Analysis of Java Based Open Source Projects

Cristina Videira Lopes

lopes@uci.edu

Abhinav Mukund Kulkarni

abhinav.kulkarni@uci.edu

Bren School of Informatics & Computer Sciences

University of California, Irvine

December 2012

Abstract

It is often of interest to study the evolution of a software project over time, compare two different projects by looking at their development histories and draw key summary statistics from the source code. It is also of use to know if there is any average trend in the development of projects and find out those projects that differ significantly from the average trend so that they can be studied closely to know what went different in their development proecesses. Matrices related to software development can also be used to make better estimation for upcoming projects.

In this report, we present an analysis of multiple Java based open-source projects of varying sizes and types. We draw heavily from a previous study related to Aspect Oriented Programming paradigm that suggests that software concerns and aspects can be modelled as latent topics in the source code.

1 Introduction

Those involved in software developement and management often want to measure software development related matrices for the purpose of improving software quality and productivity. Also there is often a need to visualize the trajectory of the development of the project. In the next sections we detail techniques used to measure various matrices associated with software development process and draw several conclusions. Most of the techniques are drawn from a recent study related to Aspect Oriented Programming [3].

Aspect Oriented Programming paradigm defines a concern as a cohesive area of functionality, such as logging, disk I/O or string handling. Usually a concern is spread across multiple classes and methods in source code files of a software project. Concerns are *cross-cutting* in a sense that they cut across multiple abstractions. These concerns and aspects can be modelled as latent topics occuring in the source code and their cross-cutting and tangling can be precisely measured using entropies of probability distributions of topics over files and that of files over topics respectively [3]. Key summary statistics based on these entropies provide us with holistic view of the development process.

In the following sections we give a brief information about topic modelling and details about our experiemention with 24 Java based open-source projects and conclusions from the study.

2 Topic Modelling

2.1 Latent Dirichlet Distribution

Latent Dirichlet Allocation (LDA) is an unsupervised topic modelling techniques to find occurrences of abstract topics in a collection of documents [4]. LDA treats documents as bag-of-words. It endows the entire collection with a Dirichlet distribution that picks out a multinomial probability vector of topics for each document. Within each document, for every word, a topic is selected according to the multinomial probability vector and then a word is generated according to a topic specific multinomial distribution over all the words. All of this can be expressed as following:

For each document d in D_{train}:

Chose a topic probability vector $\vec{\theta} \sim \text{Dir}(\vec{\alpha})$
Chose a word probability vector $\vec{\phi}_t \sim \text{Dir}(\vec{\beta})$ for each topic t
For each word w_{d_n} belonging to d:
    Select a topic $z_t \sim \text{Multinomial}(\vec{\theta})$
    Select a word $w_{dn} \sim \text{Multinomial}(\vec{\phi}_t)$

Usually the Dirichlet priors on topic and word given topic probability distributions are symmetric. We used Gibbs sampling to learn the model parameters. Details about inference can be found in the original LDA publication [4], [6].

2.2 LDA Software

We used couple of open-source packages to train topic models on the corpus of source code. Topic modelling toolbox by Mark Steyvers and Tom Griffiths is a MATLAB compliant software that can be used to train topic models [7]. It worked fine for all purposes, however we quickly ran into issues of scalability with bigger projects such as Eclipse, JBoss Drools and Netbeans and switched to Mallet, a Java-based open-source topic modelling toolkit [5]. Mallet has a parallel topic modelling trainer.

2.3 Finding Optimal Number of Topics

We looked at projects of varyig sizes, complexity and development histories and it was important to find out natural number of topics for each version of each project before proceeding with any analysis. LDA literature has details about quite a few methodologies to determine natural number of topics occuring in a corpus, but most widely used one is based on calculating perplexity of test documents [4]. We also considered another method that views LDA as a matrix factorization algorithm [1] but decided to use perplexity measure as it provided clearer results and pinpoint estimate of natural number of topics.

