Measuring LDA Topic Stability from Clusters of Replicated Runs

Mika V. Mantyla
M3S, University of Oulu, Finland
mika.mantyla@oulu.fi

Maelick Claes
M3S, University of Oulu, Finland
maelick.claes@oulu.fi

Umar Farooq
M3S, University of Oulu, Finland
umar.farooq@oulu.fi

ABSTRACT

Background: Unstructured and textual data is increasing rapidly and Latent Dirichlet Allocation (LDA) topic modeling is a popular data analysis methods for it. Past work suggests that instability of LDA topics may lead to systematic errors. **Aim:** We propose a method that relies on replicated LDA runs, clustering, and providing a stability metric for the topics. **Method:** We generate k LDA topics and replicate this process n times resulting in n*k topics. Then we use K-medioids to cluster the n*k topics to k clusters. The k clusters now represent the original LDA topics and we present them like normal LDA topics showing the ten most probable words. For the clusters, we try multiple stability metrics, out of which we recommend Rank-Biased Overlap, showing the stability of the topics inside the clusters. **Results:** We provide an initial validation where our method is used for 270,000 Mozilla Firefox commit messages with k=20 and n=20. We show how our topic stability metrics are related to the contents of the topics. **Conclusions:** Advances in text mining enable us to analyze large masses of text in software engineering but non-deterministic algorithms, such as LDA, may lead to unreplicable conclusions. Our approach makes LDA stability transparent and is also complementary rather than alternative to many prior works that focus on LDA parameter tuning.

CCS CONCEPTS

• **General and reference** → *Cross-computing tools and techniques*; *Empirical studies*; • **Computing methodologies** → *Natural language processing*; • **Software and its engineering**;

KEYWORDS

Latent Dirichlet Allocation, Replication, Stability, Similarity, Clustering, Commit messages, Rank-Biased Overlap

ACM Reference Format:
Mika V. Mantyla, Maelick Claes, and Umar Farooq. 2018. Measuring LDA Topic Stability from Clusters of Replicated Runs. In *ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '18), October 11–12, 2018, Oulu, Finland.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3239235.3267435

1 INTRODUCTION

Latent Dirichlet Allocation (LDA) is a topic modeling technique for textual data [5] that is widely applied in software engineering [1–4, 6, 10, 11, 14–16, 19, 24, 25] for different tasks such as requirements engineering [15], software architecture [10], source code analysis [9], defect reports [16], testing [14] and to bibliometric analysis of software engineering literature [11, 22]. A survey on topic modelling in software engineering has been conducted [24] and a book titled "The art and science of analyzing software data" [4] devoted a chapter for LDA analysis [6]. Many sources give methodological guidance on how to apply LDA topic modeling in software engineering [1, 3, 19]. Given all this, we think it is fair to say that LDA topic modelling is a relevant data analysis technique in empirical software engineering research.

The quality of the resulting topic model can be evaluated with multiple metrics some inspired by mathematics such as the posterior probability of the topic model given the data [13], perplexity of measure in the test data [13], or Silhouette coefficient of resulting topics [19]. Other target metrics are based on empirical observations such as coherence, which measures topic model quality using word co-occurrences in publicly available texts [23], or stability which investigates similarity of topics between different runs [1].

Recently, Agrawal et al. [1] published a paper titled "What is wrong with topic modeling? And how to fix it using search-based software engineering", where they claimed that the instability of topics is one major shortcoming of this technique. Indeed, studies could result in wrong conclusions if the results are based on instable topics. They proposed using a differential evolution search algorithm to find the input parameters which maximize the topic model stability measured as the similarity of topics between multiple runs. This method reduces instability by finding optimal input parameter settings, but only uses the result of one LDA run which can still have some instable topics.

In this paper, we address the stability of topic models, but rather than optimizing input parameters we propose making stability (or instability) transparent to the user. We achieve this by performing replicated runs of LDA topic modeling and clustering the results. Subsequently, we present the clustering results as any topic modeling results by adding an additional metric of stability. Our method is not an alternative to the ones presented by Agrawal et al. [1] but additive. Thus, we may use both methods at the same time. However, a benefit of our method is that the topic models may also be optimized towards other targets than stability. For example, a user may choose to optimize the topic model input parameters for coherence [23] or perplexity and still use our approach in the end to provide information about topic stability.

This paper is structured as follows. In Section 2 first we present LDA in more detail and then our method for making the stability transparent. In Section 3 we demonstrate our results while Section 4 provides conclusions and discusses future improvements.

