Taming Topic Instability For Unstructured Data in Software Engineering

Amritanshu Agrawal, Tim Menzies, Wei Fu
Computer Science
North Carolina State University, USA
{aagrawa8, tjmenzie, wfu}@ncsu.edu

***Abstract*—Topic Modelling has been widely used to identify patterns in a corpus across various fields. Latent Dirichlet Allocation (LDA) is a widely used example of a topic modelling. 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 tens, 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 modelling algorithms is now an open and pressing issue. Even we can improvise the actual clusters to get better results.**

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

I. Introduction

In the 21st century, we have been seeing loads of data floating around in the world. These data are referred to as unstructured data. 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 leading big data algorithms for finding related topics within unstructured text (an area called topic modeling) is latent Dirichlet allocation (LDA) [7]. 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 in short time and get a good stability of topics. 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.

To tackle the problem of how to tune, We have seen that one of the simplest multiobjective optimizers (differential evolution [44]) works very well for tuning in major software analytics [11]. There has been success stories where DE has helped in tuning the parameters [11], [5], [8]. 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 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, a Non-Deterministic LDA can be Deterministic. 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**: *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.
- **RQ2**: *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 22 rather than 66 [13].
- **RQ3**: ??
- **RQ4**: *Is tuning easy?* We have seen that one of the simpler multiobjective optimizers (differential evolution [44]) 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?* [11] 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 Modelling have been used in various spheres of Software Engineering. There has been various work done in

Requirements engineering were it was necessary to analyze the text and come up with the important topics [4], [45], [29]. People have used topic modelling in prioritizing test cases, and identifying the importance of test cases [18], [52], [50]. 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 tuning 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" (more details can be found at link). After sorting by the citation count and discarding the non-SE papers, we read over this sample of 50 highly-cited SE text analytics papers. Most of these cited papers are since 2012. What we found in that sample was that few authors acknowledged the impact of tunings. Most papers in our sample did not adjust the "off-the-shelf" configuration of the LDA. Of the remaining papers the authors have acknowledged the fact of topics instabilities due to LDA.

A. Problems

In LDA, learning the various distributions is a problem of Bayesian inference. It also use the alternative inference techniques like Gibbs sampling [48]. These techniques gave rise to non deterministic ways which are responsible for the instability in the topic modelling. 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 [51], [41]. Online learning [20] 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 modelling in all their use cases [24], [39]. 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, 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.

The data can have many local maximas, which 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 Gibbs sampling [22]. This randomness affects perplexity variations, word and document ratios. There is a problem of finding the optimal number of clusters, different configuration parameters which 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 GibsLDA++ written in C++ and they observed the same instability [26], [46], [17]. Some papers mentioned about using Python implementation of LDA and observed the same instability [17]. LDA implemented in JAVA language produced the same incoherent topics [27], [19]. Recently, we observed this problem is in the Scikit-Learn version of LDA, implemented in Python, as well as Spark Mllib library.

All these recently [51], [41], [24], [39], [22], [26], [46], [17], [27], [19], [37], [28], [43], [25] 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. Solutions

1) Unsupervised: We found most of the papers mentioning about manually accessing the topics generated and then using them for further experiments [28], [27], [17]. Some made use of Amazon Mechanical Turk to create gold-standard coherence judgements [24].

Users have external knowledge regarding word correlation, which can be taken into account to improve the semantic coherence of topic modeling methods. SCLDA 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 [51]. The most common parameters in the LDA are number of clusters, number of iterations, document topic prior, and word topic prior $[k, n, \alpha, \beta]$.

The other solution is tuning the above mentioned parameters of LDA. Tuning the parameters and different configurations were used [41], [25], [37]. They [12], [46] 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 [11]

2) Supervised: We will not be using these methods as we don't have labels assigned to the datasets which we are using.

Labeled LDA [42], 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. Labeled LDA outperforms SVMs by more than 3 to 1. Incorporating domain knowledge using a novel **Dirichlet Forest** prior in a Latent Dirichlet Allocation framework [2]. 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** [3] is a scalable inference technique using stochastic gradient descent. **Quad-LDA** [36] is a framework in order to improve the coherence of the keywords per topic learned by LDA. **NMF-LDA** [49] 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) [21], 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** [9] 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.

C. Evaluation

To evaluate the topics coherence or the cluster stability using LDA, there has been number of evaluation measures proposed. An automatic method for estimating topic coherence based on pairwise pointwise mutual information (PMI) between the topic words. A direct approach, by asking people about topics, indirect approach by evaluating PMI [24], [39]. For good clusters, we want less cohesion (intra-distance) between data points in a cluster and large separation (inter) between 2 different clusters.

Perplexity 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 [22].

Panichella [41] checks 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 take median of all the scores generated by evaluating topic. It makes sense to us that to have a successful topic modelling, to define each topic you need top few terms which will completely define that topic.

D. 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 [7], alternative inference techniques use Gibbs sampling [16] and expectation propagation [33].

All these methods 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 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}. 1 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}.

