42(13):5645–5657, 2015.

[46] H. Pan, M. Zheng, and X. Han. Particle swarm-simulated annealing fusion algorithm and its application in function optimization. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 1, pages 78–81. IEEE, 2008.

[47] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and 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*, pages 522–531. IEEE Press, 2013.

[48] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics, 2009.

[49] B. Ray, D. Posnett, V. Filkov, and P. Devanbu. A large scale study of programming languages and code quality in github. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 155–165. ACM, 2014.

[50] A. Restificar and S. Ananiadou. Inferring appropriate eligibility criteria in clinical trial protocols without labeled data. In *Proceedings of the ACM sixth international workshop on Data and text mining in biomedical informatics*, pages 21–28. ACM, 2012.

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

[52] X. Sun, X. Liu, B. Li, Y. Duan, H. Yang, and 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)*, pages 357–362. IEEE, 2016.

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

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

[55] G. Tsafnat, P. Glasziou, M. K. Choong, A. Dunn, F. Galgani, and E. Coiera. Systematic review automation technologies. *Systematic reviews*, 3(1):1, 2014.

[56] J. Vesterstrøm and 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*, volume 2, pages 1980–1987. IEEE, 2004.

[57] X. Wei and 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*, pages 178–185. ACM, 2006.

[58] P. Xie, D. Yang, and E. P. Xing. Incorporating word correlation knowledge into topic modeling. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2015.

[59] G. Yang and B. Lee. Predicting bug severity by utilizing topic model and bug report meta-field. *KIISE Transactions on Computing Practices*, 21(9):616–621, 2015.

[60] Y. Yang. *Improving the Usability of Topic Models*. PhD thesis, NORTHWESTERN UNIVERSITY, 2015.

[61] D. Zhang and J. J. Tsai. *Machine learning applications in software engineering*, volume 16. World Scientific, 2005.

[62] Y. Zhang, M. Harman, Y. Jia, and F. Sarro. Inferring test models from kate's bug reports using multi-objective search. In *Search-Based Software Engineering*, pages 301–307. Springer, 2015.

[63] M. Zuluaga, G. Sergent, A. Krause, and M. Püschel. Active learning for multi-objective optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pages 462–470, 2013.