Finding Associations between Natural and Computer Languages: A Case-Study of Bilingual Lda Applied to the Bleeping Computer Forum Posts

Article *in* Journal of Systems and Software · January 2022

CITATIONS
0

READS
117

6 authors, including:

Kundi Yao
Queen's University
5 PUBLICATIONS **93** CITATIONS
SEE PROFILE

Gustavo Ansaldi Oliva
Queen's University
52 PUBLICATIONS **828** CITATIONS
SEE PROFILE

Ahmed E. Hassan
Queen's University
493 PUBLICATIONS **23,395** CITATIONS
SEE PROFILE

Andrew Malton
BlackBerry
29 PUBLICATIONS **779** CITATIONS
SEE PROFILE

Finding Associations Between Natural and Computer Languages: A Case-study of Bilingual LDA Applied to the Bleeping Computer Forum Posts

Kundi Yao[a,*], Gustavo A. Oliva[a], Ahmed E. Hassan[a], Muhammad Asaduzzaman[b], Andrew J. Malton[c], Andrew Walenstein[c]

[a]*Software Analysis and Intelligence Lab (SAIL) at Queen's University, Kingston, Canada*
[b]*Lakehead University, Thunderbay, Canada*
[c]*Blackberry Ltd*

Abstract

In the context of technical support, trails of technical discussions often contain a mixture of natural language (e.g., English) and software log excerpts. Uncovering latent links between certain problems and log excerpts that are often requested during the discussions of those problems enables the construction of a valuable knowledge base. Nevertheless, uncovering such latent links is challenging because English and software logs are two fundamentally different languages. In this paper, we investigate the suitability of multilingual LDA models to address the problem at hand. We study three models, namely: *enriched LDA* (M_+), *two-layer LDA* (M_{2L}), and *off-the-shelf bilingual LDA* (M_{bi}). We use approximately 8K discussion threads from a Bleeping Computer forum as our dataset. We observe that M_{2L} performs the best overall, although it yields a substantially coarser-grained view of the discussed themes in the threads (20 topics, 0.3% of the documents). We also note that M_+ outperforms M_{bi} achieving higher coherence, lower perplexity, and higher cross-lingual coverage ratio. We invite future studies to qualitatively assess the quality of the topics produced by the LDA models, such that the feasibility of employing such models

*Corresponding author
Email addresses: kundi@cs.queensu.ca (Kundi Yao), gustavo@cs.queensu.ca (Gustavo A. Oliva), ahmed@cs.queensu.ca (Ahmed E. Hassan), md.asaduzzaman@lakeheadu.ca (Muhammad Asaduzzaman), amalton@blackberry.com (Andrew J. Malton), awalenstein@blackberry.com (Andrew Walenstein)

in practice can be better determined.

Keywords: Technical support, logs, LDA, multilingual LDA, topic models

1. Introduction

Software logs are a valuable resource in the diagnosis of failures in software systems. When end-users describe software failures, they typically combine natural language (e.g., English) and logs. Examples of channels in which end-users describe technical problems with a combination of natural language and logs include (i) tickets in IT ticket systems (e.g., Freshdesk[1]), (ii) online chat with customer technical support (e.g., LiveChat[2]), and (iii) posts in technical support forums (e.g., Bleeping Computer[3]). We refer to discussion threads happening through these channels as *support threads*.

There might be latent associations between pieces of text written in natural language and certain log excerpts in support thread datasets. For instance, there might be multiple reports of a specific operating system crash that is associated with the log excerpt `VIDEO_TDR_FAILURE (igdkmd64.sys)`. Discovering such latent associations enables the construction of a knowledge base that links problem discussions to log excerpts that are typically associated (requested/provided) with those discussions. Ultimately, such a knowledge base speeds up failure debugging and technical support service quality. The central question that we address in this paper is thus **how can one find associations between natural language and logs?**

Similar to previous research where source code and natural language are treated as separate channels [1], the problem of establishing connections between text and logs can be seen as the problem of establishing connections between *pieces of text that are written in different languages*. More specifically,

[1] `http://freshdesk.com`

[2] `https://www.livechat.com`

[3] `https://www.bleepingcomputer.com`

system logs produced by a given tool (e.g., MiniToolBox[4]) can be seen as a semi-formal language that has its own syntax, vocabulary, and semantics. Recently, bilingual LDA models [2, 3, 4] have been proposed to address this very problem. Although monolingual LDA has been extensively used in Software Engineering (SE) research [5], the extent to which bilingual LDA works for SE data is unknown. In particular, in the context of support threads, natural language and logs are often loosely related (as opposed to being direct *translations* of one another).

In this work, we conduct an exploratory study where we investigate whether bilingual LDA models are suitable for uncovering associations between English text and system logs. We evaluate three candidate bilingual models, namely: (i) enriched LDA (M_+), (ii) two-layer LDA (M_{2L}), and (iii) Mallet Bilingual LDA (M_{bi}). The first two models were designed by us and the third one is an off-the-shelf solution provided by the Mallet topic modeling toolkit[5]. More specifically, we evaluate and compare these models along with three perspectives: internal quality, inferential power, and language alignment. The internal quality measures the semantic similarities of top words within the topic. The inferential power refers to the competency of a model in estimating unseen documents. And the language alignment measures the proportion of topics with top words from different languages. To evaluate the candidate models, we use data from the Bleeping Computer forum. Bleeping Computer is an information security and technology news website that offers a technical support forum where end-users can post their problems and get help from senior Bleeping Computer community members. Our research questions and associated key results are as follows:

RQ1) What is the achieved level of internal quality by each candidate model? LDA models need to produce a coherent set of topics (i.e., the set of words that describe a topic should be semantically connected). We evaluated

[4] `https://www.bleepingcomputer.com/download/minitoolbox`

[5] `http://mallet.cs.umass.edu`

the candidate models using the *topic coherence* metric (the higher, the better), which scores each topic by measuring the degree of semantic similarity between high scoring words in that topic. We observed that:

The M_{2L} model achieves the highest level of internal quality (highest coherence).

RQ2) What is the achieved level of inferential power by each candidate model? LDA models are useful when they can correctly guess the topic of a new, unseen document. We evaluated the candidate models using the *perplexity* metric (the lower the better), which measures how "perplexed" an LDA model gets when it encounters the words of a new, unseen document. We observed that:

The M_{2L} model achieves the highest inferential power (lowest perplexity).

RQ3) What is the achieved level of language alignment by each candidate model? In the context of support threads, LDA models become more useful when they produce topics that contain both English words and log words. We evaluated our candidate LDA models using our proposed *language alignment* metric. This metric corresponds to the ratio of LDA topics that contain both English words and log words. We observed that:

The M_{2L} model achieves the highest language alignment (highest cross-lingual coverage ratio).

We conducted a follow-up study to determine which model yields the best balance between the three quality attributes of interest. To quantify such a balance, we define an *aggregated score* that ranges from zero to one. Our conclusions are as follows. Although M_{2L} unsurprisingly achieves the highest aggregated score (0.87), it provides a substantially coarser-grained view of the discussed themes in the documents. Comparison between M_{+} (aggregated score of 0.77) and M_{bi} (aggregated score of 0.69) reveals that M_{+} achieves a higher overall performance. When single measure is preferred, M_{+} produces topics with higher coherence, while M_{bi} produces topics with lower perplexity. Therefore, the choice between those two models depends on which measure one prioritizes.

Paper structure. Section 2 describes how we collected data from the Bleep-

ing Computer forum. Section 3 describes our candidate LDA models. Section 4 shows the motivation, the approach, and the results associated with each of our research questions. Section 5 describes a follow-up study in which we determine what candidate model achieves the best balance between internal quality, inferential power, and language alignment. Section 6 describes related work revolving around multilingual LDA as well as LDA being applied to Software Engineering. Section 7 discusses the threats to the validity of our study. Finally, Section 8 states our concluding remarks.

2. Data Collection

We collect data from the community question-answering forum of the Bleeping Computer website. In the following, we describe this data source as well as our approach for collecting the data.

2.1. Data Source

The Bleeping Computer forum is organized into a number of *sections* and *subsections*. For instance, there exists a *Microsoft Windows Support* section, which centers discussion around that operating system. Its subsections include (but are not limited to) *Windows Crashes and Blue Screen of Death (BSOD) Help and Support*, *Windows 8 and 8.1*, and *Windows 10 Support*. A subsection might contain other subsections. For instance, the *Windows 10 Support* subsection contains two other subsections: *Windows 10 Discussion* and *Windows 10 Insider Preview Builds*.

Users can ask technical *questions* by starting a *thread* on a given forum subsection. Any user can respond to a question by providing one or more *replies*. In a reply, forum members typically ask for question clarifications, suggest the question-poster to run a diagnosis software tool, examine the produced logs by a diagnosis software tool, and offer potential solutions to the problem. We use the term *post* to refer to either the question or any reply to it. A thread thus comprises the set of posts corresponding to the question and its associated

replies. A thread contains English text and *may* contain system logs. For the sake of simplicity, we henceforth refer to the former as *text* and to the latter as *logs*.

2.2. Approach

In the following, we describe our data collection approach. An overview is shown in Figure 1.

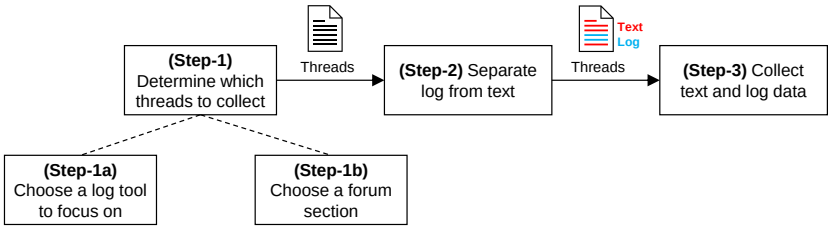


Figure 1: An overview of our the data collection process.

Step 1: Determine which threads to collect. We collect all threads that (i) contain log snippets that are produced by the MiniToolBox tool and (ii) belong to the *am-i-infected-what-do-i-do* forum section. We explain our rationale below.

