Responses to reviewers

Please see our replies to reviewer comments, below. Note that anything in *italics* is from the review team.

A. Comment from Editor:

Both reviewers consider the research of great interest. LDA is a widely used techniques, and studying and improving its performance is a value research topic.

Thank you for those kind words.

Both reviewers raised a number of substantial concerns that need to be looked into. Mainly they are related to

- *More transparently differentiate from existing and closely related work.*
  As recomended by the reviewers, we now compare our proposed method to prior work that used a genetic algorithm (by Panichella, Dit, Oliveto, Di Penta, Poshyvanyk, De Lucia, et al. [4]). We find our method runs orders of magnitude faster than theirs (see Figure 19) and that their method result in worse topics stability (see Figure 18).

- *More precise and detailed description of LDADA and its evaluation.*
  As per the the feedback from reviewers, we have provided a more precised and detailed description on LDADE (see Section 4.3).

There is one matter that the coordinating editor might need to comment on. Review A.2 comments that "There is a need to expand the experiments to compare LDA and LDADE on realistic software engineering tasks". By this, we think, the reviewer means running more experiments on supervised learning tasks where there is some target variable that should be predicted for. In reply, we note that:

- We do run some experiments on supervised tasks (see Section 5.3 and Figures 8, 9);
- But, more importantly, as we show in our literature review (section 2.1 and Table 2), most of the usages of LDA is for "unsupervised tasks" where LDA is used to find a structure to help humans navigate around and gain some insight to the space within software artifacts. In fact, by our count (in Table 2). $23/28 = 82\%$ of the applications of LDA in SE are of this unsupervised nature.

Now we believe that when using LDA for such navigation purposes that unstable topics would be very misleading (see our case study on that in section 2.2 and Table 3; also seee our notes on that point in A.1) so we think our paper does address a "realistic task". Nevertheless, this is a matter that needs to be decided by the co-ordinating editor.

A. Reviewer 1

Topic modeling has been used in many software engineering papers. The paper shows that topic modeling, especially LDA, suffers from order effects – which is new and interesting. Depending on the order data is presented to LDA, it produces a different model. Search-based software engineering can be used to tune LDA parameters so that order effect is minimized. LDADE (LDA mixed with Differential Evolution) is presented. It is shown to improve standard LDA for various settings including classification of StackExchange websites into relevant and non-relevant documents – which is good.

Thank you for those kind words.

A.1. Very closely related work exists:

A prior work has demonstrated suboptimal performance of topic modeling, in particular LDA, when default parameters are used, and proposed a solution that addresses the problem using search-based software engineering [4]. Thus, the novelty of the work seems limited. The paper does not describe why LDADE is better than Panichella et al.'s work. Panichella et al.'s work (LDA-GA) should have been prominently highlighted in the introduction of the paper and a short discussion should be given as to why another search-based solution is needed.

Thank you for making us look again at that prior work. We have updated the introduction section to highlight the difference between LDA-GA and LDADE. Also, we have compared the results of LDADE against LDA-GA in Section 5.8. We note that Panichella et al. did not consider the problem of order effects. Also, their Genetic Algorithm method is much slower than our approach. We observed that LDADE gets better stable results (see Figure 18) and is much faster (see Figure 19)[5]. Now we stress that Panichella et al.'s *did not* optimize for stability. As shown in Figure 18, the topics found by Panichella et al.'s methods are far less stable than ours, This is important since, as Mark Harmon [86] said

> "In some software engineering applications, solution robustness may be as important as solution functionality. For example, it may be better to locate an area of the search space that is rich in fit solutions, rather than identifying an even better solution that is surrounded by a set of far less fit solutions."
>
> "Hitherto, research on SBSE has tended to focus on the production of the fittest possible results. However, many application areas require solutions in a search space that may be subject to change. This makes robustness a natural second order property to which the research community could and should turn its attention."

[5] Just to answer your next question, we think there is much value in speeding up widely used algorithms. A summary of our arguments for this point appears in section 2.1 of https://arxiv.org/abs/1703.00133 and pages 7 to 35 of http://tiny.cc/tim17vic. Our call is that material is not needed to be summarized here (but would take the reviewer's advice here).

Why is optimizing for stability important in the particular context of this research? For tasks such as those explored by Panichella et al.'s, it does not matter since the output LDA models are not inspected by humans. Rather, they are under-the-hood intermediaries for further analysis.

However, for *unsupervised tasks*, the situation is very different. This distinction between *unsupervised tasks* and *supervised tasks* is significant since, as shown in Table 2, 23 of the recent highly-cited LDA studies found in the SE literature are for *unsupervised tasks*. And as shown by example, (please see Table 3) that topic instability can be a major impediment to knowledge discovery in SE. Note that for *unsupervised tasks*, the LDA topics are the primary product used and shared by humans to justify certain conclusions about a domain. In such unsupervised tasks, humans have no goal, no supervised direction, rather, they are reading the topics as knowledge discovery aids to better explore some domain (or they are used by clustering algorithms to determine groupings of software artifacts). In either case, instabilities in the topics will result in different clustering and different justifications. Hence, this paper.

