Empirical Software Engineering (2021) 26: 120
https://doi.org/10.1007/s10664-021-10026-0

Topic modeling in software engineering research

Camila Costa Silva[1] · Matthias Galster[1] · Fabian Gilson[1]

Accepted: 29 July 2021 / Published online: 6 September 2021

Abstract

Topic modeling using models such as Latent Dirichlet Allocation (LDA) is a text mining technique to extract human-readable semantic "topics" (i.e., word clusters) from a corpus of textual documents. In software engineering, topic modeling has been used to analyze textual data in empirical studies (e.g., to find out what developers talk about online), but also to build new techniques to support software engineering tasks (e.g., to support source code comprehension). Topic modeling needs to be applied carefully (e.g., depending on the type of textual data analyzed and modeling parameters). Our study aims at describing how topic modeling has been applied in software engineering research with a focus on four aspects: (1) which topic models and modeling techniques have been applied, (2) which textual inputs have been used for topic modeling, (3) how textual data was "prepared" (i.e., pre-processed) for topic modeling, and (4) how generated topics (i.e., word clusters) were named to give them a human-understandable meaning. We analyzed topic modeling as applied in 111 papers from ten highly-ranked software engineering venues (five journals and five conferences) published between 2009 and 2020. We found that (1) LDA and LDA-based techniques are the most frequent topic modeling techniques, (2) developer communication and bug reports have been modelled most, (3) data pre-processing and modeling parameters vary quite a bit and are often vaguely reported, and (4) manual topic naming (such as deducting names based on frequent words in a topic) is common.

Keywords Topic modeling · Text mining · Natural language processing · Literature analysis

Communicated by: Andrea De Lucia

✉ Camila Costa Silva
camila.costasilva@pg.canterbury.ac.nz

Matthias Galster
mgalster@ieee.org

Fabian Gilson
fabian.gilson@canterbury.ac.nz

[1] University of Canterbury, Christchurch, New Zealand

1 Introduction

Text mining is about searching, extracting and processing text to provide meaningful insights from the text based on a certain goal. Techniques for text mining include natural language processing (NLP) to process, search and understand the structure of text (e.g., part-of-speech tagging), web mining to discover information resources on the web (e.g., web crawling), and information extraction to extract structured information from unstructured text and relationships between pieces of information (e.g., co-reference, entity extraction) (Miner et al. 2012). Text mining has been widely used in software engineering research (Bi et al. 2018), for example, to uncover architectural design decisions in developer communication (Soliman et al. 2016) or to link software artifacts to source code (Asuncion et al. 2010).

Topic modeling is a text mining and concept extraction method that extracts *topics* (i.e., coherent word clusters) from large corpora of textual documents to discovery hidden semantic structures in text (Miner et al. 2012). An advantage of topic modeling over other techniques is that it helps analyzing long texts (Treude and Wagner 2019; Miner et al. 2012), creates clusters as "topics" (rather than individual words) and is unsupervised (Miner et al. 2012).

Topic modeling has become popular in software engineering research (Sun et al. 2016; Chen et al. 2016). For example, Sun et al. (2016) found that topic modeling had been used to support source code comprehension, feature location and defect prediction. Additionally, Chen et al. (2016) found that many repository mining studies apply topic modeling to textual data such as source code and log messages to recommend code refactoring (Bavota et al. 2014b) or to localize bugs (Lukins et al. 2010).

Probabilistic topic models such as Latent Semantic Indexing (LSI) (Deerwester et al. 1990) and Latent Dirichlet Allocation (LDA) (Blei et al. 2003b) discover topics in a corpus of textual documents, using the statistical properties of word frequencies and co-occurrences (Lin et al. 2014). However, Agrawal et al. (2018) warn about systematic errors in the analysis of LDA topic models that limit the validity of topics. Lin et al. (2014) also advise that classical topic models usually generate sub-optimal topics when applied "as is" to small amounts or short text documents.

Considering the limitations of topic modeling techniques and topic models on the one hand and their potential usefulness in software engineering on the other hand, our goal is to describe how topic modeling has been applied in software engineering research. In detail, we explore the following research questions:

- **RQ1. Which topic modeling techniques have been used and for what purpose?** There are different topic modeling techniques (see Section 2), each with their own limitations and constraints (Chen et al. 2016). This RQ aims at understanding which topic modeling techniques have been used (e.g., LDA, LSI) and for what purpose studies applied such techniques (e.g., to support software maintenance tasks). Furthermore, we analyze the types of contributions in studies that used topic modeling (e.g., a new approach as a solution proposal, or an exploratory study).
- **RQ2. What are the inputs into topic modeling?** Topic modeling techniques accept different types of textual documents and require the configuration of parameters (see Section 2.1). Carefully choosing parameters (such as the number of topics to be generated) is essential for obtaining valuable and reliable topics (Agrawal et al. 2018; Treude and Wagner 2019). This RQ aims at analysing types of textual data (e.g., source code), actual documents (e.g., a Java class or an individual Java method) and configured parameters used for topic modeling to address software engineering problems.

- **RQ3: How are data pre-processed for topic modeling?** Topic modeling requires that the analyzed text is pre-processed (e.g., by removing stop words) to improve the quality of the produced output (Aggarwal and Zhai 2012; Bi et al. 2018). This RQ aims at analysing how previous studies pre-processed textual data for topic modeling, including the steps for cleaning and transforming text. This will help us understand if there are specific pre-processing steps for a certain topic modeling technique or types of textual data.
- **RQ4. How are generated topics named?** This RQ aims at analyzing if and how topics (word clusters) were named in studies. Giving meaningful names to topics may be difficult but may be required to help humans comprehend topics. For example, naming topics can provide a high-level view on topics discussed by developers in Stack Overflow (a Q&A website) (Barua et al. 2014) or by end mobile app users in tweets (Mezouar et al. 2018). Analysts (e.g., developers interested in what topics are discussed on Stack Overflow or app reviews) can then look at the name of the topic (i.e., its "label") rather than the cluster of words. These labels or names must capture the overarching meaning of all words in a topic. We describe different approaches to naming topics generated by a topic model, such as manual or automated labeling of clusters with names based on the most frequent words of a topic (Hindle et al. 2013).

In this paper, we provide an overview of the use of topic modeling in 111 papers published between 2009 and 2020 in highly ranked venues of software engineering (five journals and five conferences). We identify characteristics and limitations in the use of topic models and discuss (a) the appropriateness of topic modeling techniques, (b) the importance of pre-processing, (c) challenges related to defining meaningful topics, and (d) the importance of context when manually naming topics.

The rest of the paper is organized as follows. In Section 2 we provide an overview of topic modeling. In Section 3 we describe other literature reviews on the topic as well as "meta-studies" that discuss topic modeling more generally. We describe the research method in Section 4 and present the results in Section 5. In Section 6, we summarize our findings and discuss implications and threats to validity. Finally, in Section 7 we present concluding remarks and future work.

2 Topic Modeling

Topic modeling aims at automatically finding topics, typically represented as clusters of words, in a given textual document (Bi et al. 2018). Unlike (supervised) machine learning-based techniques that solve classification problems, topic modeling does not use tags, training data or predefined taxonomies of concepts (Bi et al. 2018). Based on the frequencies of words and frequencies of co-occurrence of words within one or more documents, topic modeling clusters words that are often used together (Barua et al. 2014; Treude and Wagner 2019). Figure 1 illustrates the general process of topic modeling, from a raw corpus of documents ("Data input") to topics generated for these documents ("Output"). Below we briefly introduce the basic concepts and terminology of topic modeling (based on Chen et al. (2016)):

- Word w: a string of one or more alphanumeric characters (e.g., "software" or "management");
- Document d: a set of n words (e.g., a text snippet with five words: w_1 to w_5);

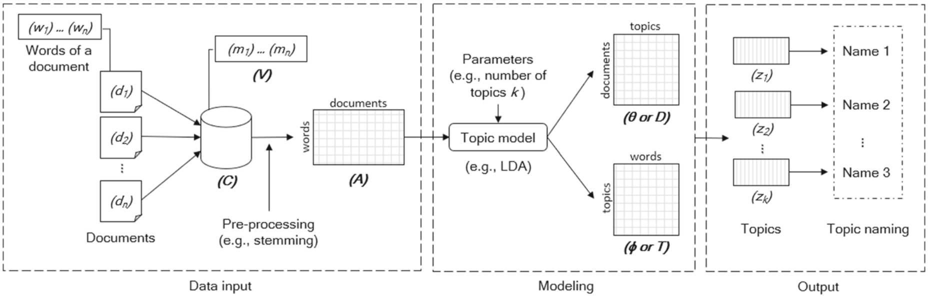

Fig. 1 General topic modeling process

- Corpus C: a set of t documents (e.g., nine text snippets: d_1 to d_9);
- Vocabulary V: a set of m unique words that appear in a corpus (e.g., $m = 80$ unique words across nine documents);
- Term-document matrix A: an m by t matrix whose $A_{i,j}$ entry is the weight (according to some weighting function, such as term-frequency) of word w_i in document d_j. For example, given a matrix A with three words and three documents as

	d_1	d_2	d_3
$w_1 = code$	5	1	4
$w_2 = develop$	4	3	1
$w_3 = test$	1	2	1

  $A_{1,1} = 5$ indicates that "code" appears five times in d_1, etc.;
- Topic z: a collection of terms that co-occur frequently in the documents of a corpus. Considering probabilistic topic models (e.g., LDA), z refers to an m-length vector of probabilities over the vocabulary of a corpus. For example, in a vector $z_1 = (code : 0.35; test : 0.17; bug : 0.08)$,

  0.35 indicates that when a word is picked from a topic z_1, there is a 35% chance of drawing the word "code", etc.;
- Topic-term matrix ϕ (or T): a k by m matrix with k as the number of topics and $\phi_{i,j}$ the probability of word w_j in topic z_i. Row i of ϕ corresponds to z_i. For example, given a matrix ϕ as

	$w_1 = code$	$w_2 = develop$	$w_3 = test$
z_1	0.25	0.10	0.80
z_2	0.10	0.00	0.05
z_3	0.05	0.20	0.09

  0.05 in the first column indicates that the word "code" appears with a probability of 0.5% in topic z_3, etc.;
- Topic membership vector θ_d: for document d_i, a k-length vector of probabilities of the k topics. For example, given a vector $\theta_{d_i} = (z_1 : 0.25; z_2 : 0.10; z_3 : 0.08)$,

  0.25 indicates that there is a 25% chance of selecting topic z_1 in d_i;
- Document-topic matrix θ (or D): an n by k matrix with $\theta_{i,j}$ as the probability of topic z_j in document d_i. Row i of θ corresponds to θ_{d_i}. For example, given a matrix θ as

	z_1	z_2	z_3
d_1	0.25	0.10	0.80
d_2	0.10	0.00	0.05
d_3	0.05	0.00	0.00

0.10 in the first column indicates that document d_2 contains topic z_1 with probability of 10%, etc.

2.1 Data Input

Data used as input into topic modeling can take many forms. This requires decisions on what exactly are documents and what the scope of individual documents is (Miner et al. 2012). Therefore, we need to determine which unit of text shall be analyzed (e.g., subject lines of e-mails from a mailing list or the body of e-mails).

To model topics from raw text in a corpus C (see Fig. 1), the data needs to be converted into a structured vector-space model, such as the term-document matrix A. This typically also requires some pre-processing. Although each text mining approach (including topic modeling) may require specific pre-processing steps, there are some common steps, such as tokenization, stemming and removing stop words (Miner et al. 2012). We discuss pre-processing for topic modeling in more detail when presenting the results for RQ3 in Section 5.4.

2.2 Modeling

Different models can be used for topic modeling. Models typically differ in how they model topics and underlying assumptions. For example, besides LDA and LSI mentioned before, other examples of topic modeling techniques include Probabilistic Latent Semantic Indexing (pLSI) (Hofmann 1999). LSI and pLSI reduce the dimensionality of A using Singular Value Decomposition (SVD) (Hofmann 1999). Furthermore, variants of LDA have been proposed, such as Relational Topic Models (RTM) (Chang and Blei 2010) and Hierarchical Topic Models (HLDA) (Blei et al. 2003a). RTM finds relationships between documents based on the generated topics (e.g., if document d_1 contains the topic “microservices”, document d_2 contains the topic “containers” and document d_n contains the topic “user interface”, RTM will find a link between documents d_1 and d_2 (Chang and Blei 2010)). HLDA discovers a hierarchy of topics within a corpus, where each lower level in the hierarchy is more specific than the previous one (e.g., a higher topic “web development” may have subtopics such as “front-end” and “back-end”).

Topic modeling techniques need to be configured for a specific problem, objectives and characteristics of the analyzed text (Treude and Wagner 2019; Agrawal et al. 2018). For example, Treude and Wagner (2019) studied parameters, characteristics of text corpora and how the characteristics of a corpus impact the development of a topic modeling technique using LDA. Treude and Wagner (2019) found that textual data from Stack Overflow (e.g., threads of questions and answers) and GitHub (e.g., README files) require different configurations for the number of generated topics (k). Similarly, Barua et al. (2014) argued that the number of topics depends on the characteristics of the analyzed corpora. Furthermore, the values of modeling parameters (e.g., LDA’s hyperparameters α and β which control an initial topic distribution) can also be adjusted depending on the corpus to improve the quality of topics (Agrawal et al. 2018).

2.3 Output

By finding words that are often used together in documents in a corpus, a topic modeling technique creates clusters of words or *topics* z_k. Words in such a cluster are usually related in some way, therefore giving the topic a meaning. For example, we can use a topic modeling technique to extract five topics from unstructured document such as a combination of Stack Overflow posts. One of the clusters generated could include the co-occurring words “error”, “debug” and “warn”. We can then manually inspect this cluster and by inference suggest the label “Exceptions” to name this topic (Barua et al. 2014).

3 Related Work

3.1 Previous Literature Reviews

Sun et al. (2016) and Chen et al. (2016), similar to our study, surveyed software engineering papers that applied topic modeling. Table 1 shows a comparison between our study and prior reviews. As shown in the table, Sun et al. (2016) focused on finding which software engineering tasks have been supported by topic models (e.g., support source code comprehension, feature location, traceability link recovery, refactoring, software testing, developer recommendations, software defects prediction and software history comprehension), and Chen et al. (2016) focused on characterizing how studies used topic modeling to mine software repositories.

Furthermore, as shown in Table 1, in comparison to Sun et al. (2016) and Chen et al. (2016), our study surveys the literature considering other aspects of topic modeling such as data inputs (RQ2), data pre-processing (RQ3), and topic naming (RQ4). Additionally, we searched for papers that applied topic models to any type of data (e.g., Q&A websites) rather than to data in software repositories. We also applied a different search process to identify relevant papers.

Although some of the search venues of these two previous studies and our study overlap, our search focused on specific venues. We also searched papers published between 2009 and 2020, a period which only partially overlaps with the searches presented by Sun et al. (2016) and Chen et al. (2016).

Regarding the data analysed in previous studies, Chen et al. (2016) analyzed two aspects not covered in our study: (a) tools to implement topic models in papers, and (b) how papers evaluated topic models (note that even though we did not cover this aspect explicitly, we checked whether papers compared different topic models, and if so, what metrics they used to compare topic models). However, different to Chen et al. (2016) we analyzed (a) the types of contribution of papers (e.g., a new approach); (b) details about the types of data and documents used in topic modeling techniques, and (c) whether and how topics were named. Additionally, we extend the survey of Chen et al. (2016) by investigating hyperparameters (see Section 2.1) of topic models and data pre-processing in more detail. We provide more details and a justification of our research method in Section 4.

3.2 Meta-studies on Topic Modeling

In addition to literature surveys, there are “meta-studies” on topic modeling that address and reflect on different aspects of topic modeling more generally (and are not considered primary studies for the purpose of our review, see our inclusion and exclusion criteria in Section 4). In the following paragraphs we organized their discussion into three parts: (1) studies about parameters for topic modeling, (2) studies on topic models based on the type of analyzed data, and (3) studies about metrics and procedures to evaluate the performance of topic models. We refer to these studies throughout this manuscript when reflecting on the findings of our study.

Table 1 Comparison to previous reviews

	(Sun et al. 2016)	(Chen et al. 2016)	This study
Reviewed time range	2003-2015	1999-2014	2009-2020
Search venues	4 journals	6 journals	5 journals
	9 conferences	9 conferences	5 conferences
Papers analysed	38	167	111
Analysed data items			
Topic modeling technique	✓	✓	✓
Supported tasks	Specific (e.g., feature localization)	Specific and high-level (e.g., feature localization (specific) under concept localization (high-level))	High-level (e.g., documentation)
Type of contribution	–	–	✓
Tools used	–	✓	–
Types of data and documents	–	–	✓
Parameters used	–	Number of topics	Number of topics Hyperparameters
Data pre–processing		General analysis	Detailed analysis
Topic naming	–	–	✓
Evaluation of topic models	–	✓	–

Regarding parameters used for topic modeling, Treude and Wagner (2019) performed a broad study on LDA parameters to find optimal settings when analyzing GitHub and Stack Overflow text corpora. The authors found that popular rules of thumb for topic modeling parameter configuration were not applicable to their corpora, which required different configurations to achieve good model fit. They also found that it is possible to predict good configurations for unseen corpora reliably. Agrawal et al. (2018) also performed experiments on LDA parameter configurations and proposed LDADE, a tool to tune the LDA parameters. The authors found that due to LDA topic model instability, using standard LDA with "off-the-shelf" settings is not advisable. We also discuss parameters for topic modeling in Section 2.2.

For studies on topic models based on the analyzed data, researchers have investigated topic modeling involving short texts (e.g., a tweet) and how to improve the performance of topic models that work well with longer text (e.g., a book chapter) (Lin et al. 2014). For example, the study of Jipeng et al. (2020) compared short-text topic modeling techniques and developed an open-source library of the short-text models. Another example is the work of Mahmoud and Bradshaw (2017) who discussed topic modeling techniques specific for source code.

Finally, regarding metrics and procedures to evaluate the performance of topic models, some works have explored how semantically meaningful topics are for humans (Chang et al. 2009). For example, Poursabzi-Sangdeh et al. (2021) discuss the importance of interpretability of models in general (also considering other text mining techniques). Another example is the work of Chang et al. (2009) who presented a method for measuring the interpretability of a topic model based on how well words within topics are related and how different topics are between each other. On the other hand, as an effort to quantify the interpretability of topics without human evaluation, some studies developed *topic coherence metrics*. These metrics score the probability of a pair of words from topics being found together in (a) external data sources (e.g., Wikipedia pages) or (b) in the documents used by the model that generated those topics (Röder et al. 2015). Röder et al. (2015) combined different implementations of coherence metrics in a framework. *Perplexity* is another measure of performance for statistical models in natural language processing, which indicates the uncertainty in predicting a single word (Blei et al. 2003b). This metric is often applied to compare the configurations of a topic modeling technique (e.g., Zhao et al. (2020)). Other studies use perplexity as an indicator of model quality (such as Chen et al. 2019 and Yan et al. 2016b).

4 Research Method

We conducted a literature survey to describe how topic modeling has been applied in software engineering research. To answer the research questions introduced in Section 1, we followed general guidelines for systematic literature review (Kitchenham 2004) and mapping study methods (Petersen et al. 2015). This was to systematically identify relevant works, and to ensure traceability of our findings as well as the repeatability of our study. However, we do not claim to present a fully-fledged systematic literature review (e.g., we did not assess the quality of primary studies) or a mapping study (e.g., we only analyzed papers from carefully selected venues). Furthermore, we used parts of the procedures from other literature surveys on similar topics (Bi et al. 2018; Chen et al. 2016; Sun et al. 2016) as discussed throughout this section.

4.1 Search Procedure

To identify relevant research, we selected high-quality software engineering publication venues. This was to ensure that our literature survey includes studies of high quality and described at sufficient level of detail. We identified venues rated as A and A^* for Computer Science and Information Systems research in the Excellence Research for Australia (CORE) ranking (ARC 2012). Only one journal was rated B (IST), but we included it due to its relevance for software engineering research. These venues are a subset of venues also searched by related previous literature surveys (Chen et al. 2016; Sun et al. 2016), see Section 3. The list of searched venues includes five journals: (1) Empirical Software Engineering (EMSE); (2) Information and Software Technology (IST); (3) Journal of Systems and Software (JSS); (4) ACM Transactions on Software Engineering & Methodology (TOSEM); (5) IEEE Transaction on Software Engineering (TSE). Furthermore, we included five conferences: (1) International Conference on Automated Software Engineering (ASE); (2) ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM); (3) International Symposium on the Foundations of Software Engineering / European Software Engineering Conference (ESEC/FSE); (4) International Conference on Software Engineering (ICSE); (5) International Workshop/Working Conference on Mining Software Repositories (MSR).

We performed a generic search on SpringerLink (EMSE), Science Direct (IST, JSS), ACM DL (TOSEM, ESEC/FSE, ASE, ESEM, ICSE, MSR) and IEEE Xplore (TSE, ASE, ESEM, ICSE, MSR) using the venue (journal or conference) as a high-level filtering criterion. Considering that the proceedings of ASE, ESEM, ICSE and, MSR are published by

ACM and IEEE, we searched these venues on ACM DL and IEEE Xplore to avoid missing relevant papers. We used a generic search string ("topic model[l]ing" and "topic model"). Furthermore, in order to find studies that apply specific topic models but do not mention the term "topic model", we used a second search string with topic model names ("lsi" or "lda" or "plsi" or "latent dirichlet allocation" or "latent semantic"). This second string was based on the search string used by Chen et al. (2016), who also present a review and analysis of topic modeling techniques in software engineering (see Section 3). We applied both strings to the full text and metadata of papers. We considered works published between 2009 and 2020. The search was performed in March 2021. Limiting the search to the last twelve years allowed us to focus on more mature and recent works.

