Evaluating Probabilistic Topic Models for Bug Triaging Tasks

Daniel Atzberger*, Jonathan Schneider*,
Willy Scheibel, Matthias Trapp, and Jürgen Döllner

Hasso Plattner Institute, Digital Engineering Faculty, University of Potsdam

Abstract. During the software development process, occurring problems are collected and managed as bug reports using bug tracking systems. Usually, a bug report is specified by a title, a more detailed description, and additional categorical information, e.g., the affected component or the reporter. It is the task of the triage owner to assign open bug reports to developers with the required skills to fix them. However, the bug assignment task is time-consuming, especially in large software projects with many involved developers. This observation motivates using (semi-)automatic algorithms for assigning bugs to developers. Various approaches have been developed that rely on a machine learning model trained on historical bug reports. Thereby, the modeling of the textual components is mainly done using topic models, mainly Latent Dirichlet Allocation (LDA). Although different variants, inference techniques, and libraries for LDA exist and various hyperparameters can be specified, most works treat topic models as a black box without exploring them in detail. In this work, we extend a study of Atzberger and Schneider et al. on the use of the Author-Topic Model (ATM) for bug triaging tasks. We demonstrate the influence of the underlying topic model, the used library and inference techniques, and the hyperparameters on the bug triaging results. The results of our conducted experiments on a dataset from the *Mozilla Firefox* project provide guidelines for applying LDA for bug triaging tasks effectively.

Keywords: Bug Triaging · Topic Models · Latent Dirichlet Allocation · Inference Techniques.

1 Introduction

The complexity of modern software development processes requires the systematic coordination of the communication and work steps of the involved stakeholders. For example, changes to the source code are managed within a version control system, tasks are coordinated in issue tracking systems, or occurring errors are documented in bug tracking systems. Therefore, throughout the entire development process, data is generated and archived in the various repositories. The

* The first two authors contributed equally to this work. This work is mainly based on a former publication of the two main authors and their co-authors and the master thesis of the second author.

Mining Software Repositories research field is now concerned with gaining insights from this data, e.g., by applying algorithms from the machine learning domain. One issue that is often addressed in related work is the aim to increase efficiency during the software development process is the (semi-)automatic assignment of open bug reports to suitable developers. This is especially of interest in large distributed development teams, where the triage owner may not be informed about the specific skills of each developer. Here and in the following, we always refer to bug reports, though the assignment of issue reports or change requests are treated equally in our considerations. Precisely, we refer to the problem defined as follows: "Given a new bug report, identify a ranked list of developers whose expertise qualifies them to fix the bug" [27].

Various approaches for automatic bug triaging have been proposed utilizing different data sources, e.g., source code [11,13,14,17,31], question-and-answer platforms [26], or solely historical bugs [6]. Thereby the textual components of the bug reports are usually modeled using LDA [6]. LDA is a probabilistic topic model proposed by Blei et al., often used to capture a intrinsic semantic structure of a collection of documents [5]. Given a set of documents, a so-called corpus, LDA detects patterns of co-occurring words and derives topics as multinomial distributions over the vocabulary. An underlying human-understandable concept of a topic can then often be derived from its most probable words, thus supporting explainability to the user. Furthermore, the semantic structure of a document is modeled as a distribution over the extracted topics, thus allowing a mathematical treatment of the documents. When applying topic models, different specifications can be made by the user:

Variant of LDA. The baseline LDA model can be extended by meta-information about the documents, e.g., the ATM, which incorporates knowledge about the authorship of the documents [25], or Labeled LDA for documents associated with discrete categories [23].

Inference Technique. Since exact inference for LDA is intractable, an approximation algorithm has to be taken into account when training LDA, e.g., Collapsed Gibbs Sampling (CGS) [9] or Variational Bayes (VB) [5].

Implementation. Different software libraries implement different inference algorithms and also differ in their default settings.

Hyperparameters. Besides the number of topics, the Dirichlet priors for the document-topic distribution and the topic-term distribution can be specified by the user.

Though the application of LDA incorporates many specifications by the user, in most related work, the model is treated as a black box without fully exploring the effect of the parameters [6].

Atzberger and Schneider et al. proposed three novel bug triaging algorithms that rely on the ATM, a variant of LDA, and compared them against an approach proposed by Xie et al. [2]. In their work, the authors evaluated the effect of the inference technique, as well as the Dirichlet prior and the number of topics as hyperparameters of the underlying LDA model. We extend their work by

additionally comparing the effect of an online training method with the batch training algorithm and detailing the effect of the used topic modeling library. We further discuss statements made by Xie et al. about a hyperparameter in their proposed approach. Our results are derived from experiments on a large dataset of bug reports from the *Mozilla Firefox* project. As a result of our work, we formulate different guidelines for applying LDA effectively for the bug triaging task, which can be seen as a starting point for other software engineering tasks that rely on topic models.

The remainder of this work is structured as follows: Section 2 presents existing bug triaging approaches utilizing probabilistic topic models. The techniques of Xie et al. and Atzberger and Schneider et al. are explained in Section 3, together with an introduction to LDA and the ATM. Our experimental setup for conducting our study is shown in Section 4. The results of our experiments are presented in Section 5 and their implications are discussed in Section 6. We conclude this work and point out directions for future work in Section 7.

2 Related Work

In this section, we present existing approaches for assigning bug reports to developers. We focus our presentation of related work on approaches that rely on LDA for modeling the textual components of a bug report as this is the focus of our work and therefore ignore approaches utilizing information from the version control system [11,13,14,17,31], or question-and-answer platforms [26]. An overview of the approaches that we consider is shown in Table 1.

Xie et al. were the first who applied LDA for the bug triaging task [39]. Their approach Developer Recommendation based on Topic Models (DRETOM) comprises three parts. First, an LDA model is trained on a corpus of historical bug reports with known resolvers. Then, for an incoming bug report, for each developer, a score is computed that is meant to capture their familiarity with the dominant topic of the bug report. In the last step, the developers are ranked according to their scores. In a later work, Atzberger and Schneider et al. built up on the idea of Xie et al. and proposed three algorithms based on the ATM. An advantage of the ATM is its interpretability, as each developer can be modeled as a distribution over the topics. This sort of explainability is particularly interesting for its use in real-world settings [1,45]. The modifications Developer Recommendation based on the Author-Topic Model (DRATOM) and Developer Recommendation based on the Author-Topic Model and Bayes Formula (DRATOMBayes) replace the LDA core with the ATM. Their third approach Developer Recommendation based on Author Similarity (DRASIM) exploits the fact that the ATM can describe bug reports and developers in a joint feature space and thus allows them to draw associations direct from the model itself.

The approach presented by Xia et al. takes a bug report's title and description as textual components, the product and component affected by the bug, and the developers who participated in resolving the bug into account [38]. For an incoming bug report, similar historical bug reports are detected based on an

Table 1: Comparison of related work that used probabilistic topic models.
All approaches (in chronological order) used the natural language title and description for topic modeling (LDA). Further attributes are used to improve the predictions of ground truth developers with regard to the chosen evaluation scheme and metrics covered in Section 4. We also examined the work in relation to the following aspects: (Q1) Mentioned inference technique, (Q2) Optimized number of Gibbs iterations, (Q3) Optimized number of topics K, and (Q4) Optimized Dirichlet priors (α, β). Depending on whether it is true or not, we use ✓or ✗, respectively. ✍ denotes that the authors claimed but did not report this optimization and ? that the publication did not report the parameters at all.

Authors	Topic Model	Q1	Q2	Q3	Q4	Further used Features	Ground Truth	Evaluation Scheme	Evaluation Metric
Xie et al. [39]	LDA	✗	✗	✗	✗		assignee, commenters	fixed train/test split	Precision@k, Recall@k, F_1
Xia et al. [38]	LDA	✗	✗	✗	✗	product, component	assignee, commenters	10-fold time-split	Precision@k, Recall@k, F_1
Naguib et al. [20]	LDA	✓	?	?	?	component	assignee, reviewer, resolver	fixed train/test split	Accuracy@k
Zhang et al. [43]	LDA	✗	✍	✗	✗	#comments, #assignments	assignee, commenters	fixed train/test split	Precision@k, Recall@k, F_1, MRR
Yang et al. [40]	LDA	✗	✗	✗	✗	product, component, priority, severity, #comments, #commits, #assignments, #attachments	assignee	fixed train/test split	Precision@k, Recall@k, F_1, MRR
Zhang et al. [44]	LDA	✓	✗	✍	✗	commenters, submitters	commenters	8-fold time-split	Recall@k
Nguyen et al. [22]	LDA	✗	?	✗	?	fixer, time to repair, severity	commenters	fixed train/test split	Median absolute error for time to repair
Zhang et al. [42]	LDA	✗	✗	✓	✗	product, component, priority, #comments, #fixed bugs, #reopened bugs	assignee, commenters	10-fold time-split	Precision@k, Recall@k, F_1, MRR
Xia et al. [37]	MTM	✓	✓	✓	✗	product, component	assignee	10-fold time-split	Accuracy@k

abstraction based on the terms occurring in the bug textual components, its topic distribution derived from LDA, and both the categorical features component and product. The developers are then associated with an affinity score. Similarly, a second affinity score is derived from comparing the new bug report with developers by considering their past activities. Combining the two affinity scores results in a list of potential developers.

Naguib et al. presented an approach for bug triaging based on LDA and the affected system component [20]. In their approach, each developer's activity profile is computed based on past activities as an assignee, reviewer, or resolver of a bug. Given the topic distribution of a new bug, the developers are then ranked according to their activity profiles.

Zhang et al. proposed an algorithm utilizing the textual components of a bug and the developers' social network [43]. First, an LDA model is trained, which allows for comparing a new bug report with a historical data basis. A developer's expertise in a topic is modeled from past bug reports' activities. Besides actual assignments, also comments are taken into account. Furthermore, a metric is derived that represents the role of a developer in the social network by relating a potential candidate to the most active reporter. The combination of both metrics leads to a ranking for a new bug based on its semantic composition.

The approach of Yang et al. also trains an LDA model on a corpus of historical bug reports [40]. Their approach detects similar bug reports for a new incoming bug report by considering its topics, product, component, severity, and priority. Then, from those similar bug reports, their approach extracts potential candidates. Based on the number of assignments, attachments, commits, and comments, the candidates are ranked according to their potential to fix the bug.

Zhang et al. presented BUTTER, another approach based on training an LDA model on a corpus of historical bug reports [44]. For each developer, a score in a topic is derived from a heterogeneous network consisting of submitters, developers, and bugs using the RankClass model. Combining the topic distribution from the RankClass model with the topic distribution derived from the LDA model, a candidate list of developers for a new bug is computed.

