The Journal of Systems and Software 216 (2024) 112111

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Applying short text topic models to instant messaging communication of software developers☆

Camila Costa Silva *, Matthias Galster, Fabian Gilson

University of Canterbury, Christchurch, New Zealand

ARTICLE INFO

Keywords:
Short text topic modeling
Instant messaging
Topic coherence
Word intrusion
Topic intrusion
Topic naming

ABSTRACT

When modeling topics from chat messages of developer instant messaging communication, individual chat messages are short text documents. Our study aims at understanding how short text topic models perform with conversations from developer instant messaging. We applied four models to nine Gitter chat rooms (with sizes ranging from ≈100 to ≈160,000 messages). To assess the quality of topics and identify the best performing models, we compared topics based on four metrics for topic coherence. Furthermore, for a subset of Gitter chat rooms we used two human-based assessments: *intrusion tasks* with 18 experts analyzing 40 topics each, and *topic naming* (assigning a name to a topic that summarizes its main concept) with eight additional experts naming 60 topics each. Models performed differently in terms of coherence metrics *and* human assessment depending on the corpus (small, medium or large chat room). Our findings offer recommendations for the selection and use of short text topic models with developer chat messages based on characteristics of models and their performance with different sizes of corpora, and based on different strategies to assess topic quality.

1. Introduction

1.1. Context and background

Software developers communicate via different communication channels (Storey et al., 2017) including *instant messaging tools*, such as Slack and Microsoft Teams. Instant messaging allows developers to create chat rooms to discuss problems, share experiences and knowledge, and to collaborate in projects (Dittrich and Giuffrida, 2011). This can lead to large amounts of information in the form of chat messages (Storey et al., 2017) that may contain relevant knowledge potentially useful for other developers, such as code snippets and API information (Chatterjee et al., 2017, 2019).

To identify relevant information in instant messaging communication, previous works applied *topic modeling* to conversations in chat rooms, for instance, Chatterjee et al. (2019) modeled topics with conversation threads from Slack communities and Stack Overflow posts. Topic modeling, e.g., using Latent Semantic Indexing (LSI) (Deerwester et al., 1990) or Latent Dirichlet Allocation (LDA) (Blei et al., 2003), is a text mining and concept extraction method that extracts topics from large corpora of textual documents (such as chat messages) to discover hidden semantic structures in text (Miner et al., 2012). Topics are represented as clusters of words based on their co-occurrence in different documents.

Previous studies (see Silva et al. (2021) for an overview) applied topic modeling to support software engineering tasks such as software maintenance (e.g., by identifying code similarities Capiluppi et al., 2020) and bug handling (by identifying topics in issue reports that can be used to link bugs to source code Tantithamthavorn et al., 2018 or to assign a bug to a developer Sun et al., 2017). Similarly, topic modeling has been used in exploratory studies as a data analysis technique to explore, for example, what topics developers discuss in different forums (Silva et al., 2021), what developers discuss related to specific topics (e.g., discussion about bots on Stack Overflow Abdellatif et al., 2020), or to map Stack Overflow tags to topics of posts (Chen et al., 2019).

1.2. Problem and motivation

According to Silva et al. (2021), the most popular topic model in software engineering studies and to analyze developer communication is LDA. As LDA discovers topics using word occurrences and co-occurrences, larger documents generally provide more word frequencies (Blei et al., 2003; Lin et al., 2014). However, Lin et al. (2014) argue that LDA may not be appropriate for informal communication text such as tweets, comments on blog posts, instant messages and Q&A posts. For example, considering that each chat message would be a

☆ Editor:Nicole Novielli.

* Corresponding author.
E-mail addresses: camila.costasilva@pg.canterbury.ac.nz (C.C. Silva), mgalster@ieee.org (M. Galster), fabian.gilson@canterbury.ac.nz (F. Gilson).

https://doi.org/10.1016/j.jss.2024.112111
Received 1 March 2023; Received in revised form 9 May 2024; Accepted 21 May 2024
Available online 27 May 2024

short text document (typically a couple of words long), the quality of the generated topics would be compromised if there are not enough words in these documents to create meaningful word clusters (Lin et al., 2014).

To generate meaningful topics from short text, studies developed topic modeling techniques specifically for short text documents. For example, Qiang et al. (2022) analyzed and implemented eight short text topic models and used them to experiment with several types of short text (e.g., tweets). However, the authors did not use their models with discussions from developers instant messaging communication and we have not found studies that applied short text topic modeling technique to instant messages of developer discussions. Similarly to Stack Overflow posts (Chatterjee et al., 2019), topic modeling applied to instant messages could help uncover topics that can be used to trace information for augmenting software documentation (Henß et al., 2012; Souza et al., 2019), identify tasks for software maintenance (Sun et al., 2015) and localize bugs and/or requirements (Ahasanuzzaman et al., 2020). Additionally, topics identified in instant messaging communication can be used as labels to train automated techniques such as classifiers (Chatterjee et al., 2021), or be applied in approaches for augmenting software documentation or tagging code snippets in conversation threads (Souza et al., 2019). Therefore, the goal of the study is to *understand how short text topic models perform with chat messages from developer instant messaging communication*.

1.3. Contributions

In detail, we provide the following contributions:

- We generate topics from conversations on Gitter[1] using four existing topic modeling techniques for short text. We analyze nine public Gitter chat rooms of different sizes: three small (e.g., one with ≈100 messages), three medium-sized (e.g., one with ≈9400 messages) and three large chat rooms (e.g., one with ≈160,000 messages).
- We evaluate the quality of short text topic models applied to software developer instant communication based on four established topic coherence metrics (Röder et al., 2015) (e.g., normalized point-wise mutual information - NPMI (Bouma, 2009)). We also compare the coherence of topic models with two text preprocessing techniques: stemming and lemmatization. We check whether more human readable topics based on a lemmatized corpus compare favorably to a stemmed corpus as frequently used for topic modeling (Silva et al., 2021). We also compare the results of short text topic models to results from LDA. While LDA is not a short text topic model, it is the most commonly used topic model in software engineering.
- To understand how comprehensible topics are for humans (the "users" of the topics), we recruit practitioners to evaluate the quality of topics based on two types of human assessment: intrusion tasks (word and topic intrusion) (Chang et al., 2009) and topic naming (i.e. assigning a name to a topic that summarizes its concept) (Hindle et al., 2015). Unlike other studies that asked humans to evaluate the usefulness of topics based on a Likert scale (Aletras and Stevenson, 2013; Newman et al., 2009; Lau et al., 2014), we apply intrusion tasks and topic naming since these require assessors to engage in actual tasks with topics. This helps us understand how humans understand topics rather than how they perceive them (e.g., as useful or not useful). Topic naming has been used in software engineering studies before (e.g., to name the concepts discussed by developers in Stack Overflow Barua et al., 2014; Haque and Ali Babar, 2020) (Silva et al., 2021). However, to the best of our knowledge, only our study uses topic naming as a strategy for measuring the quality of topics.
- To understand whether human assessment can be replaced by automatically calculated coherence scores and if theoretical coherence aligns with human comprehension of topics, we compare topic coherence metrics and human assessments.

We found that, in general, short text topic models generate coherent topics with instant messaging communication (considering corpora with different number of messages and vocabulary size). However, some models generated coherent topics based on human comprehension while others generated coherent topics based on automatically calculated metrics. Therefore, we did not find a model that performed best for all of our topic quality metrics (automated topic coherence and human judgement). Among our selection of short text topic models, GPU-PDMM had the best overall performance (see details in Section 4.4).

We did not intend to develop a *new* short text topic model, to customize or pre-train existing models with domain-specific data to optimize their performance, but to explore how existing topic models perform with developer discussions. In this sense, our study is more of a comparative study rather than a study to develop a solution per se. Therefore, our study is of interest to researchers and practitioners as it discusses implications and insights related to the practical use and potential usefulness of short text topic modeling in software engineering (see more in Section 5.4). Our findings offer guidance to select short text topic models to dealt with instant messages based on characteristics of models and their performance with different sizes of corpora, and strategies for assessing topic quality.

This paper is organized as follows. In Section 2, we introduce the background of topic modeling and studies related to our research. We describe the research method in Section 3 and present the results in Section 4. In Section 5, we further discuss our findings, and present implications and threats to validity. Finally, in Section 6 we present concluding remarks and future work.

2. Background and related work

2.1. Topic modeling – an overview

Topic modeling is a text mining technique that automatically finds topics, typically represented as clusters of words, in a given textual document (Bi et al., 2018). 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 often appear together (Barua et al., 2014; Treude and Wagner, 2019). Details about topic modeling are described in previous studies (Silva et al., 2021; Chen et al., 2016). Below we briefly introduce the basic concepts of topic modeling:

- 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);
- 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);
- Topic z: a collection of words that co-occur frequently in the documents of a corpus. Considering topic models such as LSI (Deerwester et al., 1990) and LDA (Blei et al., 2003), z refers to an m-length vector of probabilities over the vocabulary of a corpus (the probabilities indicate the likelihood that a unique word m belongs to the topic).
- Number of topics k: the number of topics (or word clusters) to be generated by the topic model.

[1] Gitter is an instant messaging tool originally used by users of Git repositories, see https://gitter.im/.

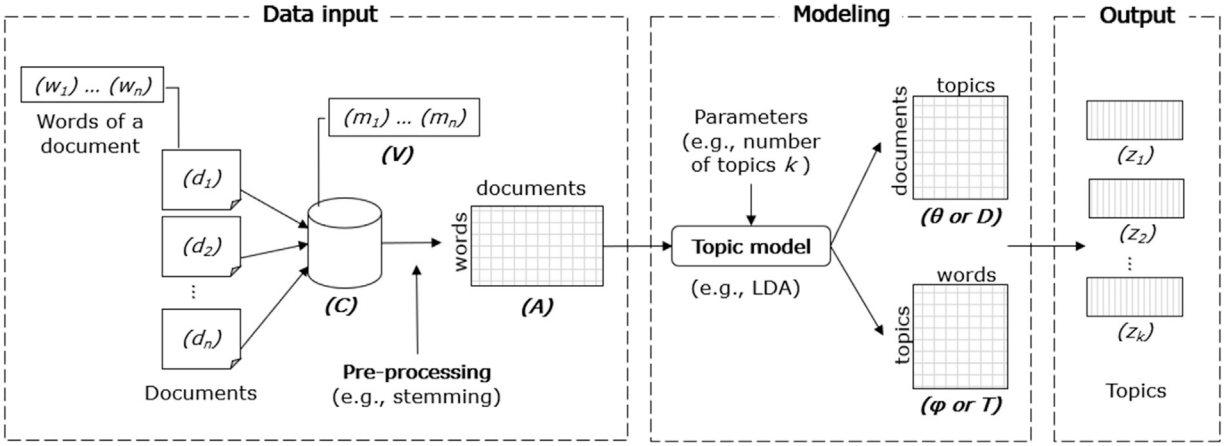

Fig. 1. General topic modeling process (Silva et al., 2021).

The general process of topic modeling does not differ when dealing with long or short documents. As shown in Fig. 1, this process takes the raw text to be used by the model (*Data Input*), then runs the topic modeling technique (*Modeling*) and analyzes the topics generated by the model (*Output*) (Silva et al., 2021):

- **Data Input:** Raw textual documents have to be organized into a corpus C from which a topic model will infer topics. This raw data is converted into a structured vector-space model such as the word-document matrix A.[2] This typically requires data pre-processing, which commonly involves tokenization, stemming and removing stop words (Miner et al., 2012) (details about how we pre-processed data in our study are in Section 3.2).
- **Modeling:** With the raw data pre-processed, the topic model is set to infer topics from corpus C. A topic modeling technique creates clusters of words or *topics* z_k by finding words that are often used together in documents in a corpus. Depending on the topic modeling technique, parameter setting is necessary, such as the number of topics (k) to be generated. LDA, for example, has the hyperparameters α and β which control an initial topic distribution. In the modeling, the topic model also generates two matrices: document-topic matrix θ (or D)[3] and topic-word matrix ϕ (or T).[4]
- **Output:** The quality of topics can be evaluated at this point of the process, considering the output of the topic modeling (e.g., resulting vectors θ and ϕ). For example, the coherence of topics can be measured with topic coherence metrics (Röder et al., 2015). Moreover, the topics generated can be applied to solve a problem (e.g., in software engineering, topics have been used to locate bugs in source code based on topics generated from issue reports Tantithamthavorn et al., 2018; Thomas et al., 2013; Rao and Kak, 2011) or to obtain empirical insights about analyzed text (e.g., topics can uncover the concepts discussed by developers around a specific subject such as deep learning, non-functional requirements or Big Data Han et al., 2020; Zou et al., 2017; Bagherzadeh and Khatchadourian, 2019).

[2] A $m \times n$ matrix whose $A_{i,j}$ entry is the weight (according to some weighting function, such as word-frequency) of word w_i in document d_j (Silva et al., 2021; Chen et al., 2016).

[3] A $n \times k$ matrix with $\theta_{i,j}$ as the probability of topic z_j in document d_i (Silva et al., 2021; Chen et al., 2016).

[4] A $k \times m$ matrix with $\phi_{i,j}$ as the probability of word w_j in topic z_i (Silva et al., 2021; Chen et al., 2016).

2.2. Topic modeling and developer communication

Previous studies on developer communication applied topic modeling to develop techniques to support software engineering tasks, such as bug handling, software documentation and software maintenance. For example, Souza et al. (2019) used LDA to develop a technique that automatically creates cookbooks for APIs using topics modeled from Stack Overflow posts; Ahasanuzzaman et al. (2020) proposed a technique that uses topics modeled with LDA to classify API issue posts in Stack Overflow.

Additionally, previous studies applied topic modeling as a data analysis technique in exploratory studies. For example, Pagano and Maalej (2013) examined posts of blogs in large open source communities of developers to identify the themes posted by different types of users; Abdellatif et al. (2020) explored Stack Overflow posts to provide insights on the topics that chat bot developers are interested in and challenges developers face.

We found one study that applied topic modeling to instant messaging discussions of developers. Chatterjee et al. (2019) used LDA to model topics from Stack Overflow posts and Slack conversations to explore what is discussed in these developer communications. Based on generated topics, the authors found that Stack Overflow and Slack are venues where developers share similar types of information (e.g., APIs and code). The authors used the entire conversation of four programming-related Slack chat rooms (e.g., ≈1000 messages) as documents; therefore, the authors did not have to use strategies for short text documents.

We did not find software engineering studies that compare short text topics models. However, we found the study of Qiang et al. (2022), who implemented and compared existing short text topic models using different sizes of corpora (from publicly available data sets). Their contributions were an open-source Java library, named STTM,[5] with the implementation of short text topic models and the comparison of these models' strengths and weaknesses. We also found studies about short text topic models in the context of artificial intelligence (Murshed et al., 2023), information systems (Li et al., 2018) and neurocomputing (Xiong et al., 2018).

2.3. Short text topic modeling

2.3.1. Overview

Lengthy documents provide context and a higher number of word counts to probabilistic topic models, while short documents do not (Yan et al., 2013). Qiang et al. (2022) identified four main characteristics of short text documents:

[5] https://github.com/qiang2100/STTM.

1. Each document has low word co-occurrences.
2. As documents have few words, sometimes only one topic can be inferred from all documents.
3. Sometimes only few (less than k) topics can be inferred from all documents.
4. Word counts from short text documents cannot fully capture semantically related words that rarely co-occur (e.g., "software testing" and "automated test" can be related but may not be identified as related).

Several previous studies adapted LDA to improve its performance with short documents via *pooling* and *contextualization* (Lin et al., 2014; Mehrotra et al., 2013; Tang et al., 2013). Pooling refers to aggregating similar (e.g., semantically or temporally) documents into a single document (Mehrotra et al., 2013). For example, Pettinato et al. (2019) 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 (e.g., the type of context for tweets can refer to time, user and hashtags associated with tweets Tang et al., 2013). For example, Weng et al. (2010) combined all individual tweets of an author into one pseudo-document (rather than treating each tweet as a document). Therefore, with contextualization, the topic model uses word co-occurrences at a context level instead of document level.

Other studies proposed topic modeling techniques and adapted LDA for specific types of data. For example, Zhao et al. (2011) developed Twitter-LDA, a model that infers topics from tweets by considering each tweet as a document which contains a single topic. Hu et al. (2019, 2018) applied Twitter-LDA to user reviews. For our study we experimented with short text topic models not designed to deal with specific types of data, such as tweets.

2.3.2. Short text topic models

Short text topic models were created to reduce the sparsity problem (i.e. the sparse word co-occurrence patterns in each short document) of probabilistic topic models, such as LDA. Qiang et al. (2022) argue that short text topic modeling techniques deal with one or two of the main characteristics of short documents (see these in Section 2.3.1). The authors divided short text topic models into three types (Qiang et al., 2022):

- *Dirichlet multinomial mixture-based models*: Techniques that follow the assumption that each text is sampled from only one latent topic. Models based on this type address characteristic (1) of short text documents and, depending on the model, characteristics (2) or (3) of short text documents.
- *Global word co-occurrences-based models*: Techniques that group the words of all inputted documents into a rich global word co-occurrence patterns and use it for inferring latent topics. Models based in this type of topic model address characteristic (1) of short text documents.
- *Self-aggregation-based models*: Techniques that merge documents into long pseudo-documents during topic inference. Topics are inferred from these long pseudo-documents without depending on auxiliary information (e.g., pre-trained word embeddings) or metadata (e.g., hashtags or timestamps). Models based in this type of topic model address characteristic (4) of short text documents.

In our study, we experimented with four models: (1) DMM (or GSDMM, Gibbs Sampling Dirichlet Multinomial Mixture model) and (2) GPU-PDMM (Pólya urn Poisson-based Dirichlet Multinomial Mixture model) which are based on *Dirichlet multinomial mixture*; (3) BTM (Biterm topic model) and (4) WNTM (Word Network topic model) which are based on *Global word co-occurrences*. As described in Section 3.3, we selected these four short text topic models from the study of Qiang et al. (2022) because they performed best with the titles of Stack Overflow posts (which are similar documents to ours) rather than the best performing models for each of the types of topic model. Since *Self-aggregation*-based models did not present a high performance with the titles of Stack Overflow (Qiang et al., 2022) posts, considering all topic quality metrics used in their study, we did not experiment with them. The details about the implementation of these models are in Qiang et al. (2022); here, we provide an overview of each:

- **DMM**: This model was developed by Yin and Wang (2014). DMM samples a topic z_d for document d by Multinomial distribution θ, and then generates all words in the document d from topic z_d by Multinomial distribution ϕ_{z_d}. The sampling, with Gibbs sampling, is based on the assumption that each document is sampled by a single topic.
- **GPU-PDMM**: This model was developed by Li et al. (2017). They proposed a Poisson-based Dirichlet Multinomial Mixture model (PDMM) that assumes that each document can generate one or more (but less than k) topics. This model was extended by incorporating generalized Pólya urn model (GPU) during the sampling process. This sampling uses pre-trained word embeddings (training based on semantic relationships between words Mikolov et al., 2013) to check the semantic similarity (with cosine similarity) between words pairs. In GPU-PDMM, each document is represented by t_d ($0 < t_d \leq \zeta$) topics, where ζ is the maximum number of topics to be generated for a document (considering the characteristic (4) of short text documents).
- **BTM**: This model was created by Yan et al. (2013). BTM first generates biterms from corpus C, where any two words in a document are treated as a biterm: $b_i = (w_{i,1}, w_{i,2})$. For example, in the short text document "I visit the apple store" and ignoring the stop words "I" and "the", we will have three biterms, i.e. "visit apple", "visit store", "apple store". Then, the model infers topics over all biterms, i.e. a "larger" corpus B with biterms.
- **WNTM**: This model was developed by Zuo et al. (2016). WNTM uses global word co-occurrence to construct a word co-occurrence network and using LDA, infers topics from this word co-occurrence network. To create the network, WNTM sets the size of a sliding window[6] to identify co-occurring words; then, it constructs a network, where each node represents a word and the weight of each edge is the co-occurrence of the two connected words. The number of nodes in the network refers to the size of the vocabulary V.

In Section 3.3 we describe how we used these short text topic models in our study.

2.4. Topic quality

Previous studies used different approaches to check the quality of topics considering human judgement (e.g., intrusion tasks, topic naming or evaluating a topic on a Likert scale); after all, if topics are to be used by humans, they need to be comprehensible and meaningful (Chang et al., 2009). Furthermore, since topics are based on the probabilities of word co-occurrences in documents that can be compared to word co-occurrences in natural language text, "intrinsic" metrics for topic coherence (not based on human judgement) exist.