2.3.1 Perplexity

Document corpus is divided into training and test documents and perplexity is defined as:

$$Perplexity(D_{test}|D_{train}) = exp\left\{-\frac{\sum_{d=1}^{M^{test}} \log p(\vec{w}_d|\mathbf{\Phi}, \alpha, \beta)}{\sum_{d=1}^{M^{test}} N_d}\right\}$$

$$= exp\left\{-\frac{\sum_{d=1}^{M_{test}} \log \int_{\vec{\theta}} p(\vec{w}_d|\vec{\theta}, \mathbf{\Phi}, \alpha) \cdot p(\vec{\theta}|\beta)\, \mathrm{d}\vec{\theta}}{\sum_{d=1}^{M^{test}} N_d}\right\}$$

$$= exp\left\{-\frac{\sum_{d=1}^{M_{test}} \log \int_{\vec{\theta}} \left\{p(\vec{\theta}|\beta) \prod_{n=1}^{N_d} p(w_{dn}|\vec{\theta}, \mathbf{\Phi}, \alpha)\right\}\, \mathrm{d}\vec{\theta}}{\sum_{d=1}^{M^{test}} N_d}\right\}$$

where $\vec{w}_d$ is d^{th} document vector in the test corpus, z_t is the t^{th} topic, M_{test} is the total number of documents in test corpus and last step follows from the fact that words in a document are independent give topic mixture for the document.

Above integration is hard to carry out over all possible multinomial topic assignments. With Stevyer's and Griffith's MATLAB topic modelling toolkit, we resorted to calculating "half-document perplexity" [2]. Instead of dividing corpus into training and test documents, we divide each document in training and test sections, learn topic mixture multinomial probability for each document during training phase and calculate perplexity on the test part of the document.

$$Perplexity(D) = exp\left\{-\frac{\sum_{d=1}^{M} \log p(\vec{w}_d|\vec{\theta}_d, \mathbf{\Phi}, \alpha, \beta)}{\sum_{d=1}^{M} N_d}\right\}$$

$$= exp\left\{-\frac{\sum_{d=1}^{M} \sum_{n=1}^{N_d} \log p(w_{dn}|\vec{\theta}_d, \mathbf{\Phi}, \alpha, \beta)}{\sum_{d=1}^{M} N_d}\right\}$$

$$= exp\left\{-\frac{\sum_{d=1}^{M} \sum_{n=1}^{N_d} \log \left\{\sum_{t=1}^{T} p(w_{dn}|\mathbf{\Phi}, \beta) \cdot p(z_t|\vec{\theta}_d, \alpha)\right\}}{\sum_{d=1}^{M} N_d}\right\}$$

Here topic probabilities for each document are known from training phase, hence there is no integration over $\vec{\theta}$. One alternative to above is to have separate training and test corpuses and run few iterations of Gibbs or Importance

sampling on each test document to obtain corresponding topic mixtures [8]. We used fixed values of α and β for each project. While manually fixing their values introduces bias in the natural number of topics, it is a fairly common practice in topic modelling community to do so.

With Mallet, we used its in-build functionality for calculating perplexity on test corpus.

Wallach, et. al. 2009 has more details about evalutation methods for topic models [8].

Figure 1a and 1b show perplexity graphs for selected versoins of Apache Ant and Lucene. It can be easily seen that perplexity graphs stabilize or rise upwards after reaching natural number of topics.

2.3.2 Symmetric KL Divergence

The other method to find out natural number of topics is based on an idea that views LDA as a matrix factorization method and computes natural number of topics by maximizing "quality of split" (i.e. matrix factorization) [1]. Unlike preplexity, this method highlights a range in which natural number of topics lies. Figure 1c shows plot for Apache Maven. It is easy to see a dip for each version in the range 30-90.

2.4 LDA Results

We analyzed Below are listed some of the topics extracted from different projects. It is not difficult to recognize that they represent specific functionalities such as string handling, I/O, etc. in Java.