2 METHOD

2.1 LDA Topic Modelling

LDA (Latent Dirichlet Allocation) is a soft clustering algorithm that is ideal for text [5] but also for other purposes such as genetics [21]

ESEM '18, October 11–12, 2018, Oulu, Finland

where a relationship between a gene and a genotype can be considered similar to a relationship between a word and a document. Given a set of documents, LDA models from what topics this set of documents may have been created from. As opposed to hard clustering where each document would be assigned to a single topic only, LDA soft clustering assigns each document a list of topics and probabilities for the topics. A topic in LDA is a collection of words and their probability estimates for each topic. In order to summarize, after running LDA we have the following.

- For all documents m there is a vector θ which is the topic distribution for that document.
- For all topics k there is a vector ϕ which is the word distribution for that topic.

Before topic generation, LDA requires that we set the input parameters such as the number of topics k, and hyper priors α and β. Past work in software engineering has used different techniques to find optimal input parameters such as genetic algorithms [19] or differential evolution [1]. As pointed out in Section 1, what is optimal can be measured with many metrics such as perplexity [13], stability [1], or coherence [23].

The stability of a topic model can be defined as the model's ability to replicate its solutions [8]. Instability (the lack of stability) is caused by the non-deterministic nature of Monte-Carlo simulation that is part of the LDA algorithm [1]. Past work has shown different stability measures and how to optimize the input parameters to provide a stable topic model [1, 8, 12]. We think using the results of a single LDA run, whether optimized for stability or not, is dangerous as perfect stability is impossible to reach. The next section shows a method that can be used to make more informed decisions.

2.2 Transparent Stability

To make LDA topic stability transparent, we suggest performing replicated LDA runs, clustering the topics, and giving a measure of stability. R-code of our approach is available[1]. Section 2.2.1, describes the approach, Section 2.2.2 explains how to show the clusters, and finally Section 2.2.3 presents different stability measures.

2.2.1 Clustering LDA topics. As previously described, an LDA topic is a list of words with the probabilities of each word appearing in that topic. When we cluster replicated LDA runs we have n replicated runs, and each run contains k number of topics. Therefore the total number of topics is $t = nk$. Our word list is represented by w where ϕ is the vector of word distribution for each topic. Thus, we have a topic-word matrix T with dimensions $t \times w$ that we want to cluster back to k clusters as k was the number of topics in our LDA setting. We wanted to take an advantage of the word embeddings produced by GloVE [20] where our entire word list w, which has typically thousands of words, is converted to a word vector space with typically 200-400 elements. It has been shown that in this word vector space, semantically similar words appear close to one another [20] and we have previously use it for searching software engineering specific synonyms [17]. Thus, we form a word vector space with w words and v vectors as matrix V with dimensions $(w \times v)$. Then we convert our topic-word matrix T $(t \times w)$ to topic-vector matrix W $(t \times v)$ via matrix multiplication $T(t \times w)V(w \times v)$. An additional benefit is that the W $(t \times v)$ matrix is much smaller than the T $(t \times w)$ matrix, resulting in faster clustering. Finally, we use K-medioids clustering to cluster our topics W $(t \times v)$ to k topics.

2.2.2 Showing Clustered LDA topics. We want to deviate as little as possible from standard LDA topic modeling when presenting the results. We form the list of top ten words for each cluster, in LDA each topic is typically represented by the top ten words. To compute the top 10 words, for a cluster that has multiple topics, we sum up word distributions ϕ for all topics in a particular cluster and the top ten words are the ones with the highest sums.

2.2.3 Topic stability measure. At this point, our results would appear like any LDA topic model to the user. However, as we want to give the user transparency to topic stability, we need to add a measure describing topic stability. Obviously, user can investigate each cluster in detail but the topic stability measure can help the user to focus on specific clusters. We propose several measures of topic stability, i.e. whether a set of topics are actually about the same content. When two topics contain the same top 10 words in the same order, then we can think that they are exactly about the same content and should result in a maximum score. On the other hand, any deviations from this should result in a lower score.

First, *Silhouette* is a well-established measure for cluster validation that considers both how similar each object is to its own cluster (cohesion) and how different it is to other clusters (separation). It has been used in LDA optimization before [18, 19]. The average silhouette is produced by the K-medioids clustering performed earlier. However, the cluster separation is not interesting for the user as the user mainly cares about whether a particular cluster has similar elements, i.e. high stability. Furthermore, this measure is based on the absolute values of word probabilities rather than the ranks what are presented to the user.

