Sistema de Extracción y Análisis Tópicos sobre Historial de Reportes de Software

Propuesta de trabajo final para la carrera Ingeniería de Sistemas

Autor: Martin I. Pacheco
Director: Dra. Daniela Godoy, Co-Director: Ing. Alejandro Corbellini
Facultad de Cs. Exactas, Universidad Nacional del Centro de la Pcia. de Bs. As.

1 de Julio, 2013

1. Introducción

El software es una parte integral de nuestra vida cotidiana y por lo tanto la calidad del mismo es muy importante. Sin embargo, la mejora y mantenimiento de su calidad es una tarea difícil e implica el empleo de una cantidad significativa de recursos para arreglar sus defectos. Trabajos previos sobre calidad de software han sido estudiados empleando varios aspectos medibles sobre la calidad del software, como métricas de producto (por ejemplo tamaño), métricas de proceso (por ejemplo tiempo para arreglar un defecto) y métricas de proyecto (por ejemplo tamaño del equipo) [3, 4]. Sin embargo, estas clases de métricas no toman en cuenta las preocupaciones (concerns) conceptuales del sistema de software (los requerimientos principales y las decisiones de diseño) que pueden afectar los detalles de implementación.

En la actualidad, investigaciones recientes en Ingeniería de Software proponen una nueva clase de enfoques para predecir defectos basados en preocupaciones (concerns) generales [2, 6, 8, 9, 10]. Estos estudios aproximan las preocupaciones (concerns) como tópicos usando modelos de tópicos (o Latent Topic Model o Statistical Topic Model), como por ejemplo Dirichlet Allocation (LDA) [1]. Es un modelo diseñado para extraer tópicos desde el cuerpo de texto de los documentos. Un tópico es una colección de términos que co-ocurren frecuentemente en el cuerpo de un documento, por ejemplo un tópico podría ser {mouse, click, drag, right, left} o {user, account, password, authentication}. Debido a la naturaleza del uso del lenguaje, los términos que constituyen un tópico a menudo se relacionan semánticamente [1].

2. Motivación

Analizar y caracterizar cómo un sistema de software cambia en el tiempo, o la evolución del software de un proyecto [5], ha sido de interés para los investigadores durante muchos años. Ambas, cómo y porqué un sistema de software cambia puede ayudar a mejorar el rendimiento de los procesos empleados en un determinado proyecto de software. Se ha aplicado LDA [7] a varias versiones de código fuente de un proyecto en un esfuerzo por identificar las tendencias de los tópicos a través del tiempo. Cuando documentos relativos a un tópico son agregados por primera vez al sistema, los tópicos experimentaran un aumento en su probabilidad total. Similarmente se ha evaluado [12] la efectividad de la evolución de los modelos de tópicos para detectar tendencias en el proceso de desarrollo de software. Los autores aplicaron LDA a una serie de versiones del código fuente y calcularon la popularidad de un tópico a lo largo del tiempo. Luego estos autores verificaron estos aumentos o caídas de popularidad de los tópicos y en efecto coincidieron con la actividad mencionada en las notas de las versiones y otros documentos de los proyectos, otorgando evidencia de que los modelos de evolución de tópicos proporcionan un buen resumen sobre la historia de un sistema de software.

3. Objetivo y Metodología

El objetivo de este trabajo es implementar una aplicación que permita extraer tópicos de un conjunto de datos y realizar su posterior análisis mediante el empleo de métricas. Para ello, en primer lugar se creara un índice a partir de la fuente de datos a utilizar para posteriormente facilitar el acceso y manipulación de los mismos. A partir de este índice se realizara la extracción de tópicos mediante el empleo de las técnicas apropiadas como LDA a partir de las consultas. Finalmente se llevara a cabo un análisis de la evolución de los tópicos obtenido desde el índice mediante el empleo de métricas como assignment, weight, scattering, entre otras [13].

El conjunto de datos sobre el cual se basara este trabajo proviene de un data set propuesto en un desafío sobre minería de datos que se realiza desde el año 2006 en la International Working Conference on Mining Software Repositories (MSR). Este conjunto se centra en reportes de bugs de diferentes aplicaciones que corren sobre la plataforma Android. La información brindada en cada bug es: identificador, titulo, estado, tipo, prioridad, componente, fecha de inicio, persona que lo reporta, descripción y una lista de comentarios [11].

A continuación, se enumeran las actividades a realizar en el marco del proyecto final de carrera:

1. Relevamiento bibliográfico sobre la aplicación de modelos estadísticos y de evolución de tópicos. Estudio del arte en la aplicación de esos algoritmos y sus respectivas métricas para reflejar sus características y evoluciones.

2. Análisis de los algoritmos a utilizar en el enfoque de los modelos de tópicos.

3. Generación de un índice pre-procesado a partir del data set anteriormente mencionado.

4. Diseño e implementación de los algoritmos y métricas seleccionados.

5. Extracción de las métricas o indicadores a partir de los resultados obtenidos.

6. Evaluación y desempeño de los modelos aplicados.

7. Construcción de una aplicación open source que permita utilizar los modelos y visualizar los resultados obtenidos.

8. Documentación del trabajo realizado.

Referencias

[1] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *the Journal of machine Learning research 3* (2003), 993–1022.

[2] Chen, T.-H., Thomas, S. W., Nagappan, M., and Hassan, A. E. Explaining software defects using topic models. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on* (2012), IEEE, pp. 189–198.

[3] Hall, T., Beecham, S., Bowes, D., Gray, D., and Counsell, S. A systematic review of fault prediction performance in software engineering.

[4] Kan, S. H. *Metrics and Models in Software Quality Engineering (2nd Edition)*. Addison-Wesley Professional, 2002.

[5] Lehman, M. M. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE 68*, 9 (1980), 1060–1076.

[6] Linstead, E., Lopes, C., and Baldi, P. An application of latent dirichlet allocation to analyzing software evolution. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on* (2008), IEEE, pp. 813–818.

[7] Linstead, E., Lopes, C., and Baldi, P. An application of latent dirichlet allocation to analyzing software evolution. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on* (2008), IEEE, pp. 813–818.

[8] Liu, Y., Poshyvanyk, D., Ferenc, R., Gyimóthy, T., and Chrisochoides, N. Modeling class cohesion as mixtures of latent topics. In *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on* (2009), IEEE, pp. 233–242.

[9] Maskeri, G., Sarkar, S., and Heafield, K. Mining business topics in source code using latent dirichlet allocation. In *Proceedings of the 1st India software engineering conference* (2008), ACM, pp. 113–120.

[10] Nguyen, T. T., Nguyen, T. N., and Phuong, T. M. Topic-based defect prediction: Nier track. In *Software Engineering (ICSE), 2011 33rd International Conference on* (2011), IEEE, pp. 932–935.

[11] Shihab, E., Kamei, Y., and Bhattacharya, P. Mining challenge 2012: The android platform. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on* (2012), IEEE, pp. 112–115.

[12] Thomas, S. W., Adams, B., Hassan, A. E., and Blostein, D. Validating the use of topic models for software evolution. In *Source Code Analysis and Manipulation (SCAM), 2010 10th IEEE Working Conference on* (2010), IEEE, pp. 55–64.

[13] Thomas, S. W., Adams, B., Hassan, A. E., and Blostein, D. Studying software evolution using topic models. *Science of Computer Programming* (2012).