Top Words	Topic	Project
file buildexception io dest fileinputstream close length line	File I/O	Apache Ant
stringtokenizer tok nexttoken hasmoretokens token max path	String Tokenization	Apache Ant
swt bounds width x height y rectangles left	SWT GUI Functionalities	Eclipse
server port fetchpage mortbay startserver listener getserver socketlistener	Server Messaging	Apache Nutch

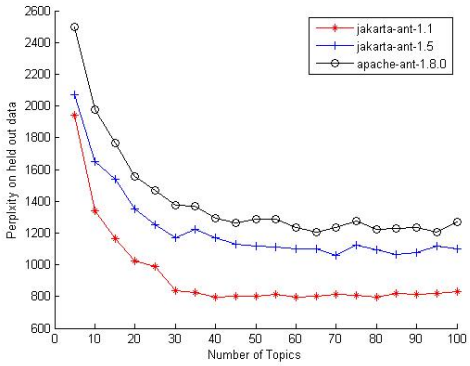
Top Words	Topic	Project
message cause protocolexception protocol suppresswarnings httpauthenticationexception ftpexception urlfilterexception parseexception	Protocol Exception Handling	Apache Nutch
dateformat timezone date format dateformattype dateformatoption timezoneid absolutetimedateformat equalsignorecase time settimezone	Date/Timezone Operations	Log4J
assertequals test assert conf asserttrue junit configuration class mockito org assertfalse fail set testcase	Assert/Testing	Apache Hadoop
taskid taskattemptid jobid job this taskstatus status task tasktracker jvmid counters id tip jobinprogress get jobstatus state tracker taskinprogress	Hadoop Job/Task Tracking	Apache Hadoop
mapper reducer name outputpath log inputpath argmap outputformat inputformat writable catfile mainclasses mappervalue mapperkey	Hadoop Map/Reduce	Apache Mahout

3 Source Code Analysis

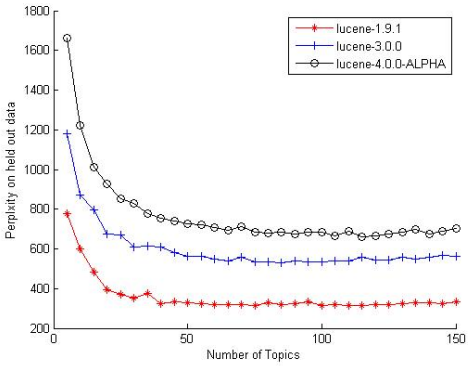
We draw a number of key summary statistics from the corpus. In particular, we focussed on minor releases for each project (such as v1.1, v1.2, v2.1, v2.2, etc.). Source code for each version was tokenized and obvious Java and project-specific stopwords were removed. A topic model for each version was fit with natural number of topics obtained from previous analysis. The details about each project are presented in Appendix A. It can be seen that the natural number of topics roughly correlates with project size.