2.4.1. Intrusion tasks

Unsupervised learning methods such as topic models give no guarantees on the interpretability of their output (Röder et al., 2015). Chang et al. (2009) proposed two types of intrusion tasks to evaluate the quality of topics models based on human judgement:

[6] A window is formed over consecutive words and slides over the data to capture different portions of the data. The size of the window defines the number of consecutive words to be used at a time.

1. With **word intrusion tasks**, humans are asked to identify intruder words in topics (i.e. the word with the lowest probability in the topic) to check how well the topics match human concepts. Chang et al. (2009) evaluated the model coherence by calculating *model precision* - MP for each model m and topic k. MP_k^m represents how well the intruders in topics detected by humans correspond to the actual intruder. This metric is defined as follows:

$$MP_k^m = \sum_p 1(i_{k,p}^m = w_k^m)/P$$

where w_k^m represents the intruding word selected from the kth topic inferred by model m. The intruder selected by a human p on the set of words from the kth topic inferred by model m is represented by $i_{k,p}^m$. The number of human evaluators is represented by P. The higher MP the higher the model coherence (Chang et al., 2009).

2. With **topic intrusion tasks**, humans are asked to identify intruder topics in selected documents (i.e. the topic with the lowest probability of belonging to that document) to check how well a topic model assigns topics to documents. For topic intrusion tasks, Chang et al. (2009) evaluated the model coherence using *topic log odds* - TLO. TLO_d^m is defined as how well humans identify the true intruder topic. It uses the log ratio of the probabilities of a topic in a document to compare true intruder topics and the topics selected by humans as intruders. This metric is defined as follows:

$$TLO_d^m = (\sum_p \log \hat{\theta}_{d,j_{d,*}^m}^m - \log \hat{\theta}_{d,j_{d,p}^m}^m)/P$$

where the intruding topic selected by human p for document d on model m is represented by $\hat{\theta}_{d,j_{d,p}^m}^m$, and the real intruding topic selected for document d on model m is represented by $\hat{\theta}_{d,j_{d,*}^m}^m$. The number of human evaluators is represented by P. The higher TLO, the greater the agreement between the topics created by the model and the judgements of humans. The upper bound of TLO is 0, which is achieved when humans choose intruders with a probability no higher than the probability of the real intruder (more details about this metric can be found in the study of Chang et al. (2009)). Nguyen and Hovy (2019) describe TLO as an "error function" because it considers that each topic has a probability of belonging to document d. This is more accurate than calculating coherence only based on whether humans selected the right intruder or not.

Examples of studies that assessed the quality of topics using intrusion tasks include Nguyen and Hovy (2019) and Bhatia et al. (2018) who applied intrusion tasks to measure the quality of topics generated with LDA using long text documents (e.g., a group of user reviews and news articles). Guzman et al. (2017) applied BTM and LDA to three corpora of tweets about Slack, Dropbox and Spotify. By comparing the results of the intrusion tasks of BTM and LDA, Guzman et al. (2017) found that BTM outperformed LDA.

2.4.2. Topic naming

Topic naming requires humans to assign a "label" to a topic (word cluster). As with word and topic intrusion tasks, assigning a name or a label to a topic requires understanding the common concept in the words of that topic. The more coherent and cohesive the words in a topic, the easier it is to name a topic. Silva et al. (2021) identified three approaches to name 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 judgement;
- 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.

Manual topic naming can help evaluate topics, i.e. the more topics can be named and the more consistent the names of topics assigned by different experts, the more comprehensible a topic is.

2.4.3. Topic coherence metrics

Due to the cost of running human judgement tasks, coherence metrics calculated based on topics have been proposed (Röder et al., 2015). Previous studies such as Lau et al. (2014), Aletras and Stevenson (2013), Mimno et al. (2011), Newman et al. (2009) and Newman et al. (2010) proposed automatic coherence metrics to rate topics regarding their understandability. Röder et al. (2015) implemented a set of coherence metrics.[7] We provide a brief overview of some of the metrics that were used in our study (see details about this selection in Section 3.4):

- c_{npmi} defines coherence as a score (normalized point-wise mutual information - NPMI) between top word[8] pairs from the topics and word pairs from an external corpus of general natural language. Word co-occurrences with the external corpus are derived using a sliding window[6] of size 10 (Bouma, 2009; Stevens et al., 2012). For c_{npmi} the scores are expected to be between −1 and +1, the higher the score, the better. Scores of −1 mean that the word pairs compared never occur together, while scores of +1 mean that the word pairs compared always occur together. The score 0 means that words in word pairs are independent of each other (Bouma, 2009; Choi et al., 2015). For example, when c_{npmi} scores −1 for a topic, it means that the word pairs from that topic never appear together in a general natural language text, implying that it is less coherent (or would make less sense to humans).
- c_a defines coherence as a score (NPMI and cosines similarity[9]) between top word[8] pairs from the topics and word pairs from an external corpus of general natural language. Word co-occurrences with the external corpus are derived using a sliding window[6] of size 5 (Aletras and Stevenson, 2013; Röder et al., 2015). The scores for c_a, similar to c_{npmi}, are expected to be between −1 (word pairs never occur together) and +1 (word pairs always occur together. The score 0 means that words in word pairs are independent of each other (Aletras and Stevenson, 2013). The higher the scores obtained for c_a the better the coherence of that topic or model.
- c_p defines coherence as a score between top word[8] pairs from the topics and word pairs from an external corpus of general natural language. Word co-occurrences with the external corpus are derived using a sliding window[6] of size 70. The confirmation measure in this metric is calculated with Fitelson's coherence, which scores the probability of a word in comparison to preceding subsets of words that do not contain such word (Röder et al., 2015). The higher the scores obtained for c_p the better the coherence of that topic or model (Newman et al., 2010; Aletras and Stevenson, 2013).
- u_{Mass} defines coherence as a score (smoothed conditional probability) between top word[8] pairs where one word in the pair is frequent and the other word is less frequent. Word probabilities are estimated based on the frequency of their occurrence in documents from the corpus used as input for the topic model. This metric attempts to confirm that the model learned topics known

[7] https://aksw.org/Projects/Palmetto.html.

[8] Most frequent word in a topic based on word probabilities obtained with the topic model.

[9] Measure of similarity between two non-zero vectors of an inner product space.

to be in the corpus (Mimno et al., 2011; Stevens et al., 2012). For u_{Mass}, the scores obtained are generally negative values and the closer to zero, the better the coherence (Mimno et al., 2011). For example, u_{Mass} scoring −0.50 is less coherent than u_{Mass} scoring −0.05 because higher scores for this metric represents that word pairs occurred frequently together in the corpus applied to the topic model producing an "internally" consistent topic.

2.4.4. Comparison of coherence metrics and human judgement

Previous studies compared the performance of coherence metrics and coherence based on human judgement. For example, Newman et al. (2009) modeled topics using LDA and asked humans to decide whether these topics are "useful" or not (defined rather loosely in the study as a combination of coherence, meaningfulness, interpretability, something easy to label) based on a 3-point Likert scale. They found that the metric point-wise mutual information (PMI) distinguished useful topics from useless ones based on the high correlation between PMI and human judgements on the same topics. In our study, we used the normalized version of PMI, c_{npmi} which is described in Section 2.4.

Similarly, Aletras and Stevenson (2013) compared human judgement of the quality of topic from LDA based on a Likert scale ("Useless", "Average" and "Useful") and two metrics: c_{npmi} and u_{Mass}. The study found that c_{npmi} correlates well with human judgements, while u_{Mass} does not correlates well.

Lau et al. (2014) performed two approaches for human judgement: humans who performed word intrusion tasks (using MP to assess the human evaluation) and humans who assessed topics using a 3-point Likert scale (e.g., "Useless", "Average" and "Useful"). The authors compared the scores of the two human judgement approaches with the scores of automated coherence metrics, such as c_{npmi}, u_{Mass} and c_a. At the topic level, Lau et al. (2014) found that their automated coherence metrics align better with the scores obtained from assessing topics on a 3-point Likert scale than with the scores of word intrusion tasks.

To check the u_{Mass} metric, Mimno et al. (2011) compared it to the word intrusion tasks and MP (Chang et al., 2009). They found that topics with low MP (i.e. low accuracy) tend to be "low quality", which means that humans could not group together the frequent words in that topic as a single coherent concept; on the other hand, some "low quality" topics can have a high MP (i.e. high accuracy). This means that a topic with unrelated words (e.g., "database", "query", "sql", "table", "excel", [intruder:] "exception") can be considered good, if the intruder word differs too much to the others (e.g., "excel" is not related to database theme, but it differs less than the intruder word "exception"). Mimno et al. (2011) concluded that some "low quality" topics cannot be detected using word intrusion tasks, but their metric can identify such topics. This is because the metric relates words in a topic to the corpus based on which the topics were created, while the humans would evaluate intruders based on a limited understanding of the full context and corpus.

Similar to these previous studies, we compared the performance of topic models using the scores of intrusion tasks and topic coherence metrics. Our study differs from these studies by mainly comparing short text topic models and by using developer instant messaging communication (i.e. messages from Gitter chat rooms).

3. Research method

In this section we describe how we studied how short text topic models perform with chat messages from developer instant messaging communication. First, we describe how we modeled topics using short text topic modeling techniques (Sections 3.1, 3.2, 3.3), and then we discuss how we evaluated the topics (Sections 3.4 and 3.5) and compare their results (Section 3.6). Fig. 2 illustrates the steps of our study.

3.1. Selection of data sets

We used messages exchanged by developers in public chat rooms of Gitter, a free open source instant messaging tool.[10] The messages of each chat room represent a data set (or *corpus*). We selected Gitter chat rooms for the following reasons (Costa Silva et al., 2019):

- Most of the discussions on Gitter focus on software development (Ehsan et al., 2021).
- Gitter offers comprehensive APIs to access developer communication free of charge.[11] Gitter's data use policy and privacy regulations do not restrict the use of data by other users[12];
- Gitter does not restrict access to chat rooms (e.g., no authorization required from chat room administrators).

To identify chat rooms for this study, we first used the Gitter search to search for generic terms like ["software" or "development" or "management" or "quality" or "technology"], words usually found in textbooks (Bass et al., 2003). Note that to be selected, chat rooms had to have one of these keywords rather than all. At the time we were selecting chat rooms, Gitter did not provide a list of all chat rooms but a communities page with predefined categories (e.g., "Android" and "Java Script") that could be used to filter the most popular chat rooms in each category. We checked that all categories referred to software development tools or technologies, and that not all chat rooms were tagged into one of these categories to have a balanced sample. Therefore, we performed such general searches to find more chat rooms that could discuss themes not necessarily related to tools or technologies.

We obtained 87 chat rooms with conversations in English. Based on the number of messages of these 87 chat rooms (min: 76; max: 182,853; stdev: 37,852; mean: 24,359; median: 6374), we selected three groups of chat rooms of different sizes for a representative sample. Therefore, our data sets were represented by three small rooms: Android[13] with ≈100 messages; WebpackDocs[14] with ≈900 messages; and ConsenSys[15] with ≈1600 messages. Three medium-sized rooms: Jenkinsci[16] with ≈9400 messages; Locomotive[17] with ≈6900 messages; and SpringSecurity[18] with ≈7900 messages. Three large rooms: Flutter[19] with ≈160,000; Laravel[20] with ≈88,800 messages; and GitterHQ[21] with ≈42,000 messages. Then, we downloaded all messages posted until August 2019 (time of data collection) using the Gitter API.[22] The files downloaded are available online.[23] We used 89% of the original messages from Flutter, our largest chat room, due to computational constraints when modeling topics from individual messages. To reduce the number of messages, we randomly excluded blocks of sequential messages until we reached a data set size we were able to process. Table 1 describes these data sets in terms their size.

3.2. Data pre-processing

We pre-processed data based on the data pre-processing performed by other studies that modeled topics from developer and end user short

[10] https://gitter.im/.
[11] https://spec.matrix.org/latest/client--server-api/.
[12] https://static.element.io/legal/online-EULA.pdf.
[13] https://gitter.im/googlesamples/android-architecture.
[14] https://gitter.im/webpack/docs
[15] https://gitter.im/ConsenSys/smart-contract-best-practices.
[16] https://gitter.im/jenkinsci/configuration-as-code-plugin.
[17] https://gitter.im/locomotivecms/engine.
[18] https://gitter.im/spring-projects/spring-security.
[19] https://gitter.im/flutter/flutter.
[20] https://gitter.im/laravel/laravel.
[21] https://gitter.im/gitterHQ/developers.
[22] https://spec.matrix.org/latest/client--server-api/.
[23] https://zenodo.org/records/11362380.

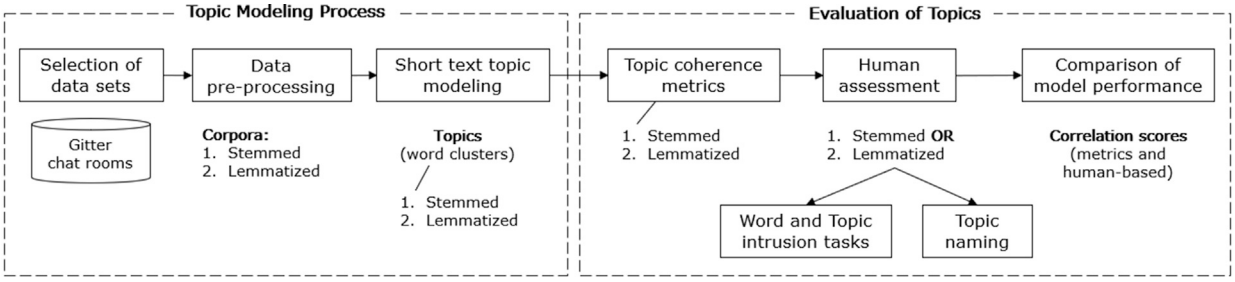

Fig. 2. Steps of the study.

Table 1
Size of data sets (number of messages) and messages length (number of words) before data pre-processing.

Length of messages	Small			Medium			Large		
	Android	WebpackDocs	ConsenSys	Jenkinsci	Locomotive	Spring security	Flutter	Laravel	GitterHQ
Mean	15	13	20	17	12	20	17	14	15
Median	11	9	12	12	8	12	12	9	9
Std deviation	14	17	31	19	16	25	20	20	23
Minimum	1	1	1	1	1	1	1	1	1
Maximum	87	221	579	276	229	472	1373	1640	558
Number of messages	119	938	1611	9408	6950	7945	162,426	88,801	41,983

text communication (Silva et al., 2021). We did not remove source code tokens (e.g., terms like "if", "when"). This would have reduced the vocabulary of our corpora and would have impacted the generation of distinct topics (i.e., less significant co-occurring words or the same words may be relevant for multiple topics). On the other hand, the occurrence of source code tokens (e.g., "begin", "end", "for") may be less informative for topic generation than the occurrence of tokens such as "test", "compile" or "error", which could be most likely be associated to software engineering subjects. Note that some of these tokens may have been removed based on our stop words list (e.g., "for" and "while"):

1. Remove noise (URLs, e-mail addresses, mentions of users, new line characters, non-English characters such as Chinese or Japanese characters, punctuation and HTML tags) and extra white spaces;
2. Tokenize messages to break the text of a document (message) into individual words, referred to as *tokens*;
3. Convert tokens to lowercase;
4. Remove tokens with fewer than three letters;
5. Remove stop words (e.g., "the", "a", "in") from tokenized messages using a customized stop word list.[24] Our list of stop words is available online[23];
6. Stemming (with Porter Stemmer - package from NLTK 3.6.2[25]) **or** lemmatizing (with *spaCy* lemmatizer library - version 3.0[26]) to normalize tokens.
7. Remove empty documents (documents that ended up empty after the previous steps).

For step (6) of the pre-processing, we duplicated our three data sets: one to be normalized with stemming and another to be normalized with lemmatization (see Fig. 2). Stemming removes prefixes and suffixes to normalize words into their morphological root, e.g., the words "tested" and "testing" can be stemmed to "test", and lemmatization captures only the base or dictionary form of words or bigrams. One benefit of lemmatization over stemming is that lemmatization retains the readable form of words (e.g., the lemmatized version of the words "programming" and "programmer" are "programming" and "programmer", while their stemmed version is "programm"). Since lemmatization keeps different "versions" of the same word, it leads to a larger vocabulary that, on the other hand, lowers the chances of re-occurring words. We experimented with stemming and lemmatization because previous studies (that modeled topics in short documents to be interpreted by humans) did not agree on using stemming or lemmatization. For example, several previous studies used stemming as a default normalization procedure (Bagherzadeh and Khatchadourian, 2019; Ahmed and Bagherzadeh, 2018; Noei et al., 2019; Barua et al., 2014; Haque and Ali Babar, 2020), while some used lemmatization (Nayebi et al., 2018; Martin et al., 2015; Abdellatif et al., 2020). After these steps, we transformed our corpora (nine stemmed and nine lemmatized) into 18 *bags of words* to be submitted to our short text topic models. In our study, these *bags of words* were text files (.txt) where each line in the file referred to one message (i.e. one document), and each document was a sequence of words separated by white spaces. Table 2 describes our pre-processed data sets. Regarding the differences in number of documents across similar corpora (e.g., Laravel_stem and Laravel_lemma), we found that more documents were left empty after being pre-processed with stemming than with lemmatization. We noticed that lemmatization kept more documents with at least one word in comparison to stemming, which impacted the total number of documents considering that we removed all empty documents for the modeling.

3.3. Short text topic modeling

3.3.1. Selection of models

We modeled topics using four short text topic models reported by Qiang et al. (2022) (BTM, DMM, GPU-PDMM and WNTM, see Section 2.3). To select these topic models, we first checked which of the data sets used by Qiang et al. (2022) was similar to ours in terms of document length. Qiang et al. (2022) used different data sets, such as titles of Stack Overflow posts, tweets, and strings used in web search queries. The data set with titles of Stack Overflow posts was the most similar to our data sets after data pre-processing (average of 6 words in each document). In general, titles of Stack Overflow posts have an average number of 10 words (Liu et al., 2022) Then, we identified which were the best performing models for the "Stack Overflow" data

[24] This list is a combination of the stop words from Ranks NL website https://www.ranks.nl/stopwords, the SMART (Layman et al., 2016; Pettinato et al., 2019), Mallet (Ahmed and Bagherzadeh, 2018; Bagherzadeh and Khatchadourian, 2019; Yan et al., 2016) lists, and words added by us (e.g., text abbreviations like "lol" or "idk").

[25] https://www.nltk.org/howto/stem.html.

[26] https://spacy.io/api/lemmatizer.

Table 2
Description of corpora.

Corpus	Number of documents	Vocabulary size
Android_stem	84	253
Android_lemma	84	262
WebpackDocs_stem	864	1269
WebpackDocs_lemma	861	1369
ConsenSys_stem	1406	2258
ConsenSys_lemma	1406	2509
Jenkinsci_stem	8163	5299
Jenkinsci_lemma	8146	6134
Locomotive_stem	5504	3844
Locomotive_lemma	5502	4325
SpringSecurity_stem	7077	4897
SpringSecurity_lemma	7092	5665
Flutter_stem	141,040	36,627
Flutter_lemma	140,757	41,565
Laravel_stem	70,459	28,626
Laravel_lemma	76,228	31,899
GitterHQ_stem	33,955	20,728
GitterHQ_lemma	34,081	23,262

set in the study of Qiang et al. (2022) considering the results of their strategies for topic quality assessment (classification, clustering and topic coherence). We also modeled topics with LDA as the baseline for the comparison of the performance of our selection of short text topic models.

3.3.2. Parameter setting and execution of models

We ran our selection of models (including LDA) on each of our 18 corpora using the implementation of the STTM library[5]. We set the hyperparameters α, β and i (i.e. number of sampling iterations) as suggested by the authors who proposed the selected models rather than grid searched values (see details in Section 5.5), except by LDA, which was used as the baseline for the performance comparison of our short text topic models. We grid-searched LDA hyperparameters (using sklearn GridSearchCV library[27]) considering that we would model a fixed number topics for all corpora. Therefore, we set LDA with the hyperparameters: $\alpha = 0.1; \beta = 0.1; i = 1000$.

