Modeling Changeset Topics

Christopher S. Corley, Kelly L. Kashuda
The University of Alabama
Tuscaloosa, AL, USA
{cscorley, klkashuda}@ua.edu

Daniel S. May
Swarthmore College
Swarthmore, PA, USA
dmay1@swarthmore.edu

Nicholas A. Kraft
ABB Corporate Research
Raleigh, NC, USA
nicholas.a.kraft@us.abb.com

***Abstract*—Topic modeling has been applied to several areas of software engineering, such as bug localization, feature location, triaging change requests, and traceability link recovery. Many of these approaches combine mining unstructured data, such as bug reports, with topic modeling a snapshot (or release) of source code. However, source code evolves, which causes models to become obsolete. In this paper, we explore the approach of topic modeling *changesets* over the traditional release approach. We conduct an exploratory study of four open source systems. We investigate the differences in corpora in each project, and evaluate the topic distinctness of the models.**

***Keywords*—Mining software repositories; changesets; topic modeling; latent Dirichlet allocation**

I. Introduction

Software developers are often confronted with maintenance tasks that involve navigation of repositories that preserve vast amounts of project history. Navigating these software repositories can be a time-consuming task, because their organization can be difficult to understand. Fortunately, topic models such as latent Dirichlet allocation (LDA) [1] can help developers to navigate and understand software repositories by discovering topics (word distributions) that reveal the thematic structure of the data [2]–[4].

When modeling a source code repository, the corpus typically represents a snapshot of the code. That is, a topic model is often trained on a corpus that contains documents that represent files from a particular version of the software. Keeping such a model up-to-date is expensive, because the frequency and scope of source code changes necessitate retraining the model on the updated corpus. However, it may be possible to automate certain maintenance tasks without a model of the complete source code. For example, when assigning a developer to a change task, a topic model can be used to associate developers with topics that characterize their previous changes. In this scenario, a model of the changesets created by each developer may be more useful than a model of the files changed by each developer. Moreover, as a typical changeset is smaller than a typical file, a changeset-based model is less expensive to keep current than a file-based model.

Toward the goal of automating software maintenance tasks using changeset-based models, in this paper we qualitatively compare topic models trained on corpora of changesets to those trained on files. For our comparison we consider vocabulary measures, which indicate whether term distributions in the changeset corpora match those in the file corpora, and topic distinctness [3], [5], [6], which measures how distinct one topic in a model is from another. Models with higher topic distinctness values are desirable, because distinct topics are more useful in differentiating among the documents in a corpus than are similar topics.

II. Background & Related Work

In this section we provide an overview of latent Dirichlet allocation and review closely related work.

A. Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) [1] is a generative topic model. LDA models each document in a corpus of discrete data as a finite mixture over a set of topics and models each topic as an infinite mixture over a set of topic probabilities. That is, LDA models each document as a probability distribution indicating the likelihood that it expresses each topic and models each topic that it infers as a probability distribution indicating the likelihood of a word from the corpus being assigned to the topic.

B. Topic Models in Software Maintenance

Thomas et al. [3] describe an approach to modeling the evolution of source code topics using LDA. Their *Diff* model outperforms the Hall topic evolution model [7] in the context of software repositories, because the *Diff* model trains topic models on the changesets between two snapshots, rather than on individual snapshots. That is, for a particular source code file, *Diff* trains a topic model on a document that represent the changes between consecutive versions of the file. Consequently, the *Diff* model eliminates the issues related to data duplication that arise in the Hall model, which trains a topic model on all versions of the (whole) file. Thomas et al. demonstrate the efficacy of the *Diff* model via a comparative study with the Hall model. Their evaluation measures include topic distinctness, which we define in Section III.

Hindle et al. [4] validate the use LDA topics during software maintenance via a study at Microsoft. Their focus is on stakeholder validation of topics — i.e., they seek confirmation that LDA topics are interpretable by stakeholders (e.g., developers or managers) and relevant to the requirements implemented by the modeled source code. Previous work by Hindle et al. [8] describes an approach to modeling the evolution of software topics using commit messages rather than source code.