– *Step 1a: Choose a log tool to focus on.* The produced logs from each log tool differ in terms of structure, vocabulary, and grammar. We thus consider that each log tool produces logs in a different language. Therefore, we choose a log system to focus on. While inspecting a random sample of threads from Bleeping Computer, we notice that users typically submit logs that are produced by the following log tools: Trend Micro HijackThis, DDS, MiniToolBox, ComboFix, and OTL. To choose the tool, we take a step back and reflect on the goals of our study. We wish to uncover associations between text and logs. As such, it does not make sense to pose constraints on how the text or logs should look like. In the most general case, logs will contain collected troubleshooting data from several sources (e.g., different hardware components and different parts of the operating system). Hence, we opt to choose the log tool that produces the most complete type of log. Upon an analysis of each tool, we selected

MiniToolBox. This tool focuses on troubleshooting network-related problems. It is also capable of producing logs that pack information from several parts of the Microsoft Windows operating system, including: content of the host files, IP configurations, Winsock entries, last 10 Event Viewer log errors, installed programs, installed devices, registered user accounts, disk partitions, memory size, and Windows restore points. In contrast, the logs of other tools tend to be simpler and contain less heterogeneous types of information. For instance, Trend Micro HijackThis aids in the detection of Windows malware by scanning a computer and producing a log of the running processes and suspicious Windows registry keys.

– *Step 1b: Choose a forum section.* We collect data from the *am-i-infected-what-do-i-do* forum section, which contains 36,477 threads. The rationale is that this section has the largest ratio of threads containing log excerpts (21.8%) among all sections in the Bleeping Computer forum. Next, since MiniToolBox logs typically start with either of the two following sentences: "MiniToolBox by Farbar Version:dd-mm-yyyy" or "Scan result of Farbar Recovery Scan Tool (FRST) (x64) Version:dd-mm-yyyy", we select the threads that contain MiniToolBox logs based on such patterns. In total, we collect 7,969 threads in this step.

Step-2: Separate log from text. Since we only consider MiniToolBox logs and text, logs generated from other diagnosis software tools are considered noise data which shall be removed. In this step, we discuss our approaches to identifying both *complete* and *incomplete* log excerpts.

– *Identify complete log excerpt.* Diagnosis software diagnosis tools usually include a prefix pattern with the tool's name in its generate logs, as we discussed in Step 1b. In this step, we summarize patterns from different diagnosis software tools and leverage such patterns to separate logs from text. First, we study the diagnosis software tools that are both hosted on the Bleeping Computer web-

site [6] (*internal tools*) and referred from external sources (e.g., ESET Online Scanner [7]) (*external tools*). From all *internal tools*, we choose those that occur in studied threads through keyword matching. We then extract and examine external links in all posts to summarize all *external tools*. Finally, we create regular expression rules for each tool based on the characteristics and availability of their corresponding logs, and such rules are leveraged to identify logs from texts.

– *Identify incomplete log excerpt.* We observe incomplete log excerpts (e.g., part of a diagnosis log) which are not identified by regular expression rules in certain posts. Inspired by the work of Bettenburg et al. [6], we leverage a spellchecker to determine the proportion of unknown words (i.e., words not included in the English dictionary) in a data excerpt. Data excerpts with $n > t$ unknown terms are considered log data and vice versa, where t is some predefined threshold. In order to determine a suitable threshold t, we first manually build a dataset from our studied threads containing 500 text excerpts and 500 log excerpts. Next, we (i) select 10 random samples with 50 text excerpts and 50 log excerpts, (ii) run *pyspellchecker* [8], (iii) and search for the threshold that maximizes the accuracy of log and text identification on each sample (from 0.3 to 1.0, with steps of 0.01). We use the median of the 10 threshold values as our optimal threshold (0.685).

Step 3: Collect text and log data. For each selected thread, we collect text and log data. Text data comprises the English text found in all posts of a given thread, including the question being asked (thread name). Log data is obtained as per Step-2. We collect all MiniToolBox log snippets that appear in a given thread.

[6] `https://www.bleepingcomputer.com/download/windows/`
[7] `https://www.eset.com/int/home/online-scanner/`
[8] `https://pypi.org/project/pyspellchecker/`

Summary

- Data source: Bleeping Computer.
- Dataset: All data in the *am-i-infected-what-do-i-do* forum as of December 15th, 2020.
- Pieces of collected data: Discussion threads containing both English text and log snippets produced by MiniToolBox.
- Total number of collected threads: 7,969.

3. Computation of the LDA Models

In this section, we first introduce the data preprocessing procedures that we performed according to the different natures of log and text. Next, we describe how we compute our candidate LDA models. Finally, we demonstrate how we implement these models.

3.1. LDA-Oriented Data Preprocessing

Each input document corresponds to a thread that we collected from the Bleeping Computer forum. We remove line breaks and any HTML tags from both log and text parts of each input document. We also replace sequences of whitespaces with a single space and remove special characters (e.g., '). Next, we apply tokenization and separate the words.

Text. We perform *stop word removal* and *lemmatization*, which are common LDA-oriented data preprocessing techniques for natural languages, to further process text words.

Logs. Logs are computer-generated and thus have their own structure, vocabulary, and grammar. Compared to natural language, log data tends to be more repetitive and with fewer unique words [7]. Logs also comprise system-specific words (e.g., *block_9*) that consist of more than alphabetical characters (which form most natural language words). Furthermore, log words (e.g., *getProductID*) do not follow the same composition and syntax as natural language.

Therefore, the common text preprocessing rules such as *lemmatization* are not applicable to logs. Hence, we choose *not* to preprocess logs.

3.2. Candidate LDA Models

In the context of topic modeling, a topic model is meant to *discover* topics in a corpus (collection of documents), and a topic is a distribution over terms that is biased around those associated with a single theme [8]. Specifically, Latent Dirichlet Allocation (LDA) [9] is a probabilistic topic model that has been recurrently studied and adopted in prior software engineering research [5]. Figure 2 shows a standard topic modeling pipeline with monolingual LDA as a reference model.

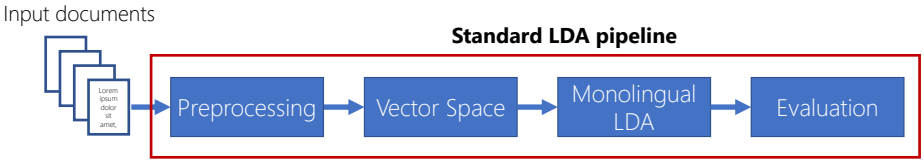
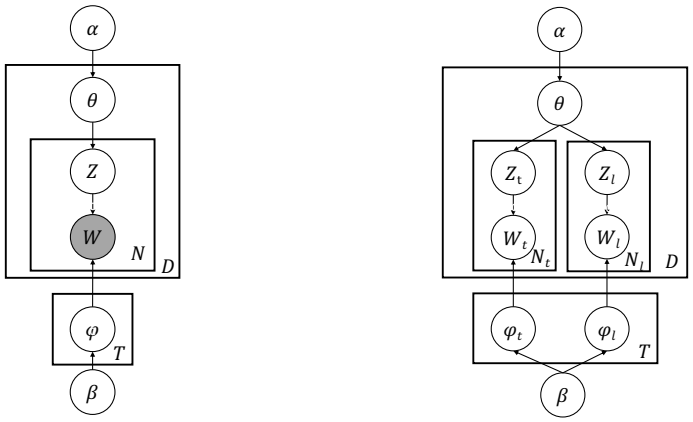


Figure 2: Standard LDA pipeline.

To handle documents containing both natural language (English) and logs, we design three new LDA model pipelines. Our first two pipelines use monolingual LDA models, while the third uses a multilingual LDA model. The plate notation of monolingual and multilingual LDA models is shown in Figure 3.

Our three candidate LDA-based topic model pipelines are shown below (for brevity purposes, we refer to them as our *candidate LDA models*):

- *Enriched LDA (M_+):* adds a prefix to the words of the original documents, such that text and log words can be differentiated.
- *Two-layer LDA (M_{2L}):* uses internal LDA runs to summarize documents in an effort to counterbalance the prevalence of text in comparison to logs.
- *Off-the-shelf bilingual LDA (M_{bi}):* treats text and logs as different languages and uses Mallet's bilingual LDA implementation.



(a) Monolingual LDA (b) Multilingual LDA

Figure 3: Plate notation of adopted LDA models. α and β are the parameters of the Dirichlet prior over θ (per-document topic distribution) and φ (per-topic word distribution). W and Z denotes each word and the topic of each word. (e.g., $W_{d,n}$ is the n^{th} word of the d^{th} document and $Z_{i,j}$ is the topic of the j^{th} word of the i^{th} document). N, D, and T represents the total words in a document, total number of documents in a corpus, and the total number of topics, respectively.

In the following, we describe our candidate models in detail. For each candidate model, we describe (i) the rationale behind its design and (ii) how we construct it.

3.2.1. Enriched LDA (M_+)

An overview of our Enriched LDA (M_+) candidate model is shown in Figure 4.

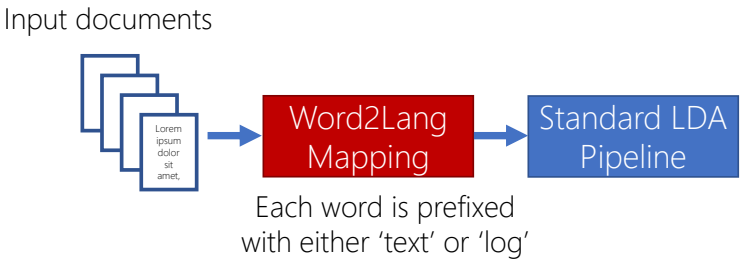


Figure 4: Enriched LDA (M_+) pipeline.

• *Rationale*: Standard LDA models are monolingual, which means that they cannot align topics across languages. Nevertheless, standard LDA models can still produce topics containing words from multiple languages if run on multilingual input documents. Such multilingual topics can uncover relationships between words from different languages (e.g., text and logs). The key drawback in this scenario is the lack of an explicit indication of the language in which a topic word is written in. Our Enriched LDA model addresses such a drawback.

• *Construction:* Our Enriched LDA model performs a preemptive data preprocessing step before pushing the input documents to the standard LDA pipeline. Each text word is prefixed with the term *text_* and each log word is prefixed with the term *log_*. We refer to this prefixing process as *Word2Lang* mapping. As a result, the standard LDA pipeline produces topics containing words that explicitly indicate the language in which they are written.

Pros and cons:

+ Simple and straightforward to interpret

+ Topics tend to be coherent (i.e., represent themes).

− If text is more prevalent than logs, then topics are more likely to contain text words only.

3.2.2. Two-layer LDA (M_{2L})

An overview of our two-layer LDA (M_{2L}) candidate model is shown in Figure 5.