A.2.

The experiments need to be improved in the following ways/ First, LDA has been used to help many realistic software engineering tasks (for example, tasks considered by papers included in Table 2. There is a need to expand the experiments to compare LDA and LDADE on those realistic software engineering tasks. It is unclear if the task considered in the experiments (Section 5.3) is realistic (why categorizing StackExchange data into relevant and non relevant categories useful?). More than one tasks should have been considered (similar like Panichella et al.'s work).

Our goal in this revision was to be as responsive as possible to your suggestions (as as you can see below, in A.4, A.5 and A.7, we were able to implement much of your advice.)

But as to applying this to "more realistic SE task", we might have a different perspective on what is a "valid" SE task. We say this since it sounds like you are saying the unsupervised tasks conducted by 23 of 28 recent highly cited LDA papers are not "realistic"? And that we should evaluate this paper only via the supervised tasks seen in 4 of the 28 papers? That is not our view, please see the notes above in A.1 on the need for stability in unsupervised SE tasks.

A.3.

Second, there is a need to compare Panichella's work (LDAGA) with LDADE on some realistic software engineering tasks. It is unclear if LDADE is better than LDAGA.

We agree on your suggestion and now we have provided that comparison. Please see section 5.8. We find that LDA-GA is much slower than our approach. Also, it does not generate anywhere the stable topics we do.

A.4.

The results reported in Section 5.3 are questionable. Why not use LDADE to tune all 3 parameters of LDA (k, alpha, beta)? f k is fixed and alpha and beta are fixed too, what parameter(s) is tuned then?

That is a great catch, that was our mistake in the text. We did not tune separately. We tuned for all 3 at once. We have corrected the text to make that clearer. See Section 5.3.

A.5.

Why untuned (with $k = 10$) is compared with tuned (with k = 20, 40, 80, and 200)? Shouldn't they be compared under the same setting of k?

Thank you for providing us this insight to present our results in a better way. We have updated the Section 5.3 to show the results under the same settings of k. This result shows that, if we keep k fixed, then tuning α and β produces somewhat better improvement. As shown in 8, for some datasets $k = 39$ matters and for some $k = 82$ (and LDADE now finds this kind of settings automatically).

A.6. Is k-fold cross validation used? Is training data kept separate from test data (including during the tuning phase)?

Thank you for pointing out this, we did perform a 5-fold stratified cross validation but missed out onto include that in the paper. Now this detail is added in Section 5.3.

A.7.

Why F2 rather than F1 or F1/2? Is the difference statistically significant?

Our industrial partners prefer F2 but you right, we should have used more standard measures. Hence, we now include the F2 and F1 results in Section 5.3.

As to statistical tests, as said at the end of Section 5.3, the tuned results are indeed different and better to the untuned.

A.8.

Minor issues:

Why need to evaluate across different platforms (Linux, Macintosh)? Wouldn't the platform be irrelevant?

That is our mistake in text, we wanted to say that we performed the experiments on any kind of Linux platform.

- *Related work is revised $->$ is reviewed*
- *S2 $->$ Section 2 (similarly for S1, S3, and so on)*
- *The period of time considered for drawing Table 1 needs to be mentioned.*
- *Should Table headings be at the top of tables? Need to confirm with IST guidelines.*
- *Reference 69 is incomplete.*

Thank you for pointing out these, we have fixed them now.

B. Reviewer 2

SUMMARY: The authors empirically analyze the instability of LDA and the importance of its parameters' tuning. Then the authors present LDADE, a search-based software engineering tool that tunes LDA parameters using DE (Differential Evolution). An empirical study indicate that by using LDADE, the topic modeling technique is much more stable and reliable.

EVALUATION: This is a very interesting paper. It is in general well-written and easy to follow. I really like the goal of the paper. Having worked with LDA I totally agree that using the technique as a "black-box" is not recommended. So, the idea of using LDADE to (i) improve the stability of LDA and (ii) reduce the threats that could affect the validity of the results achieved due to an incorrect configuration of the technique is really important. Thus, I think that the paper has a great potential.

Thank you for those kind words

B.1.

The first issue is related to the description of LDADE. I have to admit that Section 4.3 of the paper is quite difficult to follow. First of all, I strongly suggest to the authors to provide a "bird-eye-view" of the approach before providing the details. Which is the output of LDADE? How can I use such an output? If I understood correctly, the output of LDADE is just a set of topics (similar to the output of LDA). Is this correct? If so, the authors should explicit mention this. Also, an usage scenario of LDADE could be worthwhile.