Second, to model whether the same top words are present and that they are in the same order, we can use *Spearman* rank correlation between the top words of any two topics. Any words that are present in the top word list of one topic, but not the other, are assigned the lowest rank in the other topic. A problem occurs if two topics have the same words but in reverse order, the rank correlation between the topics would be -1 while one would still consider these two topics somewhat similar due to the same top words. Another anomaly is that for two topics with no intersecting top ten words, we would get a better Spearman correlation value than -1 (-0.86).

Third, we can measure *Jaccard* similarity between the top words of any two topics. Extended Jaccard measures have been used in LDA stability task optimization [1, 12]. When two topics have all the same top words, the Jaccard similarity would be 1. On the other hand, the worst case (when all the top words are different) would result in a Jaccard similarity of 0. The undesirable property of the Jaccard similarity is that any variations in ordering would not be reflected in the measure. Obviously, a measure of topic stability should take into account both differences in word intersection and rank of two topics. Luckily, a paper published in 2010 has presented such a measure known as rank-biased overlap (*RBO*) [26] that seems

[1] https://github.com/M3SOulu/Measuring-LDA-Topic-Stability

ideal for LDA topic comparisons. We use the extrapolated version of RBO from R-package Bioconductor[2] that is computed as follows

$$RBO_{\text{EXT}}(T1, T2, p, d) = \frac{X_d}{d} \cdot p^d + \frac{1-p}{p} \sum_{i=1}^{d} \frac{X_i}{i} \cdot p^i$$

$T1$ and $T2$ are two ranked lists and in our case they are two topics represented by their top words, d is the evaluation depth and in our case it is 10 as we wish to compare top ten words, X_d is the intersection of $T1$ and $T2$ at depth d. RBO ranges between 0 and 1. RBO is zero when none of the top words are the same and one when all top words are the same and in the same order. The effect of order, i.e. top-weightedness, is controlled by p. When p is 1, the order has no effect and only the intersection is considered. Smaller p gives more weight to order of words. We set p to 0.9 as such value is suggested by RBO authors [26] and as it seems to offer good balance on impact of different ranks of the top ten words. For an illustration let us consider two topics with the same top ten words but in reverse order. This pair would result in the opposite of Spearman correlation (-1) and Jaccard similarity (1) while the RBO (p=0.9) has value close to the middle of its 0 to 1 range (0.51).

3 RESULTS

3.1 Data, Parameter Tuning, and LDA Runs

We demonstrate our approach on 271,236 commit messages from Mozilla Firefox. We did some minimal preprocessing by excluding the words appearing fewer than ten times and the ones that appeared in 30% or more documents. Additionally, we removed common stopwords and individuals’ names that appeared as Firefox commit messages are also used for assigning reviewers. We consider this preprocessing very conservative.

We used a fixed number of topics ($k = 20$) and differential evolution algorithm (population=20, $CR = 0.5$, $F = 0.8$, iterations = 10) from the R-package *DEoptim* to optimize for hyper parameters α (0.167) and β (0.076) with the target of maximizing perplexity in the holdout set. As previously said, our approach is not affected by hyper parameter tuning so future works may use any training targets or algorithms they see appropriate.

We performed 20 LDA run replications with the R-package *text2vec* (1,000 iterations). This package includes very efficient LDA implementation (WarpLDA [7]) and training a single model with our data takes less than 2 minutes with a laptop computer.

3.2 Stable and Unstable Topics

Table 1 shows the best, worst, and median (eleventh) clusters in terms of LDA topic stability out of the 20 clusters ranked with the RBO stability metric. All four stability metrics are highly correlated with each other in our data and the Pearson correlation range is between 0.91 and 0.98. To demonstrate the details, Tables 2 to 4 shows five topics from the best, worst and median cluster with their five top words. All computations of Table 1 were performed with all topics and the top 10 words but Tables 2 to 4 only shows a smaller sample to keep the paper within the four page limit.

Table 1 shows that for the cluster with the best topic stability, all topics have all the words nearly in the same order as the average Spearman rank correlation is very close to 0.95. The average Jaccard similarity in topics is only 0.813, however, we need to remember that if two topics differ by one word of the top ten words this already results in Jaccard similarity of 0.818 (9/11). Manual inspection of the details in Table 2 shows that for the five topics shown, the top 5 words appear in the same order. We can further confirm that this is true for all topics in this cluster. Deviations in word rank and occurrence exist in words in places 6 to 10. This topic is about commits that revert (back out) previous changes.