• *Rationale:* Our Enriched LDA candidate model (M_{+}) is biased towards producing topics that frequently include text words only when text is more prevalent than logs. Indeed, logs tend to be repetitive and contain a much lower number of unique words compared to text [7]. Our two-layer LDA candidate model aims to tackle the imbalance between text and logs.

• *Construction:* We design a two-layer LDA pipeline. In the first layer, we prefix words with either text or log (analogously to M_{+}). Next, we generate

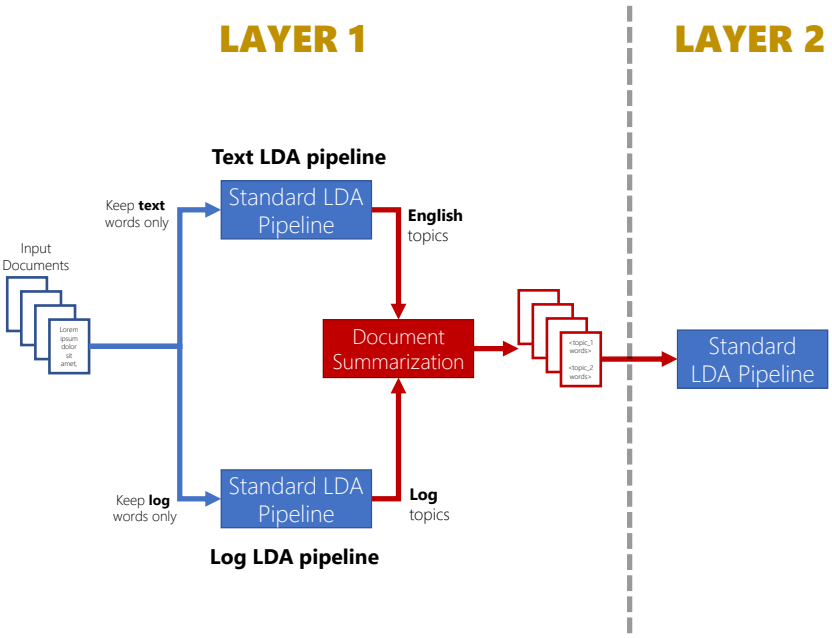


Figure 5: Two-layer LDA (M_{2L}) pipeline.

two topic models: one using text words only (*LDA-LY1-text*) and another using log words only (*LDA-LY1-log*). Once these models are built using the standard LDA pipeline, we proceed to the *document summarization* step. In this step, we replace the contents of the original documents with their text topics and log topics. More specifically, for each topic model, we first identify the document-specific topic according to the topic probabilities, then we extract the top 10 associated terms by term probability. To identify the origin of each term, we also append the topic indexes at the end of each term (e.g., LOG:stamp;TOPIC:399). The top terms extracted from both models are then merged by documents. Finally, the newly created documents now act as input to a third LDA topic model (layer 2).

Pros and cons:

+ Designed to counter-balance the prevalence of English words.
− More complicated.
− Higher computational cost (three LDA models across two layers).

3.2.3. Off-the-shelf bilingual LDA (M_{bi})

An overview of the off-the-shelf bilingual LDA (M_{bi}) candidate model is shown in Figure 6.

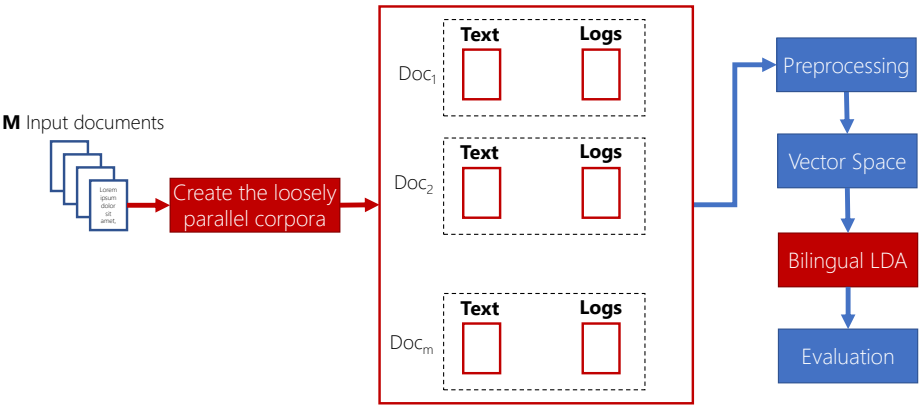


Figure 6: Off-the-shelf bilingual LDA (M_{bi}) pipeline.

• *Rationale:* Text and logs can be seen as belonging to two inherently different languages. Yet, we expect that the text and logs in a given forum thread will have some relationship. Multilingual LDA models are designed to find topic alignments across documents written in different languages. However, practitioners should be cautious when choosing a proper multilingual LDA model since it usually requires comparable corpora. The weak alignment between different languages may introduce noise and thus impair the effectiveness of the topic modeling process.

• *Construction:* We use Mallet's multilingual LDA implementation, which is one of the most popular, readily available multilingual LDA implementations. Most importantly, Mallet's multilingual LDA implements the pLDA (a.k.a.,

pLTM) algorithm [10], which is arguably suitable for *loosely parallel corpora* (a.k.a., *loosely comparable corpora*). Loosely parallel corpora can be formalized as a set of document pairs $\langle d_1, d_2 \rangle$ where d_1 and d_2 are each written in a different language, yet they share *similar* content (i.e., similar themes) [11]. In other words, Mallet's implementation does *not* require d_1 and d_2 to be direct translations of one another (i.e., a parallel corpora). The authors of pLDA observe that (i) "pLDA is appropriate for aligning topics in corpora that have a small subset of comparable documents" and (ii) "relatively small numbers of topically comparable document tuples are sufficient to align topics between languages in non-comparable corpora" [10]. Hence, we choose to build our model using Mallet's multilingual LDA implementation. The input to pLDA is the loosely parallel corpora, which we build by extracting the text and log parts of each document (Figure 6).

Pros and cons:

+ Prioritizes language alignment.
+ Easy to use and readily available (off-the-shelf solution)
+ Lower computation cost compared to our *two-layer LDA* candidate model.
− Text and logs are very loosely related. Off-the-shelf implementations such as Mallet's may thus struggle to align text and logs, which can result in fewer discovered topics and/or less coherent topics.

3.3. Implementation of Candidate Models

We modify gensim's Mallet LDA wrapper[9] to implement the candidate LDA models. We added multiple features based on the original wrapper to accommodate our requirements, including (i) supporting multilingual LDA modeling and analysis, (ii) calculating different evaluation metrics, and (iii) extending configurations for Mallet modeling. The detailed implementation is available in

[9] `https://radimrehurek.com/gensim_3.8.3/models/wrappers/ldamallet.html`

our replication package[10].

4. Results

In this section, we present our evaluation results to three proposed research questions (RQs). In each RQ, we explain the motivation and introduce corresponding approaches prior to discussing the experimental results.

4.1. RQ1: What is the achieved level of internal quality by each candidate model?

Motivation. We measure the quality of the generated topic models in an effort to understand the effectiveness of those models in capturing the underlying topics from a corpus. High-quality LDA models generate coherent and interpretable topics that can provide practitioners with valuable insights into the data for downstream tasks such as pattern discovery and text summarization – ultimately supporting practitioners in identifying the root underlying cause of failures.

Approach. We calculate the topic coherence achieved by each candidate LDA model after hyper tuning their parameters. Subsequently, we compare the coherence scores obtained. In the following, we explain our approach in more detail:

• *Step 1) Coherence calculation.* A set of statements or facts is said to be coherent if they support each other [12]. Several metrics have been proposed to quantify the coherence of a fact set (e.g., a topic in LDA) [13, 14, 15, 16]. We use the C_V measure to calculate topic coherence. In simple terms, C_V measures the relative distance between words within a topic and outputs an average computed over all topics. The C_V measure ranges from 0.0 to 1.0. We use *gensim* to calculate C_V.

[10] `https://bit.ly/3Tdlp81`. Once the paper is accepted, this replication package will be made publicly available on GitHub

It is challenging to interpret the absolute values of C_V, since there is no established nor commonly used guideline. In our practical experience using LDA and C_V [5, 17, 18, 19, 20], we observe that extreme values for C_V are unlikely. Values too close to zero (0.0 to 0.2) often indicate a data collection problem (e.g., corrupt data). Similarly, values too close to one (0.8 to 1.0) often indicate that the top words in a topic always appear in the input documents as bigrams or trigrams, which is uncommon. The values in between tend to occur much more frequently, especially those between 0.4 and 0.6. Some LDA practitioners seem to share a similar interpretation of the C_V values[11]. Hence, we use the following thresholds to interpret the C_V measure.

$$\text{Coherence}(C_V) = \begin{cases} \text{Abnormally low}, & \text{if } 0.0 \leq |C_V| < 0.2 \\ \text{Very low}, & \text{if } 0.2 \leq |C_V| < 0.4 \\ \text{Low}, & \text{if } 0.4 \leq |C_V| < 0.5 \\ \text{Medium}, & \text{if } 0.5 \leq |C_V| < 0.55 \\ \text{High}, & \text{if } 0.55 \leq |C_V| < 0.6 \\ \text{Very high}, & \text{if } 0.6 \leq |C_V| < 0.8 \\ \text{Abnormally high}, & \text{if } 0.8 \leq |C_V| \leq 1.0 \end{cases}$$

• *Step 2) Finding the LDA parameter set that optimizes coherence.* LDA models have three key parameters: *alpha*, *beta*, and *the number of topics*. LDA uses Bayesian inference, which relies on the notion of *prior distributions*. Alpha is a parameter of the prior distribution of topics over documents. A low alpha value puts more weight on having each document composed of only a few dominant topics. Beta is a parameter of the prior distribution of words over topics. A low beta value puts more weight on having each topic composed of only a few dominant words. The number of topics indicates how many topics the LDA

[11] https://stackoverflow.com/questions/54762690/what-is-the-meaning-of-coherence-score-0-4-is-it-good-or-bad

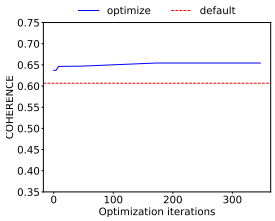
model should produce.