Although our work is preliminary, we believe that it is the first to consider modeling changesets in lieu of snapshots to support software maintenance. Like Rao et al. [9], we are targeting problems that require an up-to-date topic model. Thus, the expense of training a topic model is a key consideration for us, unlike for Thomas et al. [3] or Hindle et al. [4], [8].

TABLE I: Subject systems version and corpora description

System	Snapshot Version	Commit SHA	Snapshot No. Documents	Changeset No. Documents	Snapshot No. Unique Terms	Changeset No. Unique Terms	Snapshot No. Total Terms	Changeset No. Total Terms
Ant	1.9.4	1c927b15	2208	12,996	17,986	74,681	1,066,446	11,801,353
AspectJ	1.8.0	5a5bef1e	10130	7,650	22,855	25,071	4,825,289	10,583,008
Joda-Time	2.3	b0fcbb95	402	1,750	9,298	11,385	493,131	5,541,330
PostgreSQL	9.3.4	d4f8dde3	4080	36,870	84,591	164,703	6,644,409	59,850,328

III. Case Study

In this section we describe the design of a case study in which we compare topic models trained on changesets to those trained on snapshots. We describe the case study using the Goal-Question-Metric approach [10].

A. Definition and Context

Our *goal* is to explore the relationship between changeset topics and snapshot topics. The *quality focus* of the study is on informing development decisions and policy changes that could lead to software with fewer defects. The *perspective* of the study is of a researcher, developer, or project manager who wishes to gain understanding of the concepts or features implemented in the source code. The *context* of the study spans the version histories of four open source systems.

Toward achievement of our goal, we pose the following research questions:

RQ1 Do changeset- and snapshot-based corpora express the same terms?

RQ2 Are topic models trained on changesets more distinct than topic models trained on a snapshot?

At a high level, we want to determine whether topic modeling changesets can perform as well as, or better than, topic modeling a snapshot.

In the remainder of this section we introduce the subjects of our study, describe the setting of our study, and report our data collection and analysis procedures.

B. Subject software systems

The four subjects of our study — Apache Ant[1], AspectJ[2], Joda-Time[3], and PostgreSQL[4] — vary in language, size and application domain. Ant is a library and command-line tool for managing builds, AspectJ is an aspect-oriented programming extension for the Java language, Joda-Time is a library for replacing the Java standard library `date` and `time` classes, and PostgreSQL is an object-relational database management system. Each system is stored in a Git repository, and the developers of each system use descriptive commit messages. Further, the developers store bug reports in an issue tracker. All systems are written in Java with the exception of PostgreSQL, which is written in C. Table I outlines the releases for each system studied.

[1]http://ant.apache.org/

[2]http://eclipse.org/aspectj/

[3]http://www.joda.org/joda-time/

[4]http://www.postgresql.org/

C. Setting

Our document extraction process is shown on the left side of Figure 1. We implemented our document extractor in Python v2.7 using the Dulwich library[5]. We extract documents from both a snapshot of the repository at a tagged release and each commit reachable from that tag's commit. The same preprocessing steps are employed on all documents extracted.

For our document extraction from a snapshot, we use the entire contents of the document. We do not parse the source code documents for classes, methods, and so on. We do this to leave our technique language-independent, and to also allow for a fair comparison between the two approaches.

To extract text from the changesets, we look at the output of viewing the `git diff` between two commits. Figure 2 shows an example of what a changeset might look like in Git. In our changeset text extractor, we only extract text from removed or added lines. Context and metadata lines are ignored. Note that we do not consider where the text originates from, only that it is text changed by the commit.

After extracting tokens, we split them based on camel case, underscores, and non-letters. We normalize to lower case before filtering non-letters, English stop words [11], Java keywords, and words shorter than three characters long. We do not stem words.

Our modeling generation is shown on the right side of Figure 1. We implemented our modeling using the Python library Gensim [12]. Gensim's LDA implementation is based on an Online LDA by Hoffman et al. [13] and uses variational inference instead of a Collapsed Gibbs Sampler. Unlike Gibbs sampling, in order to ensure that the model converges for each document, we allow LDA to see each document 10 times by setting Gensim's initialization parameter `passes` to this value. We set the following LDA parameters for all four systems: 100 topics (K), a symmetric $\alpha = 0.01$, β is left as a default value of $1/K$ (also 0.01).