Table 1: Best (most stable), median and worst topic clusters

	Best	Median	Worst
Silhouette	0.954	0.618	0.092
Spearman	0.953	0.255	-0.295
Jaccard	0.813	0.462	0.289
RBO	0.948	0.605	0.335
Topics	20	22	18
1	backed	build	update
2	changeset	files	add
3	tree	use	https
4	closed	builds	style
5	backout	file	servo
6	bustage	support	changes
7	failures	add	source
8	changesets	update	patch
9	build	version	css
10	mochitest	windows	support

Table 2: Topics of the best topic cluster

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
1	backed	backed	backed	backed	backed
2	changeset	changeset	changeset	changeset	changeset
3	tree	tree	tree	tree	tree
4	closed	closed	closed	closed	closed
5	backout	backout	backout	backout	backout

Table 3: Topics of the median topic cluster

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
1	build	build	build	build	build
2	files	files	files	files	files
3	builds	update	file	file	use
4	file	version	use	use	file
5	use	builds	builds	builds	builds

As expected, the cluster with median stability has a lower topic stability than the best cluster in all stability metrics. We can also notice that number of topics in this cluster is higher than 20, i.e. the number of replicated runs we performed. This means that from a single LDA run, more than one topic is part of this cluster. Our K-medioids clustering took all 400 (20*20) topics as input and we did not try to force it to pick one topic from one LDA run to each cluster. This is something we may want to investigate in the future. Table 3 shows that this cluster is about build file usage or updates.

Table 4 shows detailed sample of the worst cluster and we see variations in the word order and occurrence. It appears that this

[2] https://rdrr.io/bioc/gespeR/man/rbo.html

Table 4: Topics of the worst topic cluster

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
1	update	update	style	update	style
2	add	https	animation	fix	css
3	version	changes	element	patch	text
4	support	source	patch	x	servo
5	changes	add	https	changes	https

topic is about updates, additions, fixes and changes. Since all these are very common words in a version control context, it is hard to make a meaningful interpretation of the topics in this cluster.

4 CONCLUSIONS AND FUTURE WORK

Past work in software engineering [1] and machine learning [12] point out that LDA instability may lead to incorrect conclusions and proposes input parameter optimization to alleviate the problem. This paper suggests performing replicated runs, clustering the results and measuring the topic stability. These approach are not alternative but additive. Our approach can be combined with any LDA optimization technique that relies on input parameter optimization. Finally, our approach shows topic stability by providing a metric of topic stability and allowing further investigation of the clusters when desired.

This paper presents multiple metrics of topic stability in Table 1 that are highly correlated with each other in our data set. Based on theoretical metric properties (see Section 2.2.3) we recommend using RBO [26]. In the future, one should empirically establish what p value setting of RBO metric most accurately matches the user expectation on topic stability as in this paper we only used the default ($p = 0.9$). We should also study how the topic clusters can be used in the downstream NLP tasks in software engineering. Furthermore, to demonstrate our idea we also made other design choices but didn't investigate their impact. For example, we considered only 20 replication runs which might be too little and we only generated 20 LDA topics for each run. We also clustered our topics in the word vector space produced by GLoVe as prior work suggested it would produce better results than clustering in word space. All these choices could be challenged.

Zeller's 2018 ICSE talk [27] has warned us about the dangers of adding complexity. Our approach adds complexity but eventually hides it behind an RBO metric showing the stability of each topic cluster. If the stability of topics is an issue, users of topics models should be made aware of it but with minimal added complexity as we have tried to do in this paper.

ACKNOWLEDGMENTS

This work has been supported by Academy of Finland grant 298020.

REFERENCES

[1] Amritanshu Agrawal, Wei Fu, and Tim Menzies. 2018. What is wrong with topic modeling? And how to fix it using search-based software engineering. *Information and Software Technology* 98 (2018), 74 – 88.

[2] Hazeline U Asuncion, Arthur U Asuncion, and Richard N Taylor. 2010. Software traceability with topic modeling. In *Proceedings of the 32nd ACM/IEEE international conference on Software Engineering-Volume 1*. ACM, 95–104.