In this step, we aim to find the parameter set that optimizes the coherence of each candidate model (this set is likely different for each model). There are multiple hyperparameter tuning algorithms that can be used to achieve this goal. For instance, Agrawal et al. [21] employed Differential Evolution (DE), Panichella et al. [22] used Genetic Algorithms (GA), and popular LDA tools such as Mallet use Grid Search. Hence, there is no clear guideline indicating which algorithm should be used in each situation. Therefore, we choose to delegate this decision to OpenTuner [23]. OpenTuner uses ensembles of disparate search techniques simultaneously. Techniques that perform well receive larger testing budgets and techniques that perform poorly are disabled. As such, OpenTuner adapts to the problem at hand. Additionally, OpenTuner is extensible by design and supports flexible specification of hyperparameters and objective functions. We run OpenTuner on a Canada Compute cluster called Cedar[12]. In terms of resource allocation, we requested 32 CPU and 64Gb of RAM in total. We optimize one model at a time with a separate job for each. The execution time budget that we allocated for each OpenTuner job is LDA-dependent: 48 hours for M_{+}, 48 hours for M_{bi}, and 48 x 3 = 144 hours for M_{2L} (since this model entails training three LDAs). We use the following search space (parameter range) in all cases: $0.001 \leq \alpha \leq 50.0$; $0.001 \leq \beta \leq 50.0$; $10 \leq numTopics \leq 800$.

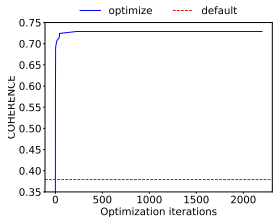
Findings. Observation 1) ***The M_{2L} model achieves the highest level of internal quality (highest coherence). However, it produces 4 to 65 times more topics than the other models.*** Figure 7 depicts the coherence scores that we obtained from the hyperparameter tuning of our three candidate models. In all three cases, hyperparameter tuning led to a higher coherence score compared to that produced with the parameters fixed at default values. M_{2L} ranks first with a final coherence of 0.73 (*very high*), M_{+} ranks second with a final coherence of 0.65 (*very high*), and M_{bi} ranks last with a coherence

[12] `https://docs.computecanada.ca/wiki/Cedar`

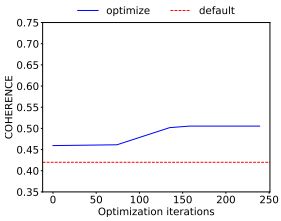
of 0.51 (*medium*).



(a) Coherence for the M_+ model



(b) Coherence for the M_{2L} model



(c) Coherence for the M_{bi} model

Figure 7: Coherence scores of our candidate LDA models (higher is better). The red dotted line serves as a reference and indicates the coherence score achieved when gensim's default values for alpha and beta are used to build the LDA model.

The optimal parameter-set for each model is shown in Table 1. As the table indicates, M_{2L} produces 4-65x more topics than the other candidate models. Hence, if practitioners care about coherence only, they may select either M_{2L} or M_+ depending on how coarse-grained they want the topics to be.

Table 1: Optimal parameter set for each candidate model when maximizing coherence.

	Alpha	**Beta**	**# Topics**	**Optimal Coherence**
M_+	24.02	6.25	157	0.65
M_{2L}	49.04	19.03	651	0.73
M_{bi}	6.07	11.59	10	0.51

4.2. RQ2: What is the achieved level of inferential power by each candidate model?

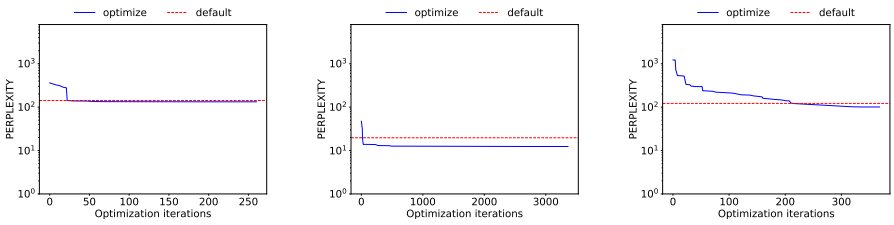
Motivation. LDA models are useful when they can correctly guess the topic of a new, unseen document. For instance, assume that an LDA found produced a topic T that contains both English words and log words. Now, assume that a forum member posts a problem P in a Bleeping Computer forum. If the LDA topic of P happens to be T and the LDA model in question has good inferential

power, then log-specific information (i.e., the log words in T) can be requested to the poster of `P`. The provision of such log-specific information is likely to help to address P.

Approach. We computed the *perplexity* of each candidate model. Perplexity is the geometric mean of the inverse marginal probability of each word in a held-out set of documents. Perplexity is considered a suitable measure to determine the predictive power of a topic model. Low perplexity indicates that the model is good at predicting unseen data. To determine the LDA parameter-set that minimizes perplexity, we employed the same approach of RQ1 (i.e., use of OpenTuner in a *Canada Compute* cluster).

Findings. **Observation 2)** ***The M_{2L} model achieves the highest inferential power (lowest perplexity).*** Figure 8 depicts the perplexity scores that we obtained from the hyperparameter tuning of our three candidate models. The hyperparameter tuning led to major perplexity drops for models M_+ and M_{bi}. M_{2L}'s perplexity remained fairly still, but it already started at a remarkably low level. In all cases, hyperparameter tuning led to a lower perplexity compared to that produced with the parameters fixed at their default values. M_{2L} ranks first with a final perplexity of 13.15, M_{bi} ranks second with a final perplexity of 100.88 (7.7 times higher than M_{2L}), and M_+ ranks last with a final perplexity score of 132.62 (10.1 times higher than M_{2L}, 1.3 times higher than M_{bi}).

The optimal parameter-set for each model is shown in Table 2. As the table indicates, M_{2L} and M_{bi} produce remarkably fewer topics than M_+. Different from the coherence optimization (RQ1), M_{2L} now produces the least number of topics. Hence, although coherence and perplexity are both popular LDA evaluation measures, we conclude that optimizing one in lieu of the other can lead to very different models.



(a) Perplexity for the M_+ model (b) Perplexity for the M_{2L} model (c) Perplexity for the M_{bi} model

Figure 8: Perplexity scores of our candidate LDA models (lower is better). The red dotted line serves as a reference and indicates the coherence score achieved when gensim's default values for alpha and beta are used to build the LDA model.

Table 2: Optimal parameter set for each candidate model when minimizing perplexity.

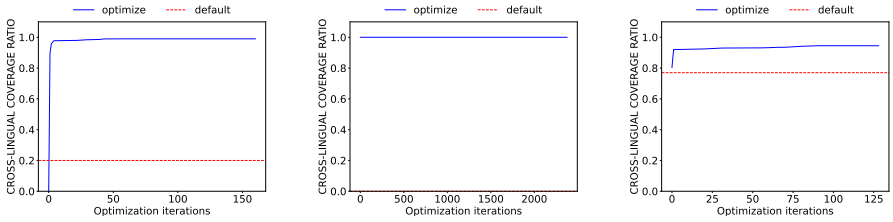
	Alpha	**Beta**	**# Topics**	**Optimal Perplexity**
M_+	9.25	0.06	184	132.62
M_{2L}	0.46	13.15	10	13.15
M_{bi}	5.12	0.00	502	100.88

4.3. RQ3: What is the achieved level of language alignment by each candidate model?

Motivation. In this work, we are interested in discovering associations between text and logs. Therefore, we assess the ability of our candidate models in discovering topics that contain both text and log words.

Approach. We refer to topics containing both log and text words as *cross-lingual topics*. Our approach consists of determining the coverage ratio of cross-lingual topics produced by each candidate model. We proceed analogously to RQ1 and RQ2.

Findings. **Observation 3)** ***The M_{2L} model achieves the highest cross-lingual coverage ratio.*** Figure 9 depicts the cross-lingual coverage ratios that we obtained from the hyperparameter tuning of our three candidate models. Hyperparameter tuning led to very high cross-lingual coverage ratios overall (at least 0.95 in all three models), clearly outperforming the default parameter configurations. More specifically, M_{2L} ranked first with a final cross-lingual coverage ratio of 1.0 which is very close to M_{+} with a final cross-lingual coverage ratio of 0.99, and M_{bi} ranked last with a final cross-lingual coverage ratio of 0.95.



(a) Cross-lingual coverage ratio for the M_{+} model

(b) Cross-lingual coverage ratio for the M_{2L} model

(c) Cross-lingual coverage ratio for the M_{bi} model

Figure 9: Cross-lingual coverage ratio scores for our candidate LDA models.

The optimal parameter-set for each model is shown in Table 3. The numbers of topics produced by the models are all generally high (at least 490). This result contrasts to those obtained in RQ1 and RQ2, where M_{bi} and M_{2L} produced 10 topics only.

Table 3: Optimal parameter set for each candidate model when maximizing cross-lingual coverage ratio.

	Alpha	Beta	# Topics	Optimal Crossl. Ratio
M_+	16.87	47.67	490	0.99
M_{2L}	44.77	33.70	694	1.00
M_{bi}	37.95	48.78	800	0.95

4.4. Discussion

In prior sections, we present the experimental results of our proposed candidate models. In this section, we further study the extracted topics from the candidate models to discover the associations between log and text words. Such a relationship shall provide extra information to assist in failure debugging and improving the technical support service quality.

We follow a similar approach as Panichella et al. [22] to understand the extracted topics using representative words. First, we choose the candidate models that achieve different optimal evaluation metric scores, respectively. Then for each model, we pick the most representative topics of each thread based on their probability in the corresponding topic distribution over documents. After mapping threads to topics, we then choose the top 20 words for each topic with the highest probability in its topic distribution. Finally, we manually inspect the acquired topics from each model. This analysis serves as an example to show the associations between extracted topics and the discussion threads. A selection of our manual study is showcased in Table 4.

[13] https://www.bleepingcomputer.com/forums/t/452705/windows-wont-update/

[14] https://www.bleepingcomputer.com/forums/t/422583/deleted-ipsecsysi-now-have-no-internet-connection/

[15] https://www.bleepingcomputer.com/forums/t/525169/unknown-chrome-infection/

[16] https://www.bleepingcomputer.com/forums/t/454549/vista-is-falling-apart/

[17] https://www.bleepingcomputer.com/forums/t/421442/unable-to-update-antivirus-definitions-and-windows-updates/

Table 4: Examples of selected threads to associate topics with representative words.