Nguyen et al. translated the task of assigning a bug report to the task of predicting the resolution time for fixing a bug report [22]. By assuming a log-normal regression model for the defect resolution time, their approach predicts a bug report's resolution time based on its topic distribution along with the bug fixer and the severity level [22]. Also in their approach, the topic distribution of a bug is inferred from a previously trained LDA model. The bug report is then assigned to the developer, who is most likely the fastest to fix the bug.

Similar to Xia et al. [38], the approach presented by Zhang et al. detects the K-nearest neighbors for a new bug report based on its textual components, product and component, and its topic distribution [42]. From the K most similar bug reports, the developers are then ranked according to their activities as assignees and commenters. The approach allows for the choice of 18 hyperparameters, of which only the number of topics concerns the underlying LDA model, the remaining hyperparameters of the LDA model remain on the default settings.

This is an example of a general trend whereby the algorithm for developer recommendations is becoming more complex, but the underlying topic models remain unchanged. Researchers usually treat topic models as black boxes without fully exploring their underlying assumptions and hyperparameter values [6]. The approaches presented so far used LDA without considering the impact of all of its hyperparameters or the used inference technique.

A significant exception is the work by Xia et al., who extend the basic topic modeling algorithm LDA and propose the Multi-feature Topic Model (MTM) [37]. The MTM includes a bug report's feature combination, e.g., its product and component, as an additionally observed variable and models the topic distributions for each feature combination. They recommend the developers based on the affinity scores of a developer towards a topic and the feature combination. Although they systematically evaluated the number of topics and iterations required for Gibbs sampling, they omit the influence of the Dirichlet priors, which remain at the tool's default settings.

All presented works use Gibbs sampling as the inference technique for LDA, although most do not mention it, only the used topic modeling tool. Therefore, it is likely that this choice was made unconsciously, especially since the authors often explicitly emphasize using the default hyperparameters for LDA. Only two of the previous works reported the influence of the most critical hyperparameter – the number of topics – on the quality of the developer recommendations [37,42]. Others set this value arbitrarily [22,38,39], kept the number of topics at the default value of the used implementation [40,43], or did not report this value [20].

Although much of the work compares directly with the basic DRETOM approach by Xie et al. , comparability of study results is limited due to different definitions of ground truth developers, evaluation setups, evaluation metrics, varying projects selected for evaluation, and chosen preprocessing techniques. Among those works that reimplement DRETOM [40,42,43,44], none of them describe how DRETOM's trade-off parameter θ was chosen. In contrast, these approaches introduce their own hyperparameters to weigh various influencing factors. The impact of their hyperparameters on the quality of developer recommendations is considered and optimized for, but not those of the methods compared with, let alone the hyperparameters of LDA. Therefore, in our work, we put particular emphasis on the influence of the hyperparameters of LDA and the ATM as well as the used inference techniques on the hyperparameters of the proposed approaches.

A detailed overview of how topic models are used for software engineering tasks, in general, is presented by Chen et al. [6]. In this systematic literature study, the authors show that only a small number of related work fully uses the possibilities of tuning the hyperparameters of a topic model or its implementation. Our work follows their suggestion of reimplementing existing approaches and taking various preprocessing steps and hyperparameter tuning into account.

3 Assigning Bug Reports using LDA

In this section, we present four approaches proposed by Xie et al. and Atzberger and Schneider et al. for assigning bug reports to developers. Both works rely solely on analyzing bug reports' textual components using probabilistic topic models. We, therefore, present details about the preprocessing steps, the underlying structure of the topic model, and its hyperparameters before detailing the ranking schemes in the different bug triaging approaches.

3.1 Preprocessing

We combine each bug report's title and more detailed description to form a document. The resulting corpus $\mathcal{D} = \{d_1, \ldots, d_M\}$ consists of documents that contain the textual components of the given bug reports. In our considerations, we ignore the discussions attached to a historical bug report as they are not available at the time when a bug report is assigned. In order to remove words that carry no meaning, we need to undertake the corpus $\mathcal{D}$ several preprocessing steps. The work of Atzberger and Schneider et al., follows the best practices studies by Schofield et al. [28,29,30], specifically:

1. Removal of URL, hex code, stack trace information, timestamps, line numbers, tokens starting with numerics, tokens consisting of only one character and punctuation,
2. Lower casing the entire document,
3. Removal of all words from the English *NLTK Stopwords Corpus*[1]
4. Removal of words with a frequency less than 5 as well as a maximum word occurrence of no more than 20% across all bug reports.

In contrast to Xie et al., we intentionally do not apply stemming. Topic model inference often groups words sharing morphological roots in the same topics, making stemming redundant and potentially damaging to the resulting model [29]. After preprocessing, we store each document as a Bag-of-Words (BOW), i.e., we neglect the ordering of the terms within a document and only keep their frequencies. The entire corpus $\mathcal{D}$ is written as a document-term matrix, whose rows are the BOW vectors of the documents.

3.2 Latent Dirichlet Allocation and its Variants

Starting from the document-term matrix, whose rows contain the term frequencies of the corresponding preprocessed documents, LDA detects clusters within the vocabulary by observing patterns of co-occurring words [5]. These clusters $\varphi_1, \ldots, \varphi_K$ are called topics and are formally given by multinomial distributions over the vocabulary $\mathcal{V}$. Here K, denotes the number of topics and is a hyperparameter of the model which needs to be set by the user initially. Table 2 shows three exemplary topics extracted from the *Mozilla Firefox* dataset that will be

[1] http://www.nltk.org/nltk_data/

Table 2: Three extracted topics with the highest probability from the exemplary bug report. The example is taken from Atzberger et al. [2].

Topic #1	**Topic #2**	**Topic #3**
private	preferences	bar
browsing	pref	url
mode	options	location
clear	dialog	autocomplete
cookies	default	text
history	set	address
window	prefs	results
data	option	type
cookie	preference	enter
cache	change	result

presented in detail in Section 4.1. From its ten most probable words, we derive that topic #1 concerns browsing in private mode, topic #2 concerns preference settings, and topic #3 is about the user interface. Besides a description of the topics, LDA results in a document-specific topic distributions $\theta_1, \ldots, \theta_M$ that capture the semantic composition of a document in a vector of size K. LDA assumes a generative process underlying a corpus $\mathcal{D}$, which is given by:

1. For each topic $\varphi_1, \ldots, \varphi_K$ choose a distribution according to the Distribution Dirichlet(β)
2. For each document d in the corpus $\mathcal{D}$
   (a) Choose a document-topic distribution θ according to Dirichlet(α)
   (b) For each term w in d
      i. Choose a topic $z \sim$ Multinomial(θ)
      ii. Choose the word w according to the probability $p(w|z, \beta)$

The generative process as a graphical model is shown in Figure 1.

Here, α and β denote the Dirichlet priors for the document-topic and topic-term distribution, respectively. The meaning of the parameter $\alpha = (\alpha_1, \ldots, \alpha_K)$, where $0 < \alpha_i$ for all $1 \leq i \leq K$, is best understood when written as the product $\alpha = a_c \cdot m$ of its concentration parameter $a_c \in \mathbb{R}$ and its base measure $m = (m_1, \ldots, m_K)$, whose components sum up to 1. Depending on whether the base measure is uniform, i.e., $m = (1/K, \ldots, 1/K)$, or not, we call the Dirichlet distribution symmetrical or asymmetrical, respectively. In the case of a symmetric prior, small values of a_c, the Dirichlet distribution would favor points in the simplex that are close to one edge, i.e., LDA would try to describe a document with a minimum of topics. The larger the value of a_c, the more likely that LDA is to fit all topics with a non-zero probability for a document. Figure 2 illustrates the effect of the chosen concentration parameter of a symmetric prior.

A symmetrical prior distribution of topics within documents assumes that all topics have an equal prior probability of being assigned to a document. However, setting a symmetrical α prior ignores that specific topics are more prominent

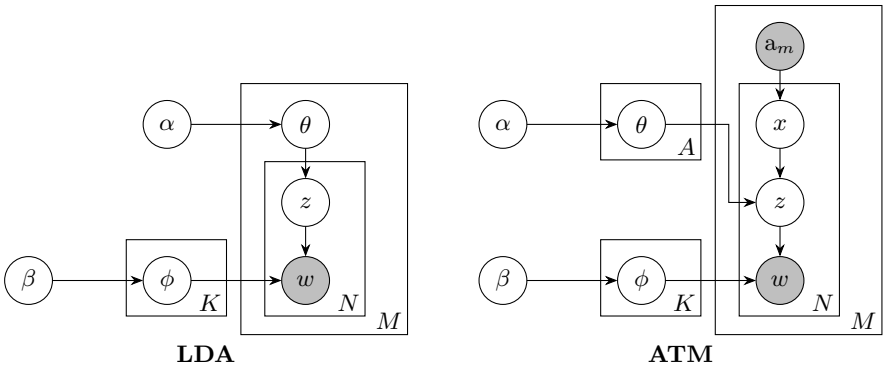
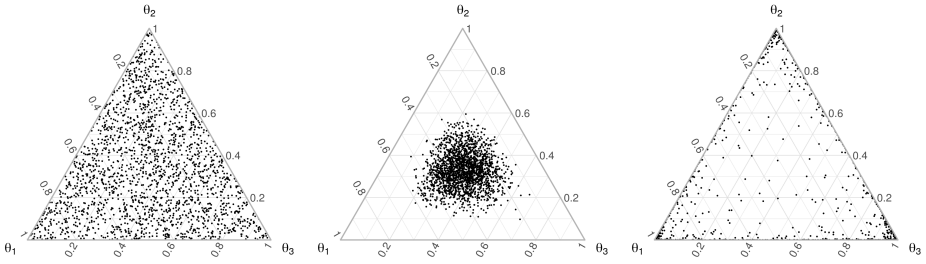


Fig. 1: (Left) Graphical model underlying LDA in plate notation, (Right) Graphical model underlying the ATM in plate notation. The shaded circles denote observed variables.



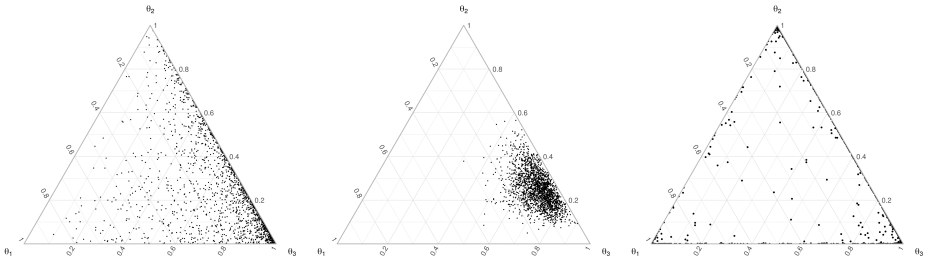
(a) $a_c = 3$ (b) $a_c = 30$ (c) $a_c = 0.3$

Fig. 2: 2,000 randomly sampled topic distributions θ using varying concentration parameters a_c for symmetrical Dirichlet prior α. We illustrate the distributions for $K = 3$ topics in the three-dimensional topic space $\theta = (\theta_1, \theta_2, \theta_3)$. The base measure $m = (1/3, 1/3, 1/3)$ is uniform for every symmetrical Dirichlet distribution.

in a corpus and, therefore, would naturally have a higher probability of being assigned to a document. Conversely, some topics that are less common and, thus, not appropriately reflected with a symmetrical prior for the document-topic distributions. In contrast, an asymmetrical β prior over the topic-term distributions is not beneficial [32,35]. The case of an asymmetric Dirichlet prior is shown in Figure 3.