[3] David Binkley, Daniel Heinz, Dawn Lawrie, and Justin Overfelt. 2014. Understanding LDA in source code analysis. In *Proceedings of the 22nd International Conference on Program Comprehension*. ACM, 26–36.

[4] Christian Bird, Tim Menzies, and Thomas Zimmermann. 2015. *The art and science of analyzing software data*. Elsevier.

[5] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[6] Joshua Charles Campbell, Abram Hindle, and Eleni Stroulia. 2016. Latent Dirichlet allocation: extracting topics from software engineering data. In *The art and science of analyzing software data*. Elsevier, 139–159.

[7] Jianfei Chen, Kaiwei Li, Jun Zhu, and Wenguang Chen. 2016. Warplda: a cache efficient o (1) algorithm for latent dirichlet allocation. *Proceedings of the VLDB Endowment* 9, 10 (2016), 744–755.

[8] Alta De Waal and Etienne Barnard. 2008. Evaluating topic models with stability. (2008).

[9] Bogdan Dit, Meghan Revelle, Malcom Gethers, and Denys Poshyvanyk. 2013. Feature location in source code: a taxonomy and survey. *Journal of software: Evolution and Process* 25, 1 (2013), 53–95.

[10] Joshua Garcia, Daniel Popescu, Chris Mattmann, Nenad Medvidovic, and Yuanfang Cai. 2011. Enhancing architectural recovery using concerns. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 552–555.

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

[12] Derek Greene, Derek O'Callaghan, and Pádraig Cunningham. 2014. How many topics? stability analysis for topic models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 498–513.

[13] Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences* 101, suppl 1 (2004), 5228–5235.

[14] Hadi Hemmati, Zhihan Fang, Mika V Mäntylä, and Bram Adams. 2017. Prioritizing manual test cases in rapid release environments. *Software Testing, Verification and Reliability* 27, 6 (2017).

[15] Abram Hindle, Christian Bird, Thomas Zimmermann, and Nachiappan Nagappan. 2012. Relating requirements to implementation via topic analysis: Do topics extracted from requirements make sense to managers and developers?. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*. IEEE, 243–252.

[16] Lucas Layman, Allen P Nikora, Joshua Meek, and Tim Menzies. 2016. Topic modeling of NASA space system problem reports: research in practice. In *Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on*. IEEE, 303–314.

[17] Mika V Mäntylä, Nicole Novielli, Filippo Lanubile, Maëlick Claes, and Miikka Kuutila. 2017. Bootstrapping a lexicon for emotional arousal in software engineering. In *Mining Software Repositories (MSR), 2017 IEEE/ACM 14th International Conference on*. IEEE, 198–202.

[18] Vineet Mehta, Rajmonda S Caceres, and Kevin M Carter. 2014. Evaluating topic quality using model clustering. In *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on*. IEEE, 178–185.

[19] Annibale Panichella, Bogdan Dit, Rocco Oliveto, Massimilano Di Penta, Denys Poshynanyk, and Andrea De Lucia. 2013. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *Software Engineering (ICSE), 2013 35th International Conference on*. IEEE, 522–531.

[20] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[21] Jonathan K Pritchard, Matthew Stephens, and Peter Donnelly. 2000. Inference of population structure using multilocus genotype data. *Genetics* 155, 2 (2000), 945–959.

[22] Paivi Raulamo-Jurvanen, Mika V Mantyla, and Vahid Garousi. 2015. Citation and Topic Analysis of the ESEM papers. In *Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE International Symposium on*. IEEE, 1–4.

[23] Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*. ACM, 399–408.

[24] Xiaobing Sun, Xiangyue Liu, Bin Li, Yucong Duan, Hui Yang, and Jiajun Hu. 2016. Exploring topic models in software engineering data analysis: A survey. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2016 17th IEEE/ACIS International Conference on*. 357–362.

[25] Stephen W Thomas, Bram Adams, Ahmed E Hassan, and Dorothea Blostein. 2010. Validating the use of topic models for software evolution. In *Source Code Analysis and Manipulation (SCAM), 2010 10th IEEE Working Conference on*. IEEE, 55–64.

[26] William Webber, Alistair Moffat, and Justin Zobel. 2010. A Similarity Measure for Indefinite Rankings. *ACM Trans. Inf. Syst.* 28, 4, Article 20 (Nov. 2010), 38 pages.

[27] Andreas Zeller. 2018. ICSE 2018 - Plenary Sessions - Andreas Zeller. https://www.youtube.com/watch?v=U5jLjcxnwfU.