	Measure	Topic	Top Text Terms	Top Log Terms	Threads
M_+	Coherence	A	java,md5,document, gmer,dll,registri, repair,system32	update,fault,windows, hang,server,address, search,dcom,network, dll,bonjour,device	[T1] Windows Won't Update[13]
M_{2L}	Perplexity	B	registri,java,error, system32,repair,dll, adwclean,document, googl,md5	fault,windows,path, code,program,update, targetinst,schedule, stamp,bonjour	[T2] deleted ipsec.sys... I now have no internet connection[14]
					[T3] Unknown Chrome infection[15]
M_{bi}	Aggregated	C	essenti,connect,access, error,point,network, mse,java,machin,think	signature,type,update, engine,stage,source, current,previous, authority,user	[T4] Vista is falling apart[16]
					[T5] Unable to update antivirus definitions and windows updates[17]

As shown in Table 4, topic A contains a list of text and log words that produce latent connections where words can be used to identify and locate related issues. For instance, log word hang is related to system interruption or responding errors. Bonjour indicates logged errors related to the bonjour service of windows, which discovers and services devices from a local network. As for text words, the co-ocuurence of dll, system32, and md5 could indicate the procedure of integrity checking of critical windows system files. GMER is a rootkit detector that mines potentially malicious processes that are secretly running in the background. Such a word indicates a potential solution or at least a must-try step when issues related to those log words exist. For example, thread T1 [13] reports an issue that windows failed to update. Throughout the diagnosis, the fixing includes searching for a missing dll file, solving application hang, running GMER, etc.

From the top text words of Topic B, we infer that this topic may relate to registry and dll files under the system32 folder. Adwclean refers to leveraging Malwarebytes AdwCleaner, a junkware and malware removal tool, for

system cleaning. The corresponding log words such targetinst, fault, stamp are related to errors generated from different windows applications. For example, thread T2 [14] describes an internet connection error and a google search redirect error. Such issues are also reflected in system error logs. To resolve such issues, the expert suggests to copy a missing DLL file from a cache folder (C:/WINDOWS/system32/dllcache). Similarly, thread T3 [15] explains a malware infection in google chrome. The user performs diagnosis through multiple tools, with Adwcleaner used as the first option. We believe such knowledge assists users in quickly locating issues and finding solutions from others' expertise.

In Topic C, the top text words network and connect indicate that the topic is likely related to internet connection issues, and MSE (Microsoft Security Essentials) suggests that such issues may relate to virus or malware. For example, thread T5 [17] states that the user is unable to connect to or download updates from anti-malware sites. The system error logs indicate that Windows Anti-Malwarefails fail to update its signatures. Another example in thread T4 [16] mentions connection errors to local devices and internet with the similar signature issues in the system error log.

Our observation indicates that our approach is capable of revealing hidden associations between log and text through designated topic modeling models. Such information can be leveraged to better support and facilitate issue diagnosis procedures.

5. Follow-up study

The results that we obtained indicate that M_{2L} consistently performs best. Yet, we are left with three open issues: (i) the selection of the quality attribute of interest (i.e., internal quality, inferential power, and language alignment) has a huge impact on the final number of topics that M_{2L} produces, (ii) it is unclear which model ranks second, and possibly more importantly (iii) it is unclear how many topics would have been produced by each model if we chose the parameter set that yields the best balance between our three quality attributes of interest.

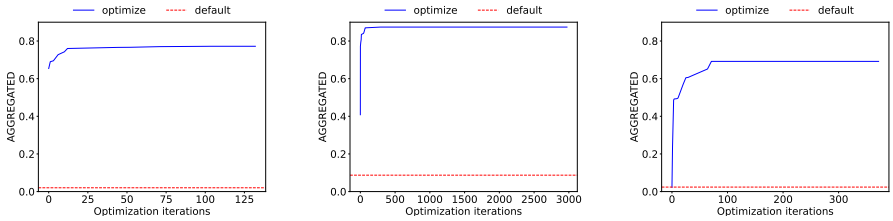
To determine the best balance between the three quality attributes, we (i) design a composite metric that takes into account coherence, perplexity, and inter-corpus and (ii) use that metric as the objective function during hyperparameter tuning. However, designing such a composite metric is not trivial. While both *coherence* and *inter-corpus ratio* range from zero to one, *perplexity* does not have boundaries. In other words, aggregating the base metrics in such a way that each base metric contributes the same to the final value is not trivial. To circumvent the problem, we artificially define the boundaries for perplexity based on empirical data. More specifically, we use the perplexity values from the previous perplexity-based hyperparameter optimization as a reference range to create a normalized perplexity value, as shown below.

$$\textit{Normalized Perplexity} = 1 - \frac{p - p\_ref_{min}}{p\_ref_{max} - p\_ref_{min}} \quad (1)$$

p is the current perplexity value, while p_ref_{max} and p_ref_{min} are the maximum and minimum perplexity values of the reference range. The perplexity values outside this range will be rounded to the nearest range boundary. Next, we composite the aggregated scores using the harmonic mean of coherence, cross-lingual coverage ratio, and the normalized perplexity.

Findings. Observation 4) ***The M_+ model outperforms the M_{bi} model, but by a small margin.*** Figure 10 depicts the aggregated scores that we obtained from the hyperparameter tuning of our three candidate models. As expected, the M_{2L} model ranks first with an aggregated score of 0.87. M_+ ranks seconds with an aggregated score of 0.77, which is 11.6% higher than that of M_{bi}. The figure also indicates that using the default parameter-set would lead generally lead to very poor models.

The optimal parameter-set for each model is shown in Table 5. The number of topics produced by the models differs substantially: 20 (0.3% of the documents), 64 (0.8% of the documents), and 780 (9.8% of the documents). Hence, although M_{2L} has the highest aggregated score, it is also providing a substantially coarser-grained view of the themes discussed in the documents. In



(a) Aggregated score for the M_+ model

(b) Aggregated score for the M_{2L} model

(c) Aggregated score for the M_{bi} model

Figure 10: Aggregated scores for our candidate LDA models.

addition, the high coherence and the low perplexity suggest that the *document summarization* step in the M_{2L} construction pipeline (Figure 5) is possibly too aggressive.

We now compare M_+ and M_{bi}. Analysis of the optimal parameter set for each model reveals that M_+ has a higher overall performance than M_{bi} in coherenece, perplexity, and cross-lingual coverage ratio. On the other hand, previous findings indicate that these two models outperform another when a single evaluation metric (i.e., coherence, perplexity) is perferred. Hence, in a practical scenario, a choice between these two models would be driven by a preference for coherence over perplexity or vice-versa (provided that the data has similar characteristics to the one that we investigate in this study).

Table 5: Optimal parameter set for each candidate model when maximizing aggregated score.

	Alpha	**Beta**	**# Topics**	**Coherence**	**Perplexity**	**Crossl. Ratio**	**Optimal Aggreg. Score**
M_+	20.60	2.44	780	0.63	231.87	0.87	0.77
M_{2L}	24.62	32.40	64	0.70	71.26	1.00	0.87
M_{bi}	13.13	28.07	20	0.52	400.59	0.80	0.69

6. Related Work

LDA models are widely studied and adopted in the field of software engineering for topic modeling in mining software repositories [5, 19, 21]. In this section,

we discuss the research related to our study from four areas of study: interplay between natural language and code in software engineering, multilingual LDA, tuning LDA for software engineering tasks, and applications of LDA in software engineering.

Interplay between natural language and code in software engineering. Natural language information in source code such as proper comments and identifier names makes it easier for humans to understand and evolve a given piece of code. Yet, from an algorithmic/computation perspective, such information is irrelevant. That is, the semantics of the code dictate the computation. Understanding this interplay between code and natural language has led to several research studies [1, 24, 25, 26]. For instance, Casalnuovo et al. [1] define a theory called *Dual Channel Constraints* (DCC) to highlight the relevance and interplay between the *algorithmic channel* and the *natural language* channel during software development. The theory's name stems from the realization that the two channels interact and constrain each other. Indeed, several problems arising from the interplay between the two channels have been studied in the software engineering literature.

The dual-channel nature of source code is usually leveraged for automated code generation and suggestion. Hindle et al. [24] find that source code is analogous to natural language that is likely to be repetitive and predictable. Based on this observation, the authors propose a statistical language model that outperforms a state-of-the-art IDE in code completion. Iyer et al. [27] propose an approach that automatically generates Java class member functions from dual information channels, which are the method documentation in natural language and the programmatic context from the rest of the code in a given class. Chen et al. [28] introduced *Codex*, a fine-tuned GPT language model that generates standalone Python functions based on their docstrings that are documented in natural language. *Codex* now empowers GitHub Copilot [18] to assist developers'

[18] `https://github.com/features/copilot`

daily programming tasks with automated coding suggestions. The dual-channel nature of source code is also corroborated and leveraged by Chakraborty et al. [29] to improve the performance of NLP models that are designated for source code. Their approach learns to write coherent code that is similar to developers' code, and it achieves state-of-the-art performance in multiple code generative tasks.

Another line of research focuses on *natural language* information to enhance the quality and understanding of programs. Tan et al. [30] mine implicit rules from code comments written in natural language to detect comments that are inconsistent with the corresponding source code that can lead to misunderstanding. Movshovitz-Attias and Cohen [31] predict comments from Java source files using a statistical language model. They find using comment completion approaches can save up to 47% comment typing. To assist developers in writing informative and self-explanatory comments, Louis et al. [32, 33] build a machine learning framework that takes code snippets and natural language comments as input to detect and remove redundant comments from source code and to predict locations in source code that would benefit from adding comments. Hu et al. [34, 35] capture the source code semantics and structure to automatically generate code comments for Java methods. The following research by Zhou et al. [36] introduces a novel approach to improve the effectiveness and robustness of existing deep learning models that generate code comments. Similarly, few studies focus on the challenge of automated logging text generation. He et al. [37] investigate the characteristics of logging descriptive text and propose an information-retrieval approach to automatically generate logging description texts. Ding et al. [38] propose a novel approach that generates texts in logging code by translating corresponding code snippets into meaningful descriptive texts with a neural machine translation model.

The dual-channel nature of source code is also leveraged to address other DevOps tasks, including predicting function types [39, 40, 41], recovering original code from obfuscating source code [42, 43], predicting defection with comments and bug reports [30, 44, 35], identifying and managing self-admitted technical

debt (SATD) [45], etc. In this vein, our work can be seen as a DCC study in which we aim to understand the relationship between log excerpts and natural language excerpts by means of a topic model.

Multilingual LDA. Multilingual topic models have been an active research area since the nineties [2, 3, 4, 10, 11, 46, 47, 48, 49, 50, 51]. Earlier studies in the area of multilingual topic models tried to align topics across languages by assuming that the documents written in different languages are parallel (e.g., direct translations of one another) or *highly comparable*. However, such an assumption seldom holds in practice. It is far more common for corpora to only have some degree of comparability (a.k.a., comparable corpora).

