Changeset Topics for Feature Location

Christopher S. Corley
The University of Alabama
Tuscaloosa, AL, USA
cscorley@ua.edu

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

***Abstract*—Abstractly say things**

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

I. Introduction

intro

II. Background & Related Work

In this section we provide an overview of two topic models, latent semantic indexing (LSI) and latent Dirichlet allocation (LDA), and review closely related work.

A. Latent Semantic Indexing

B. Latent Dirichlet Allocation

Latent Dirichlet allocation [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.

Inputs to LDA include a corpus and K, the number of topics. LDA represents each document in the corpus as a bag-of-word (multiset) and thus disregards word order and structure. Outputs of LDA include ϕ, the term-topic probability distribution, and θ, the topic-document probability distribution.

C. Feature Location

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 [2]. The data and source code for the case study is available in this paper's online appendix[1].

A. Definition and Context

Our *goal* is to evaluate the usefulness of topic models built from changesets. 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 fifteen open source systems.

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

RQ1 How do changeset-based topic models perform for feature location?
RQ2 How do *temporal* simulations of changeset-based topic models perform for feature location?

At a high level, we want to determine the feasibility in using changesets to train topic models for feature location. 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

All of our subject software systems come from two publicly-available datasets. The first is a dataset of four software systems by Dit et al. [3] and contains method-level goldsets. The second is a dataset of fifteen software systems by Moreno et al. [4] and contains class-level goldsets. The four software systems in the first dataset also appear in the second, supplying us with both class- and method-level goldsets for the queries.

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[2]. 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 ...

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 all text related to the changed file, e.g., context, removed, and added lines. 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 the tokens based on camel case, underscores, and non-letters. We only keep the split

[1] **REVIEWER URL ONLY:** http://cscorley.students.cs.ua.edu/cfl/

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

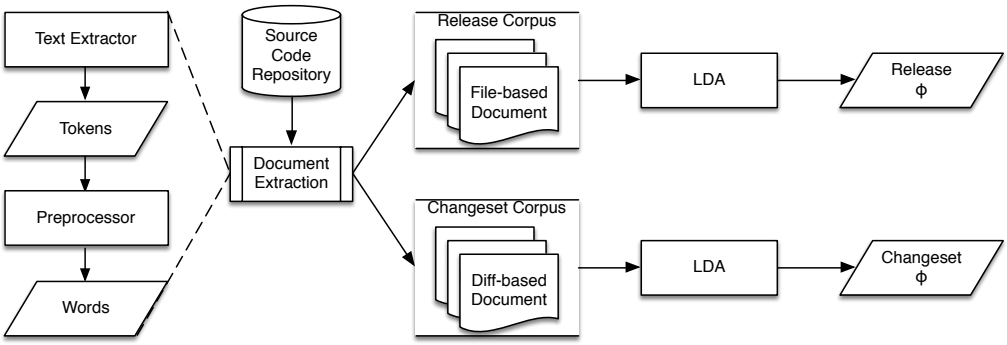

Fig. 1: Extraction and Modeling Process

tokens; original tokens are discarded. We normalize to lower case before filtering non-letters, English stop words [5], Java keywords, and words shorter than three characters long. We do not stem words. cite reason?

Our modeling generation is shown on the right side of Figure 1. We implemented our modeling using the Python library Gensim [6]. Our evaluation is two-fold: we compare results for both LDA and LSI.

1) LDA configuration: Gensim's LDA implementation is based on an online LDA by Hoffman et al. [7] 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 mini-batch 5 times by setting Gensim's initialization parameter `passes` to this value and allowing the inference step 1000 iterations over a document. We set the following LDA parameters for all fifteen systems: 500 topics (K), a symmetric $\alpha = 1/K$, and a symmetric $\beta = 1/K$. These are default values for α and β in Gensim.

For the temporal evaluation, we found it beneficial to consider two other parameters: κ and τ_0. As noted in Hoffman et al. [7], it is beneficial to adjust κ and τ_0 to higher values for smaller mini-batches. These two parameters control how much influence a new mini-batch has on the model when training. However, the implementation of Gensim at the time of this paper did not include a way to adjust τ_0, so we only adjust κ. We chose $\kappa = 2.0$ for all systems, because the temporal evaluation would often have mini-batch sizes in single digits.

2) LSI configuration: Gensim's LSI implementation is based on an online LSI by Řehůřek [8]. We set the number of topics the same as LDA for each project. The remaining parameters are left at default values, including those used during the temporal evaluation.

D. Data Collection and Analysis

We create two corpora for each of our fifteen subject systems. We then model the documents into topics.

To answer RQ1, we take the traditional approach to evaluation:

1) Build a topic model from a corpus
2) Query the model

```
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.

3) Rank the source code entities by how similar their doc-topic is to the query's.

We build a topic model for two corpora:

1) Source code entities of a release
2) All changesets leading up to the release

The release-based model is our baseline for comparison in our study.

To answer RQ2, we must take a different approach. Since both LDA and LSI can be used as online topic models, we can "simulate" the results of a feature location technique over time.

Summary:

1) Determine which commits to stop at and create mini-batches out of the commits in between
2) For each mini-batch:
   a) Update the model with the new document
   b) Extract a snapshot of the source code at that commit.

c) Infer from the model the document-topics for each entity in the snapshot
d) Query the model with the feature request text.
e) Rank the source code entities by how similar their doc-topic is to the query's.

However, only the Dit et al. dataset includes traceability links between the queries and the commits the goldsets are extracted from[3]. This limits our temporal study to those four systems.

E. Results

RQ1 asks ...

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.

Additionaly, we found errors within the datasets themselves that would be a threat to internal validity. In particular, the Moreno et al. dataset included classes that had package names that were not valid. For example, the Bookkeeper goldset for issue report 29 listed `bookkeeper-server.src.main.java.org.apache.bookkeeper.bookie.EntryLogger` for the class fixed by this change, while the actual fully-qualified name of this class is `org.apache.bookkeeper.bookie.EntryLogger`. We make the assumption that the authors of this dataset used the directory structure of the project to build the package names. Manual correction was required as our tool parses the files and uses the package name given in the source file, not the directory structure.

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 Java, 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.

[3]Authors of [4] did not respond to requests for this data.

V. Conclusion

conclusion

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. xxxxxx.

References

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

[2] 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

[3] B. Dit, A. Holtzhauer, D. Poshyvanyk, and H. Kagdi, "A dataset from change history to support evaluation of software maintenance tasks," in *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*. IEEE, 2013, pp. 131–134.

[4] L. Moreno, J. J. Treadway, A. Marcus, and W. Shen, "On the use of stack traces to improve text retrieval-based bug localization," in *Proc. IEEE Int'l Conf. on Software Maintenance and Evolution*. IEEE, 2014.

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

[6] 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*.

[7] 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.

[8] R. Řehůřek, "Subspace tracking for latent semantic analysis," in *Advances in Information Retrieval*, ser. Lecture Notes in Computer Science, P. Clough, C. Foley, C. Gurrin, G. Jones, W. Kraaij, H. Lee, and V. Mudoch, Eds. Springer Berlin Heidelberg, 2011, vol. 6611, pp. 289–300. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20161-5_29