BTM: $\alpha = 50/k; \beta = 0.01; i = 1000$
GPU-PDMM: $\alpha = 0.1; \beta = 0.01, i = 1000$
DMM: $\alpha = 0.1; \beta = 0.1; i = 30$
WNTM: $\alpha = 50/k; \beta = 0.01, i = 2000$

To compare the results of the models considering both automated and human-based metrics, we generated ten topics for each model, i.e., $k = 10$. Even though the best number of topics may depend on the size and vocabulary of data sets, we selected a single number of topics for our corpora as we aimed at performing both automated and human-based assessments on the same topics. We selected ten topics inspired by the study of Canfora et al. (2014), which initially modeled 30 topics from commit comments (i.e., each comment as a document) and then, reduced this number to 10 topics to avoid repetitive topics. The list of topics generated by each combination of model and corpus is available online[23].

3.4. Topic coherence metrics

We calculated the topic coherence metrics described in Section 2.4 for each combination of model and corpus. We selected four (c_{npmi}, c_a, c_p and u_{Mass}) of the six metrics analyzed in the study of Röder et al. (2015). Our selection was based on the strongest average correlation of the metrics to human ratings (Röder et al., 2015). We used the Palmetto library[28] to calculate these metrics at topic level. To get scores at model level, which are shown in Tables 4–6, we calculated the average between the scores (at topic level) for each metric (i.e. we calculated the average between the scores of the ten topics generated by each combination of model and corpus).

3.5. Human assessment

Coherence metrics only measure "lexical similarity" between the topics and a corpus, they do not capture comprehensibility or meaningfulness of topics (van der Lee et al., 2021). Therefore, we also performed an evaluation of topics using human judgement[29]: word and topic intrusion tasks, and topic naming. Considering the cost of human judgement (Newman et al., 2010; Lau et al., 2014), we evaluated topics based on human assessment only for:

- Topics based on our lemmatized corpora, but not for the stemmed corpora (see details about data pre-processing in Section 3.2). This selection, as detailed in Section 4.1.4, was made based on the higher average scores for the topic coherence metrics (see Section 3.4) of topics generated for each combination of model and lemmatized corpus.
- Topics modeled only with short text topic models. Therefore, we did not apply human assessment to topics generated with LDA (see findings regarding LDA topics in Section 4.1.3).
- Topics generated by three of our nine data sets. To keep a study with human experts feasible, for the human assessments, we randomly chose one example of each of our groups of data sets: Android (for the small group), Jenkinsci (for the medium-sized group), and Flutter (for the large group). Details about these data sets are shown in Table 1.

3.5.1. Intrusion tasks

Word Intrusion Tasks: In this task, each topic generated for each combination of model and corpus (e.g., *BTM & Android_lemma*) was presented to study participants with six randomly ordered words, where five of these words belong to the topic and one word is an "intruder" (e.g., for a topic with words "error", "fail", "glitch", "message", "problem" and "wrong", the word "message" can be the intruder), see Fig. 3 for an example task. Note that in word intrusion tasks participants only see topics, but no messages. For a word intrusion task, participants should be able to associate six individual words and understand which of these is an intruder (i.e. a word that does not make sense in that group of words). Each list of six words represents a topic generated by the model (five words from the topic and an intruder word), as previously exemplified. In this task, a topic is coherent if participants are able to select the real intruder, rather than a word that was generated by the topic model (Chang et al., 2009).

Participants were asked to identify the intruder in 20 topics of different combinations of model and corpus. We then calculated to calculated MP as described in Section 2.4.1. To select an intruder word for a topic, we followed procedures suggested by Murphy et al. (2012) and Chang et al. (2009). For each topic we first created an list of its words in ascending order based on the probabilities of words occurring in topics (according to the topic-to-word distributions). Then, from each topic, we selected the first word from the bottom half of that list. As suggested by Murphy et al. (2012), if the selected word was found in any of the top tenth percentile words in any of the other topics (from the same combination of model and corpus), it was selected as the intruder for that topic. In case we did not find the selected word in

[27] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[28] https://aksw.org/Projects/Palmetto.html.

[29] The study received approval for our institution's Human Ethics Board.

the other topics, we picked the next word in the bottom half list of words for that topic.

Topic Intrusion Tasks: To have the same number of tasks for both word and topic intrusion, we developed ten topic intrusion tasks for each combination of model and corpus. For each topic intrusion task, a document (i.e. a message) was presented to study participants together with four topics where one topic was the intruder (i.e. the topic that does not belong to the document presented), as suggested by Chang et al. (2009). Fig. 3 shows an example task. For a topic intrusion task, participants have to evaluate the relationship between topics and a document (which is about that group of topics) and select the topic that does not belong to that document. Therefore, each task is a document and four topics (three topics found in that document and an intruder topic), as previously exemplified. With the topic intrusion task, a topic is coherent if participants are able to select the real intruder topic regarding a document (Chang et al., 2009).

Note that having four topics in topic intrusion tasks is not necessarily easier for participants than having six words in word intrusion tasks. Chang et al. (2009) designed topic intrusion tasks with fewer items because participants have to evaluate the relationship between topics and a document (which is about that group of topics) and select the topic that does not belong to that document. Therefore, the topic intrusion task requires overall more effort from participants.

Participants had to identify the intruder topic in 20 documents of different combinations of model and corpus. We then calculated TLO to measure the quality of the model (see Section 2.4.1). To decide which document to show with each group of four topics, we randomly selected ten documents for each combination of model and corpus with at least 15 words, since documents in our corpora had an average of 15/17 words per message before data pre-processing (see Table 1). Fifteen words is also the average length of sentences in English language (TheAcropolitan, 2017). To decide which topics to show, we followed the guidelines of Chang et al. (2009) and used the three topics with the highest probability to appear in that document (according to the document-topic matrix for that combination of model and corpus) and one additional topic as an intruder. The intruder topic was the least-probable topic in that same document.

Execution of Tasks: To distribute intrusion tasks to participants, we created one survey in Qualtrics.[30] Fig. 3 shows an example of a word intrusion and a topic intrusion task (the text provided for the topic intrusion task shows the document provided with the list of topics). We presented the answer options for each task (i.e. the intruder word or topic) in random order. After each block of ten tasks we included an optional open question for feedback "Feel free to leave any comments about the previous tasks". All questionnaires were fully completed by participants. It took participants between 8 min and 5 h to complete all tasks (considering that the questionnaire kept a user session open until its completion).

We created 240 tasks in total (120 word intrusion and 120 topic intrusion tasks), see Table 3. To have each word and topic intrusion task to be performed by three different participants, we created 18 sub-surveys of 40 tasks each (20 word and 20 topic intrusion tasks) - see the list of sub-surveys in Appendix A. The tasks in the sub-surveys partially overlap. The 40 tasks in one sub-survey covered two combinations of model and corpus: 20 word and topic intrusion tasks for a combination of model and corpus, and 20 word and topic intrusion tasks for another combination of model and corpus. For example, by asking participant IT.01 to perform sub-survey 1 and participant IT.02 to perform sub-survey 12, they would have performed tasks related to different combinations of model and corpus.

[30] https://www.qualtrics.com/au/.

Table 3
Number of intrusion tasks[a] by model and corpus.

	BTM	DMM	GPU-PDMM	WNTM	*Total*
Android_lemma	20	20	20	20	80
Jenkinsci_lemma	20	20	20	20	80
Flutter_lemma	20	20	20	20	80
Total	60	60	60	60	*240*

[a] Ten word intrusion tasks and ten topic intrusion tasks for each combination of model and corpus.

3.5.2. Topic naming

We asked participants to analyze and inductively name the topics by deducting a name based on the meaning of words in topics (Silva et al., 2021; Barua et al., 2014) (see Section 2.4.2). We assessed the quality of the models based on:

- Number of topics named t_{named}: We recorded how many topics (out of the ten topics generated for each combination of model and corpus) were named by at least one participant, e.g., for the combination of model and corpus BTM & Android_lemma, $t_{named} = 8$ because eight topics for that combination of model and corpus were named (detailed data is available online[23]). However, the number of topics names t_{named} does not differentiate how many participants named each topic. Therefore, we complement that metric with the average ratio of names a_{ratio}.
- Average ratio of names per topic a_{ratio}: To calculate a_{ratio}, we first counted how many names were assigned to a topic n (out of the total number of participants p who named a topic). Then, we divided this number n by p to get the ratio of names per topic (n/p). For example, considering that topic01 of BTM & Android_lemma was named by three (out of four) participants, its ratio of names per topic (3/4) would be 0.75. We then calculated the average of the ratios of the ten topics t generated for each combination of model and corpus ($a_{ratio} = (n/p)/t$).
- Compatibility between two or more names assigned to the topics: A topic could be assigned up to four names (total number of participants naming each topic). As we did not give participants predefined names that they could assign to topics, we could not directly calculate agreement on the topic naming. Instead, we interpreted how "compatible" names were. To check the compatibility between the names assigned to a topic by at least two participants, we analyzed the semantic meaning of names. In other words, we considered synonyms, dictionary meanings and domain knowledge to judge if the names assigned to a topic were compatible. For example, if a participant named a topic "Errors" and another participant "Bugs", we considered them compatible. Another example is the compatibility between the names "software development" and "program development" because "software" and "program" are synonyms in this context. One researcher judged the initial compatibility of names across combinations of model and corpus and the other researchers validated this analysis by agreeing or suggesting changes. Disagreements on names compatibility were discussed until a consensus was reached between all researchers.

Execution of Tasks: Similarly to the intrusion tasks, we created a survey in Qualtrics[30] for the topic naming tasks with a total of 120 topics to be named. Fig. 4 shows a screenshot of a topic naming task in Qualtrics. For these tasks, if a topic was not named (i.e., ratio button "No, I cannot name it" is selected), we took this as a hint that its words were not comprehensible enough to be described with a short label. We assumed that participants were experienced enough in software development to be able to perform our tasks and if they could not think of a name for a topic, the quality of that topic was poor. We did not allow participants to skip topics without selecting "No, I cannot name it" because, if a topic was skipped and left blank, we would not know

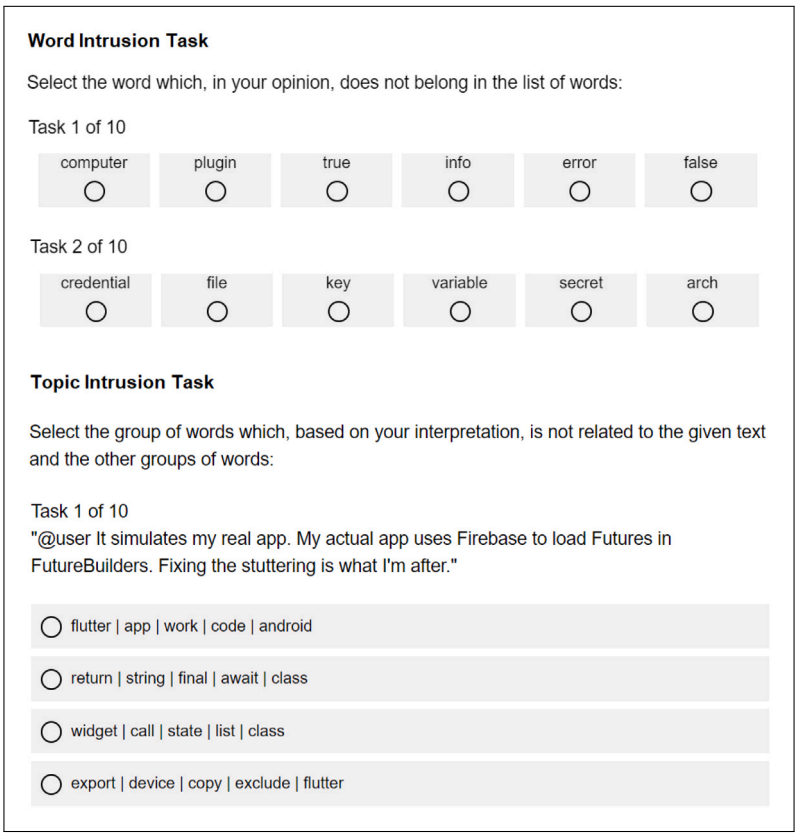

Fig. 3. Examples of word intrusion and topic intrusion tasks.

whether it would mean that (a) participants did not want to perform that task for some reason or (b) participants could not label a topic as its words did not make sense.

After each block of ten topic naming tasks, we included an optional open question for feedback "Feel free to leave any comments about the previous items". All questionnaires were fully completed by participants. It took participants between 19 min and 12 h to complete all topics (considering that the questionnaire kept a user session open until its completion).

To have four participants naming the same topics, we created six sub-surveys of 60 topics each — see the list of sub-surveys in Appendix B. This means that each sub-survey had 10 topics of six different combinations of model and corpus. For example, by asking participant TN.07 to perform sub-survey 2 and participant TN.08 to perform sub-survey 4, they would have named topics related to different combinations of model and corpus.

3.5.3. Participant selection and recruitment

We asked 18 domains experts (e.g., software developers) working in software development for at least three years (counting from their first software engineering job) to participate in word and topic intrusion tasks (Shrikanth et al., 2021 mention that professionals with three or more years of experience can be considered experts). Similarly, we asked eight domains experts also working in software development for at least three years (counting from their first software engineering job), who did not participate of the intrusion tasks, to participate in topic naming tasks. Hindle et al. (2015) explained that domain experts can name topics better than non-experts due to their familiarity to domain-specific keywords that may appear in topics. We selected these numbers of participants to have at least three participants evaluating the same intrusion tasks and four participants naming the same topics. Domain experts were identified and recruited from our contacts in industry and academia. We offered an incentive (i.e. eGift cards) for their participation, but some of them participated without accepting this incentive.

After completing all intrusion tasks or all topic naming tasks, each participant answered a survey about their professional background and a final and optional open question for general feedback: "After completing the tasks and the background questions above, is there anything else you would like to add?". Regarding participants' background, we asked:

- What country do you currently reside in?
- How many years have you worked as a software development professional in industry?
- Have you received any formal training (such an apprenticeship or a degree from a college or university) in Computer Science, Software Engineering, Information Systems or a related discipline?
- What is the highest level of formal training you received?
- If you have not received any formal training, how have you developed your skills in software engineering and development?
- How familiar are you with these technologies? (Android app development; Jenkins Continuous Integration; and Flutter development kit)

    1. I have never heard about it
    2. I have heard about it but don't use or develop with it
    3. I have used/developed with it a few times
    4. I occasionally use/develop with it
    5. I frequently use/develop with it

  Regarding the terms "use" and "develop" in the list above, we used both terms to describe that Flutter and Jenkins could be "used" in software development, and Android apps could be "developed" by participants (i.e. the meaning of "use" of Android apps as end user is different to "use" of Flutter or Jenkins as developer).

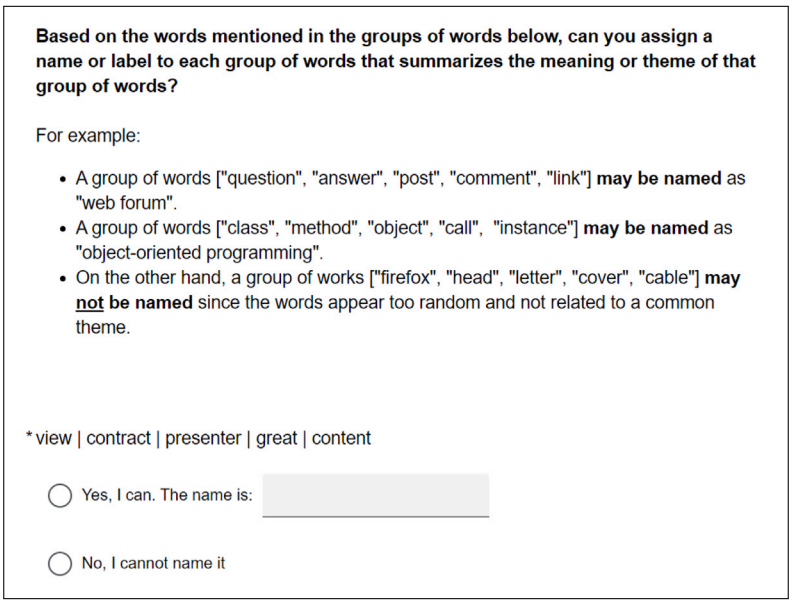

Fig. 4. Example of a topic naming task.

- Do you use instant messaging communication tools (e.g., Slack, Gitter, MS-Teams etc.)?
- Why do you use instant messaging communication tools?
    1. To chat with family and friends [Personal]
    2. To chat with people while gaming [Personal]
    3. To interact with a community (people with the same interests as you) [Personal]
    4. To keep updated with companies and brands [Personal]
    5. To discuss topics related to software development with a general community [Professional]
    6. To discuss software development issues in a particular project with other project or team members or contributors [Professional]
    7. To keep updated with software releases and new technologies for software development [Professional]
    8. Other
- How often do you use instant messaging communication tools to discuss software development?

Questions regarding familiarity with technologies and use of instant messaging communication helped us understand how comfortable participants would be with the themes discussed in our corpora and with instant messaging tools. The characterization of participants is shown in Tables 7 and 9, and detailed data is available online[23].

3.6. Comparison of model performance

We first analyzed the performance of our models by comparing the resulting scores of our three strategies for topic quality: topic coherence metrics, topic coherence based on word and topic intrusion tasks, and topic naming. Then, we calculated the correlation between scores of our topic coherence metrics and the scores of our human assessments to understand their relationship and if they are aligned with each other.

As mentioned in Section 2.4.4, studies such as Lau et al. (2014), Aletras and Stevenson (2013), Mimno et al. (2011) and Newman et al. (2009) compared how topic coherence metrics of topic models align with human evaluations of topics. These studies mostly used correlation coefficients to relate metrics and human-based scores. For example, Lau et al. (2014) calculated the Pearson correlation[31] between metrics and human evaluations using scores obtained at the topic level and at the model level, while Aletras and Stevenson (2013) used the Spearman's Rank correlation[32] to compare metrics and human evaluations. We applied Spearman correlation coefficients to compare our metrics because we did not assume a normal distribution of scores. Therefore, we checked if our automated and human-based metrics for topic quality are related (i.e. are similar) or if they provide different insights that could complement each other. All of our statistical tests were executed in Python, using implementations available in the *SciPy* library.[33]

To correlate the scores of our topic coherence metrics and human assessments at the topic level, we defined two additional metrics:

- RTI is the ratio of true intruders per topic, i.e., the total number of words selected correctly as an intruder i in a topic divided by the number of participants p who identified such intruders (i/p). Note that if the word selected by the participant was the true intruder, we counted 1; if not, we counted 0. The higher RTI the better, being 1 the highest possible score.
- RN is the ratio of names per topic, i.e., the number of unique names assigned to a topic n divided by the total number of participants p who named that topic (n/p). For this metric we are not considering the meaning of the names used to label topics, but if a name was assigned (count 1) or not (count 0). The higher RN the better, being 1 the highest possible score.

We did not use MP (word intrusion tasks) and a_{ratio} (topic naming tasks) when correlating coherence scores and human assessment because MP and a_{ratio} measure topic quality at model level rather than at topic level. Additionally, we did not compare our topic coherence metrics with the scores of topic intrusion tasks TLO because this metric evaluates the relation between documents and topics, which measures the quality of the model rather than the topics individually. Therefore, we correlated the scores of our topic coherence metrics c_a, c_p, c_{npmi} and u_{Mass} with RTI and RN.

[31] Pearson correlation measures the strength of a linear association between two variables (Lund Research Ltd, 2018a).

[32] Spearman's Rank correlation measures the strength and direction of association between two ranked variables (Lund Research Ltd, 2018b).

[33] https://docs.scipy.org/doc/scipy/reference/main_namespace.html.

Table 4
Topic coherence metrics by model and small corpus.