Algorithm 1 Pseudocode for DE with Default Parameters

Input: $terms$, $Data$
Output: $Final\_Score$

1: **function** LDASCORE($term$, $Data$)
2: $Score \leftarrow \emptyset$
3: **for** $j = 0$ to 10 **do**
4: **for** $i = 0$ to 10 **do**
5: $data \leftarrow$ **shuffle**($Data$) ▷ Data is in the form of tf scores of each word per document. Shuffling is done in order to induce maximum variance among different runs of LDA
6: $Topics$.append(lda($k = l[0]$,$\alpha = l[1]$,$\beta = l[2]$)) ▷ It is a list of list which contains topics from all the different runs.
7: **end for**
8: $Score$.append($Overlap$(Topics, $term$,$l[0]$))
9: **end for**
10: $Final\_Score \leftarrow$ median(Score)
11: **return** $Final\_Score$
12: **end function**

E. 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 [10], [31]; various genetic algorithms [15] augmented by techniques such as differential evolution [44], tabu search and scatter search [14], [6], [34], [35]; particle swarm optimization [40]; numerous decomposition approaches that use heuristics to decompose the total space into small problems, then apply a response surface methods [23], [53]. 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 advocates for differential evolution than SA. For example, Vesterstrom and Thomsen [47] found DE to be competitive with particle swarm optimization and other GAs. DEs have been applied before for parameter tuning (e.g. see [38], [8], [11]) 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 1. 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. From the table I, you can see that after 300 evaluations, there was no improvement

in the stability scores across all the datasets and it stayed flat. Each candidate solution in the Population is a pair of (Tunings, Scores). Tunings are selected from Table II 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.

Datasets\Evaluations	100	200	300	400
PitsA	0.8	0.8	0.9	0.9
PitsB	0.8	0.9	0.9	0.8
PitsC	0.7	0.8	0.9	0.9
PitsD	0.7	0.8	0.9	0.8
PitsE	0.7	0.8	0.9	0.9
PitsF	0.8	0.8	0.9	0.8
Ciemap	0.7	0.8	0.9	0.9
Stackoverflow	0.7	0.8	0.9	0.9

Table I. EVALUATIONS VS STABILITY SCORES

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 can be found in the table III. The datasets used will be:-

- **PITS**: This is a text mining data set generated from project and issue tracking system (PITS) reports [30], [32]. 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].