4.2 Study Selection Criteria

We only considered full research papers since full papers typically report (a) mature and complete research, and (b) more details about how topic modeling was applied. Furthermore, to be included, a paper should either apply, experiment with, or propose a topic modeling technique (e.g., develop a topic modeling technique that analyzes source code to recommend refactorings (Bavota et al. 2014b)), and meet none of the exclusion criteria: (a) the paper does not apply topic models (e.g., it applies other text mining techniques and only cites topic modeling in related or future work, such as the paper by Lian et al. (2020); (b) the paper focuses on theoretical foundation and configurations for topic models (e.g., it discusses how to tune and stabilize topic models, such as Agrawal et al. (2018) and other meta-studies listed in Section 3.2); and (c) the paper is a secondary study (e.g., a literature review like the studies discussed in Section 3.1). We evaluated inclusion and exclusion criteria by first reading the abstracts and then reading full texts.

The search with the first search string (see Section 4.1) resulted in 215 papers and the search with the second search string resulted in an additional 324 papers. Applying the filtering outlined above resulted in 114 papers. Furthermore, we excluded three papers from the final set of papers: (a) Hindle et al. (2011), (b) Chen et al. (2012), and (c) Alipour et al. (2013). These papers were earlier and shorter versions of follow-up publications; we considered only the latest publications of these papers (Hindle et al. 2013; Chen et al. 2017; Hindle et al. 2016). This resulted in a total of 111 papers for analysis.

4.3 Data Extraction and Synthesis

We defined data items to answer the research questions and characterize the selected papers (see Table 2). The extracted data was recorded in a spreadsheet for analysis (raw data are available online [1]). One of the authors extracted the data and the other authors reviewed it. In case of ambiguous data, all authors discussed to reach agreement. To synthesize the data, we applied descriptive statistics and qualitatively analyzed the data as follows:

- **RQ1:** Regarding the data item "Technique", we identified the topic modeling techniques applied in papers. For the data item "Supported tasks", we assigned to each paper one software engineering task. Tasks emerged during the analysis of papers (see more details in Section 5.2.2). We also identified the general study outcome in relation to its goal (data item "Type of contribution"). When analyzing the type of contribution, we also checked whether papers included a comparison of topic modeling techniques

[1] https://doi.org/10.5281/zenodo.5280890

Table 2 Data extraction form

Item	Description	RQ
Year	Publication year	n/a
Author(s)	List of all authors	n/a
Title	Title of paper	n/a
Venue	Publication venue	n/a
Technique	Topic modeling technique used	RQ1
Supported tasks	Development tasks supported by topic modeling (e.g., to predict defects)	RQ1
Type of contribution	General outcome of study (e.g., a new approach or an empirical exploration)	RQ1
Type of data	Type of data used for topic modeling (e.g., source code and commit messages)	RQ2
Document	Documents in corpus, i.e., "instances" of type of data (e.g., Java methods)	RQ2
Parameters	Topic modeling parameters and their values (e.g., number of topics)	RQ2
Pre-processing	Pre-processing of textual (e.g., tokenization and stop words removal)	RQ3
Topic naming	How topics were named (e.g., manual labeling by domain experts)	RQ4

(e.g., to select the best technique to be included in a newly proposed approach). Based on these data items we checked which techniques were the most popular, whether techniques were based on other techniques or used together, and for what purpose topic modeling was used.

- **RQ2:** We identified types of data (data item "Type of data") in selected papers as listed in Section 5.3.1. Considering that some papers addressed one, two or three different types of data, we counted the frequency of types of data and related them with the document. Regarding "Document", we identified the textual document and (if reported in the paper) its length. For the data item "Parameters", we identified whether papers described modeling parameters and if so, which values were assigned to them.
- **RQ3:** Considering that some papers may have not mentioned any pre-processing, we first checked which papers described data pre-processing. Then, we listed all pre-processing steps found and counted their frequencies.
- **RQ4:** Considering the papers that described topic naming, we analyzed how generated topics were named (see Section 5.5). We used three types of approaches to describe how topics were named: (a) Manual - manually analysis and labeling of topics; (b) Automated - use automated approaches to label names to topics; and (c) Manual & Automated - mix of both manual and automated approaches to analyse and name topics. We also described the procedures performed to name topics.

5 Results

5.1 Overview

As mentioned in Section 4.1, we analyzed 111 papers published between 2009 and 2020 (see Appendix A.1 - Papers Reviewed). Most papers were published after 2013. Furthermore, most papers were published in journals (68 papers in total, 32 in EMSE alone), while the remaining 43 papers appeared in conferences (mostly MSR with sixteen papers). Table 3 shows the number of papers by venue and year.

Table 3 Number of papers by venue and year

	Year												
Venue	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	Total
ASE	0	0	1	1	0	0	0	0	0	0	0	0	2
EMSE	2	0	1	1	3	5	2	3	4	4	4	3	32
ESEC FSE	0	0	0	0	0	1	0	2	1	1	1	1	7
ESEM	0	0	0	0	0	0	0	1	0	3	0	1	5
ICSE	0	1	0	1	2	2	0	1	3	1	1	1	13
IST	0	1	0	0	0	0	2	4	3	2	3	2	17
JSS	0	0	0	0	0	0	1	2	4	2	3	0	12
MSR	1	0	2	0	2	2	2	2	0	1	1	3	16
TOSEM	0	0	0	0	1	1	0	0	0	0	1	0	3
TSE	0	0	0	0	1	1	0	0	1	1	0	0	4
Total	3	2	4	3	9	12	7	15	16	15	14	11	111

5.2 RQ1: Topic Models Used

In this Section we first discuss *which* topic modeling techniques are used (Section 5.2.1). Then, we explore why or for what purpose these techniques were used (Section 5.2.2). Finally, we describe the general contributions of papers in relation to their goals (Section 5.2.3).

5.2.1 Topic Modeling Techniques

The majority of the papers used LDA (80 out of 111), or a LDA-based technique (30 out of 111), such as Twitter-LDA (Zhao et al. 2011). The other topic modeling technique used is LSI. Figure 2 shows the number of papers per topic modeling technique. The total number (125) exceeds the number of papers reviewed (111), because ten papers experimented with more than one technique: Thomas et al. (2013), De Lucia et al. (2014), Binkley et al. (2015), Tantithamthavorn et al. (2018), Abdellatif et al. (2019) and Liu et al. (2020) experimented with LDA and LSI; Chen et al. (2014) experimented with LDA and Aspect and Sentiment Unification Model (ASUM); Chen et al. (2019) experimented with Labeled Latent Dirichlet Allocation (LLDA) and Label-to-Hierarchy Model (L2H); Rao and Kak (2011) experimented with LDA and MLE-LDA; and Hindle et al. (2016) experimented with LDA and LLDA. ASUM, LLDA, MLE-LDA and L2H are techniques based on LDA.

The popularity of LDA in software engineering has also been discussed by others, e.g., Treude and Wagner (2019). LDA is a three-level hierarchical Bayesian model (Blei et al. 2003b). LDA defines several hyperparameters, such as α (probability of topic z_i in document d_i), β (probability of word w_i in topic z_i) and k (number of topics to be generated) (Agrawal et al. 2018).

Thirty-seven (out of 75) papers applied LDA with Gibbs Sampling (GS). Gibbs sampling is a Markov Chain Monte Carlo algorithm that samples from conditional distributions of a target distribution. Used with LDA, it is an approximate stochastic process for computing α and β (Griffiths and Steyvers 2004). According to experiments conducted by Layman et al. (2016), Gibbs sampling in LDA parameter estimation (α and β) resulted in lower perplexity

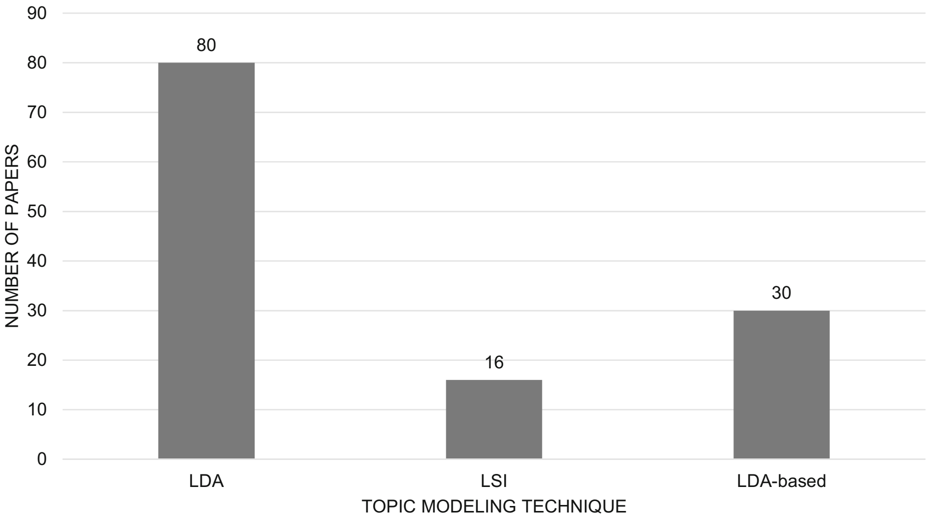

Fig. 2 Number of papers per topic modeling technique

than the Variational Expectation-Maximization (VEM) estimations. Perplexity is a standard measure of performance for statistical models of natural language, which indicates the uncertainty in predicting a single word. Therefore, lower values of perplexity mean better model performance (Griffiths and Steyvers 2004).

Thirty papers applied modified or extended versions of LDA ("LDA-based" in Fig. 2). Table 4 shows a comparison between these LDA-based techniques. Eleven papers proposed a new extension of LDA to adapt LDA to software engineering problems (hence the same reference in the third and fourth column of Table 4). For example, the Multi-feature Topic Model (MTM) technique by Xia et al. (2017b), which implements a supervised version of LDA to create a bug triaging approach. The other 19 papers applied existing modifications of LDA proposed by others (third column in Table 4). For example, Hu and Wong (2013) used the Citation Influence Topic Model (CITM), developed by Dietz et al. (2007), which models the influence of citations in a collection of publications.

The other topic modeling technique, LSI (Deerwester et al. 1990), was published in 1990, before LDA which was published in 2003. LSI is an information extraction technique that reduces the dimensionality of a term-document matrix using a reduction factor k (number of topics) (Deerwester et al. 1990). Compared to LDA, LDA follows a generative process that is statistically more rigorous than LSI (Blei et al. 2003b; Griffiths and Steyvers 2004). From the 16 papers that used LSI, seven papers compared this technique to others:

- One paper (Rosenberg and Moonen 2018) compared LSI with other two dimensionality reduction techniques: Principal Component Analysis (PCA) (Wold et al. 1987) and Non-Negative Matrix Factorization (NMF) (Lee and Seung 1999). The authors applied these models to automatically group log messages of continuous deployment runs that failed for the same reasons.
- Four papers applied LDA and LSI at the same time to compare the performance of these models to Vector Space Model (VSM) (Salton et al. 1975), an algebraic model for

Table 4 LDA-based techniques

Technique	Comparison to LDA	Proposed by	Papers
Labeled LDA (LLDA)	Supervised approach of LDA that constrains topics to a set of pre-defined labels	(Ramage et al. 2009)	(McIlroy et al. 2016; Chen et al. 2019)
Label-to-Hierarchy model (L2H)	Builds concept hierarchy from a set of documents, where each document contains multiple labels; learns from label co-occurrence and word usage to discover a hierarchy of topics associated with user-generated labels	(Nguyen et al. 2014)	(Chen et al. 2019)
Semi-supervised LDA	Uses samples of labeled documents to train model; relies on similarity between the unclassified documents and the labeled documents	(Fu et al. 2015)	(Fu et al. 2015)
Twitter-LDA	Short-text topic modeling for tweets; considers each tweet as a document that contains a single topic	(Zhao et al. 2011)	(Hu et al. 2019)
BugScout-LDA	Uses two implementations of LDA (one implementation to model topics from source code and another one to model topics in bug reports) to recommend a short list of candidate buggy files for a given bug report	(Nguyen et al. 2011)	(Nguyen et al. 2011)
O-LDA	Method for feature location that applies strategies for filtering data used as input to LDA and strategies for filtering the output (words in topics to describe domain knowledge)	(Liu et al. 2017)	(Liu et al. 2017)
DAT-LDA	Extended LDA to infer topic probability distributions from multiple data sources (Mashup description text, Web APIs and tags) to support Mashup service discovery	(Cao et al. 2017)	(Cao et al. 2017)
LDA-GA	Determines the near-optimal configuration for LDA using genetic algorithms	(Panichella et al. 2013)	(Panichella et al. 2013; Zhang et al. 2018; Sun et al. 2015; Yang et al. 2017; Catolino et al. 2019)
Aspect and Sentiment Unification Model (ASUM)	Finds topics in textual data, reflecting both aspect (i.e., a word that expresses a feeling, e.g., “disappointed”) and sentiment (i.e., a word that conveys sentiment, e.g., “positive” or “negative”)	(Jo and Oh 2011)	(Galvis Carreno and Winbladh 2012; Chen et al. 2014)
Citation Influence Topic Model (CITM)	Determines the citation influences of a citing paper in a document network based on two corpora: (a) incoming links of publications (cited papers), and (b) outgoing links of publications (citing papers); a paper can select words from topics of its own topics or from topics found in cited papers	(Dietz et al. 2007)	(Hu and Wong 2013)
Collaborative Topic Modeling (CTM)	Creates recommendations for users based on the topic modeling of two types of data: (a) libraries of users, and (b) content of publications; for each user, finds both old papers that are important to other similar users and newly written papers that are related to that user interests	(Wang and Blei 2011)	(Sun et al. 2017)

Table 4 (continued)

Technique	Comparison to LDA	Proposed by	Papers
Discriminative Probability Latent Semantic Analysis (DPLSA)	Supervised approach that recommends components for bug reports; receives assigned bug reports for training and generates a number of topics that is the same as the number of components	(Yan et al. 2016a)	(Yan et al. 2016a, b)
Multi-feature Topic Model (MTM)	Supervised approach that considers features (product and component information) of bug reports; emphasizes occurrence of words in bug reports that have the same combination of product and component	(Xia et al. 2017b)	(Xia et al. 2017b)
Relational Topic Model (RTM)	Defines probability distribution of topics among documents, but also derives semantic relationships between documents	(Chang and Blei 2009)	(Bavota et al. 2014a, b)
T-Model	Detects duplicate bug reports	(Nguyen et al. 2012)	(Nguyen et al. 2012)
Temporal LDA	Extends LDA to model document streams considering a time window	(Damevski et al. 2018)	(Damevski et al. 2018)
TopicSum	Estimates content distribution for summary extraction. Different to LDA, it generates a collection of document sets: background (background distribution over vocabulary words); content (significant content to be summarized); and docspecific (local words to a single document that do not appear across several documents)	(Haghighi and Vanderwende 2009)	(Fowkes et al. 2016)
Adaptively Online LDA (AOLDA)	Adaptively combines the topics of previous versions of an app to generate topic distributions of current versions	(Gao et al. 2018)	(Gao et al. 2018)
Hierarchical Dirichlet Process (HDP)	Implements a non-parametric Bayesian approach which iteratively groups words based on a probability distribution (i.e., the number of topics is not known a priori)	(Teh et al. 2006)	(Palomba et al. 2017)
Maximum-likelihood Representation LDA (MLE-LDA)	Represents a vocabulary-dimensional probability vector directly by its first order distribution	(Rao and Kak 2011)	(Rao and Kak 2011)
Query likelihood LDA (QL-LDA)	Combines Dirichlet smoothing (a technique to address overfitting) with LDA	(Wei and Croft 2006)	(Binkley et al. 2015)

information extraction. These studies supported documentation (De Lucia et al. 2014); bug handling (Thomas et al. 2013; Tantithamthavorn et al. 2018); and maintenance tasks (Abdellatif et al. 2019)).

- Regarding the other two papers, Binkley et al. (2015) compared LSI to Query likelihood LDA (QL-LDA) and other information extraction techniques to check the best model for locating features in source code; and Liu et al. (2020) compared LSI and LDA to Generative Vector Space Model (GVSM), a deep learning technique, to select the best performer model for documentation traceability to source code in multilingual projects.

5.2.2 Supported Tasks

As mentioned before, we aimed to understand why topic modeling was used in papers, e.g., if topic modeling was used to develop techniques to support specific software engineering tasks, or if it was used as a data analysis technique in exploratory studies to understand the content of large amounts of textual data. We found that the majority of papers aimed at supporting a particular task, but 21 papers (see Table 5) used topic modeling in empirical exploratory and descriptive studies as a data analysis technique.

We extracted the software engineering tasks described in each study (e.g., bug localization, bug assignment, bug triaging) and then grouped them into eight more generic tasks (e.g., bug handling) considering typical software development activities such as requirements, documentation and maintenance (Leach 2016). The specific tasks collected from papers are available online [1]. Note that we kept "Bug handling" and "Refactoring" separate rather than merging them into maintenance because of the number of papers (bug handling) and the cross-cutting nature (refactoring) in these categories. Each paper was related to one of these tasks:

- Architecting: tasks related to architecture decision making, such as selection of cloud or mash-up services (e.g., Belle et al. (2016));
- Bug handling: bug-related tasks, such as assigning bugs to developers, prediction of defects, finding duplicate bugs, or characterizing bugs (e.g., Naguib et al. (2013));
- Coding: tasks related to coding, e.g., detection of similar functionalities in code, reuse of code artifacts, prediction of developer behaviour (e.g., Damevski et al. (2018));
- Documentation: support software documentation, e.g., by localizing features in documentation, automatic documentation generation (e.g., Souza et al. (2019));
- Maintenance: software maintenance-related activities, such as checking consistency of versions of a software, investigate changes or use of a system (e.g., Silva et al. (2019));
- Refactoring: support refactoring, such as identifying refactoring opportunities and removing bad smell from source code (e.g., Bavota et al. (2014b));
- Requirements: related to software requirements evolution or recommendation of new features (e.g., Galvis Carreno and Winbladh (2012));
- Testing: related to identification or prioritization of test cases (e.g., Thomas et al. (2014)).

Table 5 groups papers based on the topic modeling technique and the purpose. Few papers applied topic modeling to support Testing (three papers) and Refactoring (three papers). Bug handling is the most frequent supported task (33 papers). From the 21 exploratory studies, 13 modeled topics from developer communication to identify developers' information needs: 12 analyzed posts on Stack Overflow, a Q&A website for developers (Chatterjee et al. 2019; Bajaj et al. 2014; Ye et al. 2017; Bagherzadeh and Khatchadourian 2019; Ahmed and Bagherzadeh 2018; Barua et al. 2014; Rosen and Shihab 2016; Zou et al. 2017; Chen et al. 2019; Han et al. 2020; Abdellatif et al. 2020; Haque and Ali Babar 2020) and one paper analyzed blog posts (Pagano and Maalej 2013). Regarding the other eight exploratory studies, three papers investigated web search queries to also identify developers' information needs (Xia et al. 2017a; Bajracharya and Lopes 2009; 2012); four papers investigated end user documentation to analyse users' feedback on mobile apps (Tiarks and Maalej 2014; El Zarif et al. 2020; Noei et al. 2018; Hu et al. 2018); and one paper investigated historical "bug" reports of NASA systems to extract trends in testing and operational failures (Layman et al. 2016).

Table 5 Techniques and supported tasks

	Technique					
Supported task	LDA	LDA-based	LSI	LDA-based, (LDA or LSI)	LDA, LSI	Total
Architecting	(Nabli et al. 2018; Belle et al. 2016; Demissie et al. 2020; Gopalakrishnan et al. 2017; Gorla et al. 2014)	DAT-LDA (Cao et al. 2017), LDA-GA (Yang et al. 2017) RTM (Cui et al. 2019)	(Poshyvanyk et al. 2009; Revelle et al. 2011)	–	–	10
Bug handling	(Nguyen et al. 2012; Noei et al. 2019; Hindle et al. 2015; Le et al. 2017; Choetkiertikul et al. 2017; Zhang et al. 2016; Martin et al. 2015; Murali et al. 2017; Ahasanuzzaman et al. 2019; Nayebi et al. 2018; Lukins et al. 2010; Chen et al. 2017; Naguib et al. 2013; Zhao et al. 2020; Zhao et al. 2016; Zaman et al. 2011; Mezouar et al. 2018; Silva et al. 2016)	BugScout-LDA (Nguyen et al. 2011), CITM (Hu and Wong 2013), CTM (Sun et al. 2017), DPLSA (Yan et al. 2016b), LLDA (McIlroy et al. 2016), LDA-GA (Zhang et al. 2018; Catolino et al. 2019), MTM (Xia et al. 2017b), Semi-supervised LDA (Fu et al. 2015), AOLDA (Gao et al. 2018)	–	ASUM, LDA (Chen et al. 2014), LLDA, LDA (Hindle et al. 2016), MLE-LDA, LDA (Rao and Kak 2011)	(Tantithamthavorn et al. 2018; Thomas et al. 2013)	33
Coding	(Damevski et al. 2018; Altarawy et al. 2018; Taba et al. 2017; Chen et al. 2020; Ray et al. 2014)	(Fowkes et al. 2016)	–	–	–	6
Documentation	(Asuncion et al. 2010; Jiang et al. 2017; Hindle et al. 2013; Henß et al. 2012; Moslehi et al. 2016; 2018; Souza et al. 2019; Moslehi et al. 2020; Biggers et al. 2014; Wang et al. 2015)	LDA-GA (Panichella et al. 2013), O-LDA (Liu et al. 2017)	(Dit et al. 2013; Poshyvanyk et al. 2012; Pérez et al. 2018; Noei and Heydarnoori 2016)	QL-LDA, LSI (Binkley et al. 2015)	(De Lucia et al. 2014; Liu et al. 2020)	19

Table 5 (continued)

	Technique					
Supported task	LDA	LDA-based	LSI	LDA-based, (LDA or LSI)	LDA, LSI	Total
Maintenance	(Pettinato et al. 2019; Li et al. 2018; Silva et al. 2019; Capiluppi et al. 2020; Martin et al. 2016)	DPLSA (Yan et al. 2016a), LDA-GA (Sun et al. 2015), Twiter-LDA (Hu et al. 2019), HDP (Palomba et al. 2017)	(Tairas and Gray 2009; Rosenberg and Moonen 2018)	–	(Abdellatif et al. 2019)	12
Refactoring	(Canfora et al. 2014)	RTM (Bavota et al. 2014a; Bavota et al. 2014b)	–	–	–	3
Requirements	(Jiang et al. 2019)	ASUM (Galvis Carreno and Winbladh 2012)	(Blasco et al. 2020)	–	(Ali et al. 2015)	4
Testing	(Thomas et al. 2014; Shimagaki et al. 2018; Luo et al. 2016)	–	–	–	–	3
Exploratory studies	(Chatterjee et al. 2019; Bajaj et al. 2014; Layman et al. 2016; Bajracharya and Lopes 2009; Xia et al. 2017a; Pagano and Maalej 2013; Ye et al. 2017; Bajracharya and Lopes 2012; Bagherzadeh and Khatchadourian 2019; Ahmed and Bagherzadeh 2018; Barua et al. 2014; Rosen and Shihab 2016; Zou et al. 2017; Han et al. 2020; Abdellatif et al. 2020; Haque and Ali Babar 2020; Tiarks and Maalej 2014; El Zarif et al. 2020; Noei et al. 2018)	L2H, LLDA (Chen et al. 2019), Twitter-LDA (Hu et al. 2018)	–	–	–	21

5.2.3 Types of Contribution

For each study, we identified what type of contribution it presents based on the study goal. We used three types of contributions (“Approach”, “Exploration” and “Comparison”, as described below) by analyzing the research questions and main results of each study. A study could contribute either an “Approach” or an “Exploration”, while “Comparison” is orthogonal, i.e., a study that presents a new approach could present a comparison of topic models as part of this contribution. Similarly, a comparison of topic models can also be part of an exploratory study.