Metric	Corpus	BTM	DMM	GPU-PDMM	WNTM	LDA
c_a	Android_stem	0.08	0.09	0.05	0.08	0.21
	WebpackDocs_stem	0.11	0.22	0.20	0.12	0.18
	ConsenSys_stem	0.16	0.37	0.28	0.16	0.20
	Android_lemma	0.14	0.10	0.14	0.11	0.21
	WebpackDocs_lemma	0.23	0.28	0.22	0.16	0.25
	ConsenSys_lemma	0.22	0.40	0.38	0.21	0.29
c_p	Android_stem	−0.19	−0.29	−0.54	−0.24	−0.28
	WebpackDocs_stem	−0.36	−0.36	−0.13	−0.30	−0.19
	ConsenSys_stem	−0.29	−0.18	−0.27	−0.16	−0.32
	Android_lemma	−0.01	−0.09	−0.07	−0.05	−0.03
	WebpackDocs_lemma	−0.13	−0.24	−0.14	−0.11	−0.18
	ConsenSys_lemma	−0.26	−0.14	−0.09	−0.16	−0.16
c_{npmi}	Android_stem	−0.03	−0.04	−0.07	−0.04	−0.05
	WebpackDocs_stem	−0.08	−0.03	−0.02	−0.06	−0.03
	ConsenSys_stem	−0.07	−0.05	−0.06	−0.05	−0.07
	Android_lemma	−0.06	−0.06	−0.07	−0.03	−0.06
	WebpackDocs_lemma	−0.03	−0.01	−0.04	−0.03	−0.03
	ConsenSys_lemma	−0.08	−0.04	−0.03	−0.06	−0.05
u_{Mass}	Android_stem	−4.35	−6.01	−8.50	−4.61	−3.74
	WebpackDocs_stem	−4.65	−2.96	−2.43	−4.14	−3.61
	ConsenSys_stem	−4.66	−4.06	−4.53	−3.32	−5.04
	Android_lemma	−3.16	−3.85	−3.74	−2.81	−4.73
	WebpackDocs_lemma	−3.23	−2.83	−2.79	−4.23	−3.04
	ConsenSys_lemma	−5.20	−4.05	−3.34	−3.83	−3.69

Table 5
Topic coherence metrics by model and medium-sized corpus.

Metric	Corpus	BTM	DMM	GPU-PDMM	WNTM	LDA
c_a	Jenkinsci_stem	0.08	0.12	0.10	0.14	0.21
	Locomotive_stem	0.22	0.19	0.16	0.20	0.17
	SpringSecurity_stem	0.14	0.20	0.24	0.20	0.21
	Jenkinsci_lemma	0.13	0.15	0.16	0.12	0.29
	Locomotive_lemma	0.31	0.44	0.40	0.36	0.42
	SpringSecurity_lemma	0.28	0.35	0.39	0.34	0.45
c_p	Jenkinsci_stem	−0.44	−0.27	−0.37	−0.26	−0.26
	Locomotive_stem	−0.39	−0.31	−0.30	−0.30	−0.13
	SpringSecurity_stem	−0.19	−0.14	−0.14	−0.10	−0.11
	Jenkinsci_lemma	−0.18	0.01	0.07	−0.21	0.00
	Locomotive_lemma	−0.02	0.05	0.09	0.08	0.04
	SpringSecurity_lemma	0.02	0.15	0.24	0.14	0.27
c_{npmi}	Jenkinsci_stem	−0.07	−0.06	−0.09	−0.06	−0.05
	Locomotive_stem	−0.10	−0.07	−0.06	−0.08	−0.05
	SpringSecurity_stem	−0.08	−0.04	−0.01	−0.01	−0.04
	Jenkinsci_lemma	−0.10	−0.08	−0.03	−0.09	−0.04
	Locomotive_lemma	−0.02	0.00	0.00	−0.02	−0.03
	SpringSecurity_lemma	−0.05	0.01	0.04	0.00	0.04
u_{Mass}	Jenkinsci_stem	−5.54	−3.81	−4.48	−4.43	−3.54
	Locomotive_stem	−4.96	−2.61	−4.21	−5.80	−5.22
	SpringSecurity_stem	−3.44	−2.46	−1.83	−3.66	−1.96
	Jenkinsci_lemma	−5.57	−4.54	−2.77	−5.20	−4.41
	Locomotive_lemma	−2.37	−2.08	−2.47	−2.23	−2.94
	SpringSecurity_lemma	−3.78	−2.95	−2.62	−2.90	−3.08

Table 6
Topic coherence metrics by model and large corpus.

Metric	Corpus	BTM	DMM	GPU-PDMM	WNTM	LDA
c_a	Flutter_stem	0.15	0.24	0.21	0.13	0.22
	Laravel_stem	0.29	0.39	0.31	0.30	0.39
	GitterHQ_stem	0.34	0.40	0.29	0.38	0.40
	Flutter_lemma	0.21	0.23	0.24	0.22	0.38
	Laravel_lemma	0.25	0.44	0.47	0.42	0.35
	GitterHQ_lemma	0.34	0.41	0.46	0.46	0.52
c_p	Flutter_stem	−0.25	−0.09	−0.16	−0.21	−0.22
	Laravel_stem	−0.26	−0.18	−0.14	−0.06	−0.04
	GitterHQ_stem	0.01	0.09	−0.04	−0.08	0.04
	Flutter_lemma	−0.13	−0.10	−0.06	−0.14	−0.06
	Laravel_lemma	−0.03	−0.03	0.15	0.20	0.00
	GitterHQ_lemma	0.05	0.14	0.19	0.22	0.24
c_{npmi}	Flutter_stem	−0.08	−0.04	−0.03	−0.08	−0.08
	Laravel_stem	−0.06	−0.04	−0.02	−0.05	−0.01
	GitterHQ_stem	−0.03	0.00	−0.01	−0.01	0.02
	Flutter_lemma	−0.05	−0.05	−0.03	−0.05	−0.03
	Laravel_lemma	−0.04	−0.01	0.02	0.01	−0.02
	GitterHQ_lemma	−0.01	0.01	0.03	0.03	0.06
u_{Mass}	Flutter_stem	−5.31	−3.65	−4.29	−5.56	−5.00
	Laravel_stem	−5.18	−4.18	−4.88	−3.93	−2.70
	GitterHQ_stem	−3.25	−3.77	−4.13	−5.28	−3.69
	Flutter_lemma	−3.30	−2.86	−3.43	−4.03	−3.20
	Laravel_lemma	−3.41	−2.15	−2.26	−3.27	−2.37
	GitterHQ_lemma	−4.03	−2.82	−3.01	−2.99	−2.91

4. Results

4.1. Topic coherence metrics

As described in Section 3.4, we calculated four topic coherence metrics for all combinations of models and corpus, including the stemmed and lemmatized corpora. We provide all scores at topic level online[23]. Tables 4–6 show the detailed results of these metrics (at model level) for each combination of model and corpus.

4.1.1. Statistical significance of coherence metrics

To check if our automated topic coherence scores (based on the metrics described earlier) were different or similar between models, we performed four Kruskal–Wallis' tests[34] (followed by post-hoc Dunn's tests with Bonferroni correction[35]) in model-level scores and another four in topic-level scores (i.e., two tests for each metric). For these tests, we considered the scores obtained with both lemmatized and stemmed corpora. The results of Kruskal–Wallis' tests are:

- c_a: *p-value* of 0.10 at model level and *p-value* of 6.27 at topic level.
- c_p: *p-value* of 0.67 at model level and *p-value* of 0.03 at topic level.
- c_{npmi}: *p-value* of 0.61 at model level and *p-value* of 0.00 at topic level.
- u_{Mass}: *p-value* of 0.09 at model level and *p-value* of 0.06 at topic level.

We found that there are no statistically significant differences between models' topic coherence scores at model-level (i.e., all *p-values* were higher than 0.05). At the topic-level, we found statistically significant differences between models based on the scores of metrics c_p and c_{npmi} (i.e., *p-values* were lower than 0.05). After applying Dunn's test with Bonferroni correction to topic-level scores of c_p and c_{npmi}, we found that models BTM and GPU-PDMM had statistically significant different c_p scores. For c_{npmi} scores, we identified statistically significant differences between the models BTM and DMM, BTM and GPU-PDMM, BTM and LDA. This means that BTM was outperformed by GPU-PDMM based on both of these metrics, and outperformed by DMM and LDA based on c_{npmi} scores.

Considering the results of our Kruskal–Wallis' tests, we understand that even though we found higher/lower scores at model level, the performance of our short text topic models was similar. Therefore, we describe the best performing models based on our automated metrics acknowledging that models' topic quality were not statistically significant different.

[34] Kruskal–Wallis' test is a non-parametric method for testing whether samples are originated from the same distribution. It extends the Mann–Whitney U test to more than two groups. The null hypothesis of the Kruskal–Wallis' test is that the mean ranks of the groups are the same (Kruskal and Wallis, 1952).

[35] Dunn's test performs pairwise comparisons between independent groups and tells which groups are statistically significantly different Statology, 2020.

4.1.2. Best performing models based on automated coherence metrics

By checking the top-10 scores of each metric (considering both lemmatized and stemmed corpora), we identified the highest model-level scores by group of corpus. Tables 4–6 show resulting metric scores. Note that the highest scores were frequently found in topics generated with lemmatized corpora:

- For the small corpora, WNTM had the highest scores for Android_lemma, while BTM had high scores with WebpackDocs_lemma, and GPU-PDMM with ConsenSys_lemma.
- For the medium-sized corpora, GPU-PDMM had some of the highest scores for Jenkinsci_lemma, Locomotive_lemma and SpringSecurity_lemma. DMM and WNTM also had high scores with Locomotive_lemma.
- For the large corpora, LDA had the highest scores for Flutter_lemma, while GPU-PDMM had some of the highest scores for Laravel_lemma and GitterHQ_lemma. Regarding GitterHQ_lemma, all models had some of the top-10 highest scores based on our four metrics.

As can be seen above, we did not find a single model for each group of corpus size that consistently had the highest scores. For example, WNTM, BTM and GPU-PDMM had high scores, in at least two metrics, with different small corpora while grid-searched LDA had high scores with Flutter, our largest corpus in terms of vocabulary and number of documents. Since GPU-PDMM was the model that had the highest scores with at least one corpus of each group of corpus size, we can consider that GPU-PDMM identified some of the most coherent topics across our corpora when modeling ten topics.

Additionally, we counted how many of top-10 highest scores occurred for each model to check which model had most of the highest scores considering all metrics for both lemmatized and stemmed corpora. From a total of 120 top-10 scores (being 40 for each group of corpus size), we found the following:

- For the small corpora: BTM had 8, DMM had 10, GPU-PDMM had 10, WNTM had 6 and LDA had 6.
- For the medium-sized corpora: BTM had 4, DMM had 9, GPU-PDMM had 12, WNTM had 8 and LDA had 7.
- For the large corpora: BTM had 2, DMM had 11, GPU-PDMM had 8, WNTM had 8 and LDA had 11.
- In total, BTM had 14 top-10 scores, DMM and GPU-PDMM had 30, WNTM had 22 and LDA had 24.

These totals show that, in general, DMM and GPU-PDMM frequently had some of the highest coherence scores across our corpora, followed by LDA. Even though BTM and WNTM had high scores in some combinations of model and corpus, they less frequently had top-10 scores in comparison to the other models.

4.1.3. Comparison between LDA and short text topic models

We compare the performance of our selection of short text topic models with LDA, the most popular topic model in software engineering studies (Silva et al., 2021). This is because others have argued that LDA may not be appropriate to generate topics from short text due to its sparsity issues (Lin et al., 2014; Agrawal et al., 2018). With this comparison, we aimed at identifying the models that had higher scores than LDA based on model-level metric scores. At topic-level, as described in Section 4.1.1, LDA outperformed BTM based on c_{npmi}.

- Fig. 5 shows the scores obtained with our *group of small corpora* considering both lemmatized and stemmed corpora. We found the following for each of the four coherence metrics:
  - c_a: DMM and GPU-PDMM had higher scores than LDA in general. However, DMM and GPU-PDMM had more spread out value points than LDA. This means that even though LDA had lower scores than these models, DMM and GPU-PDMM did not consistently generate coherent topics.
  - c_p: BTM had some scores higher than LDA scores for this metric. On the other hand, LDA was the second best performing model, based on its highest score, in comparison to our short text topic models.
  - c_{npmi}: DMM and GPU-PDMM also had higher scores than LDA in this metric.
  - u_{Mass}: GPU-PDMM, WNTM and DMM had higher scores than LDA. On the other hand, WNTM had the most consistent model in value points and it did not had the lowest score for this metric.
- Fig. 6 shows the scores obtained with our *group of medium-sized corpora* considering both lemmatized and stemmed corpora. We found the following:
  - c_a: LDA had the highest scores in this metric. DMM was the second best performing model.
  - c_p: LDA had the highest scores in this metric. GPU-PDMM was the second best performing model.
  - c_{npmi}: LDA also had the highest scores in this metric. However, GPU-PDMM had similarly high scores as LDA.
  - u_{Mass}: GPU-PDMM had higher scores than LDA in general. DMM also performed well based on this metric by having with the third best scores.
- Fig. 7 shows the scores obtained with our *group of large corpora* considering both lemmatized and stemmed corpora. We found the following:
  - c_a: LDA had the highest scores in this metric. GPU-PDMM and WNTM were the second and third best performing models.
  - c_p: LDA had the highest scores in this metric. WNTM and GPU-PDMM were the second and third best performing models.
  - c_{npmi}: LDA also had the highest scores in this metric. GPU-PDMM and WNTM were the second and third best performing models.
  - u_{Mass}: DMM and GPU-PDMM had higher scores than LDA in this metric.

In summary (acknowledging the statistical significance between models' scores detailed in Section 4.1.1), our short text topic models mostly had higher scores than LDA when modeling topics from our group of small corpora. For the medium-sized and large corpora, LDA only had lower scores in comparison to our short text topic models with u_{Mass}, a metric that attempts to confirm that the model learned topics known to be in the corpus used in the modeling (see details about this metric in Section 2.4.3). We also found that LDA generated more coherent topics with lemmatized corpora than with stemmed corpora, which had a smaller vocabulary size. This means that grid-searched LDA generates more coherent topics than short text topic models when modeling few topics (i.e., ten topics) from large corpora in vocabulary size.

4.1.4. Comparison of stemmed and lemmatized corpora

We used the topic coherence metrics to compare our stemmed and lemmatized corpora (see details about these normalization techniques in Section 3.2). This was to select topics for human assessment, as described in Section 3.5. Fig. 8 shows boxplot charts for each metric with the scores (at model level) of all lemmatized and stemmed corpora. We checked if there is a statistically significant difference between these model-level scores using Kruskal–Wallis' tests[34]. The results are c_a: *p-value* of 0.00, c_p: *p-value* of 4.78, c_{npmi}: *p-value* of 0.005, and u_{Mass}: *p-value* of 0.00. This means that lemmatized corpora are statistically significantly different from stemmed corpora based on automated metrics c_a, c_{npmi} and u_{Mass}. We can conclude that, in general, lemmatized corpora generated more coherent topics than stemmed corpora with

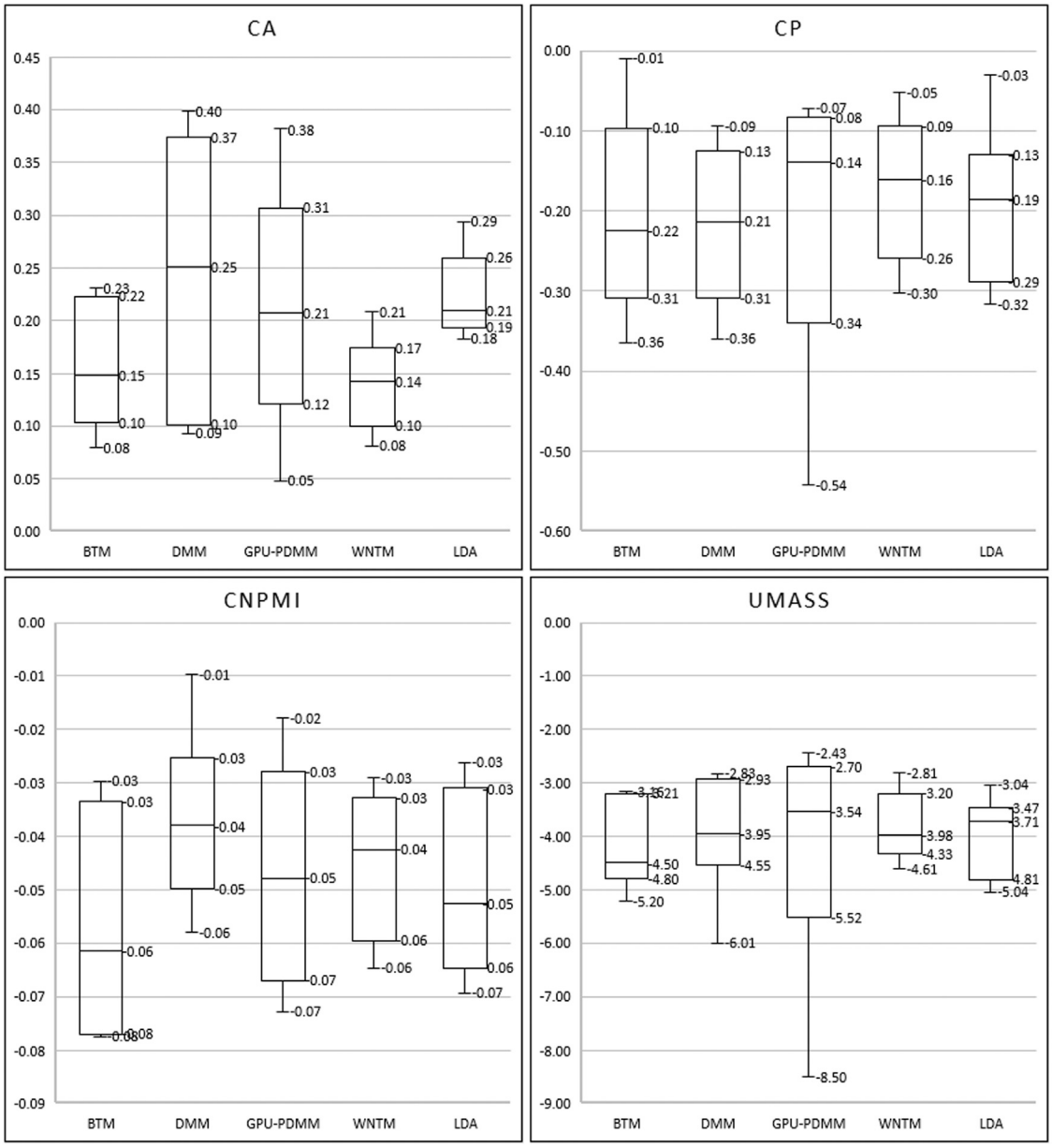

Fig. 5. Model scores with small corpora by metric.

our topic models. We also found that the top-10 best scores for each metric occurred more frequently among corpora that was lemmatized. The top-10 c_a and c_p scores occurred for lemmatized combinations of model and corpus, such as LDA & GitterHQ_lemma (0.52) and LDA & SpringSecurity_lemma (0.27). For c_{npmi}, nine scores out of the top-10 occurred for lemmatized combinations of model and corpus, and for u_{Mass}, six scores out of the top-10. Therefore, we selected the lemmatized corpora for the human-based assessments of topic quality. Additionally, with lemmatization we retained the readable form of the words from our original data sets. With readable words in topics, participants involved in the evaluation of topics would be able to understand and assess topics easier than with stemmed words (words without prefixes and suffixes).

4.2. Word and topic intrusion

The complete description of each of the 18 participants (identified as "IT.[number]") of word and topic intrusion tasks is shown in Table 7 (the categories regarding familiarity with technologies and purpose of using instant messaging are described in Section 3.5.3). The responses of all participants for all tasks are available online[23].

4.2.1. Model Precision (MP) and Topic Log Odds (TLO)

Table 8 shows the scores obtained with word and topic intrusion tasks, MP and TLO respectively. Regarding the performance of combinations of model and corpus based on **word intrusion tasks**, we found that DMM & Flutter_lemma (0.80) had the highest score in comparison to the other combinations, while GPU-PDMM & Android_lemma and BTM & Android_lemma (both 0.17) had the lowest scores. For the **topic intrusion tasks**, we found that WNTM & Flutter_lemma (−5.39), WNTM & Jenkinsci_lemma and GPU-PDMM & Flutter_lemma (both −5.82) had the highest scores for this metric, while DMM & Flutter_lemma (−36.92) had the lowest score.