Recent approaches for aligning topics across comparable corpora span a variety of techniques: the polylingual topic model – *pLDA* [10] (a seminal work that extended LDA to the multilingual context and that is implemented in Mallet), polylingual tree LDA – *ptLDA* [49] (extends the polylingual topic model with tree-based LDA – *tLDA* [52]), *JointLDA* (uses a special bilingual dictionary to mine multilingual topics) [48], Multilingual Topic Anchoring – *MTAnchor* [50] (uses the *anchor words* algorithm [53] in lieu of the more common alternatives, such as LDA [9] and LSA [54]), and Multilingual Cultural-common Topic Analysis – *MCTA* [55] (uses a technique based on *auxiliary distributions*[56] that incorporates word distributions from the other language in an attempt to capture common semantics at the topic level).

More recently, Yang et al. [2] introduced a new multilingual topic model specifically designed to find topics across *loosely comparable* corpora. In loosely comparable corpora, although some topics are shared, emphasis may diverge and some topics may lack analogs in the other language. Their proposed model works by minimizing the Euclidean distances of translation pairs' *transformed* topic distributions. Transformations are learned from the original translation pairs' topic distributions and rely on a translation dictionary. In simpler terms, their model learns *weighted* topic links and only assigns a high link weight to topic pairs whose top words have many direct translation pairs. A topic

is left unlinked if there is no matching topic in the other language (i.e., it does not enforce linking). Such a characteristic makes their model more robust for loosely comparable corpora with high topic misalignment. An example is shown in Figure 11. The authors empirically show that their model substantially outperforms ptLDA, MTAnchor, and MCTA in both intra- and cross-lingual classification tasks for loosely comparable corpora.

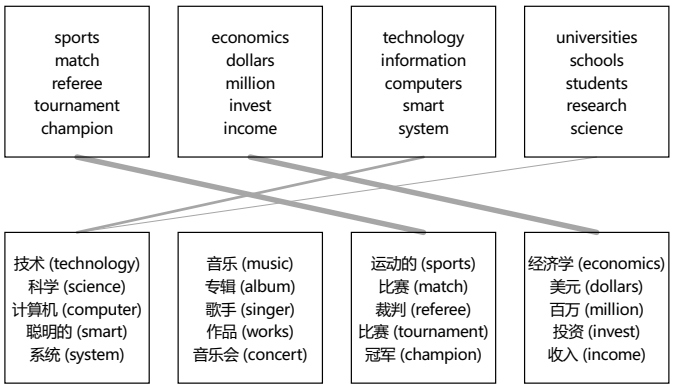


Figure 11: Illustration of topic links (adapted from the study of Yang et al. [2]). Thicker lines denote links with higher weights. Topic pairs with many word translation pairs have high link weights. Topic pairs with partial overlap receive lower weights. A topic is unlinked if there is no corresponding topic in the other language.

In our study, natural language (which is used to describe a technical problem, investigate a technique problem, and propose candidate solutions) is *very loosely* comparable to logs (which can be seen as machine language). Differently from Yang et al. [2], in our case, there is no pre-existing dictionary that can be leveraged to find direct correspondences between the words in natural language and those found in system logs. Hence, although Yang's model outperforms several multilingual models, we note that it is unsuitable in the context of our study. In addition, to the best of our knowledge, the implementation of Yang's model is not readily available.

Tuning LDA for Software Engineering tasks. Effectively applying LDA

in the software engineering context is not trivial. Panichella et al. [22] investigated how to effectively use LDA in three software engineering tasks, namely (i) traceability link recovery, (ii) feature location, and (iii) software artifact labeling. The authors conclude that LDA has a high sensitivity to different parameter settings and thus a careful calibration of those is essential. The authors propose to calibrate (hyperparameter tune) the LDA parameters using a genetic algorithm (GA). The decision variable in their study is the quality of an LDA-driven document clustering. This quality is measured using the *mean Silhouette coefficient*[57], which is a popular clustering quality metric. The authors observe that LDA parameter calibration using GA led to accuracy improvements in experiments involving the three aforementioned software engineering tasks.

Agrawal et al. [21] highlight that LDA is subject to *order effects* [58] due to its stochastic nature. Having order effects implies that the produced topics might be different every time LDA is run. The authors refer to this problem as *topic instability*. Topic instability confuses users and lowers the efficacy of text mining classifiers that rely on LDA to generate their input training data. To mitigate the problem, the authors propose an approach called LDADE that uses differential evolution (DE) [59] to hyperparameter tune LDA with the goal of making topics look more similar across runs. The authors conduct experiments with LDADE and conclude that (i) it dramatically reduces topic instability and (ii) it leads to accuracy improvements in text mining classification tasks. However, it is not clear whether optimizing for topic stability yields better topics or even an adequate number of topics.

Treude and Wagner [60] also recognize the need to calibrate LDA. They state that "there are only rough and sometimes conflicting guidelines available on how these parameters should be set." The authors conduct experiments with two large textual datasets: one from README files in GitHub and another from discussion threads on StackOverflow. The authors use the irace tool [61] to calibrate LDA hyperparameters and they choose *perplexity* to be the decision variable. The authors create the corpora by segmenting the original datasets per programming language. The authors conclude that each corpus required

different parameter settings in order to achieve good perplexity values (especially with regards to beta). They conclude that rules of thumb for topic modeling calibration are generally *not* applicable to their corpora.

In our paper, we acknowledge the high sensitivity of LDA to parameter choices. Differently from prior work, we not only optimize multiple variables (coherence, perplexity, and cross-lingual coverage ratio) but also an aggregated one. Our rationale is that all these variables are simultaneously important. In terms of the hyperparameter tuning tool/algorithm, we observe that there is no consensus in the literature at this point: researchers seem to choose the tool/algorithm based on the problem that they are trying to solve and the decision variable. To circumvent this problem, we use OpenTuner. OpenTuner is an ensemble technique that chooses the best search technique dynamically based on how they perform. More generally, evaluation of LDA models is actually research in itself [62, 63, 64, 65].

Applications of LDA in Software Engineering. LDA has been applied to support a plethora of software engineering tasks. In 2016, Chen et al. [5] published a survey that discussed enumerated these tasks: traceability link recovery, concept location, bug triaging, search engine usage analysis, requirements traceability, auditor support for exploring the implementation of requirements, analyzing bug report quality, estimating the number of topics in source code, clone detection, finding related code on the web, web service discovery, source code summarization and labeling, and refactoring. Since then, LDA has continued to be used to support software engineering tasks. More recent uses of LDA include: understanding the effects of testing on code quality [66], understanding system logs [19], and discovering the themes discussed by programmers in the context of blockchain technology [20].

7. Threats to Validity

Construct Validity. In our data collection, we extract discussion threads from the Bleeping Computer forum (Section 2). It is possible that a discussion thread

will be preemptively closed with a reference to a prior thread that discusses the same issue (a *reference thread*). Topic models such as LDA operate directly on word distributions over documents and do not account for document links (cross-references). Future work should investigate whether reference threads should be weighted higher during topic discovery.

In RQ1 (Section 4.1), we use the C_V measure to quantify topic coherence. However, there exist other popular measures for quantifying topic coherence, including: UCI coherence [14] (which relies on the notion of *pointwise mutual information* – PMI [67]), UMass coherence [15], and normalized PMI coherence (NPMI) [13, 16]. It is likely that using any of these measures would yield different coherence results compared to those that we obtained. Nevertheless, the C_V measure was found to be the top-performing measure in a systematic study of coherence measures using different datasets [12].

In this study, we adopt community-recognized thresholds for the interpretation of the C_V measure. Such thresholds derive from the knowledge and experience of practitioners in reasoning about C_V values. Hence, we believe that these thresholds are capable of providing insights into the evaluation of topic coherence. To the best of our knowledge, there is no gold standard in the literature for interpreting absolute C_V values. We thus call for future research to propose systematic ways to evaluate the C_V values of a topic model.

In our follow-up study (Section 5), we define an aggregated score that takes into account our three LDA qualities of interest (internal quality, inferential power, and language alignment). We assign the same weight to the three qualities of interest. Assigning different weights to different qualities would likely yield different results. Future research may also explore more sophisticated (including more computationally expensive) approaches for finding good trade-offs between the qualities of interest in an LDA model, such as multi-objective optimization [68]. Moreover, we normalize the perplexity measure in order to make it bounded and compatible with the other measures. Using other normalization techniques (e.g., $tanh()$ with adjustments) would possibly influence the final results as well. Ultimately, the take-home message of Section 5 is that multi-

ple quality metrics should be simultaneously considered when training an LDA model.

Internal Validity. In Section 3, we describe our three candidate LDA models. Our two-layer LDA (M_{2L}) summarizes documents in order to counterbalance the prevalence of English words compared to logs. In practice, such a summarization can be seen as an algorithm that *transforms* the original set of documents into another set of documents that have a different word distribution. That is, M_{2L} alters the data. Hence, even though we observe that LDA performs the best compared to the other candidate models, our results should be carefully scrutinized. An alternative design for M_{2L} would consist of (i) chaining the layer 1 LDA models as priors (input) to the third LDA and (ii) performing the Bayesian Inference through all three models. This is theoretically feasible, since the Dirichlet Process distribution satisfies prior conjugacy (i.e., the posterior distribution is also a Dirichlet process). Nevertheless, such an approach does not tackle the text/log imbalance problem (c.f., Section 3.2.2). We thus invite future work to adapt M_{2L} by replacing the summarization step with some other suitable preprocessing technique (e.g., term reweighing schemes [69]).

Agrawal et al. [21] discusses the problem of topic instability, which arises from the stochastic nature of LDA. In preliminary runs of our experiments, we configured OpenTuner to (i) build the same candidate model 10 times during every iteration and (ii) use the *median value* of the decision variable (e.g., C_V coherence) to inform its tuning process. We did not observe a significant change in results compared to running the tuning process in its standard form (single model construction per iteration). In other words, we did not observe significant topic instability in our candidate models. Nevertheless, future work should take topic instability into account when assessing our candidate models with other datasets.

The results from our paper heavily depend on the OpenTuner tool. OpenTuner is a robust and actively maintained autotuning tool with an active community around it. Using other autotuning tools (e.g., iRace [60, 61] or specific

tuning algorithms (e.g., genetic algorithms [22, 70]) may produce a different result.