3.1 Topic Scattering

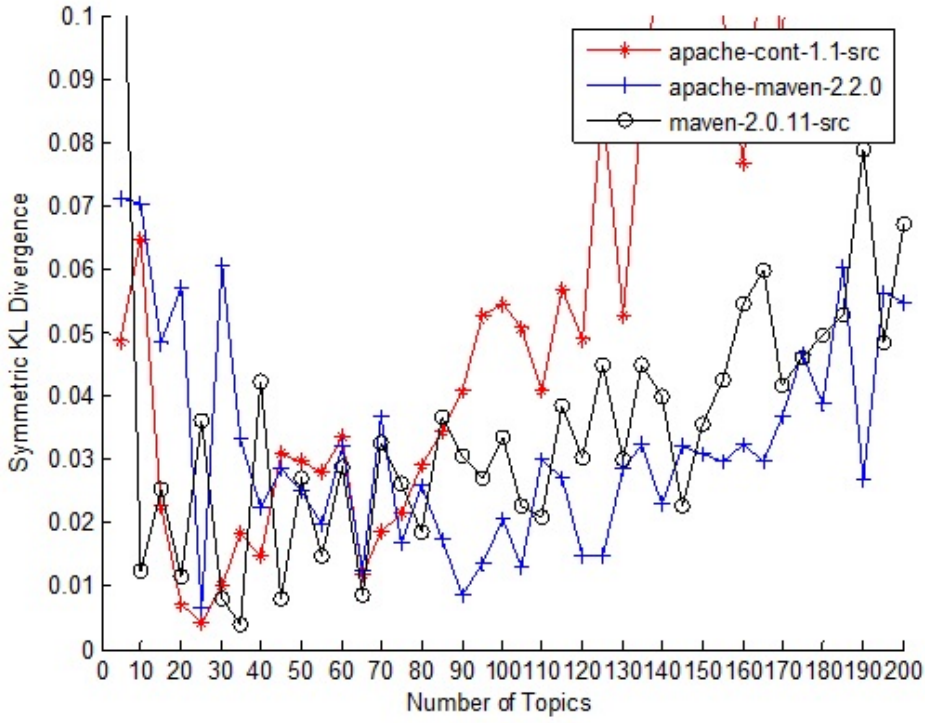
Topic scattering is the entropy of the distribution of a topic over documents. We calculated mean topic scattering for each version (summation of topic entropies weighted by probability of occurrence of the topics). Topic entropy



(a) Perplexity plot for selected versions of Apache Ant



(b) Perplexity plot for selected versions of Apache Lucene



(c) Symmetric KL divergence plot for selected versions of Apache Maven

Figure 1: Estimation of natural number of topics

can be viewed as a measure of localization of the cohesive functionlity that the topic represents. Higher the entropy, higher the spread of aspect over large parts of source code. Please refer to Baldi, Lopes, et. al. (2008) [3] for more discussion about topic scattering. For almost all projects, mean normalized scattering increases over time as projects evolve and become complex (figure 2a and 2b). There are a few exceptions such as CoffeeMud project as depicted in figure 2c and 2d.

3.2 File Tangling

File tangling is the entropy of the distribution of a file over topics. We calculated average file tangling for each version. File tangling can be viewed as a measure of heterogeneity of a souce code document in terms of concerns. Please refer to Baldi, Lopes, et. al. (2008) [3] for more discussion about file tangling. For almost all projects, average normalized file tangling increases over time as projects evolve and become complex (figure 3a and 3b). There are a few exceptions such as CoffeeMud as in figure 3c and 3d.

3.3 Overall Statistics

Figure 4 shows mean scattering and average tangling of all project versions. It is easy to spot upward trend.

4 Conclusion

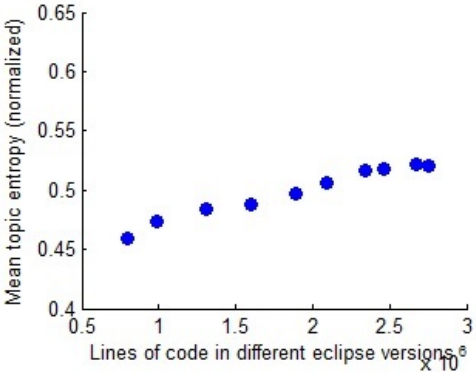
Figure 5 shows fitted curves obtained by fitting mean scattering/average tangling with log of LOC/number of files. We obtained following fit:

$$MeanScattering = -0.16 + 0.044 \cdot \log(LOC) \qquad (R^2 = 0.5498)$$

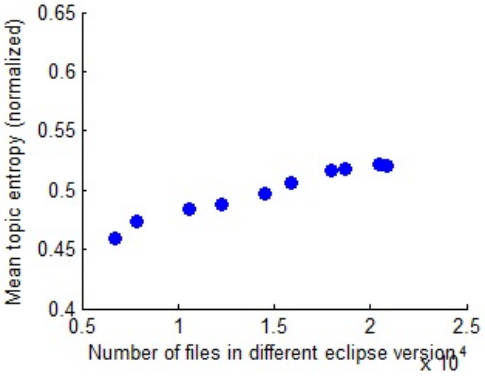
$$MeanScattering = 0.03 + 0.048 \cdot \log(NumFiles) \qquad (R^2 = 0.605)$$

The plots for average tangling do not quite lend themselves to a good linear model fit against logarithm of LOC/number of files, however the trend is not very different from that observed in case of mean scattering.