The goal of fitting such a model is to infer the latent variables from the observed terms. However, since exact inference is intractable, approximation algorithms need to be taken into account [5], e.g., CGS [9], batch VB [5], and its online version Online Variational Bayes (OVB) [10]. Geigle provides a comparison of these techniques [8].

Different variants of LDA have been developed for the case, where additional meta-information to the documents is available. One such variant is the ATM,



(a) $\alpha = \boldsymbol{\alpha_c} m$ with $\boldsymbol{\alpha_c} = 3$ (b) $\alpha = \boldsymbol{\alpha_c} m$ with $\boldsymbol{\alpha_c} = 30$ (c) $\alpha = \boldsymbol{\alpha_c} m$ with $\boldsymbol{\alpha_c} = 0.3$

Fig. 3: 2,000 randomly sampled topic distributions θ using varying concentration parameters a_c for an asymmetrical Dirichlet prior α. We illustrate the distributions for $K = 3$ topics in the three-dimensional topic space $\theta = (\theta_1, \theta_2, \theta_3)$. The base measure $m = (1/12, 1/4, 2/3)$ defines the asymmetry where the third topic is most likely independent of the concentration parameter.

proposed by Rosen-Zvi et al. [25]. By observing the authors of a document, it assumes a generative process underlying a corpus, which is given by:

1. For each author $a \in \{a_1, \ldots, a_A\}$:
   (a) Choose a distribution over topics $\theta_a \sim \text{Dirichlet}(\alpha)$,
2. For each topic $\varphi_1, \ldots, \varphi_K$ choose a distribution according to the distribution Dirichlet(β)
3. For each document d in the corpus $\mathcal{D}$:
   (a) Given a group of authors, $\mathbf{a}_m \subseteq \{a_1, \ldots, a_A\}$, for document d,
   (b) For the n^{th} word $w_{m,n}$ of the N words to be generated in document m:
      i. Choose an author $x_{m,n} \sim \text{Uniform}(\mathbf{a}_m)$,
      ii. Choose a topic $z_{m,n} \sim \text{Multinomial}(\theta_{x_{m,n}})$,
      iii. Choose a word $w_{m,n} \sim \text{Multinomial}(\varphi_{z_{m,n}})$.

Its generative process is shown in Figure 1. Rosen-Zvi et al. provided the CGS algorithm for fitting an ATM model, the batch and online VB algorithm were presented by Mortensen [19].

3.3 Ranking Developers

DRETOM. After training a LDA model on a corpus of historical bug reports with known resolvers, each bug report is associated with its dominant topic. The aptitude $P(\text{d}|\text{b})$ for a developer d to solve a bug report b can be computed using the sum rule of conditional probabilities

$$P(\text{d}|\text{b}) = \sum_{\text{z}} P(\text{d}|\text{z}) \cdot P(\text{z}|\text{b}), \tag{1}$$

where $P(\text{d}|\text{z})$ denotes the skill level of the developer d in the respective topic z, and $P(\text{z}|\text{b})$ is the probability of the topic in the given bug report. In the

considerations of Xie et al. the skill of a developer d in topic z is composed by two parts, which are balanced by a trade-off parameter $\theta \in [0, 1]$:

$$P(\mathrm{d}|\mathrm{z}) = \theta P(\mathrm{d} \to \mathrm{z}) + (1 - \theta) P(\mathrm{z} \to \mathrm{d}), \tag{2}$$

where $P(\mathrm{d} \to \mathrm{z})$ is called interest and $P(\mathrm{z} \to \mathrm{d})$ is called expertise. Given the number of bug reports $N_{d,z}$ of developer d that belong to topic z, and the number of bugs N_d, where d has contributed, the interest part is computed as

$$P(d \to z) = \frac{N_{\mathrm{d,z}}}{N_{\mathrm{d}}}, \tag{3}$$

Given the total number N_z of bug reports that are associated with topic z, the expertise component is computed as

$$P(z \to d) = \frac{N_{d,z}}{N_z}. \tag{4}$$

For an incoming bug report, its topic distribution is inferred using the trained LDA model and the developers are ranked according to their conditional probabilities from Equation (1).

DRATOM & DRATOMBayes. Atzberger and Schneider et al. proposed two adaptions of DRETOM by utilizing the ATM [2]. Their first modification DRATOM simply replaces the LDA core of DRETOM by the ATM, i.e., only the probability $P(z|b)$ is derived from another model.

Their second modification DRATOMBayes circumvents the choice of the trade-off parameter θ by taking the description of a developer as distribution over topics, learned from the ATM, into account. From the Bayes' formula, we derive

$$P(d|z) = \frac{P(z|d) \cdot P(d)}{P(z)}. \tag{5}$$

The probability $P(d)$ is approximated by the fraction

$$P(d) = \frac{N_d}{N_{total}}, \tag{6}$$

where N_{total} denotes the number of all bug reports in the training corpus. The marginal probability $P(z)$ is approximated from the base measure of the Dirichlet prior α, i.e.,

$$m = (m_1, \ldots, m_K) = (P(z_1), \ldots, P(z_K)). \tag{7}$$

DRASIM. Linstead et al. were the first to apply the ATM for software engineering tasks [15]. Atzberger and Schneider et al. adopted this approach for the case of bug reports, which results in a joint embedding of developers and bug reports in a common feature space. Their approach DRASIM therefore reduces the bug triaging tasks to a Nearest Neighbor (NN) search, i.e., for an incoming bug report the developers are ranked according to a measure $D(b, d)$, where D denotes a chosen similarity measure, e.g., the Jensen-Shannon distance, the cosine similarity, or the Manhattan distance.

4 Experimental Setup

In our experiments, we investigate the effect of the underlying topic model, the inference technique, the topic modeling library used, and the hyperparameters on the results of the presented bug triaging algorithms. In this section, we present details on our experimental setup. The dataset on which the experiments are carried out contains bugs from the *Mozilla Firefox* project for over 15 years. To take into account this long development history, we chose the Longitudinal Evaluation Scheme for quantifying the bug triaging algorithms. To compare different topic modeling libraries, we have chosen the three actively maintained libraries *Gensim*, *MALLET*, and *Vowpal Wabbit*. We present details regarding their implementations at the end of the section.

4.1 Dataset

Xie et al. evaluated their approach on a data set from the *Mozilla Firefox* and the Eclipse projects, which are widely used in related work [3,4,7,16,20,36,37,42,43]. For the *Mozilla Firefox* project, their corpus contains 3005 bug reports, collected between September 2008 to July 2009, with 96 involved developers. In contrast, we compare the approaches DRETOM, DRATOM, DRATOMBayes, and DRASIM on a dataset published by Mani et al., which contains bug reports of the *Mozilla Firefox* project collected between July 1999 until June 2016 [16]. In contrast to Xie et al., we only train our model on bug reports with an assigned developer, and a status marked as *verified fixed*, *resolved fixed*, or *closed fixed*. The final assignee is considered as ground truth. As our experiments depend on the Author-Topic Model, we ensure that each training set has at least 20 bug reports per developer. In total, our dataset contains 18,269 bug reports with 169 bug resolvers.

4.2 Evaluation Scheme

In our experiments, each bug triaging algorithm is evaluated on ten runs. For this purpose, the bug reports are ordered chronologically and divided into 11 equally sized disjoint subsets. In the k-th run, where $1 \leq k \leq 10$, the k first subsets are used for training, and the $(k+1)$-th subset is used for testing. This evaluation scheme is known as the longitudinal evaluation scheme and has been shown to increase internal validity [4,12,16,33,37]. Furthermore, the time allocation considers that the available developers change throughout the project. As Xie et al. only train on bug reports collected within one year, this circumstance is not critical for their work.

Like most related work, Xie et al. evaluate their approach using the metrics Recall@k and Precision@k. Both metrics only consider if the ground-truth assignee is among the top k recommendations but do not consider the actual position. However, a good bug triaging algorithm should place the real assignee at a higher rank, as the triager usually checks the higher positions first. In our experiments,

we consider the Mean Reciprocal Rank (MRR)@10 as an evaluation metric, which is affected more by a hit in the first rank. The MRR@10 is given by

$$MRR@10 = \frac{1}{\#bug\ reports} \sum_{i=1}^{\#bug\ reports} (RR@10)_i, \text{ where} \tag{8}$$

$$(RR@10)_i = \begin{cases} \frac{1}{\text{rank}_i}, & \text{if rank}_i \leq 10 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

and RR denotes the reciprocal rank and rank_i is the rank position of the real assignee for the i^{th} bug report.

4.3 Topic Modeling Implementations

Xie et al. used the LDA implementation provided by the *Stanford Topic Modeling Toolbox (STMT)* [23]. In our work, we use more recently maintained libraries. Among the most widely used topic modeling libraries are *MALLET* [18] for Java, *Gensim* [24] for Python, and *Vowpal Wabbit*[1] for C++. The LDA implementation of the *Gensim* library is based on the VB algorithm proposed by Blei et al. [5] and offers its online version presented by Hoffman et al. [10]. In the single-threaded mode, *Gensim* allows the automatic adjustment of both Dirichlet priors. However, in the multi-core version, this feature is not available. For the ATM we refer to *Gensim*'s implementation of the ATM, which applies VB and OVB respectively. The *Vowpal Wabbit* implementation is written in C++ and implements the same OVB as *Gensim*, though it supports neither parallelization nor hyperparameter optimization. It only allows specifying the LDA hyperparameter, i.e., the number of topics and symmetrical Dirichlet priors α and β, as well as the hyperparameters necessary for OVB. In contrast, *MALLET* is the most advanced topic modeling toolkit of those we consider. It implements the simple parallel threaded implementation proposed by Newman et al. [21] with the SparseLDA Gibbs scheme and the data structure introduced by Yao et al. [41]. Based on the work of Wallach, *MALLET* implements several strategies for hyperparameter optimizations of Dirichlet priors [34].

5 Results

In this section, we give details on the conducted experiments and present their results. In their study, the Xie et al. evaluated their proposed approach DRETOM regarding its potential for recommending bug resolvers and the influence of the hyperparameter θ on the bug triaging results. In our first three experiments, we compare different implementations of LDA and inference techniques for replicating the DRETOM approach. With the DRETOM approach, we want to discuss our first research question:

[1] https://hunch.net/?p=309

Table 3: DRETOM's default parameter settings for LDA in *Gensim*, *Vowpal Wabbit* and *MALLET*.