External Validity. Although our results derive from a specific dataset from Bleeping Computer, our proposed approach is generalizable to evaluations on other datasets. Our study should be seen as a first attempt at linking English text to system logs using LDA-based models. We strongly encourage future work to assess our models on other datasets and compare the results to those that we obtained in this paper. In particular, future work should also assess our candidate models from a more qualitative perspective (e.g., determine whether discovered topics align with the intuition of domain experts).

8. Conclusion

In this paper, we investigate the problem of linking natural language (English) to machine language (system logs) in the (technical) support threads. Uncovering such links would enable the construction of knowledge base that connects technical problem discussions to log excerpts that are often associated (requested/provided) alongside those discussions.

As an initial effort towards addressing the aforementioned problem, we evaluate three LDA candidate models. Two of them are designed by us, which we refer to *enriched LDA* (M_+) and *two-layer LDA* (M_{2L}). The third one is an *off-the-shelf bilingual LDA* (M_{bi}) implementation made available in the Mallet package.

Our evaluation of the LDA candidate models is multi-faceted. More specifically, as opposed to recent LDA research within software engineering [60, 21, 22], we evaluate models alongside three axes instead of a single one, namely: internal quality, inferential power, and language alignment. These axes are operationalized by coherence, perplexity, and cross-lingual coverage ratio respectively. We evaluate our models using discussion threads from a Bleeping Computer forum.

Our results indicate that M_{2L} performs best overall. However, it provides a substantially coarser-grained view of the themes discussed in the documents,

since it produces very few topics (20, 0.3% of the documents). M_+ performs slightly better than M_{bi}. Comparing the two reveals that M_+ produces topics with higher coherence, while M_{bi} produces topics with lower perplexity.

Given the scale of the study and the heterogeneity of the discussion threads, it is difficult for us to qualitatively determine whether the topics produced by the models are adequate to be used in practice. We invite future studies to assess the qualities of topics produced by LDA models in a practical scenario. Researchers may also refer our evaluation approach to domain-specific datasets with the availability of domain experts, to determine the feasibility of employing our proposed models in practice.

References

[1] C. Casalnuovo, E. T. Barr, S. K. Dash, P. Devanbu, E. Morgan, A theory of dual channel constraints, in: G. Rothermel, D. Bae (Eds.), ICSE-NIER 2020: 42nd International Conference on Software Engineering, New Ideas and Emerging Results, Seoul, South Korea, 27 June - 19 July, 2020, ACM, 2020, pp. 25–28. URL: https://doi.org/10.1145/3377816.3381720. doi:10.1145/3377816.3381720.

[2] W. Yang, J. Boyd-Graber, P. Resnik, A multilingual topic model for learning weighted topic links across corpora with low comparability, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 1243–1248. URL: https://www.aclweb.org/anthology/D19-1120. doi:10.18653/v1/D19-1120.

[3] S. Hao, J. Boyd-Graber, M. J. Paul, Lessons from the Bible on modern topics: Low-resource multilingual topic model evaluation, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1

(Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 1090–1100. URL: https://aclanthology.org/N18-1099. doi:10.18653/v1/N18-1099.

[4] G. Balikas, Mining and Learning from Multilingual Text Collections using Topic Models and Word Embeddings, Theses, Grenoble 1 UGA - Université Grenoble Alpes, 2017. URL: https://hal.archives-ouvertes.fr/tel-01706347.

[5] T.-H. Chen, S. W. Thomas, A. E. Hassan, A survey on the use of topic models when mining software repositories, Empirical Softw. Engg. 21 (2016) 1843–1919. URL: https://doi.org/10.1007/s10664-015-9402-8. doi:10.1007/s10664-015-9402-8.

[6] N. Bettenburg, B. Adams, A. E. Hassan, M. Smidt, A lightweight approach to uncover technical artifacts in unstructured data, in: 2011 IEEE 19th International Conference on Program Comprehension, 2011, pp. 185–188. doi:10.1109/ICPC.2011.36.

[7] K. Yao, H. Li, W. Shang, A. E. Hassan, A study of the performance of general compressors on log files, Empirical Software Engineering 25 (2020) 3043–3085. URL: https://doi.org/10.1007/s10664-020-09822-x. doi:10.1007/s10664-020-09822-x.

[8] C. Wang, D. M. Blei, Collaborative topic modeling for recommending scientific articles, in: C. Apté, J. Ghosh, P. Smyth (Eds.), Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011, ACM, 2011, pp. 448–456. URL: https://doi.org/10.1145/2020408.2020480. doi:10.1145/2020408.2020480.

[9] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.

[10] D. Mimno, H. M. Wallach, J. Naradowsky, D. A. Smith, A. McCallum, Polylingual topic models, in: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2009, pp. 880–889. URL: https://aclanthology.org/D09-1092.

[11] Z. Zhu, M. Li, L. Chen, Z. Yang, Building comparable corpora based on bilingual LDA model, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, Sofia, Bulgaria, 2013, pp. 278–282. URL: https://aclanthology.org/P13-2050.

[12] M. Röder, A. Both, A. Hinneburg, Exploring the space of topic coherence measures, in: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 399–408. URL: https://doi.org/10.1145/2684822.2685324. doi:10.1145/2684822.2685324.

[13] G. Bouma, Normalized (pointwise) mutual information in collocation extraction, in: From Form to Meaning: Processing Texts Automatically, Proceedings of the Biennial GSCL Conference 2009, volume Normalized, Tübingen, 2009, pp. 31–40.

[14] D. Newman, J. H. Lau, K. Grieser, T. Baldwin, Automatic evaluation of topic coherence, in: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, HLT '10, Association for Computational Linguistics, USA, 2010, p. 100–108.

[15] D. Mimno, H. M. Wallach, E. Talley, M. Leenders, A. McCallum, Optimizing semantic coherence in topic models, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11, Association for Computational Linguistics, USA, 2011, p. 262–272.

[16] N. Aletras, M. Stevenson, Evaluating topic coherence using distributional semantics, in: Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers, Association for Computational Linguistics, Potsdam, Germany, 2013, pp. 13–22. URL: https://aclanthology.org/W13-0102.

[17] T.-H. Chen, S. W. Thomas, M. Nagappan, A. E. Hassan, Explaining software defects using topic models, in: Proceedings of the 9th IEEE Working Conference on Mining Software Repositories, MSR '12, IEEE Press, 2012, p. 189–198.

[18] A. Barua, S. W. Thomas, A. E. Hassan, What are developers talking about? an analysis of topics and trends in stack overflow, Empirical Softw. Engg. 19 (2014) 619–654. URL: https://doi.org/10.1007/s10664-012-9231-y. doi:10.1007/s10664-012-9231-y.

[19] H. Li, T.-H. P. Chen, W. Shang, A. E. Hassan, Studying software logging using topic models, Empirical Softw. Engg. 23 (2018) 2655–2694. URL: https://doi.org/10.1007/s10664-018-9595-8. doi:10.1007/s10664-018-9595-8.

[20] Z. Wan, X. Xia, A. E. Hassan, What do programmers discuss about blockchain? a case study on the use of balanced lda and the reference architecture of a domain to capture online discussions about blockchain platforms across stack exchange communities, IEEE Transactions on Software Engineering 47 (2021) 1331–1349. doi:10.1109/TSE.2019.2921343.

[21] A. Agrawal, W. Fu, T. Menzies, What is wrong with topic modeling? and how to fix it using search-based software engineering, Information and Software Technology 98 (2018) 74–88. URL: https://www.sciencedirect.com/science/article/pii/S0950584917300861. doi:https://doi.org/10.1016/j.infsof.2018.02.005.

[22] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, A. De Lucia, How to effectively use topic models for software engineering tasks? an

approach based on genetic algorithms, in: Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, IEEE Press, 2013, p. 522–531.

[23] J. Ansel, S. Kamil, K. Veeramachaneni, J. Ragan-Kelley, J. Bosboom, U.-M. O'Reilly, S. Amarasinghe, Opentuner: An extensible framework for program autotuning, in: Proceedings of the 23rd International Conference on Parallel Architectures and Compilation, PACT '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 303–316. URL: https://doi.org/10.1145/2628071.2628092. doi:10.1145/2628071.2628092.

[24] A. Hindle, E. T. Barr, Z. Su, M. Gabel, P. T. Devanbu, On the naturalness of software, in: M. Glinz, G. C. Murphy, M. Pezzè (Eds.), 34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland, IEEE Computer Society, 2012, pp. 837–847. URL: https://doi.org/10.1109/ICSE.2012.6227135. doi:10.1109/ICSE.2012.6227135.

[25] M. Allamanis, D. Tarlow, A. D. Gordon, Y. Wei, Bimodal modelling of source code and natural language, in: F. R. Bach, D. M. Blei (Eds.), Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, volume 37 of *JMLR Workshop and Conference Proceedings*, JMLR.org, 2015, pp. 2123–2132. URL: http://proceedings.mlr.press/v37/allamanis15.html.

[26] M. Allamanis, E. T. Barr, P. T. Devanbu, C. Sutton, A survey of machine learning for big code and naturalness, ACM Comput. Surv. 51 (2018) 81:1–81:37. URL: https://doi.org/10.1145/3212695. doi:10.1145/3212695.

[27] S. Iyer, I. Konstas, A. Cheung, L. Zettlemoyer, Mapping language to code in programmatic context, in: E. Riloff, D. Chiang, J. Hockenmaier, J. Tsujii (Eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, Association for Computational Linguistics, 2018, pp. 1643–1652.

URL: https://doi.org/10.18653/v1/d18-1192. doi:10.18653/v1/d18-1192.

[28] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, W. Zaremba, Evaluating large language models trained on code, CoRR abs/2107.03374 (2021). URL: https://arxiv.org/abs/2107.03374. arXiv:2107.03374.

[29] S. Chakraborty, T. Ahmed, Y. Ding, P. T. Devanbu, B. Ray, Natgen: generative pre-training by "naturalizing" source code, in: A. Roychoudhury, C. Cadar, M. Kim (Eds.), Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022, ACM, 2022, pp. 18–30. URL: https://doi.org/10.1145/3540250.3549162. doi:10.1145/3540250.3549162.

[30] L. Tan, D. Yuan, G. Krishna, Y. Zhou, /*icomment: Bugs or bad comments?*/, SIGOPS Oper. Syst. Rev. 41 (2007) 145–158. URL: https://doi.org/10.1145/1323293.1294276. doi:10.1145/1323293.1294276.

[31] D. Movshovitz-Attias, W. W. Cohen, Natural language models for predicting programming comments, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers, The Association for Com-

puter Linguistics, 2013, pp. 35–40. URL: https://aclanthology.org/P13-2007/.