- Approach: a study develops an approach (e.g., technique, tool, or framework) to support software engineering activities based on or with the support of topic models. For example, Murali et al. (2017) developed a framework that applies LDA to Android API methods to discover types of API usage errors, while Le et al. (2017) developed a technique (APRILE+) for bug localization which combines LDA with a classifier and an artificial neural network.
- Exploration: a study applies topic modeling as the technique to analyze textual data collected in an empirical study (in contrast to for example open coding). Studies that contributed an exploration did not propose an approach as described in the previous item, but focused on getting insights from data. For example, Barua et al. (2014) applied LDA to Stack Overflow posts to discover what software engineering topics were frequently discussed by developers; Noei et al. (2018) explored the evolution of mobile applications by applying LDA to app descriptions, release notes, and user reviews.
- Comparison: the study (that can also contribute with an “Approach” or an “Exploration”) compares topic models to other approaches. For example, Xia et al. (2017b) compared their bug triaging approach (based on the so called Multi-feature Topic Model - MTM) with similar approaches that apply machine learning (Bugzie (Tamrawi et al. 2011)) and SVM-LDA (combining a classifier with LDA (Somasundaram and Murphy 2012)). On the other hand, De Lucia et al. (2014) compared LDA and LSI to define guidelines on how to build effective automatic text labeling techniques for program comprehension.

From the papers that contributed an *approach*, twenty-two combined a topic modeling technique with one or more other techniques applied for text mining:

- Information extraction (e.g., VSM) (Nguyen et al. 2012; Zhang et al. 2018; Chen et al. 2020; Thomas et al. 2013; Fowkes et al. 2016);
- Classification (e.g., Support Vector Machine - SVM) (Hindle et al. 2013; Le et al. 2017; Liu et al. 2017; Demissie et al. 2020; Zhao et al. 2020; Shimagaki et al. 2018; Gopalakrishnan et al. 2017; Thomas et al. 2013);
- Clustering (e.g., K-means) (Jiang et al. 2019; Cao et al. 2017; Liu et al. 2017; Zhang et al. 2016; Altarawy et al. 2018; Demissie et al. 2020; Gorla et al. 2014);
- Structured prediction (e.g., Conditional Random Field - CRF) (Ahasanuzzaman et al. 2019);
- Artificial neural networks (e.g., Recurrent Neural Network - RNN) (Murali et al. 2017; Le et al. 2017);
- Evolutionary algorithms (e.g., Multi-Objective Evolutionary Algorithm - MOEA) (Blasco et al. 2020; Pérez et al. 2018);
- Web crawling (Nabli et al. 2018).

Pagano and Maalej (2013) was the only study that contributed an *exploration* that combined LDA with another text mining technique. To analyze how developer communities use blogs to share information, the authors applied LDA to extract keywords from blog posts and then analyzed related "streams of events" (commit messages and releases by time in relation to blog posts), which were created with Sequential pattern mining.

Regarding *comparisons* we found that (1) 13 out of the 63 papers that contribute an approach also include some form of comparison, and (2) ten out of the 48 papers contribute an exploration also include some form of comparison. We discuss comparisons in more detail below in Section 6.1.2

5.3 RQ2: Topic Model Inputs

In this section we first discuss the type of data (Section 5.3.1). Then we discuss the actual textual documents used for topic modeling (Section 5.3.2). Finally, we describe which model parameters were used (Section 5.3.3) to configure models.

5.3.1 Types of Data

Types of data help us describe the textual software engineering content that has been analyzed with topic modeling. We identified 12 types of data in selected papers as shown in Table 6. In some papers we identified two or three of these types of data; for example, the study of Tantithamthavorn et al. (2018) dealt with issue reports, log information and source code.

Source code (37 occurrences), issue/bug reports (22 occurrences) and developer communication (20 occurrences) were the most frequent types of data used. Seventeen papers used two to four types of data in their topic modeling technique; twelve of these papers used a combination of source code with another type of data. For example, Sun et al. (2015) generated topics from source code and developer communication to support software maintenance tasks, and in another study, Sun et al. (2017) used topics found in source code and commit messages to assign bug-fixing tasks to developers.

5.3.2 Documents

A document refers to a piece of textual data that can be longer or shorter, such as a requirements document or a single e-mail subject. Documents are concrete instances of the types of data discussed above. Figure 3 shows documents (per type of data) and how often we found them in papers. The most frequent documents are bug reports (12 occurrences), methods from source code (9 occurrences), Q&A posts (9 occurrences) and user reviews (8 occurrences).

We also analyzed document length and found the following:

- In general, papers described the length of documents in number of words, see Table 7.[2] On the other hand, two papers (Moslehi et al. 2016, 2020) described their documents' length in minutes of screencast transcriptions (videos with one to ten minutes, no information about the size of transcripts). Sixteen papers mentioned the actual length of the documents, see Table 7. Ten papers that described the actual document length did

[2] This table also shows hyperparameters and the number of topics which are discussed in the following subsection.

Table 6 Types of data for topic modeling

Type of data	Description	Number of papers
"Lessons learned" as free text	Lessons learned from issues and risks of a software project (e.g., record of lessons learned from an issue of the OpenOffice project)	1
URL content	Text of a URL (e.g., URLs in a Cloud service priority queue)	1
Transcripts	Transcripts of audio or video recordings	3
Developer documentation	Documentation used by developers (e.g., Web API documentation)	4
Search query	Keywords in web search queries (e.g., "software development" used in Google search)	4
Log information	Log events of a software, such as registries of updates in a code repository	5
Commit messages	Comments of developers when committing changes to a code repository	10
End user communication	App reviews of end users in app stores	12
End user documentation	Apps and features descriptions, requirement documents, or API tutorials	15
Issue/bug reports	Reports of bugs, change requests and/or issues of a software project	22
Developer communication	Developer discussions such as Q&A websites, e-mails, and instant messaging	20
Source code	Scripts, methods and classes of a software	37

that when describing the data used for topic modeling; four papers discussed document length while describing results; and one mentioned document length as a metric for comparing different data sources;

- Most papers (80 out of 111) did not mention document length and also do not acknowledge any limitations or the impact of document length on topics.
- Fifteen papers did not mention the actual document length, but at some point acknowledge the influence of document length on topic modeling. For example, Abdellatif et al. (2019) mentioned that the documents in their data set were "not long". Similarly, Yan et al. (2016b) did not mention the length of the bug reports used but discussed the impact of the vocabulary size of their corpus on results. Moslehi et al. (2018) mentioned document length as a limitation and acknowledge that using LDA on short documents was a threat to construct validity. According to these authors, using techniques specific for short documents could have improved the outcomes of their topic modeling.

5.3.3 Model Parameters

Topic models can be configured with parameters that impact how topics are generated. For example, LDA has typically been used with symmetric Dirichlet priors over θ (document-topic distributions) and ϕ (topic-word distributions) with fixed values for α and β (Wallach et al. 2009). Wallach et al. (2009) explored the robustness of a topic model with asymmetric priors over θ (i.e., varying values for α) and a symmetric prior (fixed value for β) over ϕ. Their study found that such topic model can capture more distinct and semantically-related

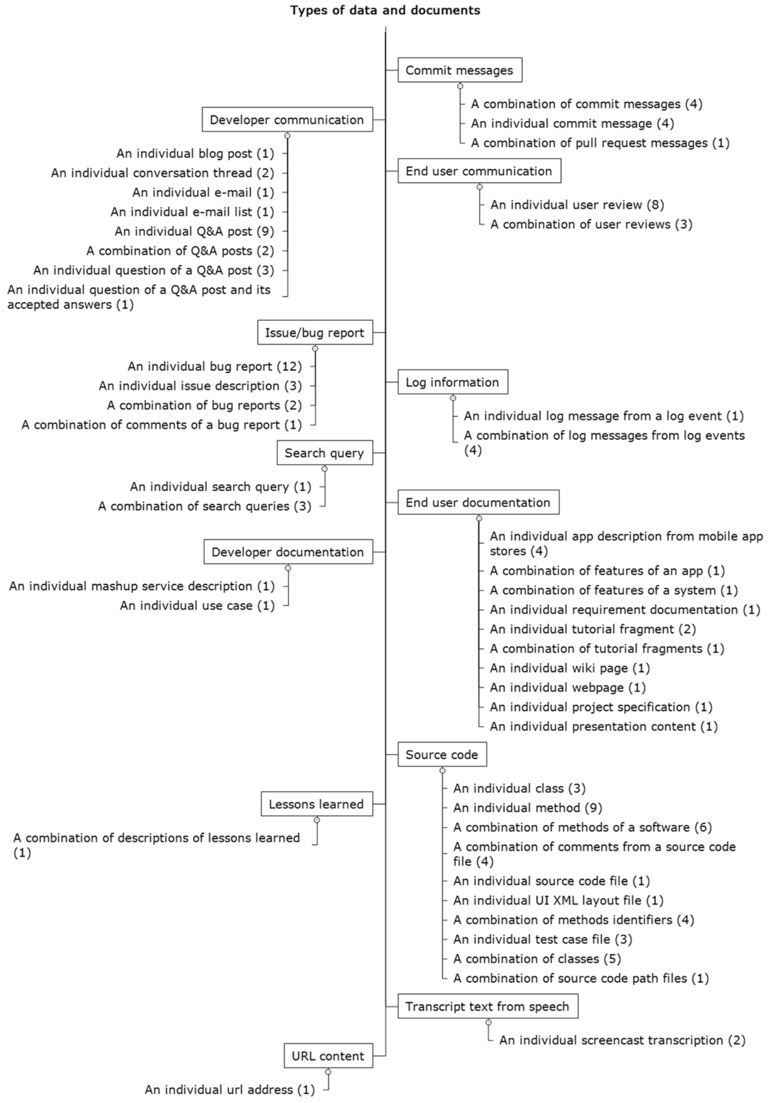
Types of data and documents
Commit messages
A combination of commit messages (4)
An individual commit message (4)
A combination of pull request messages (1)
Developer communication
An individual blog post (1)
An individual conversation thread (2)
An individual e-mail (1)
An individual e-mail list (1)
An individual Q&A post (9)
A combination of Q&A posts (2)
An individual question of a Q&A post (3)
An individual question of a Q&A post and its accepted answers (1)
End user communication
An individual user review (8)
A combination of user reviews (3)
Issue/bug report
An individual bug report (12)
An individual issue description (3)
A combination of bug reports (2)
A combination of comments of a bug report (1)
Log information
An individual log message from a log event (1)
A combination of log messages from log events (4)
Search query
An individual search query (1)
A combination of search queries (3)
End user documentation
An individual app description from mobile app stores (4)
A combination of features of an app (1)
A combination of features of a system (1)
An individual requirement documentation (1)
An individual tutorial fragment (2)
A combination of tutorial fragments (1)
An individual wiki page (1)
An individual webpage (1)
An individual project specification (1)
An individual presentation content (1)
Developer documentation
An individual mashup service description (1)
An individual use case (1)
Source code
An individual class (3)
An individual method (9)
A combination of methods of a software (6)
A combination of comments from a source code file (4)
An individual source code file (1)
An individual UI XML layout file (1)
A combination of methods identifiers (4)
An individual test case file (3)
A combination of classes (5)
A combination of source code path files (1)
Lessons learned
A combination of descriptions of lessons learned (1)
Transcript text from speech
An individual screencast transcription (2)
URL content
An individual url address (1)

Fig. 3 Documents (leaves in the figure) by type of data (nodes in the figure)

topics, i.e., the words in clusters are more distinct. Therefore, we checked which parameters and values were used in papers. Overall, we found the following:

- Eighteen of the 111 papers do not mention parameters (e.g., number of topics k, hyperparameters α and β). Thirteen of these papers use LDA or an LDA-based technique, four papers use LSI, while (Liu et al. 2020) use LDA and LSI.
- The remaining 93 papers mention at least one parameter. The most frequent parameters discussed were k, α and β:
  – Fifty-eight papers mentioned actual values for k, α and β;

Table 7 Document length as reported in papers

Document	Length	Topic model	Hyperparameters	Number of topics	Papers
An individual commit message	9 to 20 words	LDA	-	10	(Canfora et al. 2014)
An individual blog post	273 words average	LDA	-	50	(Pagano and Maalej 2013)
An individual Q&A post	500 words average	LDA	$\alpha = 50/k$, $\beta = 0.01$	40	(Barua et al. 2014)
	50 to 400 words	LLDA; L2H	$\alpha = 10$, $\beta = 1000$	3	(Chen et al. 2019)
An individual user review	65 to 155 words	Twitter-LDA	-	10	(Hu et al. 2019)
	28 to 97 words	LDA	-	85, 170	(Nayebi et al. 2018)
An individual bug report	404 words average	LDA	$\alpha = 50/k$, $\beta = 0.01$	20, [steps of 10], 100, 125, 150, [steps of 25], 225	(Layman et al. 2016)
	127 words (Eclipse data) and 146 words (Mozilla data) average	LDA; LSI	-	32, 64, 128, 256	(Tantithamthavorn et al. 2018)*
A combination of log messages	95 words (test data) and 221 words (validation data) average	LDA	$\alpha = 50/k$, $\beta = 0.1$	9	(Pettinato et al. 2019)
An individual requirement document	3,800 words average	LDA	$\alpha = 0.1$, $\beta = 0.1$	20	(Hindle et al. 2015)
An individual fragment of API tutorials	100 to 300 words	LDA	$\alpha = 0.1$, $\beta = 0.1$	-	(Jiang et al. 2017)
A combination of tutorials of an app store	3,231 words average	LDA	-	20, 50	(Tiarks and Maalej 2014)
A combination of classes from a directory	4,153 words in 922 documents (total)	LSI	-	-	(Tairas and Gray 2009)
An individual method	14 words (Eclipse data) and 35 words (Mozilla data) average	LDA; LSI	-	32, 64, 128, 256	(Tantithamthavorn et al. 2018)*
An individual screencast transcript	1 to 10 minutes	LDA	$\alpha = 50/k$, $\beta = 0.01$	20 55, 80, 130	(Moslehi et al. 2016) (Moslehi et al. 2020)

*Same study that used two different documents

- Two papers mentioned actual values for α and β, but no values for k;
- Twenty-nine papers included actual values for k but not for α and β;
- Thirty-two (out of 58) papers mentioned other parameters in addition to k, α and β. For example, Chen et al. (2019) applied L2H (in comparison to LLDA), which uses the hyperparameters γ_1 and γ_2;
- One paper (Rosenberg and Moonen 2018) that applied LSI, mentioned the parameter "similarity threshold" rather than k, α and β.

We then had a closer look at the 60 papers that mentioned actual values for hyperparameters α and β:

- α **based on** k: The most frequent setting (29 papers) was $\alpha = 50/k$ and $\beta = 0.01$ (i.e., α was depending on the number of topics, a strategy suggested by Steyvers and Griffiths (2010) and Wallach et al. (2009)). These values are a default setting in Gibbs Sampling implementations for LDA such as Mallet.[3]
- **Fixed** α **and** β: Five papers fixed 0.01 for both hyperparameters, as suggested by Hoffman et al. (2010). Another eight papers fixed 0.1 for both hyperparameters, a default setting in Stanford Topic Modeling Toolbox (TMT);[4] and three other papers fixed $\alpha = 0.1$ and $\beta = 1$ (these three studies applied RTM).
- **Varying** α **or** β: Four papers tested different values for α, where two of these papers also tested different values for β; and one paper varied β but fixed a value for α.
- **Optimized parameters**: Four papers obtained optimized values for hyperparameters (Sun et al. 2015; Catolino et al. 2019; Yang et al. 2017; Zhang et al. 2018). These papers applied LDA-GA (as proposed by Panichella et al. (2013)) which, based on genetic algorithms; finds the best values for LDA hyperparameters. In regards to the actual values chosen for optimized hyperparameters, Catolino et al. (2019) did not mention the values for hyperparameters; Sun et al. (2015) and Yang et al. (2017) mentioned only the values used for k; and Zhang et al. (2018) described the values for k, α and β.

Regarding the values for k we observed the following:

- The 90 papers that mentioned values for k modeled three (Cao et al. 2017) to 500 (Li et al. 2018; Lukins et al. 2010; Chen et al. 2017) topics;
- Twenty-four (out of 90) papers mentioned that a range of values for k was tested in order to check the performance of the technique (e.g., Xia et al. (2017b)) or as a strategy to select the best number of topics (e.g., Layman et al. (2016));
- Although the remaining 66 (out of 90) papers mentioned a single value used for k, most of them acknowledged that had tried several number of topics or used the number of topics suggested by other studies.

As can be seen in Table 7, there is no common trend of what values for hyperparameter or k depending on the document or document length.

5.4 RQ3: Pre-processing Steps

Thirteen of the papers did not mention what pre-processing steps were applied to the data before topic modeling. Seven papers only described how the data analyzed were selected,

[3] http://mallet.cs.umass.edu/topics.php

[4] https://nlp.stanford.edu/software/tmt/tmt-0.4/

but not how they were pre-processed. Table 8 shows the pre-processing steps found in the remaining 91 papers. Each of these papers mentioned at least one of these steps.

Removing noisy content (76 occurrences), Stemming terms (61 occurrences) and Splitting terms (33 occurrences) were the most used pre-processing steps. The least frequent pre-processing step (Resolving negations) was found only in the studies of Noei et al. (2019) and Noei et al. (2018). Resolving synonyms and Expanding contractions were also less frequent, with three occurrences each.

Table 9 shows the types of noise removal in papers and their frequency. Most of the papers that described pre-processing steps removed stop words (76 occurrences). Stop words are the most common words in a language, such as "a/an" and "the" in English. Removing stop words allows topic modeling techniques to focus on more meaningful words in the corpus (Miner et al. 2012). Eight papers mentioned the stop words list used:

Table 8 Pre-processing steps found in papers

Pre-processing step	Description	Number of papers
Resolving negations	Negations refer to negative sentences with positive meaning, such as "No problem"; used depending on the context of study (e.g., the paper in which we found this step removed negations in user reviews)	2
Expanding contractions	Normalizing contracted terms into expanded forms (e.g., "couldn't" into "could not")	3
Resolving synonyms	Replacing words with similar meaning with a common representative word (e.g., "bug", "error", and "glitch" can be synonyms for "exception")	3
Identifying n-grams	Words may have a more concrete meaning when used together; n-grams are a sequence of *n* words; e.g., bi-gram (n-gram of two words) *software development* can be more informative than the words "software" and "development" separately	6
Correcting typos	Replacing misspelled words with the correct ones	7
Splitting document	Breaking a long document into shorter documents (e.g., splitting long project specifications and wiki pages)	7
Lemmatizing	Reducing words to their lemmas based on the words' part of speech (e.g., words "is" and "are" can be resolved as "be")	11
Tokenizing	Breaking up text in document into individual tokens (e.g., using white space and punctuation as token delimiters)	17
Lowercasing	Entire document is converted to lowercase characters regardless of the spelling in the original document	20
Splitting words	Splitting two or more words with no separating spaces or punctuation (e.g., many papers that analyze source code separated camel cases like "processFile" into "process" and "File")	33
Stemming	Normalizing words into their single forms by identifying and removing prefixes, suffixes and pluralisation (e.g., "development", "developer", "developing" become "develop")	61
Removing noise	Noise is any text that will interfere in the topic modeling (e.g., slowing down the processing or resulting in meaningless topics); due to the different types of noise removal, we discuss noise removal separately in Table 9	76

Table 9 Noisy content removed

Noisy content	Number of papers
Empty documents	1
Long paragraphs	1
Extra white space	1
Short documents	2
Words shorter than four, three or two letters	2
URLs	4
Least frequent terms	8
Most frequent terms	8
Code snippets	9
HTML tags	9
Non-informative content	11
Numbers	17
Programming language keywords	23
Symbols and special characters	20
Punctuation	21
Stop words	75

Layman et al. (2016) and Pettinato et al. (2019) used the SMART stop words list;[5] Martin et al. (2015) and Hindle et al. (2013) used the Natural Language Toolkit English stop words list;[6] Bagherzadeh and Khatchadourian (2019), Ahmed and Bagherzadeh (2018) and Yan et al. (2016b) used the Mallet stop words list;[7] and Mezouar et al. (2018) used the Moby stop words list.[8]

As can be seen in Table 9, some papers removed words based on the frequency of their occurrence (most or least frequent terms) or length (words shorter than four, three or two letters or long terms). Other papers removed long paragraphs. For example, Henß et al. (2012) removed paragraphs longer than 800 characters because most paragraphs in their data set were shorter than that. We also found two papers that removed short documents: Gorla et al. (2014) removed documents with fewer than ten words, and Palomba et al. (2017) removed documents with fewer than three words. The concept of non-informative content depends on the context of each paper. In general, it refers to any data considered not relevant for the objective of the study. For example, Choetkiertikul et al. (2017), which aimed at predicting bugs in issue reports, removed issues that took too much time to be resolved. Noei et al. (2019) and Fu et al. (2015) removed content (end user reviews and commit messages) that did not describe feedback or cause of change.