	Gensim	*Vowpal Wabbit*	*MALLET*
α	`alpha` = 0.01	`--lda_alpha` = 0.01	`--alpha` = 0.2[2]
β	`alpha` = 0.01	`--lda_rho` = 0.01	`--beta` = 0.01
K	`num_topics` = 20	`--lda` = 20	`--num-topics` = 20
i_{CGS}	n/a	n/a	`--num-iterations` = 100
P	`workers` = 6	n/a	`--num-threads` = 6
Random seed	`random_state` = 42	`--random_seed` = 42	`--random-seed` = 42

RQ1: *To what extent does the chosen implementation of LDA and inference technique affect results for the bug triaging task?*

In the next step, we compare DRETOM with the three approaches presented by Atzberger and Schneider et al. based on the ATM. Precisely, we want to answer the following research question:

RQ2: *Are the approaches based on the ATM able to outperform DRETOM?*

Both topic models, LDA and the ATM, require to specify the number of topics K and the Dirichlet prior α for the document-topic distribution. In our fifth experiment, we varied the possible hyperparameters to address the following research question:

RQ3: *How far do the choice of hyperparameter, i.e., the number of topics and the Dirichlet prior, affect the approaches for bug triaging tasks?*

Our results further allow us to discuss statements made by Xie et al. on the hyperparameter θ, that will be discussed in the fourth research question:

RQ4: *What is the influence of the hyperparameter θ for the quality of the recommendation results?*

5.1 Naive Replication Study of DRETOM

In the first three experiments, we want to investigate how the chosen LDA implementation and the underlying inference technique affect the results of a bug triaging algorithm. In our first experiment, we therefore replicated the study by Xie et al. and adopt their parameter configurations. We assume symmetric distributions for both Dirichlet priors α and β, each with a concentration parameter of 0.01. The number of latent topics is set to $K = 20$, and the number of iterations in the Gibbs sampling is set to $i_{CGS} = 100$. To reduce computation time, we resort to the parallelized variants in *Gensim* and *MALLET* and set the number of parallel threads to 6. *Vowpal Wabbit* does not allow the use of hardware acceleration. All remaining parameters are set to their default values defined by the library. The parameter values are summarized in Table 3.

[2] *MALLET* expects the concentration parameter, i.e., $0.01 \cdot K = 0.2$, for $K = 20$.

Table 4: Impact of underlying inference techniques using default parameters for LDA implementations on MRR@10 obtained on the Mozilla Firefox project with 10-fold cross validation. The mean over the cross validation and standard deviation are reported. The optimal hyperparameter setting for θ is highlighted.

Model	θ	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Average
DRETOM Gensim (parallel)	0.0	**8.5**	**6.8**	**9.2**	**13.0**	**9.4**	4.4	**10.2**	6.9	8.4	6.1	**8.3 ± 2.4**
	0.1	7.6	6.6	8.4	12.5	9.0	**4.4**	9.2	**7.0**	**8.8**	**6.2**	8.0 ± 2.2
	0.2	7.0	6.0	7.6	11.6	6.3	4.0	7.7	6.2	7.3	5.4	6.9 ± 2.0
	0.3	6.2	5.5	6.7	9.7	5.4	3.9	7.0	5.4	5.7	5.1	6.1 ± 1.5
	0.4	5.8	5.6	5.8	7.5	4.6	3.7	6.1	4.5	4.4	4.4	5.2 ± 1.1
DRETOM Gensim	0.0	**8.4**	**6.6**	**9.4**	**13.5**	12.7	5.3	10.1	7.0	**9.0**	6.4	8.8 ± 2.7
	0.1	7.7	6.4	9.1	13.3	**13.2**	**5.6**	**10.6**	**7.3**	8.9	**6.5**	**8.9 ± 2.8**
	0.2	6.9	5.5	8.4	12.1	10.8	4.5	9.3	6.3	8.4	6.3	7.8 ± 2.4
	0.3	6.2	5.5	7.3	10.5	8.6	4.3	7.6	4.9	7.2	5.7	6.8 ± 1.9
	0.4	5.8	5.3	6.4	8.9	6.8	3.7	6.2	4.0	6.5	4.9	5.9 ± 1.5
DRETOM MALLET	0.0	**9.0**	9.8	15.0	16.0	14.3	8.5	13.7	8.8	13.1	13.1	12.1 ± 2.8
	0.1	8.8	10.6	**16.5**	**16.7**	**15.1**	**9.8**	**13.9**	**9.7**	**15.0**	**14.8**	**13.1 ± 3.0**
	0.2	8.0	11.1	15.5	16.3	12.9	9.0	10.8	8.6	13.5	14.8	12.0 ± 3.0
	0.3	7.1	**11.6**	13.5	15.6	10.3	7.8	8.4	7.6	12.1	14.1	10.8 ± 3.0
	0.4	6.4	11.0	12.2	13.0	7.8	6.6	6.5	7.0	10.0	13.1	9.4 ± 2.8
DRETOM Vowpal Wabbit	0.0	**9.0**	**7.1**	7.9	14.5	8.4	**5.5**	**8.1**	**7.0**	**7.7**	**6.9**	**8.2 ± 2.4**
	0.1	8.3	7.1	**8.7**	15.2	**8.5**	4.7	7.1	6.2	7.1	6.5	7.9 ± 2.8
	0.2	8.2	6.3	7.3	**16.8**	7.5	3.2	5.6	4.7	6.0	5.2	7.1 ± 3.7
	0.3	5.7	5.2	4.6	16.0	4.5	2.0	4.2	3.2	5.7	3.1	5.4 ± 3.9
	0.4	6.7	4.5	4.0	8.1	3.2	1.5	3.3	2.5	3.9	2.3	4.0 ± 2.0

We report the results of the first experiments over all ten runs according to the longitudinal evaluation scheme in terms of the MRR@10 in Table 4. As the optimal value for θ never exceeds a value more than 0.3, we only present the results for values $\theta \leq 0.4$. In nearly every run, the single-threaded variant of *Gensim* (8.9%) is superior to its multi-threaded version (8.3%). On average, the single-threaded variant outperforms the multi-threaded variant by 7.2%. The *MALLET*-based implementation (13.1%) outperforms all other versions in nearly every run. 47,2% gives the average advantage against the single-threaded *Gensim* implementation, 57,8% against the multi-threaded *Gensim* implementation, and 59,8% against the *Vowpal Wabbit* implementation (8.2%). This indicates the supremacy of the CGS against the VB. Furthermore, the trade-off parameter θ seems to influence all implementations strongly. Xie et al. report that the average precision and recall peak at $\theta = 0.6$ for the *Mozilla Firefox* project, which favors the developer's interest slightly more than a developer's expertise. Surprisingly, regardless of the LDA implementation, DRETOM never reaches an optimal value of θ greater than 0.3 in the first experiment of our reproduction study. The developer's expertise is much more critical concerning the bug assignment prediction accuracy measured with the MRR@10 compared to a developer's interest. Across all cross-validation splits, the MRR@10 peaks at $\theta = 0.0$ in 47.5% of the cases, which means that the recommendation is wholly based on a developer's expertise.

5.2 Multi-pass Online Variational Bayes Inference

In the second experiment, we investigated to what extent the inferiority of the VB implementations in the first experiment is because the LDA model has not converged. To determine whether the frequently used default value $i_{CGS} = 100$ is

Table 5: Additional LDA learning parameters for *Gensim*, *Vowpal Wabbit* and *MALLET*.

	Gensim	*Vowpal Wabbit*	*MALLET*
S	`chunksize` = 2000	`--minibatch` = 2000	n/a
κ	`decay` = 0.5	`--power_t` = 0.5	n/a
τ_0	`offset` = 1.0	`--initial_t` = 1.0	n/a
δ	`gamma_threshold` = 0.001	`--lda_epsilon` = 0.001	n/a
passes	`passes` = 10	`--passes` = 10	n/a
i_{VB}	`iterations` = 400	n/a[3]	n/a
i_{CGS}	n/a	n/a	`--num-iterations` = 1000

sufficient, we increase the number of Gibbs iterations to $i_{CGS} = 1000$. Furtermore, we increase the number of iterations for *Gensim*'s number of variations to $i_{VB} = 400$. *Vowpal Wabbit* does not offer a parameter to modify this stopping criterion and runs as many iterations as required to reach the convergence threshold δ. Both libraries use the learning parameters κ and τ set to 0.5 and 1.0 to guarantee convergence [10]. To allow a more direct comparison between both libraries implementing OVB we increase the batch size for *Vowpal Wabbit* from 256 to match the default value of *Gensim*, which results in updating the topic-term distributions in the M-step after processing 2,000 documents. In both tools, the parameter `passes` controls how many passes will be used to train the topic models. The remaining parameters for the LDA model are unchanged. Table 5 summarizes the parameters changed to our first experiment. The results of our second experiment are presented in Table 6.

On average, the result of the *MALLET*-based implementation does not improve. This shows that a number of $i_{CGS} = 100$ iterations is sufficient to guarantee the convergence of the LDA model. As expected, all implementations based on VB improved. The gains here are 36.1% on average for the parallelized version of *Gensim* (11.3%), 5.6% for its single-threaded version (9.4%), and 12.2% for *Vowpal Wabbit* (9.2%). On average, however, all are still worse than the CGS-based variant and inferior by at least 15.9%.

5.3 Multi-pass Batch Variational Bayes Inference

Our third experiment investigates how batch VB LDA differs from online VB LDA in their results. The primary motivation for using online VB is to study large data sets that can no longer be processed in a single step. However, using online VB entails three additional hyperparameters (S, τ_0, κ), which need to be optimized for. Moreover, OVB is not able to handle a changing vocabulary. In our case, where we analyze bug reports of more than 15 years, this might be problematic. In addition, we evaluate DRETOM based on the batch variant of *Gensim*. However, since *Vowpal Wabbit* does not allow batch learning, and its online variant is inferior to *Gensim*'s, we neglect this setup. We keep the

[3] *Vowpal Wabbit* does not offer this parameter to stop earlier, i.e., it is implicitly set to ∞.

Table 6: Impact of underlying inference techniques using multi-pass VB on MRR@10 obtained on the Mozilla Firefox project with 10-fold cross validation The mean accuracy over the cross validation and standard deviation are reported. The best hyperparameter setting for θ is highlighted.