D. Data Collection and Analysis

We create two corpora for each of our four subject systems. We then used LDA to model the documents into topics.

To answer RQ1, we investigate the term frequency in each corpus. We create two distributions from all unique terms from both corpora. That is, each vector is of the same length and contain zero values for terms not in its respective corpus. We measure the similarity of the two word-vectors using cosine distance.

[5]http://www.samba.org/~jelmer/dulwich/

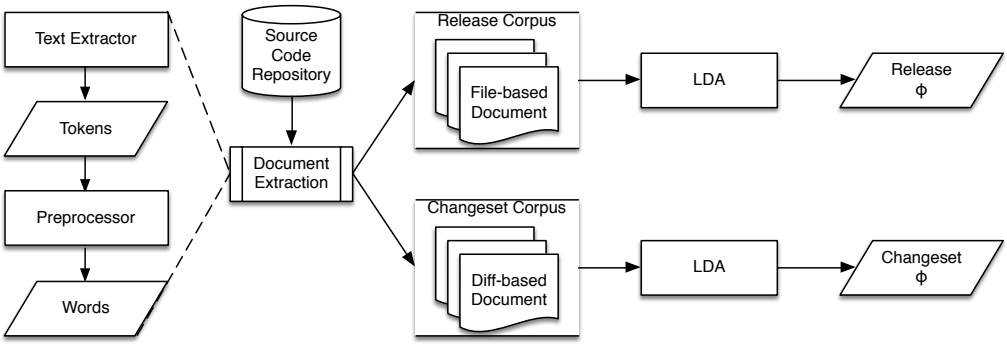


Fig. 1: Extraction and Modeling Process

```
diff --git a/lao b/tzu
index 635ef2c..5af88a8 100644
--- a/lao
+++ b/tzu
@@ -1,7 +1,6 @@
-The Way that can be told of is not the eternal Way;
-The name that can be named is not the eternal name.
 The Nameless is the origin of Heaven and Earth;
-The Named is the mother of all things.
+The named is the mother of all things.
+
 Therefore let there always be non-being,
   so we may see their subtlety,
 And let there always be being,
@@ -9,3 +8,6 @@ And let there always be being,
 The two are the same,
 But after they are produced,
   they have different names.
+They both may be called deep and profound.
+Deeper and more profound,
+The door of all subtleties!
```

Fig. 2: Example of a `git diff`. Black or blue lines denote metadata about the change useful for patching, red lines (beginning with a single −) denote line removals, and green lines (beginning with a single +) denote line additions.

To answer RQ2, we follow Thomas et al. [3] and use topic distinctness to evaluate our topic models. Distinct topics are topics with dissimilar word probabilities to all other topics in the model. Using this metric will also allow a comparison to the results of Thomas et al. [3]. Topic distinctness has also been shown to be an effective measure for qualitative comparison of topic models in context of visualization [5], [6]. Although Chang et al. [14] conclude that evaluating topic models should depend on real-world tasks over metrics, we argue that the positive results from using topic distinctness for visualization techniques qualifies the metric for this exploratory study.

Thomas et al. define topic distinctness (TD) of a topic z_i as the mean Kullback-Leibler (KL) divergence between the vectors z_i and z_j, $\forall j \neq i$:

$$TD(\phi_{z_i}) = \frac{1}{K-1} \sum_{j=1, j\neq i}^{K} KL(\phi_{z_i}, \phi_{z_j}) \qquad (1)$$

We score the overall topic distinctness of a model ϕ as the mean of its topic distinctness scores, $\forall z \in \phi$.