5.5 RQ4: Topic Naming

Topic naming is about assigning labels (names) to topics (word clusters) to give the clusters a human-understandable meaning. Seventy-five papers (out of 111) did not mention whether

[5] http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

[6] https://gist.github.com/sebleier/554280

[7] https://github.com/mengjunxie/ae-lda/blob/master/misc/mallet-stopwords-en.txt

[8] http://icon.shef.ac.uk/Moby/mwords.html

or how topics were named. These papers only used the word clusters for analysis, but did not require a name. For example, Xia et al. (2017a) and Canfora et al. (2014) did not name topics, but mapped the word clusters to the documents (search queries and source code comments) used as input for topic modeling. These papers used the probability of a document to belong to a topic (θ) to associate a document to the topic with the highest probability.

From the 36 papers (out of 111) that mentioned topic naming (see Table 10), we identified three ways of how they named topics:

- **Automated:** Assigning names to word clusters without human intervention;
- **Manual:** Manually checking the meaning and the combination of words in cluster to "deduct" a name, sometimes validated with expert judgment;
- **Manual & Automated:** Mix of manual and automated; e.g., topics are manually labeled for one set of clusters to then train a classifier for naming another set of clusters.

Most of the papers (30 out of 36) assigned one name to one topic. However, we identified six papers that used one name for multiple topics (Hindle et al. 2013; Pagano and Maalej 2013; Bajracharya and Lopes 2012; Rosen and Shihab 2016) or labeled a topic with multiple names (Zou et al. 2017; Gao et al. 2018). Two of the papers (Hindle et al. 2013; Bajracharya and Lopes 2012) that assigned one name to multiple topics used predefined labels, and in the other two papers (Pagano and Maalej 2013; Rosen and Shihab 2016) authors interpreted words in the clusters to deduct names.

Regarding the papers that assigned multiple names to a topic, Zou et al. (2017) assigned no, one or more names, depending on how many words in the predefined word list matched words in clusters. Gao et al. (2018) used an automated approach to label topics with the three most relevant phrases and sentences from the end user reviews inputted to their topic model. The relevance of phrases and sentences were obtained with the metrics Semantic and Sentiment scores proposed by these authors.

6 Discussion

6.1 RQ1: Topic Modeling Techniques

6.1.1 Summary of Findings

LDA is the most frequently used topic model. Almost all papers (95 out of 111) applied LDA or a LDA-based technique, while nine papers applied LSI to identify topics and seven papers used LDA and LSI. Regarding the papers that used LDA-based techniques, eleven (out of 30) proposed their own LDA-based technique (Fu et al. 2015; Nguyen et al. 2011; Liu et al. 2017; Cao et al. 2017; Panichella et al. 2013; Yan et al. 2016a; Xia et al. 2017b; Nguyen et al. 2012; Damevski et al. 2018; Gao et al. 2018; Rao and Kak 2011). This may indicate that the LDA default implementation may not be adequate to support specific software engineering tasks or extract meaningful topics from all types of data. We discuss more about topic modeling techniques and their inputs in Section 6.2.2. Furthermore, we found that topic modeling is used to develop tools and methods to support software engineers and concrete tasks (the most frequently supported task we found was bug handling), but also as a data analysis technique for textual data to explore empirical questions (see for example the "oldest" paper in our sample published in 2009 (Bajracharya and Lopes 2009)).

One aspect that we did not specifically address in this review, but which impacts the applicability of topics models is their computational overhead. Computational overhead refers to processing time and computational resources (e.g., memory, CPU) required

Table 10 Procedures for naming topics

		References			
Procedure	Description	Manual	Automated	Manual & Automated	Total
Deducting name based on words in clusters	Assign names to topics based on understanding of the most frequent words in topics (in one paper Pettinato et al. (2019), authors asked domain experts to validate the names)	(Bajaj et al. 2014; Layman et al. 2016; Bagherzadeh and Khatchadourian 2019; Ahmed and Bagherzadeh 2018; Pagano and Maalej 2013; Noei et al. 2019; Hindle et al. 2015; Barua et al. 2014; Rosen and Shihab 2016; Pettinato et al. 2019; Yang et al. 2017; Aggarwal and Zhai 2012; Ray et al. 2014; Haque and Ali Babar 2020; Gorla et al. 2014; Tiarks and Maalej 2014; El Zarif et al. 2020; Mezouar et al. 2018; Han et al. 2020; Abdellatif et al. 2020; Bajracharya and Lopes 2009)	–	–	21
Naming based on most frequent word(s) in cluster	The most frequent word or the combination of frequent words in the topic were used as the name of that topic	(Galvis Carreno and Winbladh 2012; Li et al. 2018)	(Panichella et al. 2013)	–	3
Assigning predefined names to clusters	A list of predefined names is related to topics based on their similarities with the most frequent words in clusters	(Martin et al. 2015; Bajracharya and Lopes 2012; Zou et al. 2017; Taba et al. 2017)	(McIlroy et al. 2016; Yan et al. 2016b; Yan et al. 2016a; Fu et al. 2015; Chen et al. 2019; Gao et al. 2018)	(Hindle et al. 2013; Hindle et al. 2016)	12

for topic modeling. As discussed by others, topic modeling can be computational intensive (Hoffman et al. 2010; Treude and Wagner 2019; Agrawal et al. 2018). However, we found that only few papers (seven out of 111) mentioned computational overhead at all. From these seven papers, five mentioned processing time (Bavota et al. 2014b; Zhao et al. 2020; Luo et al. 2016; Moslehi et al. 2016; Chen et al. 2020), one paper mentioned computational requirements and some processing times (e.g., processor, data pre-processing time, LDA processing time and clustering processing time), and one paper only mention that their technique was processed in "few seconds" (Murali et al. 2017). Hence, based on the reviewed studies we cannot provide broader insights into the practical applicability and potential constraints of topic modeling based on the computational overhead.

6.1.2 Comparative Studies

As mentioned in Sections 5.2.1 and 5.2.3, we identified studies that used more than one topic modeling technique and compared their performance. In detail, we found studies that (1) compared topic modeling techniques to information extraction techniques, such as Vector Space Model (VSM), an algebraic model (Salton et al. 1975) (see Table 11), (2) proposed an approach that uses a topic modeling technique and compared it to other approaches (which may or may not use topic models) with similar goals (see Table 12), and (3) compared the performance of different settings for a topic modeling technique or a newly proposed approach that utilizes topic models (see Table 13). In column "Metric" of Tables 11, 12 and 13 the metrics show the metrics used in the comparisons to decide which techniques performed "better" (based on the metrics' interpretation). Metrics in bold were proposed for or adapted to a specific context (e.g., SCORE and Effort reduction), while the other metrics are standard NLP metrics (e.g., Precision, Recall and Perplexity). Details about the metrics used to compare the techniques are provided in Appendix A.2 - Metrics Used in Comparative Studies.

As shown in Table 11, ten papers compared topic modeling techniques to information extraction techniques. For example, Rosenberg and Moonen (2018) compared LSI with two other dimensionality reduction techniques (PCA and NMF) to group log messages of failing continuous deployment runs. Nine out of these ten papers presented explorations, i.e., studies experimented with different models to discuss their application to specific software engineering tasks, such as bug handling, software documentation and maintenance. Thomas et al. (2013) on the other hand experimented with multiple models to propose a framework for bug localization in source code that applies the best performing model.

Four papers in Table 11 (De Lucia et al. 2014; Tantithamthavorn et al. 2018; Abdellatif et al. 2019; Thomas et al. 2013) compared the performance of LDA, LSI and VSM with source code and issue/bug reports. Except for De Lucia et al. (2014), these studies applied Top-k accuracy (see Appendix A.2 - Metrics Used in Comparative Studies) to measure the performance of models, and the best performing model was VSM. Tantithamthavorn et al. (2018) found that VSM achieves both the best Top-k performance and the least required effort for method-level bug localization. Additionally, according to De Lucia et al. (2014), VSM possibly performed better than LSI and LDA due to the nature of the corpus used in their study: LDA and LSI are ideal for heterogeneous collections of documents (e.g., user manuals from different systems), but in De Lucia et al. (2014) study each corpus was a collection of code classes from a single software system.

Ten studies proposed an approach that uses a topic modeling technique and compared it to similar approaches (shown in Table 12). In column "Approaches compared" of Table 12, the approach in bold is the one proposed by the study (e.g., Cao et al. 2017) or the topic

Table 11 Studies that include comparison of topic models

Paper	Supported task	Techniques compared	Type of data	Dataset	Type of contribution	Metrics	Best performing technique
(De Lucia et al. 2014)	Documentation	LDA, LSI, VSM	Source code	JHotDraw and eXVantage	Exploration	Term entropy; **Average overlap**	VSM
(Tantithamthavorn et al. 2018)	Bug handling	LDA, LSI, VSM	Source code; Issue/bug report	Eclipse and Mozilla	Exploration	Top-k accuracy	VSM
(Abdellatif et al. 2019)	Maintenance	LDA, LSI, VSM	Issue/bug report	Data records from an Industry partner	Exploration	Top-k accuracy; Mean average precision (MAP)	VSM
(Liu et al. 2020)	Documentation	LDA, LSI, GVSM-based techniques	Commit messages; Issue/bug report	17 open source projects	Exploration	Average precision (AP)	GVSM-based techniques
(Binkley et al. 2015)	Documentation	LSI, VSM, VSM-WS, QL-lin, QL-Dir, QL-LDA	Source code	ArgoUML 0.22, Eclipse 3.0, JabRef 2.6, jEdit 4.3 and muCommander 0.8.5	Exploration	Mean Reciprocal Rank (MRR)	QL-LDA
(Rao and Kak 2011)	Bug handling	MLE-LDA; LDA; UM; VSM; LSA; CBDM	Source code	iBUGS benchmark dataset	Exploration	MAP; **SCORE**	UM
(Rosenberg and Moonen 2018)	Maintenance	LSI, PCA, NMF	Log information	Cisco Systems Norway log base	Exploration	Adjusted mutual information (AMI); **Effort reduction**; **Homogeneity**; **Completeness**	NMF
(Silva et al. 2016)	Bug handling	LDA; XScan	Source code	Rhino and jEdit	Exploration	Precision; Recall; F-measure	XScan
(Luo et al. 2016)	Testing	Call-graph-based; String-distance-based; LDA; Greedy techniques; Adaptive random testing	Test cases	30 open source Java programs	Exploration	**Average percentage of faults detected (APFD)**	Call-graph-based
(Thomas et al. 2013) [1]	Bug handling	LDA, LSI, VSM	Source code; Issue/bug report	Eclipse, Jazz and Mozilla	Approach	Top-k accuracy	VSM

[1] This study used the best performing models to develop an approach for bug localization

Table 12 Studies that include comparison of topic-based approaches

Paper	Supported task	Approaches compared	Type of data	Dataset	Type of contribution	Metrics	Best performing approach
(Naguib et al. 2013)	Bug handling	**LDA**; LDA-SVM	Issue/bug report	Atlas, Eclipse BIRT and Unicase	Approach	**Actual assignee hit Ratio**; **Top-k hit**	LDA
(Murali et al. 2017)	Bug handling	**Salento (LDA + Probabilistic Behavior Model and Artificial Neural Networks)**; Non-Bayesian method	Software documentation	Android APIs: alert dialogs, bluetooth sockets and cryptographic ciphers	Approach	Precision; Recall; **Anomaly score**	Salento
(Xia et al. 2017b)	Bug handling	**TopicMiner (MTM)**; Bugzie; LDA-KL; SVM-LDA; LDA-Activity	Issue/bug report	GCC, OpenOffice, Netbeans, Eclipse and Mozilla	Approach	Top-k accuracy	TopicMiner
(Thomas et al. 2014)	Testing	**LDA**; Call-graph-based; String-distance-based; Adaptive random testing	Source code	Software-artifact Infrastructure Repository (SIR)	Approach	**APFD**; Mann-Whitney-Wilcoxon test; *A* measure	LDA
(Jiang et al. 2019)	Requirements	**SAFER (LDA + Clustering technique)**; KNN+; CLAP	Software documentation	100 Google Play apps	Approach	**Hit ratio**; **Normalized Discounted Cumulative Gain (NDCG)**	SAFER
(Cao et al. 2017)	Architecting	**DAT-LDA + Clustering technique**; WTCluster; WT-LDA; CDSR; OD-DMSC; CDA-DMSC; CDT-DMSC	Software documentation	6629 mashup services from ProgrammableWeb	Approach	Precision; Recall; F-Measure; Purity; Term entropy	DAT-LDA + Clustering technique

Table 12 (continued)

Paper	Supported task	Approaches compared	Type of data	Dataset	Type of contribution	Metrics	Best performing approach
(Yan et al. 2016b)	Bug handling	**DPLSA**; LDA-KL; LDA-SVM	Issue/bug report	Eclipse, Bugzilla, Mylyn, GCC and Firefox	Approach	Recall @k; Perplexity	DPLSA
(Zhang et al. 2016)	Bug handling	**LDA + Clustering technique**; INSPect; NB Multinomial; DRETOM; DREX; DevRec	Issue/bug report	GCC, OpenOffice, Eclipse, NetBeans and Mozilla	Approach	Precision; Recall; F-measure; MRR	LDA + Clustering technique
(Demissie et al. 2020)	Architecting	**PREV (LDA + Clustering and Classification techniques)**; Covert; IccTA	Software documentation	11,796 Google Plays apps	Approach	Precision; Recall	PREV
(Blasco et al. 2020)	Requirements	**CODFREL (LSI + Evolutionary algorithm)**; Regular-LSI	Source code	Kromaia video game data	Approach	Precision; Recall; F-measure	CODFREL

Table 13 Studies that include comparison of different settings for a technique

Paper	Supported task	Techniques compared	Type of data	Dataset	Type of contribution	Metrics	Outcome of comparison
Biggers et al. (2014)	Documentation	LDA (settings tested: hyperparameters α and β, document, number of topics and query (i.e., a string formulated manually or automatically by an end user or developer))	Source code	ArgoUML, JabRef, jEdit, muCommander, Mylyn, Rhino	Exploration	Effectiveness measure	Recommendation for values of LDA hyperparameters and number of topics considering the number of documents used
Poshyvanyk et al. (2012)	Documentation	LSI-based technique (settings tested: number of documents, number of attributes, stemming of corpus and queries)	Source code	ArgoUML, Freenet, iBatis, JMeter, Mylyn and Rhino	Appproach	Precision; Recall; Effectiveness; Minimal browsing area (MBA); **Maximum possible precision gain (MPG)**	Configuration settings for the proposed technique based on the characteristics of the corpora used
Chen et al. (2014)	Bug handling	**AR-Miner**: Expectation Maximization for Naive Bayes (EMNB) + LDA; EMNB + ASUM	End user communication	Apps SwiftKey Keyboard, Facebook, Temple Run 2, Tap Fish	Aproach	Precision; Recall; F-measure; **NDCG**	EMNB + LDA
Fowkes et al. (2016)	Coding	**TASSAL** + LDA; **TASSAL** + VSM	Source code	Six open source Java projects	Approach	Area Under the Curve (AUC)	**TASSAL** + LDA

modeling technique used in their approach (e.g., Thomas et al. 2014). All newly proposed approaches were the best performing ones according to the metrics used.

In addition to the papers mentioned in Tables 11 and 12, four papers compared the performance of different settings for a topic modeling technique or tested which topic modeling technique works best in their newly proposed approach (see Table 13). Biggers et al. (2014) offered specific recommendations for configuring LDA when localizing features in Java source code, and observed that certain configurations outperform others. For example, they found that commonly used heuristics for selecting LDA hyperparameter values ($beta$ = 0.01 or $beta$ = 0.1) in source code topic modeling are not optimal (similar to what has been found by others, see Section 3.2). The other three papers (Chen et al. 2014; Fowkes et al. 2016; Poshyvanyk et al. 2012) developed approaches which were tested with different settings (e.g., the approach applying LDA or ASUM (Chen et al. 2014)).

Regarding the datasets used by comparative studies, only Rao and Kak (2011) used a benchmarking dataset (iBUGS). Most of the comparative studies (13 out of 24) used source code or issue/bug reports from open source software, which are subject to evolution. The advantage of using benchmarking datasets rather than “living” datasets (e.g., an open source Java system) is that its data will be static and the same across studies. Additionally, data in benchmarking datasets are usually curated. This means that the results of replicating studies can be compared to the original study when both used the same benchmarking dataset.

Finally, we highlight that each of the above mentioned comparisons has a specific context. This means that, for example, the type of data analyzed (e.g., Java classes), the parameter setting (e.g., k = 50), the goal of the comparison (e.g., to select the best model for bug localization or for tracing documentation in source code) and pre-processing (e.g., stemming and stop word removal) were different. Therefore, it is not possible to “synthesize” the results from the comparisons across studies by aggregating the different comparisons in different papers, even for studies that appear to have similar goals or use the same topic modeling techniques, such as comparing the same models with similar types of data (such as Tantithamthavorn et al. 2018 and Abdellatif et al. 2019).

6.2 RQ2: Inputs to Topic Models

6.2.1 Summary of Findings

Source code, developer communication and issue/bug reports were the most frequent types of data used for topic modeling in the reviewed papers. Consequently, most of the documents referred to individual or groups of functions or methods, individual Q&A posts, or individual bug reports; another frequent document was an individual user review (more discussions are in Section 6.2.3). We also found that few papers (16 out of 111) mentioned the actual length of documents used for topic modeling (we discuss this more in Section 6.2.2).

Regarding modeling parameters, most of the papers (93 out of 111) explicitly mentioned the configuration of at least one parameter, e.g., k, α or β for LDA. We observed that the setting $\alpha = 50/k$ and $\beta = 0.01$ (asymmetric α and symmetric β) as suggested by Steyvers and Griffiths (2010) and Wallach et al. (2009) was frequently used (28 out of 93 papers). Additionally, papers that applied LDA mostly used the default parameters of the tools used to implement LDA (e.g., Mallet [3] with $\alpha = 50/k$ and $\beta = 0.01$ as default). This finding is similar to what has been reported by others, e.g., according to another review by Agrawal et al. (2018), LDA is frequently applied “as is out-of-the-box” or with little tuning. This means that studies may rely on the default settings of the tools used with their topic modeling technique, such as Mallet and TMT, rather than try to optimize parameters.

6.2.2 Documents and Parameters for Topic Models

Short texts: According to Lin et al. (2014), topic models such as LDA have been widely adopted and successfully used with traditional media like edited magazine articles. However, applying LDA to informal communication text such as tweets, comments on blog posts, instant messaging, Q&A posts, may be less successful. Their user-generated content is characterized by very short document length, a large vocabulary and a potentially broad range of topics. As a consequence, there are not enough words in a document to create meaningful clusters, compromising the performance of the topic modeling. This means that probabilistic topic models such as LDA perform sub-optimally when applied "as is" with short documents even when hyperparameters (α and β in LDA) are optimized (Lin et al. 2014). In our sample there were only two papers that mentioned the use of a LDA-based technique specifically for short documents (Hu et al. 2019; Hu et al. 2018). Hu et al. (2019) and Hu et al. (2018) applied Twitter-LDA with end user reviews. Furthermore, Moslehi et al. (2018) used a weighting algorithm in documents to generate topics with more relevant words, they also acknowledge that the use of a short text technique could have improved their topic model.

As shown in Table 7, few papers mentioned the actual length of documents. Considering a single document from a corpus, we observed that most papers potentially used short texts (all documents found in papers are shown in Fig. 3). For example, papers used an individual search query (Xia et al. 2017a), an individual Q&A post (Barua et al. 2014), an individual user review (Nayebi et al. 2018), or an individual commit message (Canfora et al. 2014) as a document. Among the papers that mentioned document length, the shortest documents were an individual commit message (9 to 20 words) (Canfora et al. 2014) and an individual method (14 words) (Tantithamthavorn et al. 2018). Both studies applied LDA.

Two approaches to improve the performance of LDA when analyzing short documents are *pooling* and *contextualization* (Lin et al. 2014). Pooling refers to aggregating similar (e.g., semantically or temporally) documents into a single document (Mehrotra et al. 2013). For example, among the papers analysed, Pettinato et al. (2019) used temporal pooling and combined short log messages into a single document based on a temporal order. Contextualization refers to creating subsets of documents according to a type of context; considering tweets as documents, the type of context can refer to time, user and hashtags associated with tweets (Tang et al. 2013). For example, Weng et al. (2010) combined all the individual tweets of an author into one pseudo-document (rather than treating each tweet as a document). Therefore, with the contextualization approach, the topic model uses word co-occurrences at a context level instead of at the document level to discover topics.

