Science of Computer Programming 80 (2014) 457–479

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Studying software evolution using topic models

Stephen W. Thomas[a,*], Bram Adams[b], Ahmed E. Hassan[a], Dorothea Blostein[a]

[a] *Software Analysis and Intelligence Lab (SAIL), 156 Barrie Street, School of Computing, Queen's University, Canada*

[b] *Lab on Maintenance, Construction and Intelligence of Software (MCIS), Département de Génie Informatique et Génie Logiciel (GIGL), École Polytechnique de Montréal, Canada*

A R T I C L E   I N F O

Article history:
Received 7 March 2011
Received in revised form 20 June 2012
Accepted 20 August 2012
Available online 12 September 2012

Keywords:
Software evolution
Topic model
Latent Dirichlet allocation
Mining software repositories

A B S T R A C T

Topic models are generative probabilistic models which have been applied to information retrieval to automatically organize and provide structure to a text corpus. Topic models discover *topics* in the corpus, which represent real world concepts by frequently co-occurring words. Recently, researchers found topics to be effective tools for structuring various software artifacts, such as source code, requirements documents, and bug reports. This research also hypothesized that using topics to describe the *evolution* of software repositories could be useful for maintenance and understanding tasks. However, research has yet to determine whether these automatically discovered topic evolutions describe the evolution of source code in a way that is relevant or meaningful to project stakeholders, and thus it is not clear whether topic models are a suitable tool for this task.

In this paper, we take a first step towards evaluating topic models in the analysis of software evolution by performing a detailed manual analysis on the source code histories of two well-known and well-documented systems, JHotDraw and jEdit. We define and compute various metrics on the discovered topic evolutions and manually investigate how and why the metrics evolve over time. We find that the large majority (87%–89%) of topic evolutions correspond well with actual code change activities by developers. We are thus encouraged to use topic models as tools for studying the evolution of a software system.

1. Introduction

To combat the complexities of software development, researchers realized that they could mine the repositories related to a software project to look for answers. For example, the location of bugs can be accurately predicted based on source code metrics or change activity [1–3], traceability links between requirements documents and source code can be automatically established by advanced information retrieval techniques [4,5], and developers can be automatically advised which source code documents to edit to address a new bug report [6].

One of the fundamental challenges in mining software repositories is understanding and using the unstructured natural-language text found in many of the repositories, for example email archives, commit logs, bug reports, and source code identifiers and comments. A recent advancement has been the use of statistical *topic models*, an unsupervised information retrieval technique, to automatically extract topics from textual repositories, such as a snapshot of the source code [7–10], documentation [11,12], and bug reports [6] in an effort to understand and use the natural language text embedded in these repositories. In this context, *topics* are collections of words that co-occur frequently in the documents of a repository and usually have a clear semantic relationship. For example, one topic might contain the words *{mouse move up down over left*

[*] Corresponding author. Tel.: +1 6134536162.
E-mail addresses: sthomas@cs.queensu.ca (S.W. Thomas), bram.adams@polymtl.ca (B. Adams), ahmed@cs.queensu.ca (A.E. Hassan), blostein@cs.queensu.ca (D. Blostein).

0167-6423/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2012.08.003

right} and another might contain *{file open read write close}*. In practice, it has been found that these topics serve to represent the major themes that span the corpus, providing many benefits over existing information retrieval techniques [13].

Understanding how the use of a topic evolves, i.e. changes, in a software repository over time could also provide many benefits for project stakeholders [14]. For example, stakeholders could monitor the *drift* of a topic, i.e., when the implementation of a topic in the source code gradually diverges from the original design (similar to architectural drift [15]). Because of refactoring, re-engineering, maintenance and other development activities, a topic that was once focused and modularized may become more scattered across the system over time, getting out-of-sync with the mental model that designers and architects have about the system. Automatically discovering and monitoring these topic drifts would be a useful technique for developers and project managers wishing to keep their project in good health. Topic evolutions could also be used by new developers wishing to quickly understand the history of certain aspects of the system, such as when specific features were added or removed from the project. Additionally, topic evolutions could be used to monitor the day-to-day development activities, answering questions such as "Who is working on what concepts?" and "What concepts changed since last week?"

One could measure the evolution of topics by applying a topic modeling technique to each version of the system separately, then linking topics in different versions together according to a similarity measure (for example, KL divergence [16]). Another more common method for discovering topic evolutions is to apply a topic modeling technique to all of the versions of the system at once (ignoring time), then mapping the topics back to individual versions of the system. One can then determine how the values of various topic metrics (e.g., *assignment* or *scattering* [10]) change over time. In this paper, we consider the latter approach, and use the term *topic evolution* to refer to the evolution of the topic's metrics over time.

Although previous work has applied topic models to the history of the source code of a project to recover such topic evolutions [14], it is not yet clear how or why the topics evolve as they do. Topics lie at a higher level of abstraction than other elements found in the source code, such as classes [17], and it is yet to be determined whether the automatically discovered topic evolutions are consistent with the actual changes made by developers (the *change activity*) in the source code. Topic evolution models were originally developed for and tested against natural language corpora, such as newspaper articles or conference proceedings. Our goal is to validate the use of topic evolution models to describe software change activities.

In this paper, we take a first step towards such a validation by performing a qualitative case study of topic evolution on two well-known and well-documented systems, JHotDraw and jEdit. We apply latent Dirichlet allocation (LDA) [18], a statistical topic modeling technique, to the release history of the systems' source code, compute several metrics on the discovered topics, and manually investigate the source code and project documentation to verify that the evolution of metric values is useful and consistent with the actual change activity in the source code. We use a simple characterization of topic evolutions and find that such topic evolutions are meaningful descriptions of the source code evolutions and have the potential to provide project stakeholders with valuable information for understanding and monitoring their project.

Specifically, the contributions of this paper can be summarized as follows.

- We formalize our approach for applying an existing topic evolution model to the source code history of a software project, and explore two visualizations of topic evolutions: line charts and heatmaps.
- We manually validate the topic evolutions discovered by our approach, and find that most (87%–89%) of the evolutions are useful and consistent with descriptions of the actual changes to the source code.
- We manually investigate the high-level relationship between code changes and topic evolutions, and find that topics evolve due to several different types of developer activities, including bug fixes, refactoring, and feature additions.
- We explore and quantify the patterns of topic evolutions, and find that the topic evolutions of JHotDraw, for example, are much more active than those of jEdit.

This paper is organized as follows. Section 2 provides background information on topic models and topic evolution models. Section 3 motivates and describes our research questions. Sections 4–8 present our detailed case studies on JHotDraw and jEdit. Section 9 enumerates threats to the validity of our study and outlines future work directions. Section 10 summarizes related work. Finally, Section 11 concludes the paper.

2. Background

In this section we introduce LDA, a popular topic modeling technique that is the building block of many topic evolution models. We then describe topic evolution models, introduce required notation, and present a set of metrics that can be used to measure topic evolutions.

2.1. Latent Dirichlet allocation

LDA is a popular statistical topic modeling technique [18]. *Topic models* are statistical models that infer latent topics to describe a corpus of text documents [13]. *Topics* are collections of words that co-occur frequently in the corpus. For example, a topic discovered from a newspaper collection might contain the words *{bank finance money cash loan}*, representing