[32] A. Louis, S. K. Dash, E. T. Barr, C. Sutton, Deep learning to detect redundant method comments, CoRR abs/1806.04616 (2018). URL: http://arxiv.org/abs/1806.04616. arXiv:1806.04616.

[33] A. Louis, S. K. Dash, E. T. Barr, M. D. Ernst, C. Sutton, Where should I comment my code?: a dataset and model for predicting locations that need comments, in: G. Rothermel, D. Bae (Eds.), ICSE-NIER 2020: 42nd International Conference on Software Engineering, New Ideas and Emerging Results, Seoul, South Korea, 27 June - 19 July, 2020, ACM, 2020, pp. 21–24. URL: https://doi.org/10.1145/3377816.3381736. doi:10.1145/3377816.3381736.

[34] X. Hu, G. Li, X. Xia, D. Lo, Z. Jin, Deep code comment generation, in: F. Khomh, C. K. Roy, J. Siegmund (Eds.), Proceedings of the 26th Conference on Program Comprehension, ICPC 2018, Gothenburg, Sweden, May 27-28, 2018, ACM, 2018, pp. 200–210. URL: https://doi.org/10.1145/3196321.3196334. doi:10.1145/3196321.3196334.

[35] X. Hu, G. Li, X. Xia, D. Lo, Z. Jin, Deep code comment generation with hybrid lexical and syntactical information, Empir. Softw. Eng. 25 (2020) 2179–2217. URL: https://doi.org/10.1007/s10664-019-09730-9. doi:10.1007/s10664-019-09730-9.

[36] Y. Zhou, X. Zhang, J. Shen, T. Han, T. Chen, H. C. Gall, Adversarial robustness of deep code comment generation, ACM Trans. Softw. Eng. Methodol. 31 (2022) 60:1–60:30. URL: https://doi.org/10.1145/3501256. doi:10.1145/3501256.

[37] P. He, Z. Chen, S. He, M. R. Lyu, Characterizing the natural language descriptions in software logging statements, in: M. Huchard, C. Kästner, G. Fraser (Eds.), Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France,

September 3-7, 2018, ACM, 2018, pp. 178–189. URL: https://doi.org/10.1145/3238147.3238193. doi:10.1145/3238147.3238193.

[38] Z. Ding, H. Li, W. Shang, Logentext: Automatically generating logging texts using neural machine translation, in: IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022, Honolulu, HI, USA, March 15-18, 2022, IEEE, 2022, pp. 349–360. URL: https://doi.org/10.1109/SANER53432.2022.00051. doi:10.1109/SANER53432.2022.00051.

[39] R. S. Malik, J. Patra, M. Pradel, Nl2type: inferring javascript function types from natural language information, in: J. M. Atlee, T. Bultan, J. Whittle (Eds.), Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019, IEEE / ACM, 2019, pp. 304–315. URL: https://doi.org/10.1109/ICSE.2019.00045. doi:10.1109/ICSE.2019.00045.

[40] C. C. Petrescu, S. Smith, R. Giavrimis, S. K. Dash, Do names echo semantics? A large-scale study of identifiers used in c++'s named casts, CoRR abs/2111.01577 (2021). URL: https://arxiv.org/abs/2111.01577. arXiv:2111.01577.

[41] S. K. Dash, M. Allamanis, E. T. Barr, Refinym: using names to refine types, in: G. T. Leavens, A. Garcia, C. S. Pasareanu (Eds.), Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018, ACM, 2018, pp. 107–117. URL: https://doi.org/10.1145/3236024.3236042. doi:10.1145/3236024.3236042.

[42] B. Vasilescu, C. Casalnuovo, P. T. Devanbu, Recovering clear, natural identifiers from obfuscated JS names, in: E. Bodden, W. Schäfer, A. van Deursen, A. Zisman (Eds.), Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn,

Germany, September 4-8, 2017, ACM, 2017, pp. 683–693. URL: https://doi.org/10.1145/3106237.3106289. doi:10.1145/3106237.3106289.

[43] M. Lachaux, B. Rozière, M. Szafraniec, G. Lample, DOBF: A deobfuscation pre-training objective for programming languages, in: M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, 2021, pp. 14967–14979. URL: https://proceedings.neurips.cc/paper/2021/hash/7d6548bdc0082aacc950ed35e91fcccb-Abstract.html.

[44] X. Huo, Y. Yang, M. Li, D. Zhan, Learning semantic features for software defect prediction by code comments embedding, in: IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018, IEEE Computer Society, 2018, pp. 1049–1054. URL: https://doi.org/10.1109/ICDM.2018.00133. doi:10.1109/ICDM.2018.00133.

[45] Z. Yu, F. M. Fahid, H. Tu, T. Menzies, Identifying self-admitted technical debts with jitterbug: A two-step approach, IEEE Transactions on Software Engineering 48 (2022) 1676–1691. doi:10.1109/TSE.2020.3031401.

[46] P. Fung, A statistical view on bilingual lexicon extraction: From parallel corpora to non-parallel corpora, in: Proceedings of the Third Conference of the Association for Machine Translation in the Americas on Machine Translation and the Information Soup, AMTA '98, Springer-Verlag, Berlin, Heidelberg, 1998, p. 1–17.

[47] J. Boyd-Graber, D. M. Blei, Multilingual topic models for unaligned text, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI '09, AUAI Press, Arlington, Virginia, USA, 2009, p. 75–82.

[48] J. Jagarlamudi, H. Daumé, Extracting multilingual topics from unaligned

comparable corpora, in: C. Gurrin, Y. He, G. Kazai, U. Kruschwitz, S. Little, T. Roelleke, S. Rüger, K. van Rijsbergen (Eds.), Advances in Information Retrieval, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 444–456.

[49] Y. Hu, K. Zhai, V. Eidelman, J. Boyd-Graber, Polylingual tree-based topic models for translation domain adaptation, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Baltimore, Maryland, 2014, pp. 1166–1176. URL: https://aclanthology.org/P14-1110. doi:10.3115/v1/P14-1110.

[50] M. Yuan, B. Van Durme, J. L. Ying, Multilingual anchoring: Interactive topic modeling and alignment across languages, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 31, Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/28b9f8aa9f07db88404721af4a5b6c11-Paper.pdf.

[51] Q. Xie, X. Zhang, Y. Ding, M. Song, Monolingual and multilingual topic analysis using lda and bert embeddings, Journal of Informetrics 14 (2020) 101055. URL: https://www.sciencedirect.com/science/article/pii/S1751157719305127. doi:https://doi.org/10.1016/j.joi.2020.101055.

[52] Y. Hu, J. Boyd-Graber, Efficient tree-based topic modeling, in: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2, ACL '12, Association for Computational Linguistics, USA, 2012, p. 275–279.

[53] S. Arora, R. Ge, A. Moitra, Learning topic models – going beyond svd, in: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, USA, 2012, pp. 1–10. URL:

https://doi.ieeecomputersociety.org/10.1109/FOCS.2012.49. doi:10.1109/FOCS.2012.49.

[54] T. Landauer, P. Foltz, D. Laham, An introduction to latent semantic analysis, Discourse processes 25 (1998) 259–284.

[55] B. Shi, W. Lam, L. Bing, Y. Xu, Detecting common discussion topics across culture from news reader comments, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 676–685. URL: https://aclanthology.org/P16-1064. doi:10.18653/v1/P16-1064.

[56] M. Johnson, S. Riezler, Exploiting auxiliary distributions in stochastic unification-based grammars, in: Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference, NAACL 2000, Association for Computational Linguistics, USA, 2000, p. 154–161.

[57] J. Kogan, Introduction to Clustering Large and High-Dimensional Data, Cambridge University Press, 2006.

[58] J. H. Gennari, P. Langley, D. Fisher, Models of incremental concept formation, Artificial Intelligence 40 (1989) 11–61. URL: https://www.sciencedirect.com/science/article/pii/0004370289900465. doi:https://doi.org/10.1016/0004-3702(89)90046-5.

[59] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359. URL: https://doi.org/10.1023/A:1008202821328. doi:10.1023/A:1008202821328.

[60] C. Treude, M. Wagner, Predicting good configurations for github and stack overflow topic models, in: Proceedings of the 16th International Conference on Mining Software Repositories, MSR '19, IEEE Press, 2019, p. 84–95.

URL: https://doi.org/10.1109/MSR.2019.00022. doi:10.1109/MSR.2019.00022.

[61] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58. URL: https://www.sciencedirect.com/science/article/pii/S2214716015300270. doi:https://doi.org/10.1016/j.orp.2016.09.002.

[62] H. M. Wallach, I. Murray, R. Salakhutdinov, D. Mimno, Evaluation methods for topic models, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 1105–1112. URL: https://doi.org/10.1145/1553374.1553515. doi:10.1145/1553374.1553515.

[63] J. Chang, J. Boyd-Graber, S. Gerrish, C. Wang, D. M. Blei, Reading tea leaves: How humans interpret topic models, in: Proceedings of the 22nd International Conference on Neural Information Processing Systems, NIPS'09, Curran Associates Inc., Red Hook, NY, USA, 2009, p. 288–296.

[64] K. Stevens, P. Kegelmeyer, D. Andrzejewski, D. Buttler, Exploring topic coherence over many models and many topics, in: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '12, Association for Computational Linguistics, USA, 2012, p. 952–961.

[65] L. Layman, A. P. Nikora, J. Meek, T. Menzies, Topic modeling of nasa space system problem reports: Research in practice, in: Proceedings of the 13th International Conference on Mining Software Repositories, MSR '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 303–314. URL: https://doi.org/10.1145/2901739.2901760. doi:10.1145/2901739.2901760.

[66] T.-H. P. Chen, S. W. Thomas, H. Hemmati, M. Nagappan, A. E. Hassan, An empirical study on the effect of testing on code quality using topic

models: A case study on software development systems, IEEE Transactions on Reliability 66 (2017) 806–824. doi:10.1109/TR.2017.2699938.

[67] K. W. Church, P. Hanks, Word association norms, mutual information, and lexicography, Computational Linguistics 16 (1990) 22–29. URL: https://aclanthology.org/J90-1003.

[68] K. Deb, Multi-objective optimization using evolutionary algorithms, Wiley-Interscience series in systems and optimization, Wiley, 2001.

[69] A. T. Wilson, P. A. Chew, Term weighting schemes for latent dirichlet allocation, in: Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA, The Association for Computational Linguistics, 2010, pp. 465–473. URL: https://aclanthology.org/N10-1070/.

[70] J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975. Second edition, 1992.