[1]http://openscience.us/repo/issues

[2]http://data.stackexchange.com

[3]http://blog.stackoverflow.com/tags/cc-wiki-dump/

Algorithm 2 Pseudocode for DE with a constant number of evaluations

Input: $np = 10, f = 0.7, cr = 0 : 3, iter = 3, Goal \in \{Data, term, ...\}$
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)   ▷ Data is in the form of
   tf scores of each word per document. Shuffling is done in
   order to induce maximum variance among different runs
   of LDA
27:        Topics.append(lda(k = l[0],α = l[1],β = l[2]))   ▷
   It is a list of list which contains topics from all the different
   runs.
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
```

- **Citemap**: A citation, also called a reference, uniquely

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

Table II. LIST OF PARAMETERS TUNED BY THIS PAPER

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 III. STATISTICS ON THE DATASETS

IV. EXPERIMENTAL RESULTS

A. RQ1: Does tuning improve the stability scores of LDA?

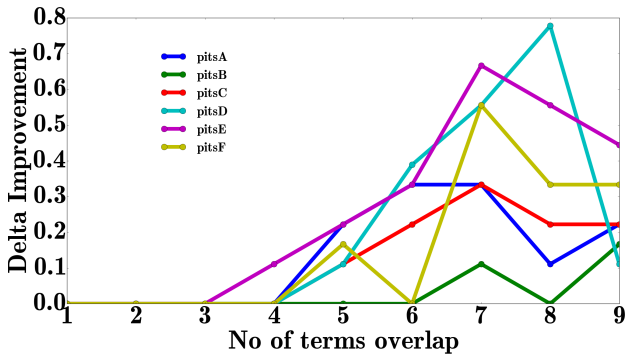

Figure 1. Terms vs Delta Improvement between tuning and default parameters

In figure 1, x-axis is number of terms overlap. Y-axis is 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 the different datasets. The figure clearly shows that the answer to RQ1 is "yes" - tuning has a positive effect on stability scores. Tuning either helped it or reamined the same but it never had any negative effect. We also recommend that we shouldn't list more than 6 terms per topic, as stability starts dropping.

B. RQ2: Does different data need different configurations to make LDA stable? Does it change some predefined parameter values of lda?

We can see these results in here. [5] From the spreadsheet, we can see that for different datasets, with different terms overlap, we are getting different median, and its IQR. This meant that we are getting different sets of configurations which are making the topic generated more constant. And if looked at R18, we are seeing 0 IQR, that means tuning helped us to found only 1 set of configurations which got us to that Stable score.

The other results which we found is in the citemap datasets. If we look from R56-64, we see that we are getting good stable scores within 30 topic size. But in the [13] paper, they found that we will need a topic size of 66. Definitely, it can change predefined parameters of LDA.

C. RQ3: ???

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, and we see that it didn't change much.

In Figure 2, 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.)

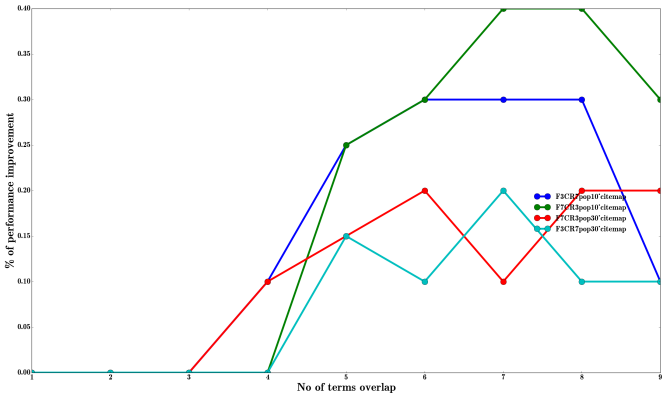

Figure 2. Terms vs Delta Improvement using Different settings of DE

E. RQ5: Is tuning impractically slow?

From the table I, we already saw that we just need 300 evaluations. Using 300 evaluations, from figure 3, and figure 4, we can see that tuning runtimes is only about 5 times the runtimes without tuning. Figure 3 is with gibbs implementation in Python, and figure 4 is with VEM implementation in python.

[4] https://github.com/ai-se/citemap/blob/master/citemap.csv
[5] https://goo.gl/Nin5pV

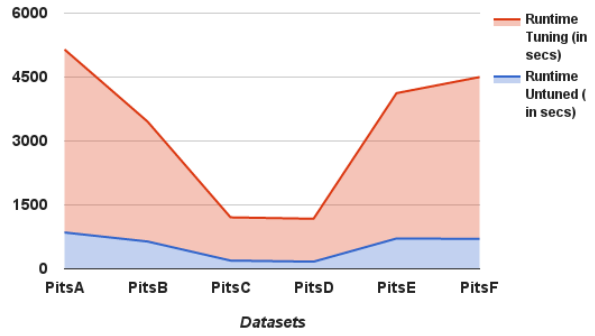

Figure 3. Datasets vs Runtimes with gibbs implementation using python

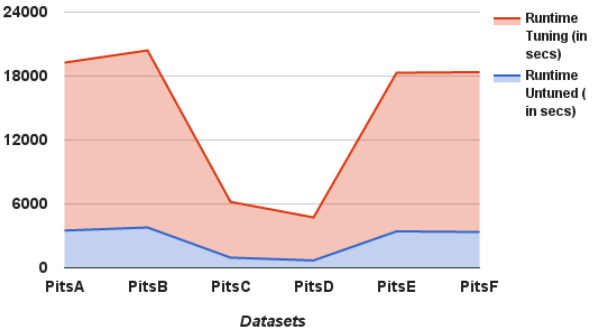

Figure 4. Datasets vs Runtimes with VEM implementation using python

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

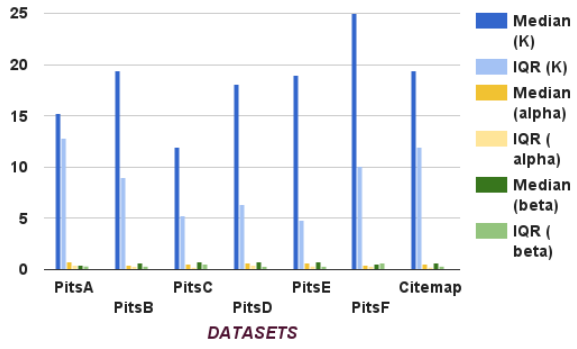

Figure 5. Datasets vs Parameters variation

In Figure 5, we show how tuning selects the different ranges of values of parameters. This result is for 5 term overlap, with VEM implementaion in Python. We observed the similar trend for other term overlaps and other sampling methods. We can easily see that for different datasets, we need different parameters. Such large IQRs show that we get quite varied ranges of parameters. For better idea, please look at this spreadsheet [6]. 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.

V. Threats to Validity

Is it a quirk of the implementation? This instability problem remained the same across any implementation of LDA. We showed the same improvement across all the implementations.

Is it a quirk of the sampling method used? In Figure 6, 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 implementation.

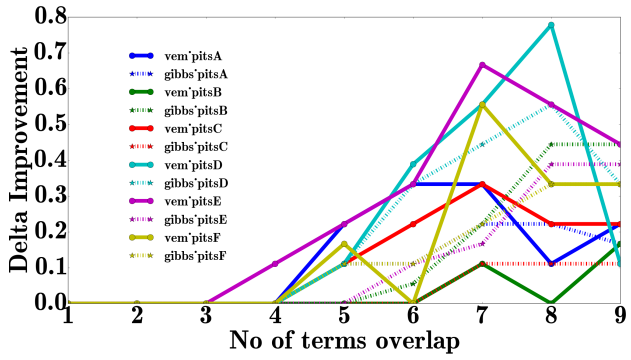

Figure 6. GIBBS vs VEM

VI. Conclusion

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

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

[6] https://goo.gl/Nin5pV

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

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

[52] 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.

[53] 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.