With these results, we can understand that a larger corpus (and consequently a larger vocabulary) may influence the quality of topics generated with short text topic models because of its more distinct words. However, we found that DMM had contradicting results with Jenkinsci_lemma and Flutter_lemma. Combined with these corpora, DMM

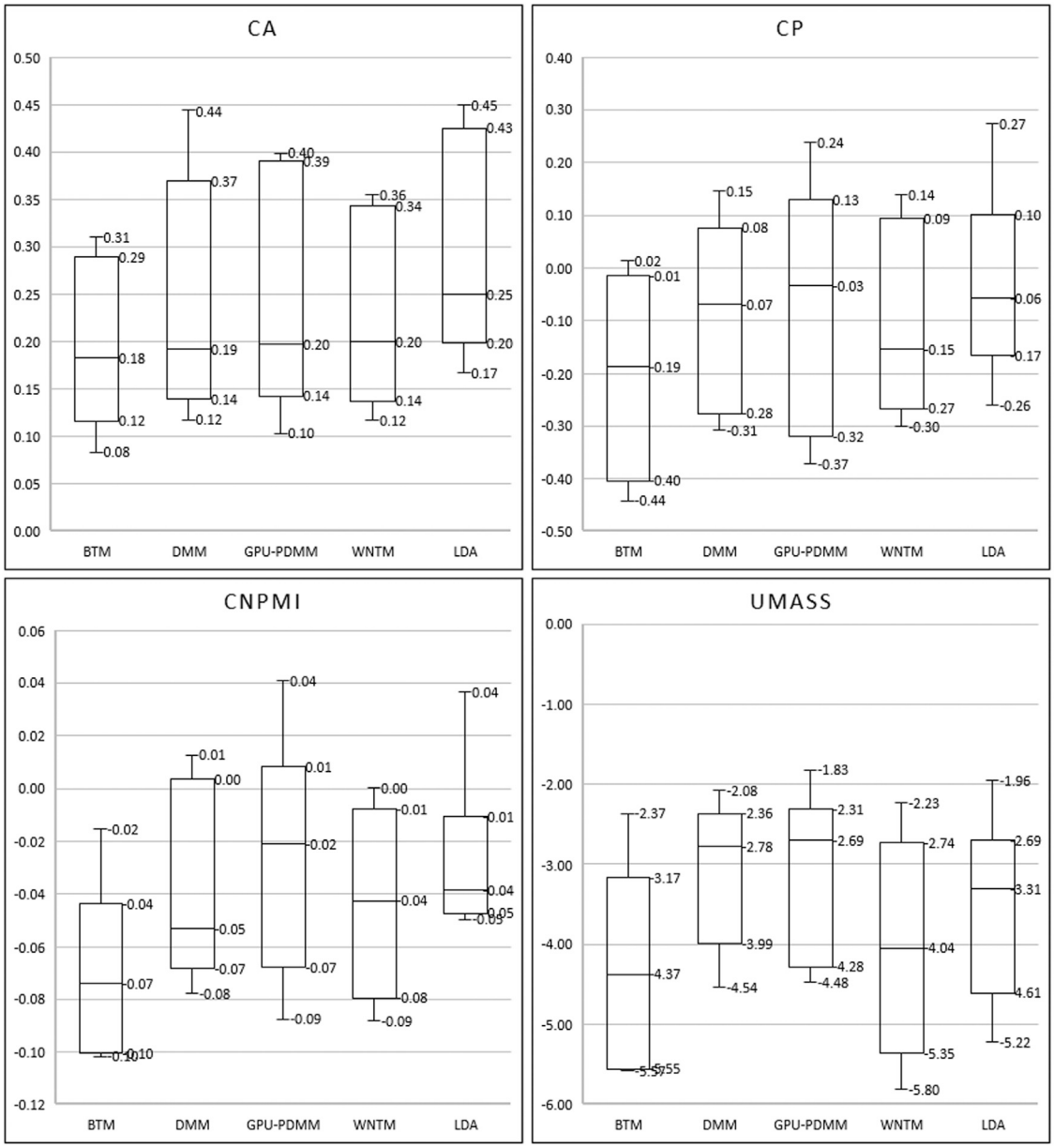

Fig. 6. Model scores with medium-sized corpora by metric.

generated coherent topics in terms of words consistency (MP), but poor perceived association between topics and documents (TLO) since intruder topics were understood as being associated to the documents presented in tasks.

4.2.2. Best performing models based on intrusion tasks

When analyzing the results of DMM, we found that it was harder for participants to identify *intruder topics* than to identify *intruder words*, which means that DMM did not perform consistently well in the intrusion tasks. The most consistent model was WNTM, which had some of the best scores for both MP and TLO with all corpora, but especially with medium-sized and large corpora (Jenkinsci_lemma and Flutter_lemma). The same consistency was observed in GPU-PDMM with Flutter_lemma, but not with Android_lemma and Jenkinsci_lemma. For example, GPU-PDMM & Android_lemma had a low score for MP (0.17) and a high score for TLO (−9.05), which means that topics of this combination are less distinct from one another (i.e. less related words) while the perceived association between topics and documents is clearer. BTM, like WNTM, had consistent scores for both MP and TLO metrics; however, in comparison to the other models, it did not generate some of the best topics. Therefore, we considered that WNTM was the best performing model on intrusion tasks.

4.2.3. Participants feedback on intrusion tasks

Only three participants responded our open-ended question for general feedback. Participant IT.13 suggested "I think this research should be done using keywords from other languages besides dart". This means that participant IT.13 understood that most of the word intrusion tasks were about dart (a programming language for web and mobile client development), which is a reasonable perception since their questionnaire contained only topics generated with our Jenkinsci corpus. The other two participants acknowledged how their experience and familiarity with the technologies discussed in our corpora influenced their responses. IT.09 acknowledged that "The tasks to identify the odd words were difficult for me, because from my perspective, I could always connect all of them together in a context. My guess is that this happens because I work with CI CD pipelines, review PRs, develop scripts and new android features (using Java and Kotlin)

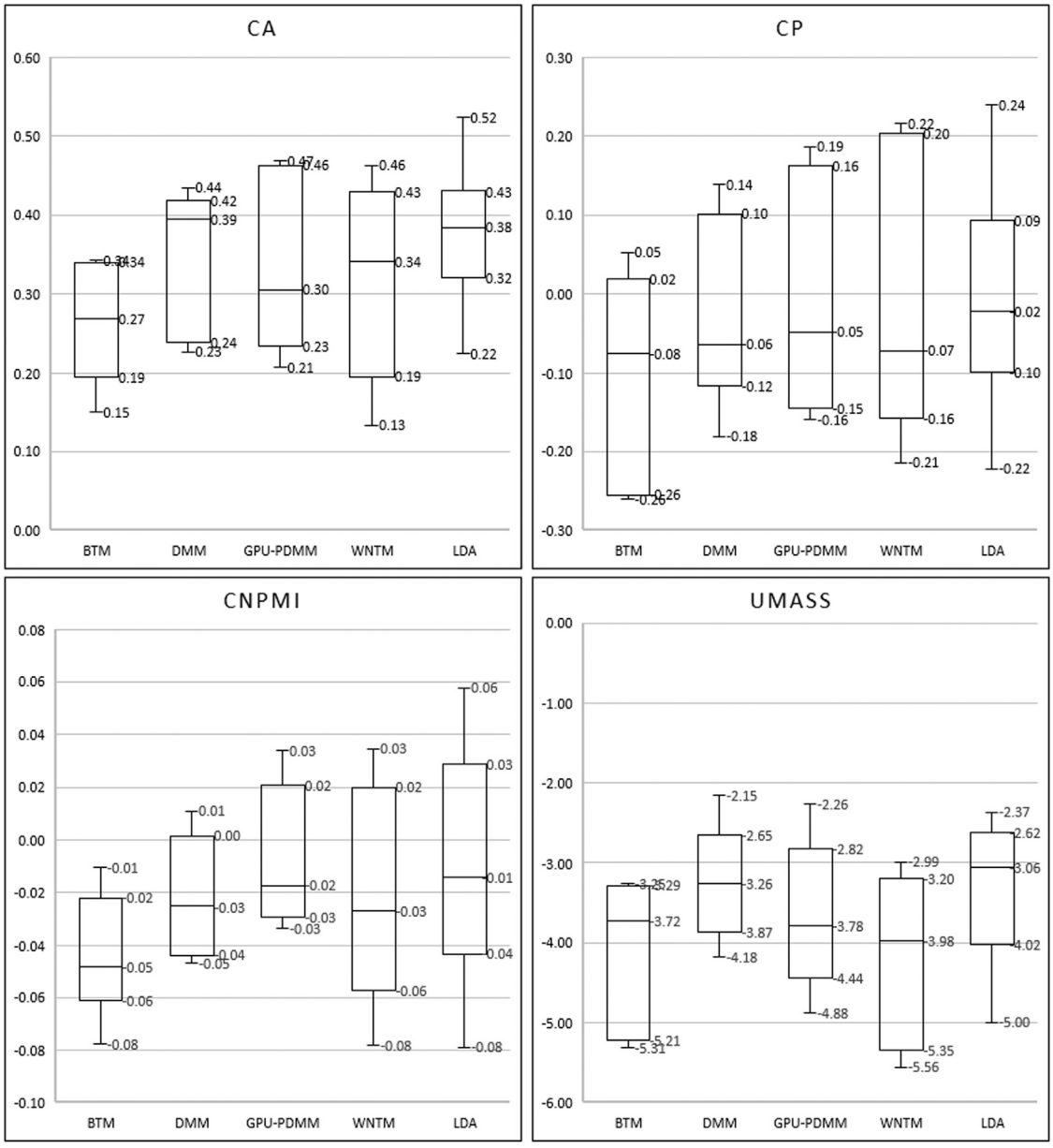

Fig. 7. Model scores with large corpora by metric.

and coordinate meetings with squads in a daily basis in my work, so all these subjects are very familiar to me, and they can always be connected. (...)". This indicates that this participant could *potentially always* find a connection between words in clusters if no specific context of the appearance of words is provided (we discuss this further in Section 5.5). IT.17 mentioned "I'm not updated about some tools of Continuous Integration and Infrastructure tools". On the other hand, according to the background questions, IT.17 used Jenkins more often than the other technologies (occasionally developed Android apps; had heard about but did not use Flutter, see Table 7).

4.3. Topic naming

We asked eight participants, who were not involved in the word and topic intrusion tasks, to name the topics with their own words (see details in Section 3.5.2). The description of these participants (identified as "TN.[number]") is shown in Table 9 (the categories regarding familiarity with technologies and purpose of using instant messaging are described in Section 3.5.3). The results of the topic naming tasks are available online[23].

4.3.1. Number of topics named and average ratio of names per topic

Table 10 shows how many topics (t_{named}) out of ten were named, and the average ratio of names per topic (a_{ratio}) by combination of model and corpus. Considering both t_{named} and a_{ratio}, we found that DMM & Flutter_lemma (t_{named} = 10 and a_{ratio} = 0.70), GPU-PDMM & Jenkinsci_lemma, GPU-PDMM & Flutter_lemma (both with t_{named} = 10 and a_{ratio} = 0.65) and DMM & Jenkinsci_lemma (t_{named} = 10 and a_{ratio} = 0.60) were the best performing combinations of model and corpus. GPU-PDMM & Android_lemma (t_{named} = 7 and a_{ratio} = 0.35) was the worst performer in comparison to the other combinations. We also found that WNTM & Jenkinsci_lemma had "conflicting" scores (t_{named} = 9 and a_{ratio} = 0.43), which means that most topics of this combination were named, but few were named by two or more participants.

4.3.2. Compatibility between names assigned to topics

As mentioned in Section 3.5.2, we analyzed the semantic compatibility of the names assigned by participants for each combination

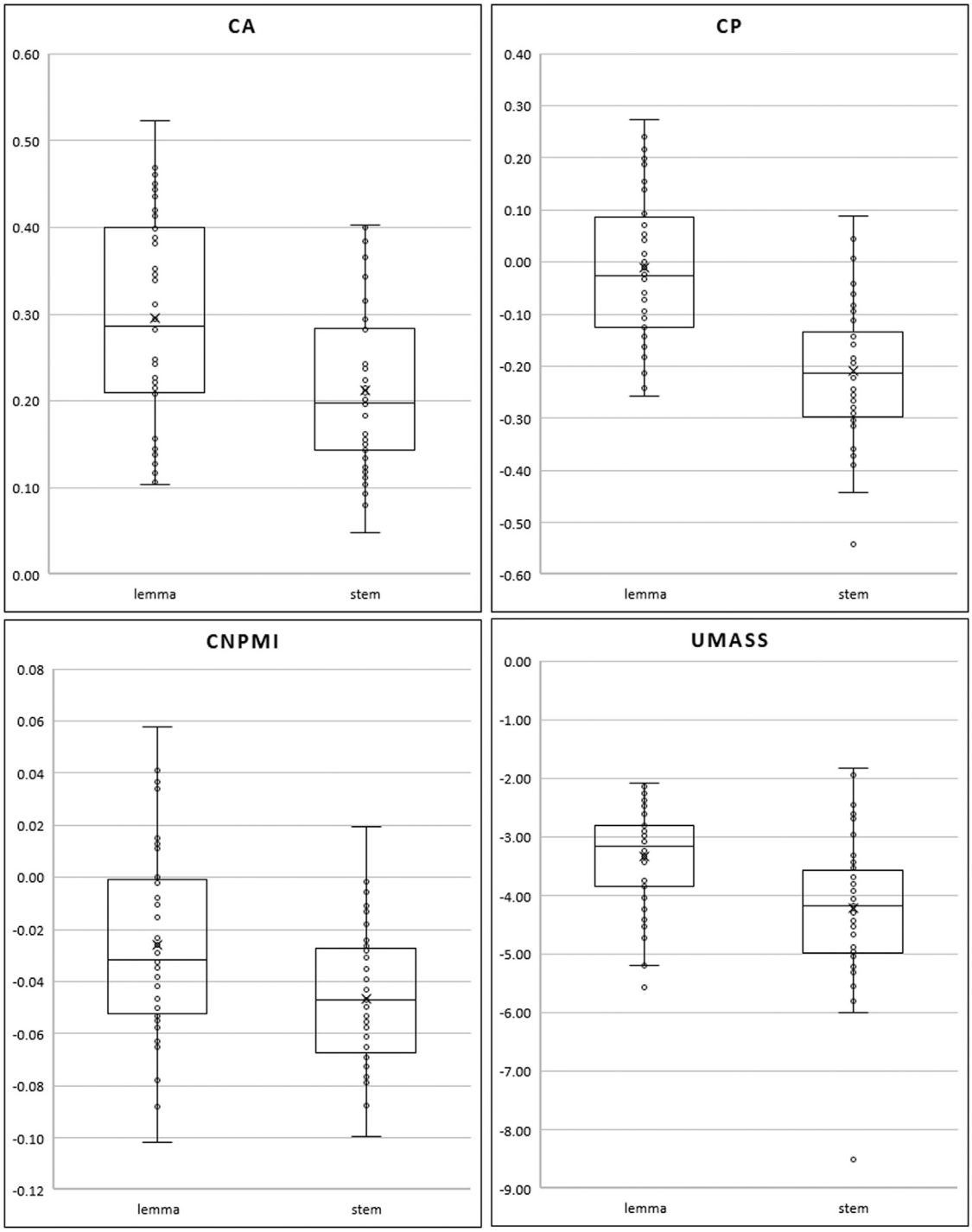

Fig. 8. Lemmatized and stemmed corpora scores by metric.

of model and corpus. Compatible names are highlighted in gray in Tables 13–16 in Appendix C. Regarding each topic model we found the following:

- BTM: In general, we found few compatible names among the topics generated with this model (see Table 13 in Appendix C). From a total of 14 topics named by at least two participants, only four were compatible. On the other hand, few BTM topics could be named by more than one participant.
- DMM: The topics generated with this model had more compatible names than the other models (see Table 14 in Appendix C). From a total of 25 topics named by at least two participants, 13 were compatible.
- GPU-PDMM: We also found few compatible names to the topics generated with this model (see Table 15 in Appendix C). From a total of 23 topics named by at least two participants, only seven were compatible. We found that, unlike the models, one participant (TN.04) could not name any of the topics generated with GPU-PDMM & Android_lemma, even though this participant had developed Android apps a few times, according to their experience. The same participant provided just one or two names to the topics generated with GPU-PDMM & Jenkinsci_lemma and GPU-PDMM & Flutter_lemma. We considered this an exceptional

Table 7
Description of participants of the intrusion tasks.

Participant	Sub-survey	Experience (in years)	Level of training	Familiarity with technologies[a]			Use of instant messaging	
				Android	Jenkins	Flutter	Purpose[a]	Frequency
IT.01	13	17	Post-graduate	3	4	4	3	Sometimes (e.g., several times per week)
IT.02	15	11	Post-graduate	3	5	1	6	Always (e.g., it is part of my daily work)
IT.03	18	10	Post-graduate	2	2	2	1, 2, 3, 6	Always (e.g., it is part of my daily work)
IT.04	16	16	Higher education	3	1	1	4, 6	Always (e.g., it is part of my daily work)
IT.05	14	5	Higher education	3	2	2	1, 2, 6	Always (e.g., it is part of my daily work)
IT.06	17	11	Masters	3	3	2	1, 3, 6, 8[b]	Always (e.g., it is part of my daily work)
IT.07	7	6	Higher education	3	2	2	2, 6, 7, 8[c]	Always (e.g., it is part of my daily work)
IT.08	3	6	Higher education	3	2	2	3, 5, 6	Always (e.g., it is part of my daily work)
IT.09	10	5	Higher education	5	5	3	1, 5, 6, 8[d]	Always (e.g., it is part of my daily work)
IT.10	9	15	Masters	2	2	1	1–7	Always (e.g., it is part of my daily work)
IT.11	2	4	Higher education	3	2	2	1, 2, 6	Sometimes (e.g., several times per week)
IT.12	6	5	Higher education	3	1	1	1–7	Always (e.g., it is part of my daily work)
IT.13	11	5	Post-graduate	4	2	5	1, 2, 6	Always (e.g., it is part of my daily work)
IT.14	5	3	Higher education	3	2	1	5, 6	Always (e.g., it is part of my daily work)
IT.15	1	14	Higher education	3	2	2	1–3, 5–7	Always (e.g., it is part of my daily work)
IT.16	12	7	Post-graduate	4	2	2	3, 6	Always (e.g., it is part of my daily work)
IT.17	8	14	Post-graduate	4	3	2	1, 3, 4, 6	Occasionally (e.g., several times per month)
IT.18	4	6	Higher education	3	2	2	1, 2, 6	Always (e.g., it is part of my daily work)

[a] See the detailed categories in Section 3.5.3.
[b] "Corporate communications".
[c] "To keep track of development logs, e.g., build/deploy logs, bug auto reporting and other similar things".
[d] "I use slack and MS Teams a lot for general work related subjects".

Table 8
Scores by intrusion tasks.

Metric	Corpus	BTM	DMM	GPU-PDMM	WNTM
MP	Android_lemma	0.17	0.27	0.17	0.33
	Jenkinsci_lemma	0.43	0.67	0.67	0.67
	Flutter_lemma	0.60	0.80	0.70	0.67
TLO	Android_lemma	−14.24	−19.22	−9.05	−22.42
	Jenkinsci_lemma	−11.55	−30.43	−11.47	−5.82
	Flutter_lemma	−16.50	−36.92	−5.82	−5.39

case because the other three participants (TN.05, TN.07 and TN.08), who also analyzed topics from GPU-PDMM, provided names for more than two topics across the corpora.

- WNTM: Like BTM, few topics generated with this model had compatible names (see Table 16 in Appendix C). From a total of 14 topics named by at least two participants, only four were compatible.

We also observed that sometimes participants assigned the same name to different topics generated with a model. For example, TN.02 assigned "devops" to 5 of the 10 topics generated with BTM & Jenkinsci_lemma. This means that the model generated topics that are understandable for humans, but which are not distinct enough (based on the words in the topics) to be assigned with different names. Sometimes different words can refer to the same high-level concept captured by the topic name. We found the same name for at least two topics in our four models. BTM and WNTM in combination with Jenkinsci_lemma, and GPU-PDMM in combination with Jenkinsci_lemma and Flutter_lemma had more than three topics assigned with the same name.

4.3.3. Best performing models based on topic naming

We found that the best performing combinations of model and corpus were DMM & Flutter_lemma (10 topics named, 0.7 average ratio of names per topic and 6 topics with compatible names) and DMM & Jenkinsci_lemma (10 topics named, 0.6 average ratio of names per topic and 4 topics with compatible names). The combinations of model and corpus GPU-PDMM & Flutter_lemma (10 topics named, 0.65 average ratio of names per topic and 3 topics with compatible names) and GPU-PDMM & Jenkinsci_lemma (10 topics named, 0.65 average ratio of names per topic and 3 topics with compatible names) also performed well. In summary, we conclude that DMM and GPU-PDMM were the best performing models based on topic naming tasks, even though the topics generated with GPU-PDMM were less distinct to humans (i.e., were sometimes assigned the same name) in comparison to the topics generated with DMM.