Hyperparameters Table 14 shows the hyperparameter settings and types of data of the papers that mentioned the value of at least one model parameter. In Table 14 we also highlight the topic modeling techniques used. Note that some topic modeling techniques (e.g., RTM) can receive more parameters that the ones mentioned in Table 14 (e.g., number of documents, similarity thresholds); all parameters mentioned in papers are available online in the raw data of our study [1]. When comparing hyperparameter settings, topic modeling techniques and types of data, we observed the following:

- Papers that used LDA-GA, an LDA-based technique that optimizes hyperparameters with Genetic algorithms, applied it to data from developer documentation or source code;

Table 14 Number of papers by type of data and hyperparameter settings

Types of Data	α based on k	Fixed α and β	Varying α or β	Optimized parameters
Commit messages	DPLSA: 1 Semi-supervised LDA: 1	LDA: 1 RTM: 1	–	–
Developer communication	LDA: 8	LDA: 3 LLDA; L2H: 1	–	–
End user communication	LDA: 1	LDA: 1 LDA; ASUM: 1 LLDA: 1 AOLDA: 1	–	–
Issue/bug report	LDA: 3 LDA; LSI: 1 DPLSA: 1 MTM: 1	LDA: 3 RTM: 1 LDA; LLDA: 1	LDA: 1 MLE-LDA: 1	–
Log information	LDA: 2	–	–	–
Search query	–	LDA: 2	–	–
End user documentation	LDA: 3	LDA: 3	LDA: 1	–
Developer documentation	–	DAT–LDA: 1	–	LDA–GA: 1
Source code	LDA: 6 LDA; LSI: 1	LDA: 3 BugScout: 1 RTM: 3 LDA; LSI: 1	LDA: 2 MLE–LDA: 1 QL–LDA; LSI: 2	LDA–GA: 2
"Lessons learned"	–	–	–	–
Transcript	LDA: 3	–	–	–
URL content	–	LDA: 1	–	–

- LDA was used with all three types of hyperparameter settings across studies. The most common setting was α based on k for developer communication and source code;
- Most of the LDA-based techniques applied fixed values for α and β.

Most of the papers that applied only LSI as the topic modeling technique did not mention hyperparameters. As LSI is a model simpler than LDA, it generally requires the number of topics k. For example, a paper that applied LSI to source code mentioned α and k (Poshyvanyk et al. 2012).

Number of topics By relating the type of data to the number of topics, we aimed at finding whether the choice of the number of topics is related to the data used in the topic modeling techniques (see also Table 7). However, the number of topics used and data in the studies are rather diverse. Therefore, synthesizing practices and offering insights from previous studies on how to choose the number topics is rather limited.

From the 90 papers that mentioned number of topics (k), we found that 66 papers selected a specific number of topics (e.g., based on previous works with similar data or addressing the same task), while 24 papers used several numbers of topics (e.g., Yan et al. (2016b) used 10 to 120 topics in steps of 10). To provide an example of how the number of topics differed even when the same type of data was analyzed with the same topic modeling technique, we looked at studies that applied LDA in textual data from developer communication (mostly Q&A posts) to propose an approach to support documentation. For these papers we found one paper that did not mention k (Henß et al. 2012), one paper that modeled different numbers of topics ($k = 10, 20, 30$) (Asuncion et al. 2010), one paper that modeled $k = 15$ (Souza et al. 2019) and another paper that modeled $k = 40$ (Wang et al. 2015). This illustrates that there is no common or recommended practice that can be derived from the papers.

Some papers mentioned that they tested several numbers of topics before selecting the most appropriate value for k (in regards to studies' goals) but did not mention the range of values tested. In regards to papers that mentioned such range, we identified four studies (Nayebi et al. 2018; Chen et al. 2014; Layman et al. 2016; Nabli et al. 2018) that tested several values for k and used *perplexity* (see details in Appendix A.2 - Metrics Used in Comparative Studies) of models to evaluate which value of k generated the best performing model; three studies (Zhao et al. 2020; Han et al. 2020; El Zarif et al. 2020) also selected the number of topics after testing several values for k; however they used *topic coherence* (Röder et al. 2015) to evaluate models. One paper (Haque and Ali Babar 2020) used both *perplexity* and *topic coherence* to select a value for k. Metrics of topic coherence score the probability of a pair of words from the resulted word clusters being found together in (a) external data sources (e.g., Wikipedia pages) or (b) in the documents used by the topic model that generated those word clusters (Röder et al. 2015).

6.2.3 Supported Tasks, Types of Data and Types of Contribution

We looked into the relationship between the tasks supported by papers, the type of data used and the types of contributions (see Table 15). We observed the following:

- Source code was a frequent type of data in papers; consequently it appeared for almost all supported tasks, except for exploratory studies;
- Considering exploratory studies, most papers used developer communication (13 out of 21), followed by search queries and end user communication (three papers each);
- Papers that supported bug handling mostly used issue/bug reports, source code and end user communication;
- Log information was used by papers that supported maintenance, bug handling, and coding;
- Considering the papers that supported documentation, three used transcript texts from speech;
- From the four papers related to the type of data developer documentation, two supported architecting tasks and the other two, documentation tasks.
- Regarding the type of data, URLs and transcripts were only used in studies that contributed an approach.

We found that most of the exploratory studies used data that is less structured. For example, developer communication, such as Q&A posts and conversation threads generally do not follow a standardized template. On the other hand, issue reports are typically submitted through forms which enforces a certain structure.

Table 15 Number of papers by types of data and supported tasks

	Supported Tasks								
Types of data	Architecting	Bug handling	Coding	Documentation	Maintenance	Refactoring	Requirements	Testing	Exploratory studies
Commit messages	Exploration: 1	Approach: 3 Exploration [C]: 1	–	Approach: 1 Exploration [C]: 1	Approach: 1	Exploration: 1	–	–	Exploration: 1
Developer communication	–	Approach: 1	–	Approach: 5	Approach: 1	–	–	–	Exploration: 13
End user communication	–	Approach: 4 Exploration: 2	–	–	Approach: 1 Exploration: 1	–	Approach: 1	–	Exploration: 3
Issue/bug report	Exploration: 1 Exploration [C]: 1	Approach: 6 Exploration: 2 Approach [C]: 5 Exploration [C]: 2	–	Approach: 2 Exploration [C]: 1	Exploration [C]: 1	–	–	–	Exploration: 1
Log information	–	Approach: 1	Approach: 1	–	Approach: 1 Exploration: 1 Exploration [C]: 1	–	–	–	–
Search query	–	–	–	Approach: 1	–	–	–	–	Exploration: 3
End user documentation	Approach: 2 Approach [C]: 1	Exploration: 1 Approach [C]: 1	Exploration: 1	Approach: 4	Approach: 1	–	Approach [C]: 1	Approach: 1	Exploration: 2
Developer documentation	Approach: 1 Approach [C]: 1	–	–	Approach: 2	–	–	–	–	–
Source code	Approach: 2 Exploration: 2	Approach: 4 Exploration: 2 Approach [C]: 1 Exploration [C]: 3	Approach: 2 Exploration: 1 Approach [C]: 1	Approach: 5 Exploration [C]: 3	Approach: 1 Exploration: 3	Approach: 2	Approach: 1 Approach [C]: 1	Approach [C]: 1 Exploration [C]: 1	–
"Lessons learned"	–	–	–	–	Exploration [C]: 1	–	–	–	–
Transcript	–	–	–	Approach: 3	–	–	–	–	–
URL content	Approach: 1	–	–	–	–	–	–	–	–

[C] Studies that also contributed with a Comparison

6.3 RQ3: Data Pre-processing

6.3.1 Summary of Findings

Most of the papers (91 out of 111) pre-processed the textual data before topic modeling. Removing noisy content was the most frequent pre-processing step (as typical for natural language processing), followed by stemming and splitting words. Miner et al. (2012) consider tokenizing as one of the basic data pre-processing steps in text mining. However, in comparison to other basic pre-processing steps such as stemming, splitting words and removing noise, tokenizing was not frequently found in papers (it was at least not mentioned in papers).

Eight papers (Henß et al. 2012; Xia et al. 2017b; Ahasanuzzaman et al. 2019; Abdellatif et al. 2019; Lukins et al. 2010; Tantithamthavorn et al. 2018; Poshyvanyk et al. 2012; Binkley et al. 2015) tested how pre-processing steps affected the performance of topic modeling or topic model-based approaches. For example, Henß et al. (2012) tested several pre-processing steps (e.g., removing stop words, long paragraphs and punctuation) in e-mail conversations analyzed with LDA. They found that removing such content increased LDA's capability to grasp the actual semantics of software mailing lists. Ahasanuzzaman et al. (2019) proposed an approach which applies LDA and Conditional Random Field (CRF) to localize concerns in Stack Overflow posts. The authors did not incorporate stemming and stop words removal in their approach because in preliminary tests these pre-processing steps decreased the performance of the approach.

6.3.2 Pre-processing Different Types of Data

Table 16 shows how different types of data were pre-processed. We observed that stemming, removing noise, lowercasing, and splitting words were commonly used for all types of data. Regarding the differences, we observed the following:

- For developer communication there were specific types of noisy content that was removed: URLs, HTML tags and code snippets. This might have happened because most of the papers used Q&A posts as documents, which frequently contain hyperlinks and code examples;
- Removing non-informative content was frequently applied to end user communication and end user documentation;
- Expanding contracted terms (e.g., “didn't” to “did not”) were applied to end user communication and issue/bug reports;
- Removing empty documents and eliminating extra white spaces were applied only in end user communication. Empty documents occurred in this type of data because after the removal of stop words no content was left (Chen et al. 2014);
- For source code there was a specific noise to be removed: program language specific keywords (e.g., “public”, “class”, “extends”, “if”, and “while”).

Table 16 shows that splitting words, stop words removal and stemming were frequently applied to source code and most of these studies (15) applied these three steps at the same time. Studies that performed these pre-processing steps to source code mostly used methods, classes, or comments in classes/methods as documents. For example, Silva et al. (2016) who applied LDA, performed these three pre-processing steps in classes from two open source systems using TopicXP (Savage et al. 2010). TopicXP is a Eclipse plug-in that extracts

Table 16 Number of papers by type of data and pre-processing steps

	Type of data											
Pre-processing steps	Commit messages	Developer communication	Developer documentation	End user communication	End user documentation	Issue/bug report	“Lessons learned”	Log information	Search query	Source code	Transcript	URL content
Resolving negations	0	0	0	2	1	0	0	0	0	0	0	0
Correcting typos	0	0	0	6	1	1	0	0	0	0	0	0
Expanding contractions	0	0	0	2	0	1	0	0	0	0	0	0
Resolving synonyms	1	0	0	2	1	0	0	0	0	1	0	0
Splitting sentences or a document into n documents	3	1	0	1	3	3	0	0	0	1	0	0
Lemmatizing	1	2	0	5	1	1	0	0	0	2	0	0
Identifying n-grams	0	3	0	2	0	0	0	0	0	1	0	0
Lowercasing	1	1	0	5	1	3	0	2	1	5	1	1
Tokenizing	1	1	0	2	2	5	0	2	1	4	0	0
Splitting words	4	0	0	0	2	8	0	0	2	24	1	0
Stemming	5	8	3	9	8	14	1	1	1	21	2	1
Removing empty documents	0	0	0	1	0	0	0	0	0	0	0	0
Removing long paragraphs	0	1	0	0	0	0	0	0	0	0	0	0
Removing short documents	0	0	0	1	1	0	0	0	0	0	0	0
Removing extra white space	0	0	0	1	0	0	0	0	0	0	0	0
Removing non-informative content	1	1	0	4	4	2	0	0	0	1	0	0
Removing words shorter than four, three or two letters	0	0	0	1	0	1	0	1	0	1	0	0
Removing least frequent terms	0	2	0	2	1	2	0	0	0	1	0	0
Removing most frequent terms	0	2	0	2	1	0	0	0	0	3	0	0
Removing code snippets	1	7	0	0	0	0	0	0	1	1	0	0
Removing HTML tags	1	6	0	0	2	1	0	0	0	0	0	0

Table 16 (continued)

Pre-processing steps	Type of data											
	Commit messages	Developer communication	Developer documentation	End user communication	End user documentation	Issue/bug report	"Lessons learned"	Log information	Search query	Source code	Transcript	URL content
Removing programming language keywords	1	3	0	0	0	4	0	0	1	19	0	0
Removing symbols and special characters	2	3	0	2	2	3	0	0	2	6	2	1
Removing punctuation	2	4	0	2	3	4	0	2	0	5	2	1
Removing stop words	6	16	2	10	8	15	1	3	0	23	2	1
Remove URL	1	4	0	0	1	0	0	0	0	0	0	0
Remove numbers	1	4	0	1	3	4	0	1	0	5	2	0

source code, pre-process it and executes LDA. This plug-in implements splitting words, stop words removal and stemming.

Splitting words was the most frequent pre-processing step in source code. Studies used this step to separate Camel Cases in methods and classes (e.g., the class constructor InvalidRequestTest produces the terms “invalid”, “request” and “test”). For example, Tantithamthavorn et al. (2018) compared LDA, LSI and VSM testing different combinations of pre-processing steps to the methods’ identifiers inputted to these techniques. The best performing approach was VSM with splitting words, stop words removal and stemming.

Removing stop words in source code refer to the exclusion of the most common words in a language (e.g., “a/an” and “the” in English), as in studies that used other types of data. Removing stop words in source code is also different from removing programming language keywords and studies mentioned these as separate steps. Lukins et al. (2010), for example, tested how removing stop words from their documents (comments and identifiers of methods) affected the topics generated by their LDA-based approach. They found that this step did not improve the results substantially.

As mentioned in Section 5.4, stemming is the process of normalizing words into their single forms by identifying and removing prefixes, suffixes and pluralisation (e.g., “development”, “developer”, “developing” become “develop”). Regarding stemming in source code, papers normalized identifiers of classes and methods, comments related to classes and methods, test cases or a source code file. Three papers tested the effect of this pre-processing step in the performance of their techniques (Tantithamthavorn et al. 2018; Poshyvanyk et al. 2012; Binkley et al. 2015), and one of these papers also tested removing stop words and splitting words (Tantithamthavorn et al. 2018). Poshyvanyk et al. (2012) tested the effect of stemming classes in the performance of their LSI-based approach. The authors concluded that stemming can positively impact features localization by producing topics (“concept lattices” in their study) that effectively organize the results of searches in source code. Binkley et al. (2015) compared the performance of LSI, QL-LDA and other techniques. They also tested the effects of stemming (with two different stemmers: Porter [9] and Krovetz [10]) and non-stemming methods from five open source systems. These authors found that they obtained better performances in terms of models’ Mean Reciprocal Rank (MRR, details in Appendix A.2 - Metrics Used in Comparative Studies) with non-stemming.

Additionally, we found that even though some papers used the same type of data, they pre-processed data differently since they had different goals and applied different techniques. For example, Ye et al. (2017), Barua et al. (2014) and Chen et al. (2019) used developer communication (Q&A posts as documents). Ye et al. (2017) and Barua et al. (2014) removed stop words, code snippets and HTML tags, while Barua et al. (2014) also stemmed words. On the other hand, Chen et al. (2019) removed stop words and the least and the most frequent words, and identified bi-grams. Some studies considered the advice on data pre-processing from previous studies (e.g., Chen et al. 2017; Li et al. 2018), while others adopted steps that are commonly used in NLP, such as noise removal and stemming (Miner et al. 2012) (e.g., Demissie et al. 2020). This means that the choice of pre-processing steps do not only depend on the characteristics of the type of data inputted to topic modeling techniques.

[9] https://tartarus.org/martin/PorterStemmer/
[10] https://pypi.org/project/krovetz/

6.4 RQ4: Assigning Names to Topics

Most papers did not mention if or how they named topics. The majority of papers that explicitly assigned names to topics (27 out of 36) used a manual approach and relied on human judgment (researchers' interpretation) of words in clusters. One paper (Rosen and Shihab 2016) justified their use of a manual approach by arguing that there was no tool that could give human readable topics based on word clusters. Thus, authors checked every word cluster generated and the documents used (an individual question of a Q&A website) to make sure they would label topics appropriately.

Table 17 shows how topics were named and the type of data analyzed. Table 18 shows how topics were named and the type of contributions they make. We observed the following:

- Studies that modeled topics from developer documentation, transcripts and URLs did not mention topic naming. Studies that contributed with both exploration and comparison also did not mention topic naming;
- Topics were mostly named in studies that used data from developer communication (ten occurrences) and in exploratory studies (22 occurrences).
- From studies that compared topic models or topic modeling-based approaches (see Section 6.1.2), only one study (Yan et al. 2016b) named topics (automatically with predefined labels).

Fourteen papers acknowledged limitations of manual topic naming:

- Twelve papers (Bagherzadeh and Khatchadourian 2019; Ahmed and Bagherzadeh 2018; Martin et al. 2015; Hindle et al. 2013; Pagano and Maalej 2013; Zou et al. 2017; Pettinato et al. 2019; Layman et al. 2016; Ray et al. 2014; Tiarks and Maalej 2014; Mezouar et al. 2018; Abdellatif et al. 2020) acknowledged that how topics were named could be a threat to validity. For example, Layman et al. (2016) mentioned that they did not evaluate the accuracy of the manual topic naming, which was based on their expertise.
- Three papers (Hindle et al. 2015; Bajracharya and Lopes 2012; Li et al. 2018) mentioned difficulties to assign names to topics. Hindle et al. (2015), for example, explained

Table 17 Number of papers by topic naming procedure and types of data

	Topic naming procedure		
Types of data	Based on word clusters	Most frequent words	Predefined names
Commit messages	Manual: 1	–	Automated: 2 Automated & Manual: 1
Developer communication	Manual: 9	Automated: 1	Automated: 1 Manual: 1
End user communication	Manual: 2	Manual: 1	Automated: 2 Manual: 1
End user documentation	Manual: 5	–	–
Issue/bug report	Manual: 3	–	Automated: 1 Automated & Manual: 1
Log information	Manual: 1	–	–
Search query	Manual: 1	–	Manual: 1
Source code	–	Automated: 1 Manual: 1	Manual: 1

Table 18 Number of papers by topic naming procedure and types of contribution

	Topic naming procedure		
Types of contribution	Based on word clusters	Most frequent words	Predefined names
Approach	Manual: 5	Automated: 1 Manual: 1	Automated: 4 Automated & Manual: 2
Approach & Comparison	–	–	Automated: 1
Exploration	Manual: 16	Manual: 1	Automated: 1 Manual: 4

that labeling topics was difficult due to many project specific and unclear terms in clusters.

- One paper (Pettinato et al. 2019) acknowledged that there is another topic naming approach that could be applied to their data: authors acknowledged that an automated extraction of topic names could replace manual labeling.

Hindle et al. (2015) provided some recommendations on topic analysis in software engineering based on their experiences. Below are some of their recommendations related to topic naming:

- Some of the generated topics will not be relevant (e.g., clusters filled with common terms may not address any particular subject) and topics may be duplicated. This means that not all topics have to be named and used for analysis;
- Domain experts can label topics better than non-experts, because they are more familiar to domain-specific keywords that may appear in word clusters;
- It is important to rely on the relationship between topics generated and the original data. Hindle et al. (2015) argued that “the content of the topic can be interpreted in many different ways and LDA does not look for the same patterns that people do”.

6.5 Implications

The goal of this study was to describe how topic modeling is applied in software engineering research. We found studies that experimented, explored data, or proposed solutions to support different software engineering tasks with topic models. Our findings help researchers and practitioners as follows:

- **Understand which topic modeling techniques to use for what purpose**. Researchers and practitioners that are going to select and apply a topic modeling technique, for example, to refactor legacy systems; may consider the experiences of other studies with similar objectives.
- **Pre-processing based on the type of data to be modeled**. Pre-processing steps depend on the type of data analyzed (e.g., removing HTML tags in developer communication, mainly Q&A posts). Researchers and practitioners who, for example, intend to model topics from source code; may consider the same pre-processing steps that other studies applied to source code.
- **Understand how to name topics**. Researchers and practitioners may check how other studies named topics to get insights on how to give meaning to their own topics.

We present some additional insights:

- **Appropriateness of topic modeling**. Although we found that most of papers applied LDA "as is", it may not be the best approach for other studies or for practical application. LDA is popular because it is an unsupervised model, i.e., it does not require previous knowledge about the data (e.g., pre-defined classes for model training), it is statistically more rigorous than other techniques (e.g., LSI), and it discovers latent relationships (i.e., topics) between documents in a large textual corpus (Griffiths and Steyvers 2004). However, LDA is an unstable and non-deterministic model. This means that generated topics cannot be replicated by others, even if the same model inputs (data pre-processing and configuration of parameters) are used. Furthermore, LDA performs poorly with short documents (Lin et al. 2014).
- **Meaningful topics**. Topic models should discover semantically meaningful topics. Chang et al. (2009) argue about the importance of the interpretability of topics generated by probabilistic topic modeling techniques such as LDA. To create meaningful and replicable topics with LDA, Mantyla et al. (2018) highlight the importance of stabilizing the topic model (e.g., through tuning (Agrawal et al. 2018)) and advocate the use of stability metrics (e.g., rank-biased overlap - RBO (Mantyla et al. 2018)).
- **Research opportunities**. Researchers interested in investigating topic modeling in software engineering may consider developing guidelines for researchers on how to use topic modeling, depending on the type of data, goals, etc. Further studies may also explore issues related to approaches for naming topics (e.g., based on domain experts), on the evaluation of the semantic accuracy of topics generated (e.g., how meaningful the topics are and if the context of document have to be considered), and on metrics to measure the performance of topic models supporting different software engineering tasks.

6.6 Threats to Validity

We analysed the validity threats to our study considering four types of threats to validity in systematic literature mapping studies (Petersen et al. 2015):

Theoretical validity This threat to validity refers to concerns related to capturing the data as intended, i.e., bias and limitations in the data selection and extraction. As we focused on the practice of topic modeling in software engineering, we restricted the search to highly ranked software engineering venues, which generally publish more mature studies. We used "topic model", "topic model[l]ing", "lsi", "lda", "plsi", "latent dirichlet allocation", "latent semantic" as search keywords to find all papers related to topic modeling. To select papers to the survey, we established inclusion and exclusion criteria. One author selected the papers and the others checked whether the selection criteria were applied appropriately. Furthermore, to minimize this threat in relation to data extraction, we first defined the data items (details are in Table 2) to be extracted from papers and the relevance of the data for each research question. Then, one author extracted the data and the others reviewed the results. Controversial data results were discussed to reach agreement.