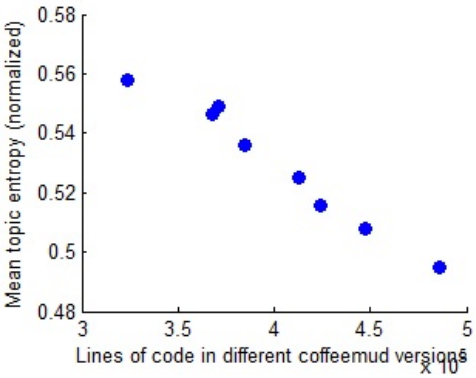
Standard statistical tests can be employed to identify outlying projects/versions for further analysis.



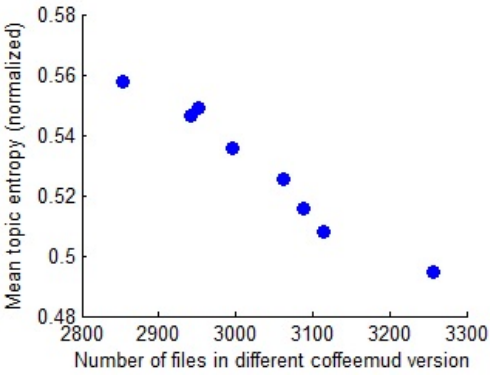
(a) Mean topic scattering against LOC of Eclipse versions



(b) Mean topic scattering against number of files in Eclipse versions

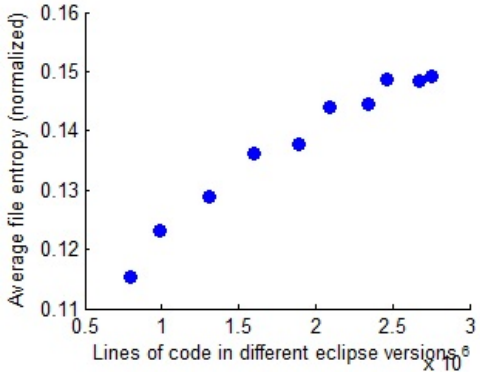


(c) Mean topic scattering against LOC of CoffeeMud versions

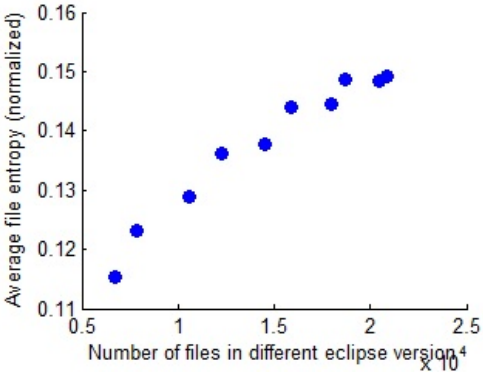


(d) Mean topic scattering against number of files in CoffeeMud versions

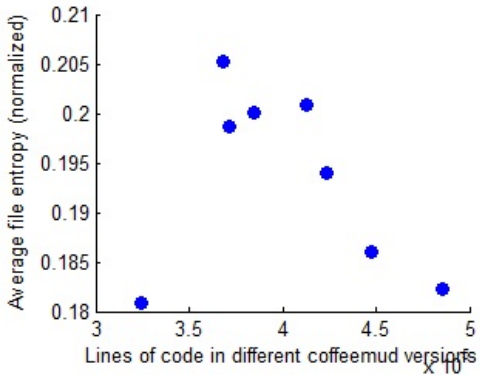
Figure 2: Mean topic scattering over time