4.3.4. Participant feedback on naming topics

Two participants mentioned in our final open-ended question that they were not so familiar with Flutter, which may have influenced their responses. TN.01 mentioned "Maybe it didn't help that I haven't used flutter or built a native android app but that was really hard", while TN.08 acknowledge that "I could recognize some stuff about java/flutter but since I have never studied or worked with them it was a little bit harder for me to recognize patterns among the words". On the other hand, the topics generated with Flutter_lemma were the most named topics across all models. In terms of compatibility, *topic04* of DMM & Flutter_lemma and *topic01* of WNTM & Flutter_lemma were the topics that received four compatible names (one name per participant), as shown in Tables 14 and 16 in Appendix C.

4.4. Overall model performance

4.4.1. Comparison of short text topic models

Table 11 shows an overview of the performance of all combinations of models and corpora. The averages in Table 11 are the mean values between the scores of our three corpora (e.g., c_a: 0.16 for BTM is the mean between model-level c_a scores of Android_lemma, Jenkinsci_lemma and Flutter_lemma in combination with BTM). The range of values for each of our metrics are described in Sections 3.4, 3.5 and 3.5.2. In Section 4.1.1, we acknowledge that, even though we discuss best performing models, there is no statistical significant differences between models' automated metric scores.

Therefore, when comparing the scores of our metrics by model, we found that:

- *BTM* did not have the highest scores for any of our metrics. The results of our Kruskal–Wallis' tests with topic-level automated metrics' scores (c_p and c_{npmi}) confirmed that BTM was outperformed (see details in Section 4.1.1).
- *DMM* was the best performing model for topic naming (t_{named}: 10, a_{ratio}: 0.62 and four topics with compatible names) and word intrusion tasks (*MP*: 0.58). This means that the topics generated by DMM are distinct and more semantically meaningful for humans.

Table 9
Description of participants of topic naming.

Participant	Sub-survey	Experience (in years)	Level of training	Familiarity with technologies[a]			Use of instant messaging	
				Android	Jenkins	Flutter	Purpose[a]	Frequency
TN.01	3	3	Honors	2	4	2	1, 3, 6, 7	Always (e.g., it is part of my daily work)
TN.02	1	5	Higher education	3	2	2	1, 2, 3, 6	Always (e.g., it is part of my daily work)
TN.03	1	6	Higher education	2	2	2	2, 3, 6	Always (e.g., it is part of my daily work)
TN.04	4	6	Higher education	3	4	2	6	Always (e.g., it is part of my daily work)
TN.05	2	15	PhD	3	4	2	1, 3, 5, 6	Sometimes (e.g., several times per week)
TN.06	6	3	Higher education	2	2	2	6	Always (e.g., it is part of my daily work)
TN.07	5	3	Masters	2	2	1	1–7	Always (e.g., it is part of my daily work)
TN.08	2	6	Post-graduate	3	2	2	1, 6	Always (e.g., it is part of my daily work)

[a] See the detailed categories in Section 3.5.3.

Table 10
Topic naming by model and corpus.

Metric	Corpus	BTM	DMM	GPU-PDMM	WNTM
t_{named}[a]	Android_lemma	8	9	7	8
	Jenkinsci_lemma	8	10	10	9
	Flutter_lemma	9	10	10	7
a_{ratio}[b]	Android_lemma	0.38	0.55	0.35	0.35
	Jenkinsci_lemma	0.35	0.60	0.65	0.43
	Flutter_lemma	0.53	0.70	0.65	0.43

[a] Number of topics named (out of 10) by corpus.
[b] Average ratio of names assigned to a topic.

Table 11
Scores of metrics for each model and corpus.

Metrics	Corpus	BTM	DMM	GPU-PDMM	WNTM
c_a	Android_lemma	0.14	0.10	0.14	0.11
	Jenkinsci_lemma	0.13	0.15	0.16	0.12
	Flutter_lemma	0.21	0.23	0.24	0.22
	Average	0.16	0.16	0.18	0.15
c_p	Android_lemma	−0.01	−0.09	−0.07	−0.05
	Jenkinsci_lemma	−0.18	0.01	0.07	−0.21
	Flutter_lemma	−0.13	−0.10	−0.06	−0.14
	Average	−0.11	−0.06	−0.02	−0.13
c_{npmi}	Android_lemma	−0.06	−0.06	−0.07	−0.03
	Jenkinsci_lemma	−0.10	−0.08	−0.03	−0.09
	Flutter_lemma	−0.05	−0.05	−0.03	−0.05
	Average	−0.07	−0.06	−0.04	−0.06
u_{Mass}	Android_lemma	−3.16	−3.85	−3.74	−2.81
	Jenkinsci_lemma	−5.57	−4.54	−2.77	−5.20
	Flutter_lemma	−3.30	−2.86	−3.43	−4.03
	Average	−4.01	−3.75	−3.31	−4.01
MP	Android_lemma	0.17	0.27	0.17	0.33
	Jenkinsci_lemma	0.43	0.67	0.67	0.67
	Flutter_lemma	0.60	0.80	0.70	0.67
	Average	0.40	0.58	0.51	0.56
TLO	Android_lemma	−14.24	−19.22	−9.05	−22.42
	Jenkinsci_lemma	−11.55	−30.43	−11.47	−5.82
	Flutter_lemma	−16.50	−36.92	−5.82	−5.39
	Average	−14.10	−28.86	−8.78	−11.21
t_{named}	Android_lemma	8	9	7	8
	Jenkinsci_lemma	8	10	10	9
	Flutter_lemma	9	10	10	7
	Average	8	10	9	8
a_{ratio}	Android_lemma	0.38	0.55	0.35	0.35
	Jenkinsci_lemma	0.35	0.60	0.65	0.43
	Flutter_lemma	0.53	0.70	0.65	0.43
	Average	0.42	0.62	0.55	0.40
Compatible names	Android_lemma	3	3	1	2
	Jenkinsci_lemma	0	4	3	0
	Flutter_lemma	1	6	3	2
	Average	1	4	2	1

- *GPU-PDMM* had high scores in most automated coherence metrics and human-based metrics (e.g., MP: 0.51, TLO: −8.78). Therefore, the topics generated with this model are both distinct, descriptive and understandable enough for humans. GPU-PDMM also generated topics that are internally consistent (based on u_{Mass}: −3.31).
- *WNTM* had the lowest performance for topic naming metrics (e.g., a_{ratio}: 0.4 and one topic with compatible names), but the most consistent performance in intrusion task metrics. WNTM had also the second highest score for the automated metric c_{npmi} (−0.06). This may indicate that WNTM generates externally coherent (topics compared to general English text) and consistent topics that are not easily named by humans.

4.4.2. Correlation between metrics

We calculated the correlation between the scores of each of our four topic coherence metrics (c_a, c_p, c_{npmi} and u_{Mass}) and human assessment metrics RTI (based on word intrusion tasks) and RN (based on topic naming tasks). We compared our metrics in pairs (automated x human-based), considering the scores of all combinations of model and corpus. We also checked if the scores our human-based metrics RTI and RN align. Our hypothesis is that automated topic coherence metrics are similar to how humans would judge topics (based on human-based topic coherence metrics); therefore, using either an automated or a human-based metric would be appropriate for assessing topic quality of short text topic models applied to instant messages.

Spearman's Rank correlation coefficients and *p-values* are described in the following. We marked with * the most statistically significant correlations, since they were lower than 0.05.

- c_a and RTI: correlation = 0.16, *p-value* = 0.08
- c_p and RTI: correlation = 0.09, *p-value* = 0.30
- c_{npmi} and RTI: correlation = 0.06, *p-value* = 0.48
- u_{Mass} and RTI: correlation = −0.47, *p-value* = 4.49
- c_a and RN: correlation = 0.25, *p-value* = 0.006*
- c_p and RN: correlation = 0.17, *p-value* = 0.06
- c_{npmi} and RN: correlation = 0.07, *p-value* = 0.45
- u_{Mass} and RN: correlation = −0.29, *p-value* = 0.001*
- RN and RTI: correlation = 0.26, *p-value* = 0.003*

We found statistically significant low positive correlations between c_a and RN, and between RN and RTI. We also found a statistically significant low negative correlation between u_{Mass} and RN. This means that the metric RN (based on topic naming) did not align with the metrics c_a, u_{Mass}, and RTI (based on intrusion tasks). Considering that the other correlation coefficients were low and did not present statistical significance (*p-value* > 0.05), there is no evidence that automated and human-based metrics are similar when measuring topic quality.

5. Discussion

5.1. Summary of findings

As described in Section 2.3, short text topic models can be divided into three types: *Dirichlet multinomial mixture*-based, *Global word co-occurrences*-based and *Self-aggregation*-based (Qiang et al., 2022). For

our study, we selected topic models based on *Dirichlet multinomial mixture* (DMM and GPU-PDMM) and based on *Global word co-occurrences* (BTM and WNTM).

In the context of our study, we found that short text topic models based on *Dirichlet multinomial mixture* generated topics that are more meaningful to humans, while models based on *Global word co-occurrences* generate topics that are coherent in comparison to general English text. Regarding the corpora size (in number of messages and in vocabulary size), we found that models based on *Global word co-occurrences* such as WNTM, performed well with our small corpus, while models based on *Dirichlet multinomial mixture* such as GPU-PDMM, performed well with our medium-sized and large corpora (see details in Section 4.4).

Considering the correlation between our automated topic coherence metrics and human assessment (based on metrics RTI and RN), we found that there is no evidence that our automated and human-based metrics align. We argue that these metrics complement each other since automated metrics evaluate coherence based on the comparison of topics (and the statistical distribution of terms) with general text while human assessment measures the perception of humans on the meaning of topics. Therefore, further studies may consider using intrusion tasks or topic naming tasks *in addition* to automated coherence metrics for further interpretation of topics.

Even though GPU-PDMM had the best overall performance across short text topic models, we cannot suggest one single model to be used with developer instant messaging communication. Additionally, as we found that model-level scores were not statistically significant different between models based on automated topic coherence metrics (see details in Section 4.1.1). We also believe that we could not identify such best performing model because we tried to cover different aspects of topic quality with: automated topic coherence, intrusion tasks and topic naming. Each of these strategies measured quality in terms of:

1. Automated metrics: "external" coherence based on the comparison between word pairs of topics and generic English files (c_a, c_p and c_{npmi}); and "internal" coherence based on the comparison between word pairs from the corpus used to infer topics (u_{mass}).
2. Intrusion tasks: how well topics match human concepts (MP) and how well a topic model assigns topics to documents (TLO).
3. Topic naming: an open-ended task focused on evaluating how comprehensible topics are, so participants would be able to create a name to summarize their concept (based on t_{named}, a_{ratio} and number of compatible names).

This means that an ideal best performing model would have to generate topics from all of our corpora with the highest scores in all of these metrics. GPU-PDMM, our overall high performing model, did not consistently have the highest scores in all metrics considering all of our corpora. Instead, we found that DMM generated the most comprehensible topics to humans based on MP, t_{named}, a_{ratio} and in number of compatible names. Another finding was that grid-searched LDA generated more "externally" coherent topics than our short text topic models when modeling few topics from larger corpora of short text. Therefore, tasks/applications that require more human comprehensive topics would benefit from DMM topics.

5.2. Practical use of topic models

To demonstrate the practical use of short text topic models, we selected one of our four models (GPU-PDMM) and applied it to the messages of 87 public Gitter chat rooms (the same rooms mentioned in Section 3.1). The goal was to check how the topics *generated automatically* with our selected model relate to the themes *identified manually* by Costa Silva et al. (2022). The topics and the themes identified for each chat room (Costa Silva et al., 2022) are available online[23]. In this online material we also include the runtime for each data set (i.e. chat room), which shows that the runtime of the topic model increases with the size of the data set.

We selected GPU-PDMM for this exercise as it was the overall best performing model. Additionally, 59 of the 87 chat rooms had more than a thousand messages and GPU-PDMM performed well with medium-sized and large corpora based on automated topic coherence and intrusion tasks (see details in Sections 4.1 and 4.2). We used the same data pre-processing steps as described in Section 3.2 and, as before, we used lemmatization instead of stemming for text normalization (step 6 of our pre-processing). We also set default hyperparameters for GPU-PDMM as suggested by Li et al. (2017), who developed this model (see values in Section 3.3). We chose to model three topics like in the study of Costa Silva et al. (2022) which identified one to three themes for each chat room. This allowed us to compare fewer but more distinct topics to themes.

To compare themes from the manual analysis and topics from the automated analysis, we manually analyzed the meaning of the words in the three topics created for each chat room in relation to the meaning of the themes associated to each of the 87 Gitter chat rooms. We identified if the meaning of the topics would (a) describe the themes in more detail, i.e. topics help improve the understanding of the themes; (b) be unrelated to themes, i.e. topics cannot be related to the meaning of themes; or (c) generalize themes, i.e. themes could be used to name the topics generated for that chat room. We also identified if there are similarities between topics across all 87 chat rooms (e.g., identify topics related to "Errors" from different chat rooms). Regarding the role of the researchers in this analysis, two researchers performed the comparison between themes and topics, and the other researcher reviewed the results and agreed with the comparison or suggested changes. Then, all researchers discussed any disagreements.

In comparison to the themes described by Costa Silva et al. (2022), we found the following (see topics and themes of the mentioned chat rooms in Table 12):

- Themes manually identified cannot directly be assigned as names to the automated topics. For example, chat room "piskvorky/gensim"[36] is about the theme *SD5.1.2.1 [Machine learning] Libraries*, while their generated topics refer to "document vectorizing" (topic01: model word vector document doc work file vec topic list), "development in python" (topic02: text python time test find small add version import size), and "training model" (topic03: gensim corpus train training code good issue similarity large fasttext). This means that themes generally represent broader subjects discussed in chat rooms while topics represent a detailed perspective of what is discussed in a chat room. In summary, we could assign a theme to at least one topic for 32% of our chat rooms.
- Topics improve the understanding of the theme identified in some chat rooms. For example, the chat rooms "FreeCodeCamp/Contributors",[37] "FreeCodeCamp/testable-projects-fcc",[38] "ga4gh/server",[39] "Hotel-Reservation-Project/requirements"[40] are related to the theme *PD2.1 Coding* which describes experiences and advice on coding practices. However, when this theme is not combined with a second theme, it is more difficult to describe what specific knowledge is discussed regarding coding practices. With the generated topics, we can identify, for example, that chat room "ga4gh/server"[39] is about coding practices associated to source code repository management (topic01: file server work error git issue version master branch case), records of server calls (topic02: reference set vcf support access vcfs callset

[36] https://gitter.im/RaRe-Technologies/gensim.
[37] https://gitter.im/FreeCodeCamp/Contributors.
[38] https://gitter.im/FreeCodeCamp/testable-projects-fcc.
[39] https://gitter.im/ga4gh/server.
[40] https://gitter.im/Hotel-Reservation-Project/requirements.

Table 12
Example of themes and topics of chat rooms.

Chat room	Themes	Topic 01	Topic 02	Topic 03
piskvorky/gensim	Application programming interface; Distributed software development	Model word vector document doc work file vec topic list	Text python time test find small add version import size	Gensim corpus train training code good issue similarity large fasttext
FreeCode-Camp/Contributors	Coding	Issue npm error challenge test code merge guide time freecodecamp	Sparkle point send brownie link good open project question people	Work file fcc branch add git commit update thing pull
FreeCodeCamp/testable-projects-fcc	Coding	Travis work build puppeteer repo stuart tracey git docker issue	Test project fail locally pass good server problem beta personal	Link cdn correct code codepen late create version step update
ga4gh/server	Coding; Application programming interface	File server work error git issue version master branch case	Reference set vcf support access vcfs callset referenceset store variant	Call test sequence add python module exception schema thing graph
Hotel-Reservation-Project /requirements	Coding	Diagram dfd ect png sdp srs good acd function chatter	Project card credit create class requirement need info invalid point	Scenario case room customer employee work call team lead specific
facebook/flow	Code quality; Web development	Function error object work property prop class issue import flowtype	Type flow code component generic var void pass test argument	String return file number export foo definition declare state const
gitlabhq/gitlabhq	Software maintenance; Version control	Work runner build docker file job set image fail test	Gitlab server git repo install run version repository update host	Project issue error create user push branch problem merge request

referenceset store variant), and exception treatment (topic03: call test sequence add python module exception schema thing graph). In summary, we found that 74% of our chat rooms generated topics that can complement the meaning of their associated themes.

- There are topics about similar subjects across all chat rooms. We found that some topics were related to "errors" in different contexts (i.e., 54 topics out of a total of 261). For example, topics discussed in chat rooms "facebook/flow",[41] "FreeCodeCamp/Contributors"[37] and "gitlabhq/gitlabhq"[42] described errors regarding, respectively: method/class use (topic01: function error object work property prop class issue import flowtype), package management (topic01: issue npm error challenge test code merge guide time freecodecamp) and project tracking (topic03: project issue error create user push branch problem merge request). We also found, for example, that out of 261 topics, 26 are related to testing, 16 are related to the development/maintenance of functions, seven are related to "commits", and six are related to installing a software/package in different contexts. This could be an indicator that topics (based on words in messages) capture problems developers face, while themes (identified by considering the broader context of a chat room) capture the contextual information beyond problems.

The added value of this analysis is the comparison between the results of two approaches for describing the content of developer conversations. By relating topics and themes, we briefly checked if our selection of models, based on our evaluation of short text topic models, would generate meaningful topics to be related to our themes. Considering that we modeled three topics for this exercise, we conclude that GPU-PDMM generated distinct topics that, in comparison to themes identified by Costa Silva et al. (2022), describe the discussions of chat rooms in more detail, complementing (rather than contradicting) the meaning of themes.

5.3. Comparison to related work

Chatterjee et al. (2019) identified topics in messages from Slack chat rooms by inputting the entire conversation of a chat room as a document to LDA, and evaluated the quality of its output by manually checking if the topics were understandable. Unlike the study of Chatterjee et al. (2019), we considered each message of a chat room as a document to our topic models. Therefore, each message would individually contribute to the generation of topics.

Similarly to the study of Guzman et al. (2017), we used the short text topic model BTM. Guzman et al. (2017) modeled topics from tweets of app end users while we modeled topics from messages of developer discussions. In comparison to our results for the word intrusion tasks with BTM, Guzman et al. (2017) got higher scores than ours, except for the topics generated by a smaller corpus (tweets about the Dropbox app) with 2435 tweets (MP: 0.30). In our study, BTM had higher scores for word intrusion tasks with topics generated from our largest corpus (see Section 4.2), while in the study of Guzman et al. (2017) the medium-sized corpus (tweets related to the Slack app) had the highest score (MP: 0.70). By comparing the performance of LDA and BTM based on intrusion, Guzman et al. (2017) found that LDA was outperformed by BTM in general.

As described in Section 2.3, Qiang et al. (2022) experimented with short text topic models using different sizes of documents (e.g., tweets and titles of Stack Overflow posts), but they did not model topics using instant messages. The authors evaluated the performance of these models using three metrics for topic quality: *topic coherence* with PMI (Newman et al., 2009); *classification* with accuracy (of how labels assigned by humans matched pre-assigned labels) and *clustering* with purity and normalized mutual information - NMI[43]). Unlike our study, the authors assessed the quality of their topics using measures that required participants to assign predefined labels to their topics to determine accuracy, purity and NMI. Since we aimed at assessing our topics with manual topic naming, a task that requires assigning open-ended names to topics, we did not measure the performance of our models based on predefined labels. As described in Section 3.4, we used the normalized version of PMI, NPMI (Bouma, 2009) (c_{npmi}) as one of our automated topic coherence metrics, while Qiang et al. (2022) used PMI. By comparing the scores of these metrics, we found that our scores with c_{npmi}, considering our nine corpora (highest scores for BTM: −0.01, DMM: 0.01, GPU-PDMM: 0.04 and WNTM: 0.03, LDA: 0.06, see details in Section 4.1.2) and the scores obtained by Qiang et al. (2022) with PMI (highest scores around 1.1 and 1.2 for all models) followed similar trends, even though the actual values differed.