Descriptive validity In the context of a literature survey, descriptive validity refers to bias and limitations in data synthesis and the accurate and objective description of the data. To mitigate this threat, we described in detail how the data was synthesized (see Section 4.3); furthermore, one of the authors synthesized the data and the others reviewed the results. Still, data and results depend on what is reported in papers which was sometimes incomplete, inconsistent or inaccurate (see for example information about document length).

Interpretive validity This threat to validity refers to bias and limitations in the results of the data analysis. We frequently reviewed the synthesized data during the data analysis and the authors with more experience in this type of study checked the occurrence of inconsistencies in results. Still, we recognize that interpretation bias may not have been removed completely.

Repeatability This threat to validity concerns whether the study and its results can be replicated. To reduce this threat, we described our search procedures in detail (Section 4), and the processes of data selection, extraction and synthesis in detail. We also followed general guidelines for systematic literature review as suggested by Kitchenham (2004) and mapping study method as suggested by Petersen et al. (2015). Furthermore, raw data of our study are available online [1].

7 Conclusions

We analyzed 111 papers that applied topic modeling. These papers were published in the last twelve years (2009-2020) in ten highly ranked software engineering venues (five conferences and five journals). Below we summarize our findings:

- LDA and LDA-based techniques are the most frequently used topic modeling techniques;
- Topic modeling was mostly used to develop techniques for handling bugs (e.g., to predict defects). Exploratory studies that use topic modeling as a data analysis technique were also frequent;
- Most papers modeled topics from source code (using methods as documents);
- Most papers used LDA "as is" and without adapting values of hyperparameters (α and β);
- Most papers describe pre-processing. Some pre-processing steps depend on the type of textual data used (e.g., removal of URL and HTML tags), while others are commonly used in NLP techniques (e.g., stop words removal or stemming);
- Only 36 (out of 111) papers named the topics. When naming topics, papers mostly adopted manual topic naming approaches such as deducting names (or labeling predefined names) based on the meaning of frequent words in that topic.

By analysing topic modeling techniques, data inputs, data pre-processing, and how topics were named, we identified characteristics and limitations in the use of topic models. Our study can provide insights and references to researchers and practitioners to make the best use of topic modeling, considering the experiences from previous studies.

Our study did not investigate all potential characteristics of topic modeling in software engineering or compared topic models to other text mining techniques. To answer our research questions, we analyzed data items shown in Table 2. Future studies may investigate other characteristics of the use of topic modeling in software engineering, for example, topic modeling tools or libraries (e.g., Mallet) used; the context of a specific supported software engineering task; or compare topic modeling techniques to other text mining techniques, such as clustering and summarization (e.g., sentence or document embeddings).

Furthermore, future work can reflect on other fields or uses of topic modeling to contrast how topic modeling is applied in software engineering. Further studies may also investigate how papers evaluate the performance of their topic modeling techniques, how papers evaluate the the quality of the generated topics, and how exactly word clusters were used when topics were not named.

Appendix A

A.1 Papers Reviewed

Year	Venue	Title	Reference
2010	ICSE	Software Traceability with Topic Modeling	(Asuncion et al. 2010)
2017	ICSE	An Unsupervised Approach for Discovering Relevant Tutorial Fragments for APIs	(Jiang et al. 2017)
2013	ICSE	How to Effectively Use Topic Models for Software Engineering Tasks? An Approach Based on Genetic Algorithms	(Panichella et al. 2013)
2013	ICSE	Analysis of User Comments: An Approach for Software Requirements Evolution	(Galvis Carreno and Winbladh 2012)
2014	ICSE	AR-miner: Mining Informative Reviews for Developers from Mobile App Marketplace	(Chen et al. 2014)
2012	ICSE	Semi-automatically extracting FAQs to improve accessibility of software development knowledge	(Henß et al. 2012)
2019	MSR	Exploratory Study of Slack Q&A Chats as a Mining Source for Software Engineering Tools	(Chatterjee et al. 2019)
2014	MSR	Mining Questions Asked by Web Developers	(Bajaj et al. 2014)
2016	MSR	Topic Modeling of NASA Space System Problem Reports: Research in Practice	(Layman et al. 2016)
2013	MSR	Using citation influence to predict software defects	(Hu and Wong 2013)
2013	MSR	Bug report assignee recommendation using activity profiles	(Naguib et al. 2013)
2018	MSR	Feature Location Using Crowd-Based Screencasts	(Moslehi et al. 2018)
2016	MSR	On Mining Crowd-Based Speech Documentation	(Moslehi et al. 2016)
2015	MSR	The App Sampling Problem for App Store Mining	(Martin et al. 2015)
2009	MSR	Mining search topics from a code search engine usage log	(Bajracharya and Lopes 2009)
2012	ASE	Duplicate Bug Report Detection with a Combination of Information Retrieval and Topic Modeling	(Nguyen et al. 2012)
2011	ASE	A Topic-based Approach for Narrowing the Search Space of Buggy Files from a Bug Report	(Nguyen et al. 2011)

Year	Venue	Title	Reference
2019	FSE	Going Big: A Large-scale Study on What Big Data Developers Ask	(Bagherzadeh and Khatchadourian 2019)
2017	FSE	Bayesian Specification Learning for Finding API Usage Errors	(Murali et al. 2017)
2013	MSR	Bug report assignee recommendation using activity profiles	(Naguib et al. 2013)
2018	MSR	Feature Location Using Crowd-Based Screencasts	(Moslehi et al. 2018)
2016	MSR	On Mining Crowd-Based Speech Documentation	(Moslehi et al. 2016)
2015	MSR	The App Sampling Problem for App Store Mining	(Martin et al. 2015)
2009	MSR	Mining search topics from a code search engine usage log	(Bajracharya and Lopes 2009)
2012	ASE	Duplicate Bug Report Detection with a Combination of Information Retrieval and Topic Modeling	(Nguyen et al. 2012)
2011	ASE	A Topic-based Approach for Narrowing the Search Space of Buggy Files from a Bug Report	(Nguyen et al. 2011)
2019	FSE	Going Big: A Large-scale Study on What Big Data Developers Ask	(Bagherzadeh and Khatchadourian 2019)
2017	FSE	Bayesian Specification Learning for Finding API Usage Errors	(Murali et al. 2017)
2018	ESEM	What Do Concurrency Developers Ask About?: A Large-scale Study Using Stack Overflow	(Ahmed and Bagherzadeh 2018)
2017	TSE	Improving Automated Bug Triaging with Specialized Topic Model	(Xia et al. 2017b)
2014	TSE	Methodbook: Recommending move method refactorings via relational topic models	(Bavota et al. 2014b)
2018	TSE	Predicting Future Developer Behavior in the IDE Using Topic Models	(Damevski et al. 2018)
2013	EMSE	Integrating information retrieval, execution and link analysis algorithms to improve feature location in software	(Dit et al. 2013)
2013	EMSE	Automated topic naming: supporting cross-project analysis of software maintenance activities	(Hindle et al. 2013)
2017	EMSE	What do developers search for on the web?	(Xia et al. 2017a)
2013	EMSE	How do open source communities blog?	(Pagano and Maalej 2013)
2014	EMSE	How changes affect software entropy: an empirical study	(Canfora et al. 2014)
2019	EMSE	Towards prioritizing user-related issue reports of mobile applications	(Noei et al. 2019)
2019	EMSE	CAPS: a supervised technique for classifying Stack Overflow posts concerning API issues	(Ahasanuzzaman et al. 2019)

Year	Venue	Title	Reference
2019	EMSE	Studying the consistency of star ratings and reviews of popular free hybrid Android and iOS apps	(Hu et al. 2019)
2015	EMSE	Do topics make sense to managers and developers?	(Hindle et al. 2015)
2017	EMSE	Predicting the delay of issues with due dates in software projects	(Choetkiertikul et al. 2017)
2017	EMSE	The structure and dynamics of knowledge network in domain-specific Q&A sites: a case study of stack overflow	(Ye et al. 2017)
2012	EMSE	Analyzing and mining a code search engine usage log	(Bajracharya and Lopes 2012)
2018	EMSE	Studying software logging using topic models	(Li et al. 2018)
2014	EMSE	Static test case prioritization using topic models	(Thomas et al. 2014)
2017	EMSE	Will this localization tool be effective for this bug? Mitigating the impact of unreliability of information retrieval based bug localization tools	(Le et al. 2017)
2016	EMSE	Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews	(McIlroy et al. 2016)
2014	EMSE	What are developers talking about? An analysis of topics and trends in Stack Overflow	(Barua et al. 2014)
2018	EMSE	App store mining is not enough for app improvement	(Nayebi et al. 2018)
2016	EMSE	What are mobile developers asking about? A large scale study using stack overflow	(Rosen and Shihab 2016)
2018	EMSE	Fusing multi-abstraction vector space models for concern localization	(Zhang et al. 2018)
2014	TOSEM	Improving Software Modularization via Automated Analysis of Latent Topics and Dependencies	(Bavota et al. 2014a)
2019	TOSEM	Recommending New Features from Mobile App Descriptions	(Jiang et al. 2019)
2016	IST	Combining lexical and structural information to reconstruct software layers	(Belle et al. 2016)
2017	IST	Towards comprehending the non-functional requirements through Developers' eyes: An exploration of Stack Overflow using topic analysis	(Zou et al. 2017)
2015	IST	MSR4SM: Using topic models to effectively mining software repositories for software maintenance tasks	(Sun et al. 2015)
2019	IST	Log mining to re-construct system behavior: An exploratory study on a large telescope system	(Pettinato et al. 2019)
2017	IST	Characterizing malicious Android apps by mining topic-specific data flow signatures	(Yang et al. 2017)
2019	IST	Automatic recall of software lessons learned for software project managers	(Abdellatif et al. 2019)
2010	IST	Bug localization using latent Dirichlet allocation	(Lukins et al. 2010)
2019	IST	Bootstrapping cookbooks for APIs from crowd knowledge on Stack Overflow	(Souza et al. 2019)

Year	Venue	Title	Reference
2017	IST	Domain-aware Mashup service clustering based on LDA topic model from multiple data sources	(Cao et al. 2017)
2018	IST	The impact of IR-based classifier configuration on the performance and the effort of method-level bug localization	(Tantithamthavorn et al. 2018)
2016	IST	A component recommender for bug reports using Discriminative Probability Latent Semantic Analysis	(Yan et al. 2016b)
2015	IST	Automated classification of software change messages by semi-supervised Latent Dirichlet Allocation	(Fu et al. 2015)
2017	JSS	Mining domain knowledge from app descriptions	(Liu et al. 2017)
2016	JSS	Towards more accurate severity prediction and fixer recommendation of software bugs	(Zhang et al. 2016)
2019	JSS	Not all bugs are the same: Understanding, characterizing, and classifying bug types	(Catolino et al. 2019)
2017	JSS	Enhancing developer recommendation with supplementary information via mining historical commits	(Sun et al. 2017)
2019	JSS	Modeling stack overflow tags and topics as a hierarchy of concepts	(Chen et al. 2019)
2017	JSS	An exploratory study on the usage of common interface elements in android applications	(Taba et al. 2017)
2017	JSS	Topic-based software defect explanation	(Chen et al. 2017)
2019	JSS	Co-change patterns: A large scale empirical study	(Silva et al. 2019)
2018	JSS	Efficient cloud service discovery approach based on LDA topic modeling	(Nabli et al. 2018)
2018	JSS	Lascad: Language-agnostic software categorization and similar application detection	(Altarawy et al. 2018)
2016	JSS	Automatically classifying software changes via discriminative topic model: Supporting multi-category and cross-project	(Yan et al. 2016a)
2013	TOSEM	Concept location using formal concept analysis and information retrieval	(Poshyvanyk et al. 2012)
2020	EMSE	A feature location approach for mapping application features extracted from crowd-based screencasts to source code	(Moslehi et al. 2020)
2020	EMSE	Security analysis of permission re-delegation vulnerabilities in Android apps	(Demissie et al. 2020)
2020	EMSE	What do Programmers Discuss about Deep Learning Frameworks	(Han et al. 2020)
2020	IST	A fine-grained requirement traceability evolutionary algorithm: Kromaia a commercial video game case study	(Blasco et al. 2020)
2020	IST	Detecting Java software similarities by using different clustering techniques	(Capiluppi et al. 2020)
2019	ICSE	Investigating The Impact Of Multiple Dependency Structures On Software Defects	(Cui et al. 2019)
2020	ICSE	Taming Behavioral Backward Incompatibilities Via Cross-Project Testing And Analysis	(Chen et al. 2020)
2020	ESEC FSE	Real-time incident prediction for online service systems	(Zhao et al. 2020)
2016	ESEC FSE	Causal impact analysis for app releases in google play	(Martin et al. 2016)

Year	Venue	Title	Reference
2016	ESEM	How Are Discussions Associated with Bug Reworking? An Empirical Study on Open Source Projects	(Zhao et al. 2016)
2011	MSR	Security versus performance bugs: a case study on Firefox	(Zaman et al. 2011)
2014	ESEC FSE	A large scale study of programming languages and code quality in github	(Ray et al. 2014)
2018	ESEM	Automatic topic classification of test cases using text mining at an Android smartphone vendor	(Shimagaki et al. 2018)
2017	ICSE	Can Latent Topics In Source Code Predict Missing Architectural Tactics?	(Gopalakrishnan et al. 2017)
2020	MSR	Challenges in Chatbot Development: A Study of Stack Overflow Posts	(Abdellatif et al. 2020)
2020	ESEM	Challenges in Docker Development: A Large-scale Study Using Stack Overflow	(Haque and Ali Babar 2020)
2014	ICSE	Checking App Behavior Against App Descriptions	(Gorla et al. 2014)
2014	MSR	How does a typical tutorial for mobile development look like?	(Tiarks and Maalej 2014)
2020	MSR	On the Relationship between User Churn and Software Issues	(El Zarif et al. 2020)
2018	ICSE	Online App Review Analysis For Identifying Emerging Issues	(Gao et al. 2018)
2017	ICSE	Recommending and Localizing Change Requests For Mobile Apps Based On User Reviews	(Palomba et al. 2017)
2015	MSR	Recommending posts concerning API issues in developer Q&A sites	(Wang et al. 2015)
2018	ESEC FSE	Winning the app production rally	(Noei et al. 2018)
2015	EMSE	An empirical study on the importance of source code entities for requirements traceability	(Ali et al. 2015)
2009	EMSE	An information retrieval process to aid in the analysis of code clones	(Tairas and Gray 2009)
2018	EMSE	Are tweets useful in the bug fixing process? An empirical study on Firefox and Chrome	(Mezouar et al. 2018)
2014	EMSE	Labeling source code with information retrieval methods: An empirical study	(De Lucia et al. 2014)
2013	TSE	The impact of classifier configuration and classifier combination on bug localization	(Thomas et al. 2013)
2016	ICSE	Autofolding for source code summarization	(Fowkes et al. 2016)
2015	JSS	Enabling improved IR-based feature location	(Binkley et al. 2015)
2014	EMSE	Configuring latent Dirichlet allocation based feature location	(Biggers et al. 2014)
2018	EMSE	Studying the consistency of star ratings and the complaints in 1 & 2-star user reviews for top free cross-platform Android and iOS apps	(Hu et al. 2018)
2016	EMSE	A contextual approach towards more accurate duplicate bug report detection and ranking	(Hindle et al. 2016)

Year	Venue	Title	Reference
2016	ESEC FSE	A large-scale empirical comparison of static and dynamic test case prioritization techniques	(Luo et al. 2016)
2016	IST	EXAF: A search engine for sample applications of object-oriented framework-provided concepts	(Noei and Heydarnoori 2016)
2018	IST	Fragment retrieval on models for model maintenance: Applying a multi-objective perspective to an industrial case study	(Pérez et al. 2018)
2018	ESEM	Improving problem identification via automated log clustering using dimensionality reduction	(Rosenberg and Moonen 2018)
2011	MSR	Retrieval from software libraries for bug localization: a comparative study of generic and composite text models	(Rao and Kak 2011)
2016	IST	The effect of automatic concern mapping strategies on conceptual cohesion measurement	(Silva et al. 2016)
2020	MSR	Traceability Support for Multi-Lingual Software Projects	(Liu et al. 2020)
2009	EMSE	Using information retrieval based coupling measures for impact analysis	(Poshyvanyk et al. 2009)
2011	EMSE	Using structural and textual information to capture feature coupling in object-oriented software	(Revelle et al. 2011)

A.2 Metrics Used in Comparative Studies

The column "Context-specific" indicates if the metric was proposed or adapted to a specific context ("Yes") or is a standard NLP metric ("No").

Metric	Definition	Context-specific	Used in
A measure	Measures difference between two populations (Vargha and Delaney 2000)	No	(Thomas et al. 2014)
Adjusted mutual information (AMI)	Compare two sets of clusters of a clustering technique, e.g., to compare gold standard labeled clusters and the clusters discovered by a technique	No	(Rosenberg and Moonen 2018)
Anomaly score	Defining program behavior as a statistical distribution, this metric represents the distance between the distribution of expected behavior and the actual program behavior (Murali et al. 2017)	Yes	(Murali et al. 2017)
Area Under the Curve (AUC)	Evaluates performance of a scoring classifier using the Receiver Operating Characteristic curve (ROC) which plots recall (true positive rate) against the fraction of false positives out of the negatives (false positive rate) (Kakas et al. 2011)	No	(Fowkes et al. 2016)
Average overlap	Average overlap between labels generated manually and labels automatically generated by the tested topic models (De Lucia et al. 2014)	Yes	(De Lucia et al. 2014)
Average percentage of faults detected (APFD)	Average percentage of faults detected by a prioritized test suite (Rothermel et al. 2001)	Yes	(Thomas et al. 2014)

Metric	Definition	Context-specific	Used in
Completeness	Extent to which all members of a given gold standard label set are assigned to the same cluster (Rosenberg and Moonen 2018)	Yes	(Rosenberg and Moonen 2018)
Homogeneity	Extent to which members of a proposed word cluster come from the same gold standard label set (Rosenberg and Moonen 2018)	Yes	(Rosenberg and Moonen 2018)
Effectiveness	Number of methods that must be investigated before the first method relevant to a feature is located (Poshyvanyk et al. 2007)	Yes	(Biggers et al. 2014; Poshyvanyk et al. 2012)
Effort reduction	Ratio between created clusters and clustered documents (log files) as a measure for the the reduced effort by analyzing clusters of log files rather than individual log files (Rosenberg and Moonen 2018)	Yes	(Rosenberg and Moonen 2018)
Precision	Fraction of documents retrieved that are relevant to the user's information need (total number of documents retrieved that are relevant divided by the total number of documents that are retrieved) (Zeugmann et al. 2011)	No	(Silva et al. 2016; Murali et al. 2017; Cao et al. 2017; Zhang et al. 2016; Demissie et al. 2020; Blasco et al. 2020; Poshyvanyk et al. 2012)
Average Precision	Average precision value for a recalled value (Zhang and Zhang 2009)	No	(Liu et al. 2020)
Mean Average Precision (MAP)	Average of the aggregated average precision (Beitzel et al. 2009)	No	(Abdellatif et al. 2019; Rao and Kak 2011)
Maximum possible precision gain (MPG)	Precision of the best possible scenarios (e.g., in a tree of concepts, the user should navigate the shortest path between the root and the node with the relevant concept) that might be obtained with a technique (Poshyvanyk et al. 2012)	Yes	(Poshyvanyk et al. 2012)
Recall	Fraction of relevant documents that are successfully retrieved (total number of documents retrieved that are relevant divided by the total number of relevant documents in the corpus) (Zeugmann et al. 2011)	No	(Silva et al. 2016; Murali et al. 2017; Cao et al. 2017; Zhang et al. 2016; Demissie et al. 2020; Blasco et al. 2020; Poshyvanyk et al. 2012)
Recall @k	Fraction of relevant documents that are successfully retrieved in top k results (Yan et al. 2016b)	No	(Yan et al. 2016b)

Metric	Definition	Context-specific	Used in
F-measure	Weighted harmonic mean of precision and recall (Brank et al. 2011)	No	(Silva et al. 2016; Cao et al. 2017; Zhang et al. 2016; Blasco et al. 2020)
Mann-Whitney-Wilcoxon test	Non-parametric test of the null hypothesis that, for randomly selected values X and Y from two populations, the probability of X being greater than Y is equal to the probability of Y being greater than X (Mann and Whitney 1947)	No	(Thomas et al. 2014)
Mean Reciprocal Rank (MRR)	Reciprocal rank is calculated using precision @k: given a rank k, precision @k is the precision calculated over the set of retrieved documents with a rank of k. Thus, MRR is the average of the reciprocal rank of a set of queries. The set of queries refer to a list of documents of interest that may be found in the ranked list of retrieved documents) (Craswell 2009)	No	(Binkley et al. 2015; Zhang et al. 2016)
Minimal browsing area (MBA)	Shortest path between root node from a tree of concepts and the node containing the relevant results of a search in such tree (Poshyvanyk et al. 2012)	No	(Poshyvanyk et al. 2012)
Hit ratio	When recommending software functionalities (e.g., features for mobile apps), evaluates how many functionalities can be successfully recommended based on a list of hit functionalities (Hariri et al. 2013)	Yes	(Jiang et al. 2019)
Actual assignee hit ratio	In the context of bug assignment to developers (referred as assignees), evaluates how much the list of recommended assignees contains the actual assignee (Naguib et al. 2013)	Yes	(Naguib et al. 2013)
Top-k hit	In the context of bug assignment to developers (referred as assignees), measures if the ranked list of recommended assignees contains any assignee who has performed either assigning, reviewing, or resolving a bug report (Naguib et al. 2013)	Yes	(Naguib et al. 2013)
Normalized Discounted Cumulative Gain (NDCG)	Quality of Top-k Accuracy ranking (Croft and Metzler 2010)	No	(Jiang et al. 2019; Chen et al. 2014)
SCORE	Ranking-based metric that calculates the proportion of bugs versus the proportion of the code that must be examined for the localization of the bugs (Jones and Harrold 2005)	Yes	(Rao and Kak 2011)
Perplexity	Measure of performance for statistical models of natural language, which indicates the uncertainty in predicting a single word (Blei et al. 2003b)	No	(Yan et al. 2016b)