E. Results

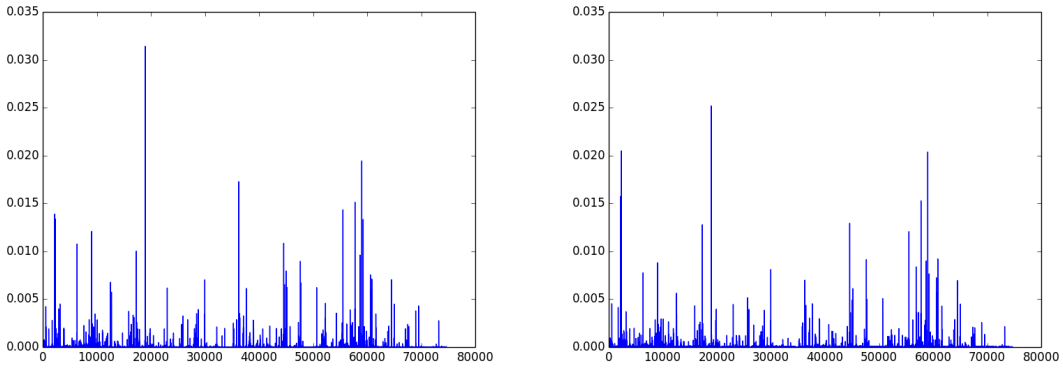
RQ1 asks whether a corpus generated from changesets have similar terms as a corpus generated from a snapshot. We expected to find that set of unique terms in the changeset corpus to be a superset of the set of unique terms in the snapshot corpus. Interestingly, this holds true for Joda-Time and AspectJ, but not for Ant and PostgreSQL. Further inspection shows that 2 terms appear in the Ant snapshot corpus that do not appear in the changeset corpus, and 19 terms appear in PostgreSQL's respective corpora. However, these anomalies appear to have been file encoding errors that had been introduced into the version history and resolved before the release. For example, one of the terms for Ant was "rapha". This is due to an encoding error in the `KEYS` file for a developer named "Raphaël Luta". Similar encoding errors were found for the remaining 20 terms.

To answer RQ1, we created two word distributions that represented the unique terms from both corpora for each system. Figure 3a shows the normalized distribution of the Ant snapshot corpus. Likewise, Figure 3b shows the distribution of the Ant changeset corpus. We measure the differences between the two distributions using cosine distance. For Ant, we had the lowest cosine distance of 0.00396. AspectJ and Joda-Time have similar distances to another of 0.06929 and 0.06540. PostgreSQL had the largest, 0.33957.

RQ2 asks if topic models trained on changeset corpora produce more distinct topics. We expected to find that they would, due to "popular" words becoming more prevalent in the corpus after appearing in several changes. Interestingly, this is true for Ant and PostgreSQL, but not for Joda-Time and AspectJ. We hypothesize this is because Ant and PostgreSQL have drastically more documents in their respective change set corpora than Joda-Time and AspectJ. That is, we think that the feasibility of using changeset topics is somewhat dependent on the amount of history in the repository.

We can compare to the results of Thomas et al. [3], as we were able to run our study on PostgreSQL[6]. We find similar topic distinctness scores for PostgreSQL in our study, even though we consider a later version of the source code. This suggests that our approach is feasible, as it captures distinct

[6]We were not able to run this study on their other subject system, JHotDraw, as no official Git mirror of the repository is available.



(a) Normalized distribution of the snapshot corpus

(b) Normalized distribution of the changeset corpus

Fig. 3: Comparison of Ant word distributions

topics while not needing post-processing and is always up-to-date with the source code repository.

TABLE II: Comparison of topic distinctness scores (RQ2)

System	Snapshot TD	Changeset TD
Ant	2.31	**3.17**
AspectJ	**3.75**	2.78
Joda-Time	**1.34**	1.03
PostgreSQL	2.59	**3.56**

IV. Threats To Validity

Our study has limitations that impact the validity of our findings, as well as our ability to generalize them. We describe some of these limitations and their impacts.

Threats to construct validity concern the adequacy of the study procedure with regard to measurement of the concepts of interest and can arise due to poor measurement design. Threats to construct validity include the use of cosine similarity as our measure of similarity for corpora and the use of topic distinctness to evaluate the topic models.

Threats to internal validity include possible defects in our tool chain and possible errors in our execution of the study procedure, the presence of which might affect the accuracy of our results and the conclusions we draw from them. We controlled for these threats by testing our tool chain and by assessing the quality of our data. Because we applied the same tool chain to all subject systems, any errors are systematic and are unlikely to affect our results substantially.

Another threat to internal validity pertains to the value of K that we selected for all models trained. We decided that the changeset and snapshot models should have the same K to help facilitate evaluation and comparison.

Threats to external validity concern the extent to which we can generalize our results. The subjects of our study comprise four open source systems in two languages, so we cannot generalize our results to systems implemented in other languages. However, the systems are of different sizes, are from different domains, and have characteristics in common with those of systems developed in industry.