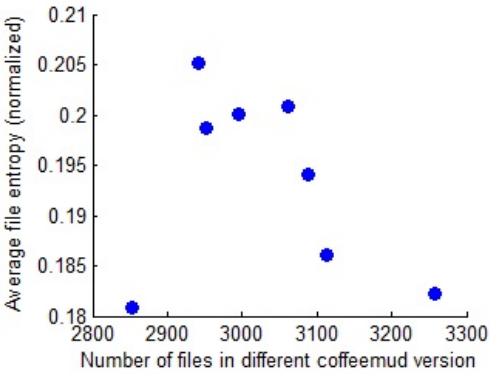
(a) Average file tangling against LOC of Eclipse versions



(b) Average file tangling against number of files in Eclipse versions

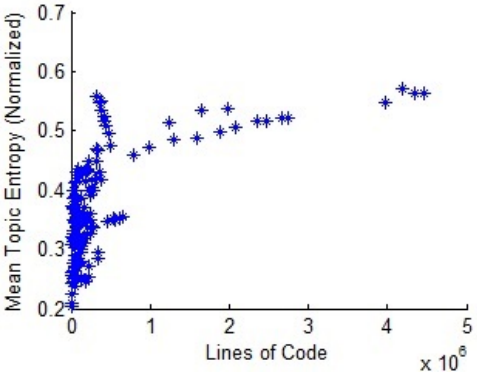


(c) Average file tangling against LOC of CoffeeMud versions

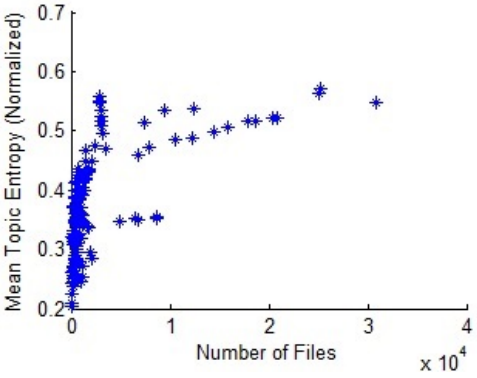


(d) Average file tangling against number of files in CoffeeMud versions

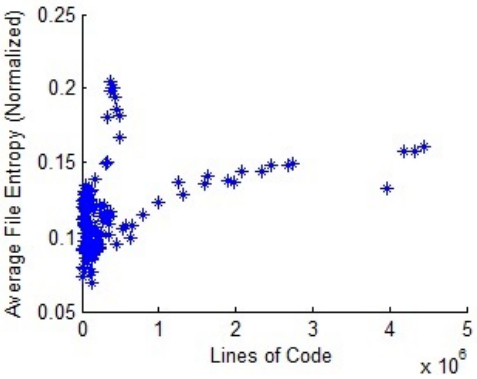
Figure 3: Average file tangling over time



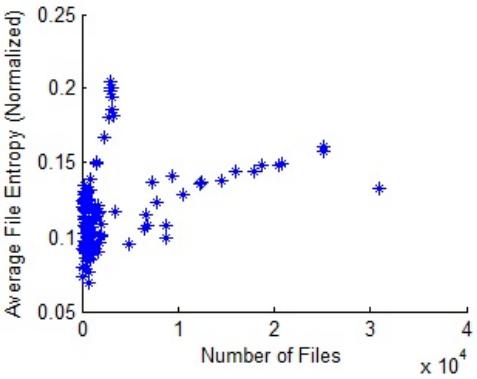
(a) Average mean scattering against LOC of different versions