Metric	Definition	Context-specific	Used in
Purity	Extent to which clusters (from a clustering technique) contain a single label (Manning et al. 2008)	No	(Cao et al. 2017)
Term Entropy	Measure of uncertainty associated with a random variable (Shannon 1948). Studies calculated entropy for distribution of terms in documents. A document with lower entropy indicates that it has few dominant terms, while a document with higher entropy presents more dominant terms	No	(De Lucia et al. 2014; Cao et al. 2017)
Top-k Accuracy	Percentage of bug reports in which at least one relevant source code entity was returned in the top k results (e.g., a top-10 accuracy value of 0.15 indicates that for 15% of the bug reports at least one relevant source code entity was returned in the top 10 results) (Nguyen et al. 2011)	No	(Thomas et al. 2013; Tantithamthavorn et al. 2018; Abdellatif et al. 2019; Xia et al. 2017b)

Acknowledgements We would like to thank the editor and the anonymous reviewers for their insightful and detailed feedback that helped us to significantly improve the manuscript.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

Abdellatif A, Costa D, Badran K, Abdalkareem R, Shihab E (2020) Challenges in Chatbot Development: A Study of Stack Overflow Posts. In: Proceedings of the 17th international conference on mining software repositories, vol 12. IEEE/ACM, Seoul, pp 174-185. https://doi.org/10.1145/3379597.3387472

Abdellatif TM, Capretz LF, Ho D (2019) Automatic recall of software lessons learned for software project managers. Inf Softw Technol 115:44–57. https://doi.org/10.1016/j.infsof.2019.07.006

Aggarwal CC, Zhai C (2012) Mining text data. Springer, New York. https://doi.org/10.1007/978-1-4614-3223-4

Agrawal A, Fu W, Menzies T (2018) What is wrong with topic modeling? And how to fix it using search-based software engineering. Inf Softw Technol 98(January 2017):74–88. https://doi.org/10.1016/j.infsof.2018.02.005

Ahasanuzzaman M, Asaduzzaman M, Roy CK, Schneider KA (2019) CAPS: a supervised technique for classifying Stack Overflow posts concerning API issues. Empir Softw Eng 25:1493–1532. https://doi.org/10.1007/s10664-019-09743-4

Ahmed S, Bagherzadeh M (2018) What do concurrency developers ask about?: A large-scale study using Stack Overflow. In: Proceedings of the international symposium on empirical software engineering and measurement. ACM, Oulu, pp 1-10. https://doi.org/10.1145/3239235.3239524

Ali N, Sharafi Z, Guéhéneuc YG, Antoniol G (2015) An empirical study on the importance of source code entities for requirements traceability. Empir Softw Eng 20(2):442–478. https://doi.org/10.1007/s10664-014-9315-y

Alipour A, Hindle A, Stroulia E (2013) A contextual approach towards more accurate duplicate bug report detection. In: IEEE international working conference on mining software repositories. pp 183–192. https://doi.org/10.1109/MSR.2013.662402

Altarawy D, Shahin H, Mohammed A, Meng N (2018) LASCAD: Language-agnostic software categorization and similar application detection. J Syst Softw 142:21–34. https://doi.org/10.1016/j.jss.2018.04.018

ARC ARC (2012) Excellence in research for australia (ERA). https://www.arc.gov.au/excellence-research-australia http://www.arc.gov.au/pdf/era12/ERAFactsheet_Jan2012_1.pdf

Asuncion HU, Asuncion AU, Taylor RN (2010) Software traceability with topic modeling. In: Proceedings of the international conference on software engineering. IEEE/ACM, Cape Town, pp 95-104

Bagherzadeh M, Khatchadourian R (2019) Going big: a large-scale study on what big data developers ask. In: Proceedings of the 27th joint european software engineering conference and symposium on the foundations of software engineering. ACM, Tallinn, pp 432-442. https://doi.org/10.1145/3338906.3338939

Bajaj K, Pattabiraman K, Mesbah A (2014) Mining questions asked by web developers. In: Proceedings of the 11th working conference on mining software repositories. ACM, Hyderabad, pp 112–121. https://doi.org/10.1145/2597073.2597083

Bajracharya S, Lopes C (2009) Mining search topics from a code search engine usage log. In: Proceedings of the 6th international working conference on mining software repositories. IEEE, Vancouver, pp 111-120. https://doi.org/10.1109/MSR.2009.5069489

Bajracharya SK, Lopes CV (2012) Analyzing and mining a code search engine usage log. Empir Softw Eng 17:424–466. https://doi.org/10.1007/s10664-010-9144-6

Barua A, Thomas SW, Hassan AE (2014) What are developers talking about? An analysis of topics and trends in Stack Overflow. Empir Softw Eng 19(3):619–654. https://doi.org/10.1007/s10664-012-9231-y

Bavota G, Gethers M, Oliveto R, Poshyvanyk D, Lucia ADE (2014a) Improving software modularization via automated analysis of latent. ACM Trans Softw Eng Methodol 23(1):1–33. https://doi.org/10.1145/2559935

Bavota G, Oliveto R, Gethers M, Poshyvanyk D, De Lucia A (2014b) Methodbook: Recommending move method refactorings via relational topic models. IEEE Trans Softw Eng 40(7):671–694. https://doi.org/10.1109/TSE.2013.60

Beitzel SM, Jensen EC, Frieder O (2009) MAP. In: Encyclopedia of database systems. Springer US, Boston, pp 1691–1692. https://doi.org/10.1007/978-0-387-39940-9_492

Belle AB, Boussaidi GE, Kpodjedo S (2016) Combining lexical and structural information to reconstruct software layers. Inf Softw Technol 74:1–16. https://doi.org/10.1016/j.infsof.2016.01.008

Bi T, Liang P, Tang A, Yang C (2018) A systematic mapping study on text analysis techniques in software architecture. J Syst Softw 144:533–558. https://doi.org/10.1016/j.jss.2018.07.055

Biggers LR, Bocovich C, Capshaw R, Eddy BP, Etzkorn LH, Kraft NA (2014) Configuring latent Dirichlet allocation based feature location. Empir Softw Eng 19(3):465–500. https://doi.org/10.1007/s10664-012-9224-x

Binkley D, Lawrie D, Uehlinger C, Heinz D (2015) Enabling improved IR-based feature location. J Syst Softw 101:30–42. https://doi.org/10.1016/j.jss.2014.11.013

Blasco D, Cetina C, Pastor O (2020) A fine-grained requirement traceability evolutionary algorithm: Kromaia, a commercial video game case study. Inf Softw Technol 119:1–12. https://doi.org/10.1016/j.infsof.2019.106235

Blei DM, Jordan MI, Griffiths TL, Tenenbaum JB (2003a) Hierarchical topic models and the nested chinese restaurant process. In: Proceedings of the 16th international conference on neural information processing systems. Neural Information Processing Systems Foundation, Vancouver, pp 17-24

Blei DM, Ng AY, Jordan MI (2003b) Latent Dirichlet allocation. J Mach Learn Res 3:993–1022. https://doi.org/10.1162/jmlr.2003.3.4-5.993

Brank J, Mladenić D, Grobelnik M, Liu H, Mladenić D, Flach PA, Garriga GC, Toivonen H, Toivonen H (2011) F 1-measure. In: Encyclopedia of machine learning. Springer US, pp 397–397. https://doi.org/10.1007/978-0-387-30164-8_298

Canfora G, Cerulo L, Cimitile M, Di Penta M (2014) How changes affect software entropy: An empirical study. Empir Softw Eng 19:1–38. https://doi.org/10.1007/s10664-012-9214-z

Cao B, Frank Liu X, Liu J, Tang M (2017) Domain-aware Mashup service clustering based on LDA topic model from multiple data sources. Inf Softw Technol 90:40–54. https://doi.org/10.1016/j.infsof.2017.05.001

Capiluppi A, Ruscio DD, Rocco JD, Nguyen PT, Ajienka N (2020) Detecting Java software similarities by using different clustering techniques. Inf Softw Technol 122. https://doi.org/10.1016/j.infsof.2020.106279

Catolino G, Palomba F, Zaidman A, Ferrucci F (2019) Not all bugs are the same: Understanding, characterizing, and classifying bug types. J Syst Softw 152:165–181. https://doi.org/10.1016/j.jss.2019.03.002

Chang J, Blei DM (2009) Relational topic models for document networks. In: Proceedings of the 12th international conference on artificial intelligence and statistics. Society for Artificial Intelligence and Statistics, Clearwater Beach, pp 81-88

Chang J, Blei DM (2010) Hierarchical relational models for document networks. Ann Appl Stat 4(1):124–150. https://doi.org/10.1214/09-AOAS309

Chang J, Boyd-Graber J, Gerrish S, Wang C, Blei DM (2009) Reading tea leaves: How humans interpret topic models. In: Proceedings of the 2009 conference advances in neural information. Neural Information Processing Systems Foundation, Vancouver, pp 288-296

Chatterjee P, Damevski K, Pollock L (2019) Exploratory study of slack q&a chats as a mining source for software engineering tools. In: Proceedings of the 16th international conference on mining software repositories. IEEE, Montreal, pp 1-12

Chen H, Coogle J, Damevski K (2019) Modeling stack overflow tags and topics as a hierarchy of concepts. J Syst Softw 156:283–299. https://doi.org/10.1016/j.jss.2019.07.033

Chen L, Hassan F, Wang X, Zhang L (2020) Taming behavioral backward incompatibilities via cross-project testing and analysis. In: Proceedings of the 42nd international conference on software engineering. IEEE/ACM, Seoul, pp 112-124. https://doi.org/10.1145/3377811.3380436

Chen N, Lin J, Hoi SC, Xiao X, Zhang B (2014) AR-miner: Mining informative reviews for developers from mobile app marketplace. In: Proceedings of the international conference on software engineering, vol 1. IEEE/ACM, Hyderabad, pp 767-778. https://doi.org/10.1145/2568225.2568263

Chen TH, Thomas SW, Nagappan M, Hassan AE (2012) Explaining software defects using topic models. In: Proceedings of the international working conference on mining software repositories. IEEE, Zurich, pp 189-198. https://doi.org/10.1109/MSR.2012.6224280

Chen TH, Thomas SW, Hassan AE (2016) A survey on the use of topic models when mining software repositories. Empir Softw Eng 21(5):1843–1919. https://doi.org/10.1007/s10664-015-9402-8

Chen TH, Shang W, Nagappan M, Hassan AE, Thomas SW (2017) Topic-based software defect explanation. J Syst Softw 129:79–106. https://doi.org/10.1016/j.jss.2016.05.015

Choetkiertikul M, Dam HK, Tran T, Ghose A (2017) Predicting the delay of issues with due dates in software projects. Empir Softw Eng 22:1223–1263. https://doi.org/10.1007/s10664-016-9496-7

Craswell N (2009) Mean reciprocal rank. In: Encyclopedia of database systems. Springer US, pp 1703–1703. https://doi.org/10.1007/978-0-387-39940-9_488

Croft WB, Metzler D (2010) Search engines: Information retrieval in practice. Addison-Wesley, Reading

Cui D, Liu T, Cai Y, Zheng Q, Feng Q, Jin W, Guo J, Qu Y (2019) Investigating the impact of multiple dependency structures on software defects, IEEE/ACM, Montreal. https://doi.org/10.1109/ICSE.2019.00069

Damevski K, Chen H, Shepherd DC, Kraft NA, Pollock L (2018) Predicting future developer behavior in the IDE using topic models. IEEE Trans Softw Eng 44(11):1100–1111. https://doi.org/10.1109/TSE.2017.2748134

De Lucia A, Di Penta M, Oliveto R, Panichella A, Panichella S (2014) Labeling source code with information retrieval methods: An empirical study. Empir Softw Eng 19(5):1383–1420. https://doi.org/10.1007/s10664-013-9285-5

Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391–407. https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

Demissie BF, Ceccato M, Shar LK (2020) Security analysis of permission re-delegation vulnerabilities in Android apps. Empir Softw Eng 25:5084–5136. https://doi.org/10.1007/s10664-020-09879-8

Dietz L, Bickel S, Scheffer T (2007) Unsupervised prediction of citation influences. In: Proceedings of the 24th international conference on machine learning. ACM, Corvallis, pp 233-240. https://doi.org/10.1145/1273496.1273526

Dit B, Revelle M, Poshyvanyk D (2013) Integrating information retrieval, execution and link analysis algorithms to improve feature location in software. Empir Softw Eng 18(2):277–309. https://doi.org/10.1007/s10664-011-9194-4

El Zarif O, Da Costa DA, Hassan S, Zou Y (2020) On the relationship between user churn and software issues. In: Proceedings of the 17th international conference on mining software repositories. ACM, New York, pp 339-349. https://doi.org/10.1145/3379597.3387456

Fowkes J, Chanthirasegaran P, Ranca R, Allamanis M, Lapata M, Sutton C (2016) Autofolding for source code summarization. Proc Int Conf Softw Eng 43(12):649–652. https://doi.org/10.1145/2889160.2889171

Fu Y, Yan M, Zhang X, Xu L, Yang D, Kymer JD (2015) Automated classification of software change messages by semi-supervised Latent Dirichlet Allocation. Inf Softw Technol 57:369–377. https://doi.org/10.1016/j.infsof.2014.05.017

Galvis Carreno LV, Winbladh K (2012) Analysis of user comments: an approach for software requirements evolution. In: Proceedings of the international conference on software engineering. IEEE/ACM, San Francisco, pp 582-591

Gao C, Zeng J, Lyu MR, King I (2018) Online app review analysis for identifying emerging issues. In: Proceedings of the 40th international conference on software engineering. IEEE/ACM, Gothenburg, pp 48-58. https://doi.org/10.1145/3180155.3180218

Gopalakrishnan R, Sharma P, Mirakhorli M, Galster M (2017) Can latent topics in source code predict missing architectural tactics? In: Proceedings of the 39th international conference on software engineering, IEEE/ACM, pp 15–26. https://doi.org/10.1109/ICSE.2017.10. http://ghtorrent.org/

Gorla A, Tavecchia I, Gross F, Zeller A (2014) Checking app behavior against app descriptions. In: Proceedings of the international conference on software engineering. IEEE/ACM, Hyderabad, pp 1025-1035. https://doi.org/10.1145/2568225.2568276

Griffiths TL, Steyvers M (2004) Finding scientific topics. In: Proceedings of the national academy of sciences, vol 101. Neural Information Processing Systems Foundation, Irvine, pp 5228-5235. https://doi.org/10.1073/pnas.0307752101

Haghighi A, Vanderwende L (2009) Exploring content models for multi-document summarization. In: Proceedings of the conference on human language technologies: the 2009 annual conference of the north american chapter of the association for computational linguistics. Association for Computational Linguistics, Boulder, pp 362–370. https://doi.org/10.3115/1620754.1620807, http://www-nlpir.nist.gov/projects/duc/data.html

Han J, Shihab E, Wan Z, Deng S, Xia X (2020) What do programmers discuss about deep learning frameworks. Empir Softw Eng 25:2694–2747. https://doi.org/10.1007/s10664-020-09819-6

Haque MU, Ali Babar M (2020) Challenges in docker development: a large-scale study using stack overflow. In: Proceedings of the 14th international symposium on empirical software engineering and measurement. IEEE/ACM, Bari, pp 1-11. https://doi.org/10.1145/3382494.3410693

Hariri N, Castro-Herrera C, Mirakhorli M, Cleland-Huang J, Mobasher B (2013) Supporting domain analysis through mining and recommending features from online product listings. IEEE Trans Softw Eng 39(12):1736–1752. https://doi.org/10.1109/TSE.2013.39

Henß S, Monperrus M, Mezini M (2012) Semi-automatically extracting FAQs to improve accessibility of software development knowledge. In: Proceedings of the international conference on software engineering. IEEE/ACM, Zurich, pp 793-803. https://doi.org/10.1109/ICSE.2012.6227139

Hindle A, Godfrey MW, Ernst NA, Mylopoulos J (2011) Automated topic naming to support cross-project analysis of software maintenance activities. In: Proceedings of the 33rd international conference on software engineering. ACM, Waikiki, pp 163-172

Hindle A, Ernst NA, Godfrey MW, Mylopoulos J (2013) Automated topic naming: Supporting cross-project analysis of software maintenance activities. Empir Softw Eng 18(6):1125–1155. https://doi.org/10.1007/s10664-012-9209-9

Hindle A, Bird C, Zimmermann T, Nagappan N (2015) Do topics make sense to managers and developers? Empir Softw Eng 20:479–515. https://doi.org/10.1007/s10664-014-9312-1

Hindle A, Alipour A, Stroulia E (2016) A contextual approach towards more accurate duplicate bug report detection and ranking. Empir Softw Eng 21(2):368–410. https://doi.org/10.1007/s10664-015-9387-3

Hoffman M, Blei D, Bach F (2010) Online learning for latent dirichlet allocation. In: Proceedings of the neural information processing systems conference. Neural Information Processing Systems Foundation, Vancouver, pp 1-9. https://doi.org/10.1.1.187.1883

Hofmann T (1999) Probabilistic latent semantic indexing. In: Proceedings of the 22nd annual international conference on research and development in information retrieval. ACM, Berkeley, pp 50-57

Hu H, Bezemer CP, Hassan AE (2018) Studying the consistency of star ratings and the complaints in 1 & 2-star user reviews for top free cross-platform Android and iOS apps. Empir Softw Eng 23(6):3442–3475. https://doi.org/10.1007/s10664-018-9604-y

Hu H, Wang S, Bezemer CP, Hassan AE (2019) Studying the consistency of star ratings and reviews of popular free hybrid Android and iOS apps. Empir Softw Eng 24:7–32. https://doi.org/10.1007/s10664-018-9617-6

Hu W, Wong K (2013) Using citation influence to predict software defects. In: Proceedings of the international working conference on mining software repositories. IEEE, San Francisco, pp 419-428. https://doi.org/10.1109/MSR.2013.6624058

Jiang H, Zhang J, Ren Z, Zhang T (2017) An unsupervised approach for discovering relevant tutorial fragments for APIs. In: Proceedings of the 39th international conference on software engineering. IEEE/ACM, Buenos Aires, pp 38-48. https://doi.org/10.1109/ICSE.2017.12

Jiang HE, Zhang J, Li X, Ren Z, Lo D, Wu X, Luo Z (2019) Recommending new features from mobile app descriptions. ACM Trans Softw Eng Methodol 28(4):1–29. https://doi.org/10.1145/3344158

Jipeng Q, Zhenyu Q, Yun L, Yunhao Y, Xindong W (2020) Short text topic modeling techniques, applications, and performance: a survey. https://doi.org/10.1109/TKDE.2020.2992485

Jo Y, Oh A (2011) Aspect and sentiment unification model for online review analysis. In: Proceedings of the fourth ACM international conference on Web search and data mining. ACM, New York, pp 815-824. https://doi.org/10.1145/1935826

Jones JA, Harrold MJ (2005) Empirical evaluation of the tarantula automatic fault-localization technique. In: Proceedings of the 20th international conference on automated software engineering. IEEE/ACM, New York, pp 273–282. https://doi.org/10.1145/1101908.1101949, http://portal.acm.org/citation.cfm?doid=1101908.1101949

Kakas AC, Cohn D, Dasgupta S, Barto AG, Carpenter GA, Grossberg S, Webb GI, Dorigo M, Birattari M, Toivonen H, Timmis J, Branke J, Toivonen H, Strehl AL, Drummond C, Coates A, Abbeel P, Ng AY, Zheng F, Webb GI, Tadepalli P (2011) Area under curve. In: Encyclopedia of machine learning. Springer US, pp 40–40. https://doi.org/10.1007/978-0-387-30164-8_28

Kitchenham BA (2004) Procedures for performing systematic reviews. Keele, UK, Keele University 33(TR/SE-0401):28. https://doi.org/10.1.1.122.3308

Layman L, Nikora AP, Meek J, Menzies T (2016) Topic modeling of NASA space system problem reports research in practice. In: Proceedings of the 13th working conference on mining software repositories. ACM, Austin, pp 303-314. https://doi.org/10.1145/2901739.2901760

Le TDB, Thung F, Lo D (2017) Will this localization tool be effective for this bug? Mitigating the impact of unreliability of information retrieval based bug localization tools. Empir Softw Eng 22:2237–2279. https://doi.org/10.1007/s10664-016-9484-y

Leach RJ (2016) Introduction to software engineering, 2nd edn. CRC Press LLC, Boca Raton. https://ebookcentral.proquest.com/lib/canterbury/detail.action?docID=4711469&query=Software+Engineering

Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401(6755):788–791

Li H, Chen THP, Shang W, Hassan AE (2018) Studying software logging using topic models. Empir Softw Eng 23:2655–2694. https://doi.org/10.1007/s10664-018-9595-8

Lian X, Liu W, Zhang L (2020) Assisting engineers extracting requirements on components from domain documents. Inf Softw Technol 118(September 2019):106196. https://doi.org/10.1016/j.infsof.2019.106196

Lin T, Tian W, Mei Q, Cheng H (2014) The dual-sparse topic model: Mining focused topics and focused terms in short text. In: Proceedings of the 23rd international conference on world wide web. ACM, Seoul, pp 539-549. https://doi.org/10.1145/2566486.2567980