Model	θ	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Average
DRETOM *Gensim* (parallel)	0.0	8.6	7.2	11.5	15.2	14.5	8.5	14.2	9.4	10.9	9.1	10.9 ± 2.8
	0.1	8.6	**7.3**	**11.8**	**15.2**	**14.8**	**9.6**	**15.2**	**10.0**	**11.2**	**9.7**	**11.3 ± 2.9**
	0.2	8.3	6.6	10.9	14.5	12.9	8.2	12.8	8.9	9.2	9.1	10.1 ± 2.5
	0.3	8.4	6.3	9.9	13.6	10.8	6.8	9.8	7.5	7.5	8.6	8.9 ± 2.2
	0.4	8.7	6.4	8.1	11.6	8.7	6.1	8.5	6.2	5.9	7.5	7.8 ± 1.8
	0.5	9.0	5.9	7.2	9.6	7.4	5.3	7.1	4.9	4.8	6.8	6.8 ± 1.6
	0.6	9.3	5.9	6.2	8.6	6.2	5.0	6.3	4.3	4.4	6.5	6.3 ± 1.6
	0.7	**9.5**	5.7	5.6	7.3	5.3	4.4	5.8	4.0	4.0	6.2	5.8 ± 1.7
DRETOM *Gensim*	0.0	8.7	7.0	10.2	15.0	11.3	5.8	10.1	7.4	8.8	6.3	9.1 ± 2.8
	0.1	8.7	**7.0**	**10.6**	**15.5**	**12.4**	**6.1**	**10.3**	**7.6**	**9.1**	6.5	**9.4 ± 2.9**
	0.2	8.2	6.7	10.0	14.8	10.7	4.7	9.7	6.6	8.6	**6.6**	8.7 ± 2.8
	0.3	8.4	6.3	8.6	13.3	8.6	3.9	8.5	5.5	7.6	6.3	7.7 ± 2.5
	0.4	8.8	6.4	7.2	11.2	7.1	3.5	7.0	4.5	6.4	5.5	6.8 ± 2.2
	0.5	8.8	6.1	6.5	9.1	5.9	3.0	5.9	3.9	5.1	4.7	5.9 ± 1.9
	0.6	9.2	5.9	5.6	7.4	4.8	2.6	4.7	3.4	4.2	4.1	5.2 ± 2.0
	0.7	**9.4**	6.0	5.1	6.3	4.0	2.3	4.2	3.0	3.7	3.9	4.8 ± 2.0
DRETOM *MALLET*	0.0	**8.9**	9.7	15.3	15.9	14.7	7.8	14.6	10.0	13.4	12.0	12.2 ± 2.9
	0.1	8.9	10.4	**16.8**	**16.3**	**16.0**	**8.8**	**15.0**	**10.7**	**14.3**	**13.7**	**13.1 ± 3.1**
	0.2	8.0	11.3	15.5	15.7	13.1	7.3	13.1	9.3	12.6	13.5	11.9 ± 2.9
	0.3	7.4	**11.5**	12.7	14.7	10.5	6.0	10.4	8.1	11.1	13.0	10.5 ± 2.7
	0.4	7.0	11.2	10.9	12.1	8.6	5.0	8.5	7.3	9.5	12.1	9.2 ± 2.4
	0.5	7.0	11.0	9.8	10.2	7.4	4.2	7.6	6.3	8.4	11.3	8.3 ± 2.2
	0.6	6.8	10.5	9.2	9.3	6.1	3.8	6.9	5.8	8.0	10.7	7.7 ± 2.2
	0.7	6.9	9.9	8.8	8.3	5.3	3.4	6.3	5.3	7.6	10.2	7.2 ± 2.2
DRETOM *Vowpal Wabbit*	0.0	**9.0**	6.2	**11.3**	14.4	**8.0**	**6.4**	8.1	7.4	**9.6**	**11.6**	**9.2 ± 2.6**
	0.1	8.2	6.6	10.8	**15.3**	6.8	5.9	**8.9**	**8.8**	8.6	9.2	8.9 ± 2.7
	0.2	5.6	6.5	8.3	14.7	4.1	4.8	6.5	8.4	6.9	7.0	7.3 ± 3.0
	0.3	4.8	6.4	6.3	15.2	2.9	3.3	5.5	6.1	4.8	5.4	6.1 ± 3.4
	0.4	4.7	8.1	5.1	8.3	2.5	2.7	4.7	3.3	3.6	3.9	4.7 ± 2.0
	0.5	5.2	8.5	3.8	6.3	2.0	2.2	1.9	2.2	2.7	3.3	3.8 ± 2.2
	0.6	5.1	8.8	4.1	5.7	1.4	1.5	1.3	1.8	2.5	3.1	3.5 ± 2.4
	0.7	5.1	**9.0**	3.9	5.6	1.0	1.3	1.1	1.6	2.3	2.6	3.4 ± 2.6

parameters the same as in the first two experiments. The results of this experiment are shown in Table 7.

The results show that the OVB variant (9.4%) performs worse than the batch VB variant (10.8%) in every run. The only exception is the first run, where OVB and batch VB practically match since the number of documents was set to 2000. On average, batch VB performs 14.9% better than its online version. Despite all the optimizations, *MALLET* (13.1%) still outperforms *Gensim*; on average, *MALLET* outperforms *Gensim* by 21.3%. In the following two experiments, we neglect OVB.

5.4 Learning Asymmetrical Dirichlet Priors

In the first three experiments, we investigated the underlying inference technique's influence on the bug triaging algorithm's results using DRETOM as an example. In the remaining two experiments, we compared DRETOM with DRATOM, DRATOMBayes, and DRASIM, all of which are based on the ATM. In particular, we want to investigate the influence of the common hyperparameters α and K. In most existing work, a symmetric Dirichlet prior α is assumed, for which the default value of the respective implementation is mainly used. However, work by Wallach et al. shows that an asymmetric Dirichlet prior in combination with a symmetric beta prior has advantages over the default setting [35]. To evaluate the impact, we set the parameter `alpha = 'auto'` in the *Gensim* implementation

Table 7: Impact of underlying inference techniques using batch VB on MRR@10 obtained on the Mozilla Firefox project with 10-fold cross validation The mean accuracy over the cross validation and standard deviation are reported. The best hyperparameter setting for θ is highlighted. *Vowpal Wabbit* does not implement batch VB LDA.

Model	θ	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Average
DRETOM *Gensim* using online VB LDA	0.0	8.7	7.0	10.2	15.0	11.3	5.8	10.1	7.4	8.8	6.3	9.1 ± 2.8
	0.1	8.7	**7.0**	**10.6**	**15.5**	**12.4**	**6.1**	**10.3**	**7.6**	**9.1**	6.5	**9.4 ± 2.9**
	0.2	8.2	6.7	10.0	14.8	10.7	4.7	9.7	6.6	8.6	**6.6**	8.7 ± 2.8
	0.3	8.4	6.3	8.6	13.3	8.6	3.9	8.5	5.5	7.6	6.3	7.7 ± 2.5
	0.4	8.8	6.4	7.2	11.2	7.1	3.5	7.0	4.5	6.4	5.5	6.8 ± 2.2
	0.5	8.8	6.1	6.5	9.1	5.9	3.0	5.9	3.9	5.1	4.7	5.9 ± 1.9
	0.6	9.2	5.9	5.6	7.4	4.8	2.6	4.7	3.4	4.2	4.1	5.2 ± 2.0
	0.7	**9.4**	6.0	5.1	6.3	4.0	2.3	4.2	3.0	3.7	3.9	4.8 ± 2.0
DRETOM *Gensim* using batch VB LDA	0.0	8.7	**7.2**	**10.8**	15.1	13.0	7.4	13.8	9.1	10.9	7.4	10.3 ± 2.9
	0.1	8.7	7.0	10.4	**15.7**	**13.6**	**8.1**	**14.8**	**9.9**	**12.0**	**7.9**	**10.8 ± 3.1**
	0.2	8.2	6.5	9.2	15.6	11.0	7.7	12.6	8.8	10.2	6.8	9.7 ± 2.8
	0.3	8.4	6.1	7.7	13.6	8.9	6.5	9.6	7.9	9.0	5.9	8.4 ± 2.2
	0.4	8.8	5.9	6.3	11.5	7.8	5.7	8.1	6.6	8.1	5.2	7.4 ± 1.9
	0.5	8.8	5.7	5.4	10.3	6.5	5.2	6.9	6.0	7.0	4.5	6.6 ± 1.8
	0.6	9.2	5.7	4.5	9.2	5.8	4.9	5.9	5.6	6.6	4.1	6.2 ± 1.8
	0.7	**9.4**	5.6	4.1	8.2	4.9	4.7	5.5	5.3	6.3	4.0	5.8 ± 1.8
DRETOM *Mallet*	0.0	**9.0**	9.8	15.3	15.9	14.5	7.9	14.7	10.0	13.6	12.0	12.3 ± 2.9
	0.1	8.9	10.4	**16.7**	**16.2**	**15.8**	**9.0**	**15.0**	**10.8**	**14.2**	**13.7**	**13.1 ± 3.0**
	0.2	8.1	11.4	15.6	15.8	13.1	7.4	12.9	9.4	12.6	13.5	12.0 ± 2.9
	0.3	7.4	**11.6**	12.6	14.8	10.6	6.1	10.4	8.0	11.6	13.0	10.6 ± 2.7
	0.4	6.9	11.3	10.9	12.2	8.6	5.0	8.5	7.2	9.9	12.0	9.3 ± 2.4
	0.5	7.0	10.9	9.9	10.3	7.4	4.2	7.5	6.4	9.0	11.3	8.4 ± 2.3
	0.6	6.8	10.5	9.1	9.3	6.2	3.8	6.9	5.7	8.5	10.8	7.8 ± 2.2
	0.7	6.9	9.9	8.7	8.4	5.4	3.4	6.2	5.4	8.2	10.2	7.2 ± 2.2

for LDA and the ATM. Starting from the symmetrical standard α prior, we let *MALLET* optimize the hyperparameters every 20 iterations after a burn-in period of 50 iterations following David Mimno's advice[4]. The results of the fourth experiment are reported in Table 8.

Changing the symmetric α prior to an asymmetric one has little effect in either implementation of DRETOM. Concretely, the *Gensim*-based implementation improves by about 0.9% and the *MALLET*-based implementation by about 4.6%. However, the selected topic model seems to have a more significant impact. The results show that DRATOM (14.2%) achieves significantly better results than DRETOM. Specifically, DRATOM outperforms the *Gensim* variant (10.9%) by about 30.3% and the *MALLET* variant (13.7%) by 3.6%. It is worth mentioning here that *MALLET* implements CGS inference, which was superior in earlier experiments. In all three cases, the ideal value for theta changes abruptly. The DRATOMBayes approach (13.3%) overcomes this problem and still outperforms the DRETOM variant based on *Gensim* by about 22.0%. Regardless of the distance metric, DRASIM seems practically unusable, as it is inferior to all three comparison methods.