Regarding the overall performance of intrinsic metrics, we found that c_a had some of the highest scores with our combinations of

[41] https://gitter.im/facebook/flow.
[42] https://gitter.im/gitlabhq/gitlabhq.
[43] https://towardsdatascience.com/evaluation-metrics-for-clustering-models-5dde821dd6cd.

models and corpus, while in the study of Röder et al. (2015), c_{npmi} had some of the highest scores. However, when considering the alignment between human judgements and automated coherence metrics, c_a had a statistical significant low positive correlation with RN (metric based on topic naming), while c_{npmi} did not present any statistical significant correlations. The results of other studies, on the other hand, indicated that c_{npmi} aligned the most with human judgements based on 3-point Likert scales rather than based on intrusion tasks metrics (Lau et al., 2014; Aletras and Stevenson, 2013; Röder et al., 2015). In comparison to Likert scales, intrusion tasks are more accurate. They require participants to (a) actively engage with topics (and comprehend them) to identify coherent topics based on altered words; and (b) look into the relationship between topics and documents. In the study of Lau et al. (2014) participants judged if topics were "Useful", "Useless" or "Average". Additionally, related studies that correlated topic coherence metrics and human-based metrics applied LDA to longer texts (e.g., articles from the New York Times and the English Wikipedia (Lau et al., 2014)) rather than short text topic models. Finally, considering our low correlation scores, we cannot conclude that intrinsic metrics and human assessments are equivalent when measuring the quality of a short text topic model (see details in Section 4.4.2).

5.4. Implications

The goal of this study was to understand how short text topic models perform with chat messages from developer instant messaging communication. We selected four short text topic models (from the study of Qiang et al. (2022)), used developer instant messages as documents and checked the quality of the resulted topics based on three strategies for measuring topic quality: (1) automated topic coherence, (2) word and topic intrusion tasks, and (3) topic naming. Based on our findings we formulated a set of recommendations to help researchers and practitioners as follows:

1. *Using both Topic Coherence Metrics and Intrusion Tasks*: Considering that intrusion tasks did not frequently align with automated coherence metrics, researchers and practitioners may measure topic quality using both intrinsic metrics and intrusion tasks. However, considering the costs of human judgement, we suggest researchers and practitioners to select an "external" and an "internal" topic coherence metric; therefore, topic quality can be evaluated in terms of "internal" consistency (u_{mass}) and if topics are distinct and descriptive in comparison to generic English texts, such as Wikipedia files (c_a, c_p and c_{npmi}). Disregarding the alignment between automated topic coherence and intrusion tasks, we suggest the use of c_a, which obtained positive scores (i.e., higher similarities between generic texts to words in topics), or c_{npmi}, which generated the most consistent scores with developer instant messages.
2. *Using Topic Naming as a Measure for Topic Quality*: We found that topic naming (RN) is not aligned with automated coherence metrics or with our intrusion tasks' metric (RTI) in general. We found statistically significant low correlations between RN and the metrics c_a, u_{mass} and RTI (see details in Section 4.4.2). This means that researchers and practitioners are not advised to use topic naming to replace an automated metric or intrusion tasks. On the other hand, topic naming can evaluate how comprehensible topics are for humans, which can add a different perspective over topic quality. Therefore, researchers and practitioners applying Manual or Manual & Automated topic naming as a methodological approach to accomplish a task/application can also consider using topic naming to measure topic quality based on the metrics we proposed in our study (t_{named}, a_{ratio} and number of compatible names).
3. *Selecting Short Text Topic Models*: Researchers and practitioners may use our study as a guide to select short text topic models to be applied to text like chat messages (e.g., based only on the results of our topic naming tasks). Our study can also be used as a reference for researchers and practitioners when applying short text topic models, considering that we described our experiences with different models applied to different sizes of corpora and using different strategies for topic quality.
4. *Pre-processing Instant Messages*: During our data pre-processing we had to customize a stop words list to remove some Internet-related acronyms such as "idk" and "lol" and expressions such as "hahahhaha". Unlike other textual data (e.g., from web pages or papers), we observed that chat messages more frequently contain emoticons, such as ":P", typos and, sometimes, non-English characters. Therefore, researchers and practitioners may consider additional cleaning steps in data pre-processing when dealing with instant messages to remove slang and acronyms.
5. *Selecting Stemming or Lemmatization*: We found that the topics generated with our lemmatized corpora, which had a larger vocabulary size, were more frequently coherent than the topics generated with our stemmed corpora (see Fig. 8). Based on automated topic coherence, the topics generated with lemmatized corpora had at least six of our top-10 highest scores. Therefore, we suggest researchers and practitioners to select lemmatization rather than stemming for the text normalization step of instant messages data pre-processing.

5.5. Threats to validity

We analyzed the limitations and threats to the validity of our study in terms of internal and external validity and regarding repeatability as suggested by Kitchenham et al. (2002) and Sjoberg et al. (2007). We also checked the alignment of our study with the ACM Empirical Standards for empirical research using its generic checklist.[44]

Internal validity: We acknowledge that our selection of short text topic models, data pre-processing steps and topic coherence metrics influenced the findings of our study. For example, we selected four of the eight short text topic models from Qiang et al. (2022) and also did not include a *Self-aggregation-based* model (for the reasons outlined in Section 3.3). Even though we made a conscious decision on what short text topic models to study (see details in Section 3.3), we acknowledge that *Self-aggregation-based* model (such as SATM and PTM), could potentially generate coherent topics with developer instant messaging communication. On the other hand, our results could mean that new short text topic models, in particular for developer communication, may need to be developed. In a preliminary exercise (not described in this study), we experimented with PTM, which is the best performing Self-aggregation-based model in the study of Qiang et al. (2022), who use the titles of Stack Overflow posts as input for topic modeling. We compared it to basic LDA (not grid-searched) and GPU-PDMM (our overall best performing model) based on automated coherence topics. We also used *spaCy*[45] to calculate the semantic similarity (range of 0-1) between topics, e.g., how topic 01 from GPU-PDMM is similar to topic 01 from PTM. We generated three topics for each of the 87 Gitter chat rooms used in the exercise described in Section 5.2, and set hyperparameters as suggested by the authors who proposed these models (like we set our short text topic models). Results show that PTM generated coherent topics in general [based on model-level scores], and its topics were also more semantically similar to basic LDA topics than GPU-PDMM topics. This could indicate that PTM is well worthy of a further analysis with instant messages. However, this was just a brief exercise and such a brief comparison might be inconclusive for our study.

[44] https://github.com/acmsigsoft/EmpiricalStandards.
[45] https://spacy.io/api/doc.

Table 13
Topic naming for BTM.

	ID of participants				Number
	TN.01	TN.02	TN.03	TN.07	of names
Android_lemma					
topic01	–	Model view presenter pattern	–	–	1
topic02	–	Issue tracker	–	–	1
topic03	Model view presenter	Front end	ViewModel	Log	4
topic04	Software artifacts	Devops	DevOps	Application	4
topic05	–	Dependency injection	Dependency injection	–	2
topic06	–	–	–	Communication tool	1
topic07	–	–	–	Backend monitoring	1
topic08	–	–	–	–	0
topic09	–	Android	–	–	1
topic10	–	–	–	–	0
Jenkinsci_lemma					
topic01	–	Devops	–	–	1
topic02	–	–	Jenkins	–	1
topic03	CI/CD tools	Devops	–	Application deployment	3
topic04	–	Devops	–	–	1
topic05	–	–	–	–	0
topic06	Build configuration stuff	Devops	–	Application deployment	3
topic07	–	Issue tracker	–	–	1
topic08	–	Devops	Linux	Setup	3
topic09	–	–	–	Variable	1
topic10	–	–	–	–	0
Flutter_lemma					
topic01	–	Asynchronous programming	–	Promise	2
topic02	–	Mobile development	Mobile development	Mobile application	3
topic03	Javascript/frontend concepts	–	Framework	–	2
topic04	–	Mobile development	Mobile development	Mobile application	3
topic05	–	Flutter language	–	–	1
topic06	Language keywords	Object-oriented programming	Method signature	Function output	4
topic07	–	Compilation	Build project	–	2
topic08	–	–	Repository	Coding language	2
topic09	–	Multiplatform development	Repository	–	2
topic10	–	–	–	–	0

Regarding the parameters setting for the topic models (e.g., α and β or the number of topics), we acknowledge that by using the default values of each model (i.e., as suggested by the authors who proposed the topic models), we may not get optimized results. On the other hand, we checked how short text topic models perform with instant messages with their original setting. Furthermore, previous studies such as Yan et al. (2013), Xiong et al. (2018), Li et al. (2018) and Murshed et al. (2023) usually set their models with default values, which was also done in the study of Qiang et al. (2022), our main reference. For LDA, we grid-searched[27] optimized hyperparameters values as suggested by Agrawal et al. (2018), considering that we would model ten topics from all corpora. However, even with optimized hyperparameters LDA did not outperform non-optimized models. Regarding the number of topics, we acknowledge that fixing it to 10 may have limited our analysis of the quality of topics generated by short text topic models with instant messages. Considering that we experimented with corpora of different sizes and that developer instant messaging may be contextually different from commit messages (our motivation for selecting ten topics), using different numbers of topics for each corpora could have produced topics of higher quality in terms of automated or human-based assessments. To check the impact of the number of topics on the results of our study, we performed a small-scale sensitivity analysis based on our automated topic coherence metrics (see detailed scores online[23]). We modeled 8, 9, 10 (number of topics used in the study), 11 and 12 topics with the models grid-searched LDA, BTM, DMM, GPU-PDMM and WNTM applied to the corpora Android_lemma, Jenkinsci_lemma and Flutter_lemma (corpora used in both automated and human-based topic quality assessments in our manuscript). To identify if there is a statistically significant difference between the scores of our automated metrics considering those different numbers of topics, we used Kruskal–Wallis' test[34], and post hoc Dunn's test with Bonferroni correction[35]. We divided our sensitivity analysis into four to understand the impact of the number of topics on the scores of each of our automated metrics (i.e., we ran a Kruskal–Wallis' test with all ca scores, then a Kruskal–Wallis' test with all cp scores and so on). For each metric, we grouped scores by the number of topics. Based on the resulting *p-values*, we found a statistically significant difference between groups of c_a metric scores (*p-value* of 0.013). After applying Dunn's test with Bonferroni correction to these groups, we identified that the differences occurred between the groups of scores generated with 9 and 10 topics, and between the groups of scores generated with 10 and 12 topics. Therefore, we can conclude that the results of our study would not be impacted if we have modeled less (e.g., 8 and 9) or more (e.g., 11 and 12) than 10 topics based on three of our four automated coherence metrics. We acknowledge that the results of our study would have been impacted, considering the metric c_a, if we had modeled 9 or 12 topics rather than 10 because we found that models generating 9 or 12 topics had higher c_a coherence scores at model-level compared to the models generating 10 topics.

Regarding the differences in scope between the analysis of intrinsic metrics and human-based assessments, we acknowledge that we could have different findings if human assessments were applied to the topics generated with LDA or to the topics from all of the original nine corpora. On the other hand, reducing the scope of our analysis in terms of number of corpus and focusing on short text topic models allowed us to recruit many experienced participants and perform two human evaluations considering the costs of human judgement.

For word and topic intrusion tasks, we followed the guidelines of Chang et al. (2009) and only provided one document for topic intrusion tasks. Chang et al. (2009) argued that humans are good at extrapolating from limited data and that participants should interpret topics based on little additional information. For example, Chang et al. (2009) only used the title and the first couple of sentences of their documents in their tasks. Providing more information was not recommended by any previous studies and may have biased the results.

Table 14
Topic naming for DMM.

	ID of participants				Number
	TN.01	TN.05	TN.06	TN.08	of names
Android_lemma					
topic01	Programming language constructs	Programming	–	–	2
topic02	–	Devops	Repository	Code versioning	3
topic03	–	Open source	–	–	1
topic04	–	Object notation	Web development	Software architecture pattern	3
topic05	–	Web	Web development	Conference	3
topic06	–	–	–	–	0
topic07	–	Continuous integration	Debug	Elastic beanstalk environment	3
topic08	–	Data requirements	Project definition	Proof of concept	3
topic09	–	Dependency injection	–	Dependency injection	2
topic10	–	Design pattern	Web pages	–	2
Jenkinsci_lemma					
topic01	Repository	Continuous integration	–	Code versioning	3
topic02	Secret management	Information security	–	Security	3
topic03	–	CI/CD (Continuous Integration/Continuous Deployment)	–	Pipelines	2
topic04	Continuous integration	Software development	Compilation	Unit testing	4
topic05	–	Planning	–	Devops	2
topic06	Build systems	Infrastructure-as-code	–	Infra as code	3
topic07	Jenkins	Workload automation	–	Automatization	3
topic08	–	Configuration as code	–	–	1
topic09	–	CI/CD	–	CI/CD	2
topic10	–	Java development	–	–	1
Flutter_lemma					
topic01	User interface	App development	–	Mobile app interface	3
topic02	–	App development	Development	Cross platform mobile framwork	3
topic03	App development	Mobile app development	–	Cross platform development	3
topic04	App development	Mobile app development	Mobile	Mobile developer	4
topic05	–	Graphical interface development	–	Mobile app	2
topic06	User interface concepts	Single-child layout widgets (Flutter)	Screen elements	–	3
topic07	–	App development	Mobile	Mobile development	3
topic08	Programming language keywords	Source code	Reserved words	Asynchronous development	4
topic09	–	Flutter	–	–	1
topic10	Programming language concepts	Single-child layout widgets (Flutter)	–	–	2

For the topic naming, we did not calculate agreement between the names assigned by participants (e.g., using Cohen's Kappa measure Sim and Wright, 2005) because we did not use predefined categories. Agreement measures are used to calculate agreement between two or more assessments using categorical data rather than open-ended names. Instead, we checked which and how many topics were named for each combination of model and corpus. We also analyzed the semantic compatibility of these names by topic. Regarding the evaluation of compatibility, we acknowledge that this is subjective and biased based on our own experiences with software development (with Android apps, Jenkins Continuous Integration and Flutter development kit). While this initial evaluation was done by one of the researchers, checking semantic compatibility of names (and performing the manual analysis described in Section 5.2) was conducted, discussed and agreed by multiple researchers.

Regarding the experience of participants for the intrusion tasks and topic naming, we recruited experienced developers in general rather than developers with specific experiences, for example, with Android app development. We acknowledge that the results of human assessments of topic models were influenced by the professional experiences of participants. On the one hand, involving developers with general software development expertise gave us potentially less biased names in comparison to having only participants with specific expertise about the subjects discussed in the analyzed conversations. On the other hand, developers with specific experiences may have led to better results (higher MP and TLO scores for the intrusion tasks and more named topics). Regarding the number of participants, we acknowledge that as our data sets referred to domain-specific content (i.e. developers' professional conversations), we recruited few experts rather than a large number of non-experts or junior professionals. Additionally, we involved more than two participants in the evaluation of topics as suggested in the literature (Shull et al., 2008).

Regarding the reliability of human assessments, we acknowledge that responses could have been influenced by circumstances of participants' daily routines. For example, if participants performed our tasks during a work break, their responses might have been rushed as they had little time. On the other hand, as we provided an incentive for participating in our study, respondents may have felt more "obliged" to perform tasks carefully. We also checked time stamps of submissions and no suspicious behavior was detected (e.g., participants completing the tasks in less than five minutes). To avoid potential learning effect bias, we randomized the distribution of tasks (i.e., sub-surveys) to participants and with Qualtrics[30] (the tool we used to manage our surveys), we set each survey to randomly present tasks, e.g., participant A would have a different task 1 than participant B if A and B responded to the same sub-survey. Additionally, participants did not answer the same survey more than once and we provided a single sub-survey for each participant.

External validity: Even though we experimented with nine diverse data sets (in terms of size and subjects, such as Android app development, Jenkins Continuous Integration and Flutter development kit) and four carefully chosen short text topic models, our results are not fully generalizable. Since we provided an in-depth analysis of generated topics involving human experts, we had to limit the number of chat rooms also for practical reasons (in particular for the human evaluation), as described in Section 3.5. We describe how we selected our data sets in Section 3.1 and our short text topic models in Section 3.3.

Repeatability: Our study can be repeated by other researchers and practitioners considering that the tools and data sets that we used are available online (see details in Section 3). On the other hand, the results may not be completely replicated because topic models do not generally generate the same topics in every processing (Blei et al., 2003; Griffiths and Steyvers, 2004). Furthermore, the selection of participants for the human assessments of topics impacts the results.

Table 15
Topic naming for GPU-PDMM.

	ID of participants				Number
	TN.04	TN.05	TN.07	TN.08	of names
Android_lemma					
topic01	–	App optimizer	–	–	1
topic02	–	–	–	–	0
topic03	–	Mobile app development	–	Responsive development	2
topic04	–	Dependency injection	Database	–	2
topic05	–	Architecture component	Backend monitoring	Angular	3
topic06	–	–	–	–	0
topic07	–	–	–	–	0
topic08	–	Software development	–	Angular framework	2
topic09	–	Web development	–	–	1
topic10	–	Continuous integration	Monitoring	CI/CD	3
Jenkinsci_lemma					
topic01	–	Configuration as code	–	–	1
topic02	–	Information security	Backend management	Security	3
topic03	–	Planning	Time organization	Happy hour	3
topic04	–	CI/CD	Deployment	CI/CD	3
topic05	Docker	Configuration as code	Setup	Configuration file	4
topic06	–	Open source development	CI/CD	Code versioning	3
topic07	–	CI/CD	Upgrade	Pipelines	3
topic08	–	Source code	Issue	Error handling	3
topic09	–	Configuration as code	–	–	1
topic10	–	Configuration as code	Backend management	–	2
Flutter_lemma					
topic01	–	Flutter	Flutter language	–	2
topic02	–	Single-child layout widgets	–	CSS	2
topic03	–	App development	Programming language	Cross platform build	3
topic04	Mobile development	Mobile app development	Mobile application	Cross platform development	4
topic05	–	App development	–	Authentication	2
topic06	–	App development	Programming language	–	2
topic07	–	Source code	Function	–	2
topic08	–	Software development	Bug	Unit testing	3
topic09	–	Graphical interface development	Oriented object	–	2
topic10	Github	Open source development	Bug to fix	Github	4

6. Conclusion

Our study analyzed how four short text topic models (Qiang et al., 2022) performed with developer instant messaging communication. We used the messages of Gitter chat rooms of different sizes. After pre-processing the messages of these chat rooms (experimenting with two types of text normalization: stemming and lemmatization) and inferring topics, we calculated four topic coherence metrics (Röder et al., 2015) for the topics generated by each of our models. We also performed two human assessments of topics: intrusion tasks (Chang et al., 2009) and topic naming (Hindle et al., 2015). By using these strategies, we aimed at evaluating how understandable, meaningful and interpretable the topics generated with models were.

Our results indicate that, in general, short text topic models generate coherent topics with instant messaging communication (considering corpora with different number of messages and size of vocabulary). Some models generated coherent topics based on human comprehension while others generated coherent topics based on intrinsic metrics. Therefore, we did not find one model that performed best with all of our corpora based on all of our topic quality metrics. Among our selection of short text topic models, GPU-PDMM had the best overall performance (see details in Section 4.4), despite the statistically significant differences between models' automated metric scores. Based on automated topic coherence, we also found that grid-searched LDA generates coherent topics with instant messages when modeling few topics from larger corpora, in terms of vocabulary size (see details in Section 4.1.3). Regarding topic naming, most models had at least one name assigned per topic and when more names were assigned to a topic (we had four experts naming each topic), these names were not always semantically compatible.

Referring to our list of contributions in Section 1, we conclude the following:

- We evaluated the quality of short text topic models applied to software developer instant communication based on four established topic coherence metrics (Röder et al., 2015). The models that frequently had the top-10 highest scores for our metrics were DMM and GPU-PDMM (see Section 4.1.2). We also found that the topics generated with lemmatized corpora were more coherent based on three of our automated metrics (c_a, c_p and c_{npmi}) than the topics generated with stemmed corpora, see Section 4.1.4.
- To understand how comprehensible topics are for humans (the "users" of the topics), we evaluated the quality of short text topic models based on intrusion tasks (word and topic) (Chang et al., 2009) and topic naming (Hindle et al., 2015). With the intrusion tasks, we found that GPU-PDMM and WNTM performed well for both intrusion tasks in comparison to the other models. For the topic naming, we found that most models had at least one topic named by at least one participant. DMM generated the most comprehensible topics for our participants, i.e., was the model with the highest average ratio of names per topic (0.62) and the highest number of topics with compatible names (four out of ten topics), see details in Section 4.3.
- To understand whether human assessment can be replaced by automatically calculated coherence scores and if theoretical coherence aligns with human comprehension of topics, we compared automated topic coherence metrics and metrics based on human assessments using Spearman's Rank correlation[32]. We found that, in the context of our study, intrinsic metrics and human-based metrics did not align. As described in Section 4.4, we could not find a consistent correlation between these metrics.