Liu Y, Liu L, Liu H, Wang X, Yang H (2017) Mining domain knowledge from app descriptions. J Syst Softw 133:126–144. https://doi.org/10.1016/j.jss.2017.08.024

Liu Y, Lin J, Cleland-Huang J (2020) Traceability support for multi-lingual software projects. In: Proceedings of the 17th international conference on mining software repositories. ACM, Seoul, pp 443-454. https://doi.org/10.1145/3379597.3387440

Lukins SK, Kraft NA, Etzkorn LH (2010) Bug localization using latent Dirichlet allocation. Inf Softw Technol 52:972–990. https://doi.org/10.1016/j.infsof.2010.04.002

Luo Q, Moran K, Poshyvanyk D (2016) A large-scale empirical comparison of static and dynamic test case prioritization techniques. In: Proceedings of the 24th international symposium on foundations of software engineering. ACM, Seattle, pp 559-570. https://doi.org/10.1145/2950290.2950344

Mahmoud A, Bradshaw G (2017) Semantic topic models for source code analysis. Empir Softw Eng 22(4):1965–2000. https://doi.org/10.1007/s10664-016-9473-1

Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. Ann Math Stat 18(1):50–60. https://doi.org/10.1214/aoms/1177730491, http://projecteuclid.org/euclid.aoms/1177730491

Manning CD, Raghavan P, Schütze H (2008) Evaluation of Clustering. In: Introduction to information retrieval. Cambridge University Press. chap 16, https://doi.org/10.33899/csmj.2008.163987. https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html, http://nlp.stanford.edu/IR?book/html/htmledition/evaluation?of?clustering?1.htmlwhereisthesetofclustersan

Mantyla MV, Claes M, Farooq U (2018) Measuring LDA topic stability from clusters of replicated runs, ACM, Oulu. https://doi.org/10.1145/3239235.3267435

Martin W, Harman M, Jia Y, Sarro F, Zhang Y (2015) The app sampling problem for app store mining. In: Proceedings of the 12th international working conference on mining software repositories. IEEE, Florence, pp 123-133. https://doi.org/10.1109/MSR.2015.19

Martin W, Sarro F, Harman M (2016) Causal impact analysis for app releases in google play. In: Proceedings of the 24th international symposium on foundations of software engineering. ACM, Seattle, pp 435-446. https://doi.org/10.1145/2950290.2950320

McIlroy S, Ali N, Khalid H, E Hassan A (2016) Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. Empir Softw Eng 21:1067–1106. https://doi.org/10.1007/s10664-015-9375-7

Mehrotra R, Sanner S, Buntine W, Xie L (2013) Improving LDA Topic Models for Microblogs via Tweet Pooling and Automatic Labeling. In: Proceedings of the 36th International Conference on Research and Development in Information Retrieval. ACM, Dublin, pp 889-892

Mezouar ME, Zhang F, Zou Y (2018) Are tweets useful in the bug fixing process? An empirical study on Firefox and Chrome. Empir Softw Eng 23(3):1704–1742. https://doi.org/10.1007/s10664-017-9559-4

Miner G, Elder J, Fast A, Hill T, Nisbet R, Delen D (2012) Practical text mining and statistical analysis for non-structured text data applications. Elsevier Science & Technology, Waltham. https://doi.org/10.1016/C2010-0-66188-8

Moslehi P, Adams B, Rilling J (2016) On mining crowd-based speech documentation. In: Proceedings of the 13th working conference on mining software repositories. ACM, Austin, pp 259-268. https://doi.org/10.1145/2901739.2901771

Moslehi P, Adams B, Rilling J (2018) Feature location using crowd-based screencasts. In: Proceedings of the 15th international conference on mining software repositories. ACM, New York, pp 192-202. https://doi.org/10.1145/3196398.3196439

Moslehi P, Adams B, Rilling J (2020) A feature location approach for mapping application features extracted from crowd-based screencasts to source code. Empir Softw Eng 25:4873–4926. https://doi.org/10.1007/s10664-020-09874-z

Murali V, Chaudhuri S, Jermaine C (2017) Bayesian specification learning for finding API usage errors. In: Proceedings of the Joint european software engineering conference and symposium on the foundations of software engineering. ACM, Paderborn, pp 151-162. https://doi.org/10.1145/3106237.3106284

Nabli H, Ben Djemaa R, Ben Amor IA (2018) Efficient cloud service discovery approach based on LDA topic modeling. J Syst Softw 146:233–248. https://doi.org/10.1016/j.jss.2018.09.069

Naguib H, Narayan N, Brügge B, Helal D (2013) Bug report assignee recommendation using activity profiles. In: Proceedings of the international working conference on mining software repositories. IEEE, San Francisco, pp 22-30. https://doi.org/10.1109/MSR.2013.6623999

Nayebi M, Cho H, Ruhe G (2018) App store mining is not enough for app improvement. Empir Softw Eng 23:2764–2794. https://doi.org/10.1007/s10664-018-9601-1

Nguyen AT, Nguyen TT, Al-Kofahi J, Nguyen HV, Nguyen TN (2011) A topic-based approach for narrowing the search space of buggy files from a bug report. In: Proceedings of the 26th international conference on automated software engineering. IEEE/ACM, Lawrence, pp 263–272. https://doi.org/10.1109/ASE.2011.6100062

Nguyen AT, Nguyen TT, Nguyen TN, Lo D, Sun C (2012) Duplicate bug report detection with a combination of information retrieval and topic modeling. In: Proceedings of the 27th international conference on automated software engineering. IEEE/ACM, Essen, pp 70–79. https://doi.org/10.1145/2351676.2351687

Nguyen VA, Boyd-Graber J, Resnik P, Chang J, Graber JB (2014) Learning a concept hierarchy from multilabeled documents. In: Proceedings of the neural information processing systems conference. Neural Information Processing Systems Foundation, Montreal, pp 1-9

Noei E, Heydarnoori A (2016) EXAF: A search engine for sample applications of object-oriented framework-provided concepts. Inf Softw Technol 75:135–147. https://doi.org/10.1016/j.infsof.2016.03.007

Noei E, Da Costa DA, Zou Y (2018) Winning the app production rally. In: Proceedings of the 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering. ACM, Lake Buena Vista, pp 283-294. https://doi.org/10.1145/3236024.3236044

Noei E, Zhang F, Wang S, Zou Y (2019) Towards prioritizing user-related issue reports of mobile applications. Empir Softw Eng 24:1964–1996. https://doi.org/10.1007/s10664-019-09684-y

Pagano D, Maalej W (2013) How do open source communities blog? Empir Softw Eng 18(6):1090–1124. https://doi.org/10.1007/s10664-012-9211-2

Palomba F, Salza P, Ciurumelea A, Panichella S, Gall H, Ferrucci F, De Lucia A (2017) Recommending and localizing change requests for mobile apps based on user reviews. In: Proceedings of the 39th international conference on software engineering. IEEE/ACM, Buenos Aires, pp 106-117. https://doi.org/10.1109/ICSE.2017.18

Panichella A, Dit B, Oliveto R, Di Penta M, Poshynanyk D, De Lucia A (2013) How to effectively use topic models for software engineering tasks? An approach based on Genetic Algorithms. In: Proceedings of the international conference on software engineering. IEEE/ACM, San Francisco, pp 522-531. https://doi.org/10.1109/ICSE.2013.6606598

Pérez F, Lapeṅa R, Font J, Cetina C (2018) Fragment retrieval on models for model maintenance: Applying a multi-objective perspective to an industrial case study. Inf Softw Technol 103:188–201. https://doi.org/10.1016/j.infsof.2018.06.017

Petersen K, Vakkalanka S, Kuzniarz L (2015) Guidelines for conducting systematic mapping studies in software engineering: An update. Inf Softw Technol 64(1):1–18. https://doi.org/10.1016/j.infsof.2015.03.007

Pettinato M, Gil JP, Galeas P, Russo B (2019) Log mining to re-construct system behavior: An exploratory study on a large telescope system. Inf Softw Technol 114:121–136. https://doi.org/10.1016/j.infsof.2019.06.011

Poshyvanyk D, Gueheneuc YG, Marcus A, Antoniol G, Rajlich V (2007) Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval, vol 33, pp 420–431. https://doi.org/10.1109/TSE.2007.1016. https://www.researchgate.net/publication/3189749

Poshyvanyk D, Marcus A, Ferenc R, Gyimóthy T (2009) Using information retrieval based coupling measures for impact analysis. Empir Softw Eng 14(1):5–32. https://doi.org/10.1007/s10664-008-9088-2, http://www.mozilla.org/

Poshyvanyk D, Gethers M, Marcus A (2012) Concept location using formal concept analysis and information retrieval. ACM Trans Softw Eng Methodol 21(4):1–34. https://doi.org/10.1145/2377656.2377660

Poursabzi-Sangdeh F, Goldstein DG, Hofman JM, Vaughan JW, Wallach H (2021) Manipulating and measuring model interpretability. In: Proceedings of the conference on human factors in computing systems. ACM, Yokohama. https://doi.org/10.1145/3411764.3445315

Ramage D, Hall D, Nallapati R, Manning CD (2009) Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In: Proceedings of the conference on empirical methods in natural language processing. ACL/AFNLP, Singapore, pp 248-256. https://doi.org/10.5555/1699510.1699543

Rao S, Kak A (2011) Retrieval from software libraries for bug localization: A comparative study of generic and composite text models. In: Proceedings of the international conference on software engineering. IEEE/ACM, Waikiki, pp 43-52. https://doi.org/10.1145/1985441.1985451

Ray B, Posnett D, Filkov V, Devanbu P (2014) A large scale study of programming languages and code quality in GitHub. In: Proceedings of the symposium on the foundations of software engineering, pp 155–165. https://doi.org/10.1145/2635868.2635922

Revelle M, Gethers M, Poshyvanyk D (2011) Using structural and textual information to capture feature coupling in object-oriented software. Empir Softw Eng 16(6):773–811. https://doi.org/10.1007/s10664-011-9159-7

Röder M, Both A, Hinneburg A (2015) Exploring the space of topic coherence measures. In: Proceedings of the eighth ACM international conference on web search and data mining - WSDM '15. ACM, Shanghai, pp 399-408. https://doi.org/10.1145/2684822.2685324

Rosen C, Shihab E (2016) What are mobile developers asking about? A large scale study using Stack Overflow. Empir Softw Eng 21:1192–1223. https://doi.org/10.1007/s10664-015-9379-3

Rosenberg CM, Moonen L (2018) Improving problem identification via automated log clustering using dimensionality reduction. In: Proceedings of the international symposium on empirical software engineering and measurement. ACM, Oulu, pp 1-10. https://doi.org/10.1145/3239235.3239248

Rothermel G, Untcn RH, Chu C, Harrold MJ (2001) Prioritizing test cases for regression testing. IEEE Trans Softw Eng 27(10):929–948. https://doi.org/10.1109/32.962562

Salton G, Wong A, Yang CS (1975) A vector space model for automatic indexing. Commun ACM 18(11):613–620. https://doi.org/10.1145/361219.361220

Savage T, Dit B, Gethers M, Poshyvanyk D (2010) TopicXP: exploring topics in source code using latent Dirichlet allocation. IEEE, Timisoara. https://doi.org/10.1109/ICSM.2010.5609654

Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27(3):379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Shimagaki J, Kamei Y, Ubayashi N, Hindle A (2018) Automatic topic classification of test cases using text mining at an android smartphone vendor. In: Proceedings of the 12th international symposium on empirical software engineering and measurement. IEEE/ACM, Oulu, pp 1-10. https://doi.org/10.1145/3239235.3268927

Silva B, Sant'anna C, Rocha N, Chavez C (2016) The effect of automatic concern mapping strategies on conceptual cohesion measurement. Inf Softw Technol 75:56–70. https://doi.org/10.1016/j.infsof.2016.03.006

Silva LL, Valente MT, Maia MA (2019) Co-change patterns: A large scale empirical study. J Syst Softw 152:196–214. https://doi.org/10.1016/j.jss.2019.03.014

Soliman M, Galster M, Salama AR, Riebisch M (2016) Architectural knowledge for technology decisions in developer communities: An exploratory study with Stack Overflow. In: Proceedings of the 13th working conference on software architecture. IEEE, Venice, pp 128-133. https://doi.org/10.1109/WICSA.2016.13

Somasundaram K, Murphy GC (2012) Automatic categorization of bug reports using latent Dirichlet allocation. In: Proceedings of the 5th India software engineering conference, vol 12. ACM, pp 125-130. https://doi.org/10.1145/2134254.2134276

Souza LB, Campos EC, Madeiral F, Paixão K, Rocha AM, Maia MdA (2019) Bootstrapping cookbooks for APIs from crowd knowledge on Stack Overflow. Inf Softw Technol 111(March 2018):37–49. https://doi.org/10.1016/j.infsof.2019.03.009

Steyvers M, Griffiths T (2010) Probalistic Topic Models. In: Landauer T, McNamara D, Dennis S, Kintsch W (eds) Latent semantic analysis: a road to meaning. University of California, Irvine, pp 993-1022. https://doi.org/10.1016/s0364-0213(01)00040-4

Sun X, Li B, Leung H, Li B, Li Y (2015) MSR4SM: Using topic models to effectively mining software repositories for software maintenance tasks. Inf Softw Technol 66:1–12. https://doi.org/10.1016/j.infsof.2015.05.003

Sun X, Liu X, Li B, Duan Y, Yang H, Hu J (2016) Exploring topic models in software engineering data analysis: A survey, IEEE, Shangai. https://doi.org/10.1109/SNPD.2016.7515925

Sun X, Yang H, Xia X, Li B (2017) Enhancing developer recommendation with supplementary information via mining historical commits. J Syst Softw 134:355–368. https://doi.org/10.1016/j.jss.2017.09.021

Taba SES, Keivanloo I, Zou Y, Wang S (2017) An exploratory study on the usage of common interface elements in android applications. J Syst Softw 131:491–504. https://doi.org/10.1016/j.jss.2016.07.010

Tairas R, Gray J (2009) An information retrieval process to aid in the analysis of code clones, vol 14, pp 33–56. https://doi.org/10.1007/s10664-008-9089-1, http://www.cis.uab.edu/tairasr/clones/literature

Tamrawi A, Nguyen TT, Al-Kofahi JM, Nguyen TN (2011) Fuzzy set and cache-based approach for bug triaging. In: Proceedings of the 19th ACM symposium on foundations of software engineering. ACM, pp 365–375. https://doi.org/10.1145/2025113.202516

Tang J, Zhang M, Mei Q (2013) One theme in all views: modeling consensus topics in multiple contexts. In: Proceedings of the 19th international conference on knowledge discovery and data mining. ACM, New York, pp 5–13

Tantithamthavorn C, Lemma Abebe S, Hassan AE, Ihara A, Matsumoto K (2018) The impact of IR-based classifier configuration on the performance and the effort of method-level bug localization. Inf Softw Technol 102(June):160–174. https://doi.org/10.1016/j.infsof.2018.06.001

Teh YW, Jordan MI, Beal MJ, Blei DM (2006) Hierarchical Dirichlet processes. J Am Stat Assoc 101(476):1566–1581. https://doi.org/10.1198/016214506000000302

Thomas SW, Nagappan M, Blostein D, Hassan AE (2013) The impact of classifier configuration and classifier combination on bug localization. IEEE Trans Softw Eng 39(10):1427–1443. https://doi.org/10.1109/TSE.2013.27

Thomas SW, Hemmati H, Hassan AE, Blostein D (2014) Static test case prioritization using topic models. Empir Softw Eng 19:182–212. https://doi.org/10.1007/s10664-012-9219-7

Tiarks R, Maalej W (2014) How does a typical tutorial for mobile development look like? In: Proceedings of the 11th international conference on mining software repositories. IEEE/ACM, Hyderabad, pp 272-281. https://doi.org/10.1145/2597073.2597106

Treude C, Wagner M (2019) Predicting good configurations for GitHub and stack overflow topic models. In: Proceedings of the 16th international conference on mining software repositories. IEEE, Montreal, pp 84-95. https://doi.org/10.1109/MSR.2019.00022

Vargha A, Delaney HD (2000) A critique and improvement of the CL common language effect size statistics of McGraw and Wong. J Educ Behav Stat 25(2):101–132. https://doi.org/10.3102/10769986025002101

Wallach HM, Mimno D, McCallum A (2009) Rethinking LDA: Why priors matter. In: Proceedings of the conference on advances in neural information processing systems. Curran Associates Inc., Vancouver, pp 1973–1981. http://rexa.info/

Wang C, Blei DM (2011) Collaborative topic modeling for recommending scientific articles. In: Proceedings of the international conference on knowledge discovery and data mining. ACM, New York, pp 448-456. https://doi.org/10.1145/2020408.2020480

Wang W, Malik H, Godfrey MW (2015) Recommending posts concerning API issues in developer Q&A sites. In: Proceedings of the international working conference on mining software repositories. IEEE/ACM, pp 224–234. https://doi.org/10.1109/MSR.2015.28. http://stackoverflow.com/questions/5358219/

Wei X, Croft WB (2006) LDA-based document models for ad-hoc retrieval. In: Proceedings of the 29th annual international conference on research and development in information retrieval. ACM, Seattle, pp 178-185. https://doi.org/10.1145/1148170.1148204

Weng J, Lim EP, Jiang J, He Q (2010) TwitterRank: Finding topic-sensitive influential twitterers. In: Proceedings of the 3rd international conference on web search and data mining. ACM, New York, pp 261-270. https://doi.org/10.1145/1718487.1718520

Wold S, Esbensen K, Geladi P (1987) Principal component analysis. Chemom Intell Lab Syst 2:37–52. https://doi.org/10.1016/0169-7439(87)80084-9

Xia X, Bao L, Lo D, Kochhar PS, Hassan AE, Xing Z (2017a) What do developers search for on the web? Empir Softw Eng 22(6):3149–3185. https://doi.org/10.1007/s10664-017-9514-4

Xia X, Lo D, Ding Y, Al-Kofahi JM, Nguyen TN, Wang X (2017b) Improving automated bug triaging with specialized topic model. IEEE Trans Softw Eng 43(3):272–297. https://doi.org/10.1109/TSE.2016.2576454

Yan M, Fu Y, Zhang X, Yang D, Xu L, Kymer JD (2016a) Automatically classifying software changes via discriminative topic model: Supporting multi-category and cross-project. J Syst Softw 113:296–308. https://doi.org/10.1016/j.jss.2015.12.019

Yan M, Zhang X, Yang D, Xu L, Kymer JD (2016b) A component recommender for bug reports using Discriminative Probability Latent Semantic Analysis. Inf Softw Technol 73:37–51. https://doi.org/10.1016/j.infsof.2016.01.005

Yang X, Lo D, Li L, Xia X, Bissyandé TF, Klein J (2017) Characterizing malicious Android apps by mining topic-specific data flow signatures. Inf Softw Technol 90:27–39. https://doi.org/10.1016/j.infsof.2017.04.007

Ye D, Xing Z, Kapre N (2017) The structure and dynamics of knowledge network in domain-specific Q&A sites: a case study of stack overflow. Empir Softw Eng 22(1):375–406. https://doi.org/10.1007/s10664-016-9430-z

Zaman S, Adams B, Hassan AE (2011) Security versus performance bugs: A case study on firefox. In: Proceedings - international conference on software engineering, pp 93–102. https://doi.org/10.1145/1985441.198545

Zeugmann T, Poupart P, Kennedy J, Jin X, Han J, Saitta L, Sebag M, Peters J, Bagnell JA, Daelemans W, Webb GI, Ting KM, Ting KM, Webb GI, Shirabad JS, Fürnkranz J, Hüllermeier E, Matwin S, Sakakibara Y, Flener P, Schmid U, Procopiuc CM, Lachiche N, Fürnkranz J (2011) Precision and recall. In: Encyclopedia of machine learning. Springer US, pp 781–781. https://doi.org/10.1007/978-0-387-30164-8_652

Zhang E, Zhang Y (2009) Average precision. In: Encyclopedia of database systems. Springer US, pp 192–193. https://doi.org/10.1007/978-0-387-39940-9_482

Zhang T, Chen J, Yang G, Lee B, Luo X (2016) Towards more accurate severity prediction and fixer recommendation of software bugs. J Syst Softw 117:166–184. https://doi.org/10.1016/j.jss.2016.02.034

Zhang Y, Lo D, Xia X, Scanniello G, Le TDB, Sun J (2018) Fusing multi-abstraction vector space models for concern localization. Empir Softw Eng 23:2279–2322. https://doi.org/10.1007/s10664-017-9585-2

Zhao N, Chen J, Wang Z, Peng X, Wang G, Wu Y, Zhou F, Feng Z, Nie X, Zhang W, Sui K, Pei D (2020) Real-time incident prediction for online service systems. In: Proceedings of the 28th ACM joint meeting european software engineering conference and symposium on the foundations of software engineering, vol 20. ACM, pp 315–326. https://doi.org/10.1145/3368089.3409672

Zhao WX, Jiang J, Weng J, He J, Lim EP, Yan H, Li X (2011) Comparing twitter and traditional media using topic models. In: Lecture Notes in Computer Science, vol 6611. Springer, Berlin, chap Advances i, pp 338-349. https://doi.org/10.1007/978-3-642-20161-5-34

Zhao Y, Zhanq F, Shlhab E, Zou Y, Hassan AE (2016) How are discussions associated with bug reworking? an empirical study on open source projects. In: Proceedings of the 10th international symposium on empirical software engineering and measurement. IEEE/ACM, Ciudad Real, pp 1–10. https://doi.org/10.1145/2961111.296259

Zou J, Xu L, Yang M, Zhang X, Yang D (2017) Towards comprehending the non-functional requirements through Developers' eyes: An exploration of Stack Overflow using topic analysis. Inf Softw Technol 84(1):19–32. https://doi.org/10.1016/j.infsof.2016.12.003

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.