5.5 Increasing the Number of Topics

It remains to investigate the effect of the hyperparameter K. Following Griffiths and Steyvers, Xie et al. fixed the number of topics to 20 [9]. However, the optimal

[4] https://stackoverflow.com/questions/47310137/mallet-hyperparameter-optimization

Table 8: Impact of an optimized, asymmetric α prior on MRR@10 obtained on the Mozilla Firefox project with 10-fold cross validation. DRATOM, DRATOMBayes and DRASIM use the same trained ATM. The mean accuracy over the cross-validation and standard deviation are reported. The best hyperparameter setting for θ and the best distance metric for DRASIM are highlighted.

Model	θ	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Average
DRETOM *Gensim*	0.0	8.8	**7.4**	10.7	15.2	12.8	7.3	13.5	9.3	11.3	7.3	10.4 ± 2.8
	0.1	8.7	7.3	**10.8**	**16.0**	**13.5**	**7.9**	**14.5**	**9.9**	**12.1**	**8.0**	**10.9 ± 3.0**
	0.2	8.2	7.3	10.1	15.7	11.0	7.5	12.4	8.9	10.2	7.0	9.8 ± 2.7
	0.3	8.2	6.9	8.9	14.0	8.2	6.4	9.3	7.9	8.8	6.1	8.5 ± 2.2
	0.4	8.6	6.9	7.8	11.4	7.2	5.7	7.9	6.9	7.9	5.2	7.5 ± 1.7
	0.5	8.5	6.7	6.8	9.9	6.5	5.3	6.5	6.4	6.9	4.5	6.8 ± 1.5
	0.6	9.0	6.6	5.7	8.7	5.6	5.1	5.6	5.8	6.5	4.0	6.3 ± 1.5
	0.7	9.0	6.7	5.2	7.8	4.7	4.9	5.1	5.5	6.2	4.0	5.9 ± 1.5
	0.8	8.9	6.7	4.6	7.2	4.1	4.7	4.9	5.3	6.2	3.8	5.6 ± 1.6
	0.9	**9.1**	6.7	4.2	6.7	4.0	4.4	4.6	5.1	5.8	3.7	5.4 ± 1.7
	1.0	8.3	6.7	3.9	6.5	3.7	4.3	4.4	4.8	5.7	3.7	5.2 ± 1.5
DRETOM *MALLET*	0.0	9.1	9.3	14.8	15.9	16.8	9.3	16.9	9.3	13.5	12.0	12.7 ± 3.3
	0.1	9.4	10.0	**16.3**	16.7	**17.7**	**10.1**	**17.9**	**10.8**	**14.3**	13.6	**13.7 ± 3.4**
	0.2	9.4	10.2	14.3	**16.8**	16.0	9.7	17.6	10.6	13.6	**14.4**	13.3 ± 3.1
	0.3	9.8	**10.7**	12.1	16.5	13.3	7.6	14.5	9.2	12.0	14.3	12.0 ± 2.7
	0.4	9.9	10.4	10.6	14.2	8.5	6.3	11.7	8.4	10.7	12.8	10.4 ± 2.3
	0.5	10.7	10.3	9.2	12.0	6.7	5.3	10.0	7.7	9.4	11.9	9.3 ± 2.2
	0.6	11.4	10.1	8.1	10.3	5.7	4.8	8.9	7.2	8.3	11.1	8.6 ± 2.2
	0.7	11.5	9.8	7.1	8.9	4.9	4.4	8.1	6.5	7.7	10.2	7.9 ± 2.3
	0.8	11.5	9.4	6.6	8.3	4.3	4.2	7.3	6.1	7.4	9.6	7.5 ± 2.3
	0.9	**11.5**	9.2	6.2	7.8	3.9	4.1	6.8	5.9	7.1	9.2	7.2 ± 2.4
	1.0	10.6	9.0	5.9	7.0	3.8	3.9	6.5	5.8	6.9	8.8	6.8 ± 2.2
DRATOM	0.0	**11.3**	11.8	**13.6**	**15.8**	17.0	10.9	17.2	11.1	**14.3**	**17.3**	14.0 ± 2.7
	0.1	11.2	12.3	12.9	15.4	18.0	12.0	**17.5**	**11.7**	14.1	17.2	**14.2 ± 2.6**
	0.2	11.0	12.6	12.3	15.6	**18.2**	**12.3**	16.6	11.5	13.7	17.0	14.1 ± 2.6
	0.3	10.9	**12.7**	11.7	15.3	17.9	12.0	14.9	11.6	13.4	16.5	13.7 ± 2.4
	0.4	10.8	12.7	11.1	14.9	17.7	11.6	13.9	11.1	12.7	16.5	13.3 ± 2.4
	0.5	10.6	12.3	9.5	15.0	16.9	10.6	13.2	10.9	13.1	16.0	12.8 ± 2.5
	0.6	10.5	12.2	8.9	14.9	17.2	10.3	11.8	10.3	12.7	16.3	12.5 ± 2.8
	0.7	10.3	11.1	8.2	11.1	16.6	9.7	10.7	10.0	12.4	15.8	11.6 ± 2.7
	0.8	10.0	10.2	7.6	10.0	15.3	9.5	8.8	9.5	10.4	14.5	10.6 ± 2.4
	0.9	10.2	9.9	6.5	9.2	12.9	9.4	8.2	9.1	10.2	13.1	9.9 ± 2.0
	1.0	9.7	8.5	6.2	8.7	13.1	9.2	8.0	8.8	9.6	12.2	9.4 ± 2.0
DRATOMBayes	n/a	**11.2**	**13.9**	**12.9**	**15.0**	**16.6**	**10.1**	**15.3**	**10.6**	**12.8**	**15.0**	**13.3 ± 2.2**
DRASIM	Jensen-Shannon	**10.2**	7.8	**8.6**	6.1	6.7	**3.8**	5.6	6.2	**5.0**	**5.3**	**6.5 ± 1.9**
	Cosine	10.2	**7.9**	6.8	**6.4**	**6.8**	3.8	5.3	**6.2**	5.0	5.1	6.3 ± 1.8
	Manhattan	6.0	7.3	7.3	4.6	5.2	3.0	**6.3**	3.0	3.7	3.2	5.0 ± 1.7

choice of topics is an open research question. Table 9 shows the results for an increasing number of topics. We do not present the results of the individual ten runs here and only give the mean values.

First, it can be observed that in almost all cases, an increase in the number of topics is accompanied by an improvement in the results, except DRATOMBayes, when changing from 40 to 50 topics. In all cases, DRATOM is superior to all other cases. In all other cases, DRATOMBayes is superior to DRETOM based on *Gensim* but does not reach the quality of the *MALLET* variant. The θ-independent variant can outperform the *MALLET* variant only for $K = 30$ and $K = 40$. Also, with an increasing number of topics, DRASIM is inferior, but this approach benefits most from an increase of K.

6 Discussion

In this section, we recapitulate the results of our previous experiments to answer the four research questions stated at the beginning of Section 5. This leads to guidelines for applying topic models for bug triaging tasks. We further discuss internal and external threats to validity.

Table 9: Impact of a varying number of topics K on MRR@10 obtained on the Mozilla Firefox project with 10-fold cross validation. DRATOM, DRATOMBayes and DRASIM use the same trained ATM. The mean accuracy over the cross-validation and standard deviation are reported. The best hyperparameter setting for θ and the best distance metric for DRASIM are highlighted.

Model	Parameter	K=20	K=30	K=40	K=50	K=60
DRETOM *Gensim*	θ=0.0	10.4 ± 2.8	11.1 ± 2.7	11.6 ± 2.7	11.9 ± 3.2	11.9 ± 2.6
	θ=0.1	**11.0 ± 3.0**	**11.6 ± 2.9**	**12.0 ± 2.8**	**12.3 ± 3.4**	12.2 ± 2.8
	θ=0.2	10.1 ± 2.7	11.3 ± 2.9	11.8 ± 2.9	12.1 ± 3.5	**12.3 ± 2.9**
	θ=0.3	8.7 ± 2.1	10.3 ± 2.9	11.0 ± 2.7	11.5 ± 3.4	11.7 ± 2.8
	θ=0.4	7.7 ± 1.6	9.3 ± 2.7	10.0 ± 2.4	10.8 ± 3.1	10.9 ± 2.7
	θ=0.5	6.9 ± 1.3	8.4 ± 2.5	9.0 ± 2.0	9.7 ± 2.7	10.0 ± 2.3
	θ=0.6	6.4 ± 1.3	7.6 ± 2.1	8.1 ± 1.7	8.7 ± 2.3	9.1 ± 2.0
	θ=0.7	6.0 ± 1.4	6.9 ± 1.7	7.4 ± 1.5	8.0 ± 1.9	8.2 ± 1.7
	θ=0.8	5.7 ± 1.4	6.4 ± 1.6	6.8 ± 1.5	7.3 ± 1.6	7.5 ± 1.4
	θ=0.9	5.5 ± 1.5	6.1 ± 1.6	6.3 ± 1.6	6.7 ± 1.5	6.9 ± 1.3
	θ=1.0	5.3 ± 1.4	5.7 ± 1.5	6.0 ± 1.5	6.4 ± 1.5	6.4 ± 1.2
DRETOM *MALLET*	θ=0.0	12.7 ± 3.3	13.4 ± 3.1	14.3 ± 3.6	15.0 ± 3.9	15.8 ± 3.7
	θ=0.1	**13.7 ± 3.4**	**14.2 ± 3.1**	15.1 ± 3.8	15.8 ± 4.1	16.6 ± 3.9
	θ=0.2	13.2 ± 3.1	14.1 ± 2.7	**15.1 ± 3.7**	**15.8 ± 3.9**	**16.7 ± 3.9**
	θ=0.3	12.0 ± 2.7	13.1 ± 2.2	14.3 ± 3.4	15.0 ± 3.3	16.4 ± 3.6
	θ=0.4	10.3 ± 2.3	11.9 ± 2.0	13.0 ± 2.9	14.0 ± 2.5	15.6 ± 3.1
	θ=0.5	9.3 ± 2.2	10.9 ± 1.7	11.9 ± 2.5	12.8 ± 2.0	14.6 ± 2.6
	θ=0.6	8.6 ± 2.2	9.9 ± 1.5	10.7 ± 2.2	11.7 ± 1.5	13.5 ± 2.2
	θ=0.7	7.9 ± 2.3	9.1 ± 1.6	9.8 ± 2.1	10.7 ± 1.3	12.3 ± 1.8
	θ=0.8	7.5 ± 2.3	8.5 ± 1.9	9.3 ± 2.1	9.9 ± 1.5	11.3 ± 1.9
	θ=0.9	7.2 ± 2.3	8.2 ± 2.0	8.7 ± 2.1	9.3 ± 1.6	10.5 ± 2.0
	θ=1.0	6.8 ± 2.2	7.7 ± 2.1	8.2 ± 2.0	8.8 ± 1.6	9.9 ± 2.3
DRATOM	θ=0.0	14.0 ± 2.7	15.5 ± 3.8	15.8 ± 3.8	15.5 ± 3.8	16.0 ± 3.3
	θ=0.1	**14.2 ± 2.6**	15.8 ± 3.8	16.5 ± 3.8	16.1 ± 3.9	16.6 ± 3.4
	θ=0.2	14.0 ± 2.5	**15.9 ± 3.8**	16.9 ± 3.9	16.5 ± 3.9	17.1 ± 3.4
	θ=0.3	13.7 ± 2.4	15.8 ± 3.7	**16.9 ± 3.7**	16.7 ± 3.8	17.4 ± 3.3
	θ=0.4	13.2 ± 2.4	15.6 ± 3.6	16.8 ± 3.6	**16.7 ± 3.6**	**17.6 ± 3.2**
	θ=0.5	12.8 ± 2.5	15.2 ± 3.3	16.7 ± 3.5	16.7 ± 3.5	17.6 ± 3.2
	θ=0.6	12.4 ± 2.7	14.9 ± 3.3	16.4 ± 3.4	16.3 ± 3.4	17.5 ± 2.9
	θ=0.7	11.6 ± 2.7	14.3 ± 3.2	15.7 ± 3.1	16.1 ± 3.2	17.0 ± 2.7
	θ=0.8	10.6 ± 2.4	13.3 ± 2.4	14.6 ± 2.2	15.5 ± 2.6	16.3 ± 2.1
	θ=0.9	9.9 ± 2.2	12.2 ± 2.3	13.7 ± 1.9	14.9 ± 2.2	15.5 ± 2.0
	θ=1.0	9.3 ± 2.0	11.4 ± 2.2	13.0 ± 1.5	14.2 ± 2.2	14.6 ± 1.8
DRATOMBayes	n/a	**13.3 ± 2.3**	**14.9 ± 2.9**	**16.0 ± 4.2**	**15.8 ± 4.2**	**16.1 ± 3.7**
DRASIM	Jensen Shannon	**6.7 ± 2.0**	**8.6 ± 1.5**	10.8 ± 2.6	**12.0 ± 2.4**	**12.7 ± 2.1**
	Cosine	6.6 ± 2.0	8.6 ± 1.6	**10.9 ± 2.7**	12.0 ± 2.3	12.6 ± 2.0
	Manhattan	5.1 ± 1.9	6.9 ± 1.7	8.5 ± 2.3	10.1 ± 2.5	10.8 ± 2.3