We carefully considered your suggestions here and also the follow-up comments, and this made us to reformat the Algorithm 2 and provide it with detailed description in the text of Section 4.3. This reformatting makes our approach of LDADE more clearer. To answer your questions, the output of LDADE is $\mathfrak{R}_n$ and the optimal configuration for a dataset to be used with LDA for further SE task. For usage, please see A.2 and Table 2. Since so many people have been using unstable LDA, LDADE could provide stability and get better results.

B.2.

In addition, I did not understand at all when and how LDADE varies the parameters of LDA (k, alpha and beta). LDA in LDADE is invoked through the function ldascore. However, in such a function k, alpha and beta are set to default values.

Sorry about that we did not clear this in the text. *Ldascore* function is invoked which is same as Algorithm 1 but this time rather than having default parameters as input, *ldascore* takes the tuned parameters generated from DE. We have made this more clearer in Section 4.3.

B.3.

I appreciated that the authors reported the pseudo-code of LDADE. However, I think that a much deeper description of each step is required. The authors should explain each step of the algorithm and more important should define each function/variable of the pseudo-code. For instance, Data is never instantiated in Algorithm 2 (probably because it is an input?). Or, which is l[0] in Algorithm 1. Another imprecision is related to the function ldascore. From Algorithm 1, ldascore takes as input n and Data. In Algorithm 2 ldascore is called at line 11 passing as parameter Si (a population?) and at line 12 Si, n and Data. Also, Cur_Gen is used as a matrix. However, when calling on Cur_Gen the method "add", four parameters are passed. I understand that it is just a pseudo-code. However, a much higher level of formality is required.

Thank you for pointing out those. We have fixed the errors of Algorithm 2 in Section 4.3.

B.4.

I still have some doubts about how DE is used. Specifically, I would like to see much more details on the technique (to make the paper self-contained) and (much important) the design of the approach. For instance, which is the encoding of each solution? Which are the operators used to evolve the solutions? Which is the quality/fitness function? Looking at the pseudo-code, it seems that the quality function is represented by ldascore. If so, why scores are encoded in the solution?

We have fixed that by providing a better explanation of Algorithm 2 (Figure 5) in section 4.3. We have also included another table 7 which provides an overview of LDADE algorithm. The quality/fitness function is represented by *ldascore*. The encoding of $\mathfrak{R}_n$ score in the solution is shown in pseudocode just to keep track of the values but it is not used to drive our *ldascore* function.

B.5.

The authors provide empirical evidence that LDA used with default configuration parameters is not stable. What about if LDA is configured properly - for instance by using LDA-GA? In other words, which are the results achieved if in the algorithm 1 instead of using LDA, the authors try to use LDA-GA?

Please see Section 5.8 for a comparison with LDA-GA. In summary, we are order of magnitude faster and our solutions are more stable.

B.6.

Could the instability be due to an incorrect configuration of the LDA parameters? This is a critical point that should be addressed in the paper.

You are correct, this is the whole point of this paper. We need to find correct LDA configurations, where "correct" means "achieved stabler results". We show in this paper that the DE algorithm explored here can achieve this, very quickly.

B.7.

Turning to the empirical evaluation, the main problem here is related to the lack of a deeper analysis of the results achieved. For instance, looking at Figure 6 it seems that the results achieved on PitsC are quite stable. Why? Did the authors have some clues on why on this particular dataset the untuned LDA provides quite acceptable results? Here a qualitative analysis of the results achieved could be worthwhile.

Sorry, I believe you mistook it for PitsD. We are seeing the largest improvements in PitsD, that is because what kind of data it represents. About 92% of the data samples report the severity of level 3. All the other Pits Datasets have mixed samples of severity level. When data is less skewed LDADE achieves the highest improvement which was not achieved when just LDA was being used. So, this makes it highly unimaginable to use just LDA for highly skewed data. This also emphasizes the use of LDADE more. And we have included the insight on the quality of data in Section 5.2.

B.8.

A qualitative analysis could be also useful to better explain why LDADE provides much more benefits on VEM than Gibbs. Note that this result also confirms the results achieved in the literature about the higher stability of Gibbs as compared to other implementation of LDA.

We have updated this qualitative analysis in Section 5.5. VEM only gets you to a local optimum that depends on initialization and other factors in the optimization problem [82]. On the other hand a good sampling algorithm like Gibbs sampling can achieve the global optimum. In practice because of finite number of samples that are used, sampling can give you a sub-optimal solution. LDADE works as another kind of sampler which selects good priors at the initialization and can achieve better sub-optimal solutions, which is why LDADE has better benefits on VEM.

B.9.

Very interesting is the analysis of the Delta score. However, in some cases the improvement in terms of stability is not so high. Also here could be worthwhile to discuss in details such cases. In addition, is there any statistical significant difference between the achieved scores? An analysis of the results supported by statistical tests could strengthen the findings.

To explain why stability is achieved better in some cases and not in some cases, please look at our explanation in B.7 and in Section 5.2. And we have also explained whether our results achieved are statistically significant or not in Section 5.3.