In further studies, we are going to apply one of the short text topic models we experimented with to a larger number of chat rooms from Gitter and develop a taxonomy of subjects discussed by developers

Table 16
Topic naming for WNTM.

	ID of participants				Number of names
	TN.02	TN.03	TN.04	TN.06	
Android_lemma					
topic01	Server side development	–	–	–	1
topic02	Issue tracker	–	–	Repository	2
topic03	Issue tracker	–	–	–	1
topic04	–	Model-view-viewmodel	–	Web development	2
topic05	Blog	–	–	Web development	2
topic06	–	–	–	–	0
topic07	Debugging	–	Deploy	Debug	3
topic08	–	–	–	Project definition	1
topic09	–	–	–	–	0
topic10	Web development	–	–	Web pages	2
Jenkinsci_lemma					
topic01	Devops	–	Continuous integration	Configuration files	3
topic02	Version control	Git/Github	Package	Commands	4
topic03	Devops	–	Cloud	–	2
topic04	–	–	–	Authorization	1
topic05	Devops	–	–	–	1
topic06	–	–	–	Support	1
topic07	Devops	–	–	Build error	2
topic08	–	–	–	–	0
topic09	Devops	–	–	Continuous integration	2
topic10	Task management	–	–	–	0
Flutter_lemma					
topic01	Android development	Mobile development	Mobile development	Mobile development	4
topic02	Front end development	–	–	–	1
topic03	Mobile development	–	–	–	1
topic04	Multiplatform development	Repository	–	Mobile development	3
topic05	–	–	–	–	0
topic06	Compilation	Build project	–	Compilation error	3
topic07	Unit testing	–	–	–	1
topic08	–	–	–	–	0
topic09	Asynchronous programming	Data type	Javascript	Reserved words	4
topic10	–	–	–	–	0

in instant messaging communication. For that, we will also relate the generated topics to the themes identified by Costa Silva et al. (2022) to check whether topics accurately describe the main themes discussed in developer chat rooms, like our exercise in Section 5.2. Note that Costa Silva et al. (2022) also applied text summarization to messages of Gitter chat rooms. Summarization techniques get the main subject of a long textual document by creating a short text based on keywords or key-sentences of this long text (Allahyari et al., 2017). Applied to instant messages, summarization can provide the summary of the discussion from a chat room, while topic modeling can provide topics that describe a discussion from a chat room or the discussions from several chat rooms at a more generic level. Unlike summaries, topics could be used, for example, as tags to describe and filter chat rooms or messages shared in instant messaging tools (Parra et al., 2018). Finally, future work can expand our study by experimenting with the other four models from the study of Qiang et al. (2022) that we did not use (e.g., *Self-aggregation*-based models) or short text topic modeling techniques proposed by other studies (e.g., Twitter-LDA by Zhao et al. (2011), mentioned in Silva et al. (2021)).

CRediT authorship contribution statement

Camila Costa Silva: Writing – review & editing, Writing – original draft, Visualization, Investigation, Data curation, Conceptualization. **Matthias Galster:** Writing – review & editing, Supervision, Investigation, Conceptualization. **Fabian Gilson:** Writing – review & editing, Validation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

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.

Appendix A. Sub-surveys distributed - Intrusion tasks

In the following list we mentioned which combinations of model and corpus were used in each of our sub-surveys.

1. Android_BTM and Android_DMM
2. Android_BTM and Android_GPU-PDMM
3. Android_BTM and Android_WNTM
4. Android_DMM and Android_GPU-PDMM
5. Android_DMM and Android_WNTM
6. Android_GPU-PDMM and Android_WNTM
7. Jenkinsci_BTM and Jenkinsci_DMM
8. Jenkinsci_BTM and Jenkinsci_GPU-PDMM
9. Jenkinsci_BTM and Jenkinsci_WNTM
10. Jenkinsci_DMM and Jenkinsci_GPU-PDMM
11. Jenkinsci_DMM and Jenkinsci_WNTM
12. Jenkinsci_GPU-PDMM and Jenkinsci_WNTM
13. Flutter_BTM and Flutter_DMM
14. Flutter_BTM and Flutter_GPU-PDMM
15. Flutter_BTM and Flutter_WNTM
16. Flutter_DMM and Flutter_GPU-PDMM
17. Flutter_DMM and Flutter_WNTM
18. Flutter_GPU-PDMM and Flutter_WNTM

Appendix B. Sub-surveys distributed - Topic naming

In the following list we mentioned which combinations of model and corpus were used in each of our sub-surveys.

1. Android_BTM, Jenkinsci_BTM, Flutter_BTM, Android_WNTM, Jenkinsci_WNTM, and Flutter_WNTM
2. Android_DMM, Jenkinsci_DMM, Flutter_DMM, Android_GPU-PDMM, Jenkinsci_GPU-PDMM, and Flutter_GPU-PDMM
3. Android_BTM, Jenkinsci_BTM, Flutter_BTM, Android_DMM, Jenkinsci_DMM, and Flutter_DMM
4. Android_WNTM, Jenkinsci_WNTM, Flutter_WNTM, Android_GPU-PDMM, Jenkinsci_GPU-PDMM and Flutter_GPU-PDMM
5. Android_BTM, Jenkinsci_BTM, Flutter_BTM, Android_GPU-PDMM, Jenkinsci_GPU-PDMM and Flutter_GPU-PDMM
6. Android_WNTM, Jenkinsci_WNTM, Flutter_WNTM, Android_DMM, Jenkinsci_DMM, and Flutter_DMM

Appendix C. Topics named by participant

See Tables 13–16.

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. http://dx.doi.org/10.1145/3379597.3387472.

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.

Ahasanuzzaman, M., Asaduzzaman, M., Roy, C.K., Schneider, K.A., 2020. CAPS: a supervised technique for classifying stack overflow posts concerning API issues. Empir. Softw. Eng. 25 (2), 1493–1532.

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. http://dx.doi.org/10.1145/3239235.3239524.

Aletras, N., Stevenson, M., 2013. Evaluating topic coherence using distributional semantics. In: Proceedings of the 10th International Conference on Computational Semantics. Association for Computational Linguistics, Potsdam, pp. 13–22.

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Elizabeth, D., Juan, B., Kochut, K., 2017. Text summarization techniques: A brief survey. Int. J. Adv. Comput. Sci. Appl. 8 (10), 397–405.

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. http://dx.doi.org/10.1145/3338906.3338939.

Barua, A., Thomas, S.W., Hassan, A.E., 2014. What are developers talking about? An analysis of topics and trends in stack overflow. Empir. Softw. Eng. 19 (3), 619–654.

Bass, L., Clements, P., Kazman, R., 2003. Software Architecture in Practice, second ed. Addison-Wesley Professional, Boston, p. 560.

Bhatia, S., Lau, J.H., Baldwin, T., 2018. Topic intrusion for automatic topic model evaluation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Brussels, pp. 844–849. http://dx.doi.org/10.18653/v1/D18-1098, URL: http://aclweb.org/anthology/D18-1098.

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.

Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent Dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022.

Bouma, G., 2009. Normalized (pointwise) mutual information in collocation extraction. In: Proceedings of the Biennial German Society for Computational Linguistics Conference. German Society for Computational Linguistics & Language Technology, Potsdam, pp. 31–40.

Canfora, G., Cerulo, L., Cimitile, M., Di Penta, M., 2014. How changes affect software entropy: An empirical study. Empir. Softw. Eng. 19, 1–38.

Capiluppi, A., Di Ruscio, D., Di Rocco, J., Nguyen, P.T., Ajienka, N., 2020. Detecting java software similarities by using different clustering techniques. Inf. Softw. Technol. 122 (106279), 1–18.

Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., Blei, D.M., 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., Kraft, N.A., Pollock, L., 2021. Automatically identifying the quality of developer chats for post hoc use. ACM Trans. Softw. Eng. Methodol. 30 (4), 1–28.

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.

Chatterjee, P., Nishi, M.A., Damevski, K., Augustine, V., Pollock, L., Kraft, N.A., 2017. What information about code snippets is available in different software-related documents? An exploratory study. In: Proceedings of the 24th International Conference on Software Analysis, Evolution, and Reengineering. IEEE, Klagenfurt, pp. 382–386. http://dx.doi.org/10.1109/SANER.2017.7884638.

Chen, H., Coogle, J., Damevski, K., 2019. Modeling stack overflow tags and topics as a hierarchy of concepts. J. Syst. Softw. 156, 283–299.

Chen, T.H., Thomas, S.W., Hassan, A.E., 2016. A survey on the use of topic models when mining software repositories. Empir. Softw. Eng. 21 (5), 1843–1919.

Choi, K., Lee, J.H., Willis, C., Downie, J.S., 2015. Topic modeling users' interpretations of songs to inform subject access in music digital libraries. In: Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries. ACM, New York, NY, USA, pp. 183–186. http://dx.doi.org/10.1145/2756406.2756936, URL: https://dl.acm.org/doi/10.1145/2756406.2756936.

Costa Silva, C., Galster, M., Gilson, F., 2022. A qualitative analysis of themes in instant messaging communication of software developers. J. Syst. Softw. 192, 111397.

Costa Silva, C., Gilson, F., Galster, M., 2019. Comparison framework for team-based communication channels. In: Franch, X., Männistö, T., Martínez-Fernández, S. (Eds.), In: Lecture Notes in Computer Science, vol. 11915, Springer, Barcelona, pp. 315–322. http://dx.doi.org/10.1007/978-3-030-35333-9{_}22, URL: http://link.springer.com/10.1007/978-3-030-35333-9_22.

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R., 1990. Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. 41 (6), 391–407.

Dittrich, Y., Giuffrida, R., 2011. Exploring the role of instant messaging in a global software development project. In: Proceedings of the Sixth International Conference on Global Software Engineering. IEEE, Helsinki, pp. 103–112. http://dx.doi.org/10.1109/ICGSE.2011.21, URL: http://ieeexplore.ieee.org/document/6063155/.

Ehsan, O., Hassan, S., Mezouar, M.E., Zou, Y., 2021. An empirical study of developer discussions in the gitter platform. ACM Trans. Softw. Eng. Methodol. 30 (1), 1–39. http://dx.doi.org/10.1145/3412378, URL: https://dl.acm.org/doi/10.1145/3412378.

Griffiths, T.L., 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. http://dx.doi.org/10.1073/pnas.0307752101.

Guzman, E., Ibrahim, M., Glinz, M., 2017. A little bird told me: Mining tweets for requirements and software evolution. In: Proceedings of the 25th International Requirements Engineering Conference. IEEE, Lisbon, pp. 11–20. http://dx.doi.org/10.1109/RE.2017.88, URL: http://ieeexplore.ieee.org/document/8048886/.

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.

Haque, M.U., 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. http://dx.doi.org/10.1145/3382494.3410693.

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. http://dx.doi.org/10.1109/ICSE.2012.6227139.

Hindle, A., Bird, C., Zimmermann, T., Nagappan, N., 2015. Do topics make sense to managers and developers? Empir. Softw. Eng. 20, 479–515.

Hu, H., Bezemer, C.P., Hassan, A.E., 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.

Hu, H., Wang, S., Bezemer, C.P., Hassan, A.E., 2019. Studying the consistency of star ratings and reviews of popular free hybrid android and iOS apps. Empir. Softw. Eng. 24, 7–32.

Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K., Rosenberg, J., 2002. Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. 28 (8), 721–734.

Kruskal, W.H., Wallis, W.A., 1952. Use of ranks in one-criterion variance analysis. J. Amer. Statist. Assoc. 47, 583–621.

Lau, J.H., Newman, D., Baldwin, T., 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In: Proceedings Ofthe 14th Conference Ofthe European Chapter Ofthe Association for Computational Linguistics. Association for Computational Linguistics, Gothenburg, pp. 530–539.

Layman, L., Nikora, A.P., 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. http://dx.doi.org/10.1145/2901739.2901760.

Li, C., Duan, Y., Wang, H., Zhang, Z., Sun, A., Ma, Z., 2017. Enhancing topic modeling for short texts with auxiliary word embeddings. ACM Trans. Inf. Syst. 36 (2), 30. http://dx.doi.org/10.1145/3091108.

Li, X., Li, C., Chi, J., Ouyang, J., 2018. Short text topic modeling by exploring original documents. Knowl. Inf. Syst. 56 (2), 443–462.

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. http://dx.doi.org/10.1145/2566486.2567980.

Liu, K., Yang, G., Chen, X., Yu, C., 2022. Sotitle: A transformer-based post title generation approach for stack overflow. In: Proceedings of the International Conference on Software Analysis, Evolution and Reengineering. IEEE, Honolulu, pp. 577–588. http://dx.doi.org/10.1109/SANER53432.2022.00075, URL: https://ieeexplore.ieee.org/document/9825881/.

Lund Research Ltd, 2018a. Pearson product-moment correlation. URL: https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php.

Lund Research Ltd, 2018b. Spearman's rank-order correlation. URL: https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php.

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. http://dx.doi.org/10.1109/MSR.2015.19.

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.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In: Proceedings of the Annual Conference on Neural Information Processing Systems. Neural Information Processing Systems Foundation, Lake Tahoe, pp. 1–9, URL: https://papers.nips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

Mimno, D., Wallach, H.M., Talley, E., Leenders, M., Mccallum, A., 2011. Optimizing semantic coherence in topic models. In: Proceedings Ofthe 2011 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Edinburgh, pp. 262–272. http://dx.doi.org/10.5555/2145432.2145462.

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, p. 1095. http://dx.doi.org/10.1016/C2010-0-66188-8.

Murphy, B., Talukdar, P.P., Mitchell, T., 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In: Proceedings of the 24th International Conference on Computational Linguistics. COLING, Mumbai, pp. 1933–1950.

Murshed, B.A.H., Mallappa, S., Abawajy, J., Saif, M.A.N., Al-ariki, H.D.E., Abdulwahab, H.M., 2023. Short text topic modelling approaches in the context of big data: taxonomy, survey, and analysis. Artif. Intell. Rev. 56 (6), 5133–5260.

Nayebi, M., Cho, H., Ruhe, G., 2018. App store mining is not enough for app improvement. Empir. Softw. Eng. 23, 2764–2794.

Newman, D., Karimi, S., Cavedon, L., 2009. External evaluation of topic models. In: Proceedings of the Fourteenth Australasian Document Computing Symposium. HCSNet, Sydney, pp. 11–18.

Newman, D., Lau, J.H., Grieser, K., Baldwin, T., 2010. Automatic evaluation of topic coherence. In: Proceedings of the Annual Conference of the North American Chapter of the ACL. ACL, Los Angeles, pp. 100–108. http://dx.doi.org/10.5555/1857999.1858011.

Nguyen, H., Hovy, D., 2019. Hey Siri. Ok google. Alexa: A topic modeling of user reviews for smart speakers. In: Proceedings of the 5th Workshop on Noisy User-Generated Text. Association for Computational Linguistics, Hong Kong, pp. 76–83. http://dx.doi.org/10.18653/v1/d19-5510.

Noei, E., Zhang, F., Wang, S., Zou, Y., 2019. Towards prioritizing user-related issue reports of mobile applications. Empir. Softw. Eng. 24, 1964–1996.

Pagano, D., Maalej, W., 2013. How do open source communities blog? Empir. Softw. Eng. 18 (6), 1090–1124.

Parra, E., Escobar-Avila, J., Haiduc, S., 2018. Automatic tag recommendation for software development video tutorials. In: Proceedings of the 26th Conference on Program Comprehension. ACM, New York, pp. 222–232. http://dx.doi.org/10.1145/3196321.3196351, URL: https://dl.acm.org/doi/10.1145/3196321.3196351.

Pettinato, M., Gil, J.P., 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. http://dx.doi.org/10.1016/j.infsof.2019.06.011.

Qiang, J., Qian, Z., Li, Y., Yuan, Y., Wu, X., 2022. Short text topic modeling techniques, applications, and performance: A survey. IEEE Trans. Knowl. Data Eng. 34 (3), 1427–1445.

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. http://dx.doi.org/10.1145/1985441.1985451.

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. http://dx.doi.org/10.1145/2684822.2685324.

Shrikanth, N.C., Nichols, W., Fahid, F.M., Menzies, T., 2021. Assessing practitioner beliefs about software engineering. Empir. Softw. Eng. 26 (73), 1–32.

Shull, F., Singer, J., Sjøberg, D.I.K. (Eds.), 2008. Guide to Advanced Empirical Software Engineering. Springer, London, p. 393. http://dx.doi.org/10.1007/978-1-84800-044-5, URL: http://link.springer.com/10.1007/978-1-84800-044-5.

Silva, C.C., Galster, M., Gilson, F., 2021. Topic modeling in software engineering research. Empir. Softw. Eng. 26 (6), 120.

Sim, J., Wright, C.C., 2005. The kappa statistic in reliability studies: Use, interpretation, and sample size requirements. Phys. Ther. 85 (3), 257–268.

Sjoberg, D.I.K., Dyba, T., Jorgensen, M., 2007. The future of empirical methods in software engineering research. In: Proceedings of the Future of Software Engineering. IEEE, Minneapolis, pp. 358–378. http://dx.doi.org/10.1109/FOSE.2007.30, URL: http://ieeexplore.ieee.org/document/4221632/.

Souza, L.B., Campos, E.C., Madeiral, F., Paixão, K., Rocha, A.M., Maia, M.d., 2019. Bootstrapping cookbooks for APIs from crowd knowledge on stack overflow. Inf. Softw. Technol. 111 (March 2018), 37–49.

Statology, 2020. Dunn's test for multiple comparisons. URL: https://www.statology.org/dunns-test/.

Stevens, K., Kegelmeyer, P., Andrzejewski, D., Buttler, D., 2012. Exploring topic coherence over many models and many topics. In: Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Association for Computational Linguistics, Jeju Island, pp. 952–961.

Storey, M.-A., Zagalsky, A., Filho, F.F., Singer, L., German, D.M., 2017. How social and communication channels shape and challenge a participatory culture in software development. IEEE Trans. Softw. Eng. 43 (2), 185–204.

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.

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.

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, A.E., 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.

TheAcropolitan, 2017. Sentence length has declined 75% in the past 500 years. URL: https://medium.com/@theacropolitan/sentence-length-has-declined-75-in-the-past-500-years-2e40f80f589f.

Thomas, S.W., Nagappan, M., Blostein, D., Hassan, A.E., 2013. The impact of classifier configuration and classifier combination on bug localization. IEEE Trans. Softw. Eng. 39 (10), 1427–1443.

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. http://dx.doi.org/10.1109/MSR.2019.00022.

van der Lee, C., Gatt, A., van Miltenburg, E., Krahmer, E., 2021. Human evaluation of automatically generated text: Current trends and best practice guidelines. Comput. Speech Lang. 67, 101151.

Weng, J., Lim, E.P., 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. http://dx.doi.org/10.1145/1718487.1718520.

Xiong, S., Wang, K., Ji, D., Wang, B., 2018. A short text sentiment-topic model for product reviews. Neurocomputing 297, 94–102.

Yan, X., Guo, J., Lan, Y., Cheng, X., 2013. A biterm topic model for short texts. In: Proceedings of the 22nd International Conference on World Wide Web. ACM, Rio de Janeiro, pp. 1445–1455.

Yan, M., Zhang, X., Yang, D., Xu, L., Kymer, J.D., 2016. A component recommender for bug reports using discriminative probability latent semantic analysis. Inf. Softw. Technol. 73, 37–51.

Yin, J., Wang, J., 2014. A Dirichlet multinomial mixture model-based approach for short text clustering. In: Proceeding of the 20th International Conference on Knowledge Discovery and Data Mining. ACM, New York, pp. 1–10. http://dx.doi.org/10.1145/2623330.2623715.

Zhao, W.X., Jiang, J., Weng, J., He, J., Lim, E.-P., Yan, H., Li, X., 2011. Comparing Twitter and traditional media using topic models. In: Lecture Notes in Computer Science, vol. 6611, Springer, Berlin, pp. 338–349. http://dx.doi.org/10.1007/978-3-642-20161-5{_}34, URL: http://link.springer.com/10.1007/978-3-642-20161-5_34.

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.
Zuo, Y., Zhao, J., Xu, K., 2016. Word network topic model: a simple but general solution for short and imbalanced texts. Knowl. Inf. Syst. 48 (2), 379–398.

Camila Costa Silva is a research assistant at the Victoria University of Wellington, New Zealand and received her Ph.D. from the University of Canterbury, New Zealand.

Matthias Galster is a Professor at the University of Canterbury, New Zealand.

Fabian Gilson is a Senior Lecturer at the University of Canterbury, New Zealand.