6.1 Main Findings

The first research question concerns the influence of the topic modeling implementation and inference technique for the bug triaging algorithm. In our first experiment, we compared three topic modeling libraries written in different programming languages. The results indicate that the CGS-based implementation *MALLET* is superior to the VB-based implementations *Vowpal Wabbit* and *Gensim*. The difference between the VB-based implementations is neglectable in our experiments. We thus recommend choosing a topic modeling library based on its underlying inference technique rather than its programming language. In our second experiment, we investigated whether the inferior results of the VB-based implementations are due to the non-convergence of the training algorithm and thus increased the number of iterations. It turned out that all VB-based implementations benefit from adjusting the parameter. Therefore, we conclude that the default values in the topic modeling libraries might not be ideal for sufficiently training a topic model. In our third experiment, the batch variant of VB outperforms its online version. We, therefore, recommend the usage of the batch variant if possible and only suggest the use of the online variant for text corpora that are too large to process.

In our fourth and fifth experiments, we compared DRETOM with the three approaches based on the ATM. We referred to *Gensim*'s implementation of the ATM based on VB. Though VB was inferior to CGS in the first three experiments, our approach DRATOM outperformed the DRETOM implementation based on *MALLET*. This effect can be traced back to the underlying topic model. The approach, DRATOMBayes, outperforms the DRETOM variant based on *Gensim* and achieves similar results as the *MALLET* implementation. However, the DRASIM approach is inferior to all algorithms compared in our study.

The last two experiments also investigate the influence of the hyperparameters α and K on the results of the bug triaging algorithms. The asymmetric Dirichlet prior α led to only a slight quality improvement in DRETOM. In contrast, all methods benefited significantly from an increasing number of topics. Overall, we conclude that the hyperparameter K significantly influences the results and should therefore be given special attention.

Xie et al. reported that their approach DRETOM is sensitive to its hyperparameter θ. Based on their experimental results, they formulated statements about its influence on the bug triaging task. As a further result of our study, we revisit four statements about the trade-off parameter θ made by Xie et al. Their first statement reads "The smaller the size of the set of recommended developers is, the larger θ should be." In our experiments, the number of developers is more likely to increase over time, because the training set grows monotonically. However, we do not observe that the parameter θ decreases. Furthermore, the evaluation of a bug triaging technique should be independent of the number of recommended developers [27]. Xie et al. evaluated the effectiveness of their proposed approach DRETOM using Precision@k, Recall@k, and its harmonic mean F_1 score [39]. Although those metrics allow taking multiple ground truth assignees into account, they do not penalize mistakes in the first ranks more than in the following ranks. Therefore, we suspect that the statement depends on the chosen evaluation metric.

The second statement is given by "The shorter the duration of a project's bug repository is, the larger θ should be." Examining this statement independently of the other parameters that affect the optimal value for θ is not easy. We see counterindications with this statement using the hyperparameters for DRETOM used by Xie et al. Because of the evaluation scheme we have chosen, we let the duration, and thus the size of the project's bug repository, grow with each cross-validation fold. Thus, a clear decreasing trend of the optimal θ should be found in the experiments. However, we observe this only very sporadically, e.g., for DRETOM using *Gensim*, but inconsistently between the individual implementations and used inference techniques. Although some experiments suggest such a relationship, a more thorough evaluation would be needed to rule out other influencing factors.

A further statement about θ reads "Prior to putting DRETOM into practice, some trials should be conducted to find out the optimal θ." This recommendation includes the hidden assumption that DRETOM has an optimal value for θ, although this is admittedly dependent on the chosen project. However, our

experiments clearly show that the identification of this *optimal* value depends several different factors, including the chosen implementation, the used inference technique and its hyperparameters, the LDA hyperparameters, and most importantly, even if we leave all other factors aside, the point in time during the project.

The last statement is given by "Setting θ in the range from 0.2 to 0.8 is appropriate." Following this recommendation, neither the developer expertise nor the interest of a developer alone is sufficient to make the best possible developer recommendation using DRETOM. However, as long as we do not adjust for the number of topics according to the dataset, on average, across all experiments and all cross-validation folds, the developer expertise has to be weighted much more to make the best possible predictions. During the experiments, the on average optimal value of θ with regard to the MRR@10 is less or equal to 0.1 for DRETOM and DRATOM. Only if we choose an appropriate number of topics, the optimal value of θ is, on average, within the recommended range. Due to the inconsistent and partially conflicting choice of an optimal value for θ, we refer to the more robust DRATOM and DRASIM, which do not require an additional trade-off hyperparameter.

6.2 Threats to Validity

The internal threat to validity mainly consists in the implementation of the studied baseline approach DRETOM. For comparison purposes, we reimplemented the algorithm DRETOM, as there is no implementation publicly available. Although we have implemented the algorithm true to the best of our knowledge, there could be some minor deviations from the original approach.

In our experiments, we were able to access three different LDA libraries. It turned out that *MALLET* generates better results than *Gensim* and *Vowpal Wabbit*. In contrast, we evaluated only a single implementation of ATM. Thus, an accurate comparison is only possible with the LDA of *Gensim*. Based on the results of the first three experiments, we expect that a CGS-based implementation of ATM would yield better results.

The main threat to external validity is the used dataset. Additional research on other projects, such as Chromium or Eclipse, should be consulted to reinforce the generalizability of our conclusions. Furthermore, we assume the final assignee as the ground truth in our experiments. However, bug triaging is a collaborative task involving various stakeholders, e.g., the reporter or other developers participating in the discussion. We see no clear way to generalize to a setting where more than one developer can be seen as ground truth.

7 Conclusions and Future Work

The analysis of software data by machine learning techniques promises to provide insights that will make future development processes more efficient. One concrete task that is particularly important in the maintenance phase of a software project

is the automated assignment of pending bugs to suitable developers. Numerous approaches apply topic models, which analyze the textual components of a bug report, and thus allow for a formal treatment of the bug triaging task. Topic models differ among themselves, their underlying inference algorithm, the software libraries used, and the choice of their hyperparameters. To deduce guidelines for the effective use of topic models for bug triaging tasks, we compared four bug triaging algorithms based on LDA and its variant ATM proposed by Xie et al. [39] and Atzberger and Schneider et al. [2]. In our experiments on a dataset taken from the *Mozilla Firefox* project, we evaluated the influence of the chosen topic modeling library and its inference technique, the influence of the topic model itself, the role of the Dirichlet prior α, and the number of topics K.

Our study results show that the topic modeling library should be chosen based on the implemented inference technique rather than its programming language. In our concrete case, the LDA implementation provided by *MALLET* achieved better results than the implementations provided by *Gensim* and *Vowpal Wabbit*. As the approach DRATOM that relies on the ATM was superior to the baseline approach DRETOM, we observe that the choice between LDA and the ATM has a significant effect. In our experiments, the choice of the Dirichlet prior α led to no significant improvement, whereas the number of topics K had a large effect on the bug triaging results. In addition to the specific use case of automated assignment of bug reports, our experiments show that a closer look at the topic models used when examining textual components of software data have the potential to improve the analyses significantly. Instead of considering topic models as black boxes, the used implementations and hyperparameters should be compared and adapted for the individual use case.

One approach for future work is to extend the presented methods by adding categorical attributes of bug reports, for example, by developing novel topic models. Alternatively, it would be conceivable to integrate other repositories, such as coding activities, into the triaging process in addition to ticket management systems. A particular challenge here is the modeling of heterogeneous data sources. To increase the acceptance of the processes in the application, thoughts should also be given to how the results can be made explainable. Although the application of ATM enables a description of developers as a distribution over the topics, concrete conclusions about the data basis have not yet been implemented.

Acknowledgements. This work is part of the "Software-DNA" project, which is funded by the European Regional Development Fund (ERDF or EFRE in German) and the State of Brandenburg (ILB). This work is part of the KMU project "KnowhowAnalyzer" (Förderkennzeichen 01IS20088B), which is funded by the German Ministry for Education and Research (Bundesministerium für Bildung und Forschung).