(b) Average mean scattering against number of files in different versions

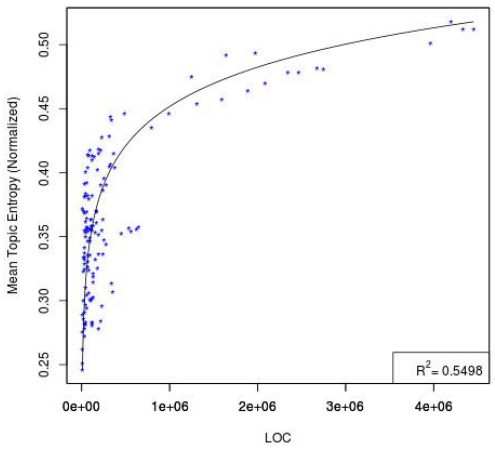


(c) Average file tangling against LOC of different versions

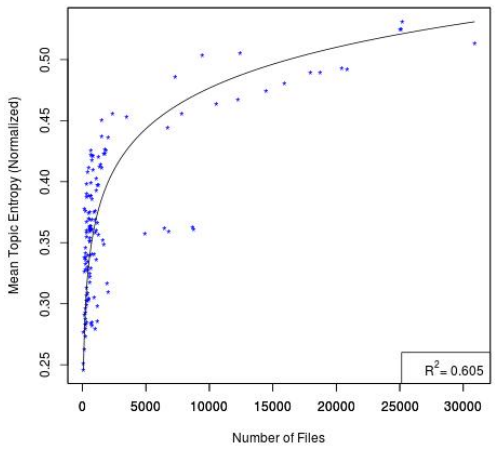


(d) Average file tangling against number of files in different versions

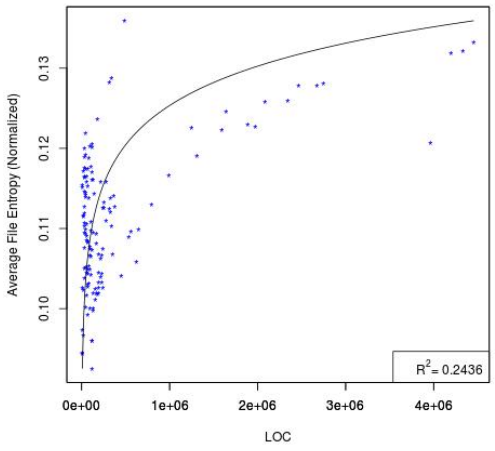
Figure 4: Overall Statistics



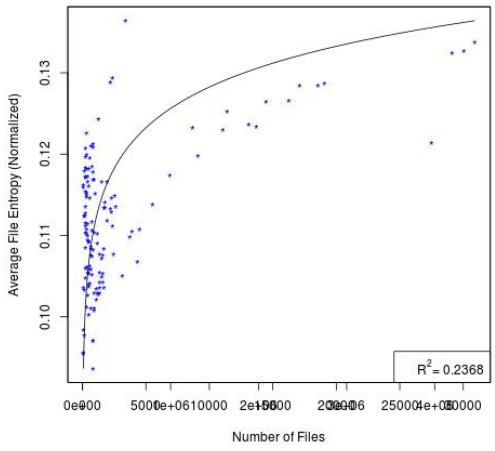
(a) Mean scattering against LOC of different versions



(b) Mean scattering against number of files in different versions



(c) Average file tangling against LOC of different versions



(d) Average file tangling against number of files in different versions

Figure 5: Fitted Curves for Overall Statistics

References

[1] R. Arun, V. Suresh, C. Veni Madhavan, and M. Narasimha Murthy. On finding the natural number of topics with latent dirichlet allocation: Some observations. *Advances in Knowledge Discovery and Data Mining*, pages 391–402, 2010.

[2] A. Asuncion, M. Welling, P. Smyth, and Y.W. Teh. On smoothing and inference for topic models. *arXiv preprint arXiv:1205.2662*, 2012.

[3] Pierre F. Baldi, Cristina V. Lopes, Erik J. Linstead, and Sushil K. Bajracharya. A theory of aspects as latent topics. *SIGPLAN Not.*, 43(10):543–562, October 2008.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[5] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[6] M. Steyvers and T. Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.

[7] M. Steyvers and T. Griffiths. Matlab topic modeling toolbox 1.4. http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm, 2011.

[8] H.M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112. ACM, 2009.