V. Conclusion

In this paper conducted an exploratory study on modeling the topics of changesets. We used latent Dirichlet allocation (LDA) to extract linguistic topics from changesets and snapshots (releases).

We addressed two research questions regarding the topic modeling of changesets. First, we investigated whether changeset copora were any different than traditional snapshot corpora, and what differences there might be. For two of the systems, we found that the changeset vocabulary was a superset to the snapshot vocabulary. We measured the cosine distance of each distribution of words, and found for 3 of the systems low (between 0.003 to 0.07), while the last was much higher than the others (over 0.33). Next, we investigated whether a topic model trained on a changeset corpus was more or less distinct than a topic model trained on a snapshot corpus. For 2 of the 4 systems, we found that the changeset corpus produced more distinct topics, while for the other 2 it did not.

Future work includes expanding our evaluation and conducting an experiment where we utilize these topic models, such as for bug localization. Additional future work includes expanding our study to other systems, particularly ones that are not Java. It seems unlikely that our results are specific to Java systems, though we cannot confirm this assumption without experimentation. This expansion should also include an investigation into why some changeset topic models are more distinct than others.

Acknowledgment

We thank the anonymous reviewers for their insightful comments and helpful suggestions. This material is based upon work supported by the U.S. Department of Education under Grant No. P200A100182 and by the National Science Foundation under Grant No. 1156563.

REFERENCES

[1] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[2] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi, "Mining eclipse developer contributions via author-topic models," in *Proc. Int'l Wksp. on Mining Software Repositories*, 2007, p. 30.

[3] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein, "Modeling the evolution of topics in source code histories," in *Proc. 8th Working Conf. on Mining Software Repositories*, 2011, pp. 173–182.

[4] A. Hindle, C. Bird, T. Zimmerman, and N. Nagappan, "Relating requirements to implementation via topic analysis: Do topics extracted from requirements make sense to managers and developers?" in *Proc. 28th IEEE Int'l Conf. on Software Maintenance*, 2012, pp. 243–252.

[5] F. Wei, S. Liu, Y. Song, S. Pan, M. X. Zhou, W. Qian, L. Shi, L. Tan, and Q. Zhang, "TIARA: a visual exploratory text analytic system," in *Proc. 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2010, pp. 153–162.

[6] J. Chuang, C. D. Manning, and J. Heer, "Termite: Visualization techniques for assessing textual topic models," in *Proc. Int'l Working Conf. on Advanced Visual Interfaces*, 2012, pp. 74–77.

[7] D. Hall, D. Jurafsky, and C. Manning, "Studying the history of ideas using topic models," in *Proc. Conf. on Empirical Methods in Natural Language Processing*, 2008, pp. 363–371.

[8] A. Hindle, M. W. Godfrey, and R. C. Holt, "What's hot and what's not: Windowed developer topic analysis," in *Proc. IEEE Int'l Conf. on Software Maintenance*, 2009, pp. 339–348.

[9] S. Rao and A. Kak, "Retrieval from software libraries for bug localization: A comparative study of generic and composite text models," in *Proceedings of the 8th Working Conference on Mining Software Repositories*, ser. MSR '11. New York, NY, USA: ACM, 2011, pp. 43–52. [Online]. Available: http://doi.acm.org/10.1145/1985441.1985451

[10] V. Basili, G. Caldiera, and H. Rombach, "The goal question metric approach," 1994. [Online]. Available: ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf

[11] C. Fox, "Lexical analysis and stoplists," in *Information Retrieval: Data Structures and Algorithms*, W. Frakes and R. Baeza-Yates, Eds. Prentice-Hall, 1992.

[12] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proc. of the LREC 2010 Wksp. on New Challenges for NLP Frameworks*.

[13] M. D. Hoffman, D. M. Blei, and F. R. Bach, "Online learning for latent Dirichlet allocation," in *Proc. 25th Annual Conf. on Neural Information Processing Systems*, 2010, pp. 856–864.

[14] J. Chang, S. Gerrish, C. Wang, J. L. Boyd-graber, and D. M. Blei, "Reading tea leaves: How humans interpret topic models," in *Advances in neural information processing systems*, 2009, pp. 288–296.