References

1. Aktas, E., Yilmaz, C.: Automated issue assignment: results and insights from an industrial case. Empirical Software Engineering **25**(5), 3544–3589 (2020).

https://doi.org/10.1007/s10664-020-09846-3
2. Atzberger, D., Schneider, J., Scheibel, W., Limberger, D., Trapp, M., Döllner, J.: Mining developer expertise from bug tracking systems using the author-topic model. In: Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering. pp. 107–118. ENASE '22, INSTICC, SciTePress (2022). https://doi.org/10.5220/0011045100003176
3. Banitaan, S., Alenezi, M.: TRAM: An approach for assigning bug reports using their metadata. In: Proc. 3rd International Conference on Communications and Information Technology. pp. 215–219. ICCIT '13, IEEE (2013). https://doi.org/10.1109/ICCITechnology.2013.6579552
4. Bhattacharya, P., Neamtiu, I.: Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. In: Proc. International Conference on Software Maintenance. pp. 1–10. ICSM '10, IEEE (2010). https://doi.org/10.1109/ICSM.2010.5609736
5. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet Allocation. Journal of Machine Learning Research **3**, 993–1022 (2003)
6. Chen, T.H., Thomas, S.W., Hassan, A.E.: A survey on the use of topic models when mining software repositories. Empirical Software Engineering **21**(5), 1843–1919 (2016). https://doi.org/10.1007/s10664-015-9402-8
7. Dedik, V., Rossi, B.: Automated bug triaging in an industrial context. In: Proc. 42th Euromicro Conference on Software Engineering and Advanced Applications. pp. 363–367. SEAA '16, IEEE (2016). https://doi.org/10.1109/SEAA.2016.20
8. Geigle, C.: Inference methods for Latent Dirichlet Allocation (course notes in CS 598 CXZ: Advanced topics in information retrieval). Tech. rep., Department of Computer Science, University of Illinois at Urbana-Champaign (2016)
9. Griffiths, T.L., Steyvers, M.: Finding scientific topics. In: Proc. National Academy of Sciences. vol. 101, pp. 5228–5235 (4 2004). https://doi.org/10.1073/pnas.0307752101
10. Hoffman, M., Bach, F., Blei, D.: Online learning for Latent Dirichlet Allocation. In: Advances in Neural Information Processing Systems. NIPS '10, vol. 23, pp. 856–864. Curran Associates, Inc. (2010)
11. Hu, H., Zhang, H., Xuan, J., Sun, W.: Effective bug triage based on historical bug-fix information. In: Proc. 25th International Symposium on Software Reliability Engineering. pp. 122–132. ISSRE '14, IEEE (2014). https://doi.org/10.1109/ISSRE.2014.17
12. Jonsson, L., Borg, M., Broman, D., Sandahl, K., Eldh, S., Runeson, P.: Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts. Empirical Software Engineering **21**(4), 1533–1578 (2016). https://doi.org/10.1007/s10664-015-9401-9
13. Kagdi, H., Gethers, M., Poshyvanyk, D., Hammad, M.: Assigning change requests to software developers. Journal of Software: Evolution and Process **24**(1), 3–33 (2012). https://doi.org/10.1002/smr.530
14. Khatun, A., Sakib, K.: A bug assignment technique based on bug fixing expertise and source commit recency of developers. In: Proc. 19th International Conference on Computer and Information Technology. pp. 592–597. ICCIT '16, IEEE (2016). https://doi.org/10.1109/ICCITECHN.2016.7860265
15. Linstead, E., Rigor, P., Bajracharya, S., Lopes, C., Baldi, P.: Mining Eclipse developer contributions via Author-Topic Models. In: Proc. 4th International Workshop on Mining Software Repositories. pp. 1–4. MSR '07, IEEE (2007). https://doi.org/10.1109/MSR.2007.20

16. Mani, S., Sankaran, A., Aralikatte, R.: Deeptriage: Exploring the effectiveness of deep learning for bug triaging. In: Proc. India Joint International Conference on Data Science and Management of Data. pp. 171–179. ACM (2019). https://doi.org/10.1145/3297001.3297023
17. Matter, D., Kuhn, A., Nierstrasz, O.: Assigning bug reports using a vocabulary-based expertise model of developers. In: Proc. 6th International Working Conference on Mining Software Repositories. pp. 131–140. MSR '09, IEEE (2009). https://doi.org/10.1109/MSR.2009.5069491
18. McCallum, A.K.: MALLET: A Machine Learning for Language Toolkit (2002), http://www.cs.umass.edu/%7Emccallum/mallet
19. Mortensen, O.: The Author-Topic Model. Master's thesis, Technical University of Denmark, Department of Applied Mathematics and Computer Science (2017)
20. Naguib, H., Narayan, N., Brügge, B., Helal, D.: Bug report assignee recommendation using activity profiles. In: Proc. 10th Working Conference on Mining Software Repositories. pp. 22–30. MSR '13, IEEE (2013). https://doi.org/10.1109/MSR.2013.6623999
21. Newman, D., Asuncion, A., Smyth, P., Welling, M.: Distributed algorithms for topic models. Journal of Machine Learning Research **10**, 1801–1828 (2009)
22. Nguyen, T.T., Nguyen, A.T., Nguyen, T.N.: Topic-based, time-aware bug assignment. SIGSOFT Software Engineering Notes **39**(1), 1–4 (2014). https://doi.org/10.1145/2557833.2560585
23. Ramage, D., Rosen, E., Chuang, J., Manning, C.D., McFarland, D.A.: Topic modeling for the social sciences. In: Proc. Workshop on Applications for Topic Models: Text and Beyond. pp. 23:1–4 (2009)
24. Rehůřek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Proc. Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA (2010)
25. Rosen-Zvi, M., Griffiths, T., Steyvers, M., Smyth, P.: The author-topic model for authors and documents. In: Proc. 20th Conference on Uncertainty in Artificial Intelligence. pp. 487–494. UAI '04, AUAI Press (2004)
26. Sajedi-Badashian, A., Hindle, A., Stroulia, E.: Crowdsourced bug triaging. pp. 506–510. ICSME '15, IEEE (2015). https://doi.org/10.1109/ICSM.2015.7332503
27. Sajedi-Badashian, A., Stroulia, E.: Guidelines for evaluating bug-assignment research. Journal of Software: Evolution and Process **32**(9) (2020). https://doi.org/10.1002/smr.2250
28. Schofield, A., Magnusson, M., Mimno, D.: Pulling out the stops: Rethinking stopword removal for topic models. In: Proc. 15th Conference of the European Chapter of the Association for Computational Linguistics. pp. 432–436. EACL '17, ACL (2017). https://doi.org/10.18653/v1/E17-2069
29. Schofield, A., Mimno, D.: Comparing apples to apple: The effects of stemmers on topic models. Transactions of the Association for Computational Linguistics **4**, 287–300 (2016). https://doi.org/10.1162/tacl_a_00099
30. Schofield, A., Thompson, L., Mimno, D.: Quantifying the effects of text duplication on semantic models. In: Proc. Conference on Empirical Methods in Natural Language Processing. pp. 2737–2747. EMNLP '17, ACL (2017). https://doi.org/10.18653/v1/D17-1290
31. Shokripour, R., Anvik, J., Kasirun, Z., Zamani, S.: Why so complicated? simple term filtering and weighting for location-based bug report assignment recommendation. In: Proc. 10th Working Conference on Mining Software Repositories. pp. 2–11. MSR '13, IEEE (2013). https://doi.org/10.1109/MSR.2013.6623997

32. Syed, S., Spruit, M.: Exploring symmetrical and asymmetrical dirichlet priors for latent dirichlet allocation. International Journal of Semantic Computing **12**(3), 399–423 (2018). https://doi.org/10.1142/S1793351X18400184
33. Tamrawi, A., Nguyen, T.T., Al-Kofahi, J.M., Nguyen, T.N.: Fuzzy set and cache-based approach for bug triaging. In: Proc. 19th SIGSOFT Symposium on Foundations of Software Engineering. pp. 365–375. FSE/ESEC '11, ACM (2011). https://doi.org/10.1145/2025113.2025163
34. Wallach, H.M.: Structured Topic Models for Language. Ph.D. thesis, Newnham College, University of Cambridge (2008)
35. Wallach, H.M., Mimno, D., McCallum, A.K.: Rethinking LDA: Why priors matter. In: Proc. 22nd International Conference on Neural Information Processing Systems. pp. 1973–1981. NIPS '09, Curran Associates, Inc. (2009)
36. Wu, W., Zhang, W., Yang, Y., Wang, Q.: DREX: Developer recommendation with k-nearest-neighbor search and expertise ranking. In: Proc. 18th Asia-Pacific Software Engineering Conference. pp. 389–396. APSEC '11, IEEE (2011). https://doi.org/10.1109/APSEC.2011.15
37. Xia, X., Lo, D., Ding, Y., Al-Kofahi, J.M., Nguyen, T.N., Wang, X.: Improving automated bug triaging with specialized topic model. Transactions on Software Engineering **43**(3), 272–297 (2016). https://doi.org/10.1109/TSE.2016.2576454
38. Xia, X., Lo, D., Wang, X., Zhou, B.: Accurate developer recommendation for bug resolution. In: Proc. 20th Working Conference on Reverse Engineering. pp. 72–81. WCRE '13, IEEE (2013). https://doi.org/10.1109/WCRE.2013.6671282
39. Xie, X., Zhang, W., Yang, Y., Wang, Q.: DRETOM: Developer recommendation based on topic models for bug resolution. In: Proc. 8th International Conference on Predictive Models in Software Engineering. pp. 19–28. PROMISE '12, ACM (2012). https://doi.org/10.1145/2365324.2365329
40. Yang, G., Zhang, T., Lee, B.: Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports. In: Proc. 38th Annual Computer Software and Applications Conference. pp. 97–106. COMPSAC '14, IEEE (2014). https://doi.org/10.1109/COMPSAC.2014.16
41. Yao, L., Mimno, D., McCallum, A.K.: Efficient methods for topic model inference on streaming document collections. In: Proc. SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 937–945. ACM (2009). https://doi.org/10.1145/1557019.1557121
42. Zhang, T., Chen, J., Yang, G., Lee, B., Luo, X.: Towards more accurate severity prediction and fixer recommendation of software bugs. Journal of Systems and Software **117**, 166–184 (2016). https://doi.org/10.1016/j.jss.2016.02.034
43. Zhang, T., Yang, G., Lee, B., Lua, E.K.: A novel developer ranking algorithm for automatic bug triage using topic model and developer relations. In: Proc. 21st Asia-Pacific Software Engineering Conference. pp. 223–230. APSEC '14, IEEE (2014). https://doi.org/10.1109/APSEC.2014.43
44. Zhang, W., Han, G., Wang, Q.: BUTTER: An approach to bug triage with topic modeling and heterogeneous network analysis. In: Proc. International Conference on Cloud Computing and Big Data. pp. 62–69. CCBD '14, IEEE (2014). https://doi.org/10.1109/CCBD.2014.14
45. Zou, W., Lo, D., Chen, Z., Xia, X., Feng, Y., Xu, B.: How practitioners perceive automated bug report management techniques. Transactions on Software Engineering **46**(8), 836–862 (2020). https://doi.org/10.1109/TSE.2018.2870414