Using Latent Dirichlet Allocation for Automatic Categorization of Software

Kai Tian, Meghan Revelle, and Denys Poshyvanyk

Department of Computer Science, The College of William and Mary, Williamsburg, Virginia

Abstract

In this paper, we propose a technique called *LACT* for automatically categorizing software systems in open-source repositories. *LACT* is based on Latent Dirichlet Allocation, an information retrieval method which is used to index and analyze source code documents as mixtures of probabilistic topics. For an initial evaluation, we performed two studies. In the first study, *LACT* was compared against an existing tool, *MUDABlue*, for classifying 41 software systems written in C into problem domain categories. The results indicate that *LACT* can automatically produce meaningful category names and yield classification results comparable to *MUDABlue*. In the second study, we applied *LACT* to 43 software systems written in different programming languages such as C/C++, Java, C#, PHP, and Perl. The results indicate that *LACT* can be used effectively for the automatic categorization of software systems regardless of the underlying programming language or paradigm. Moreover, both studies indicate that *LACT* can identify several new categories that are based on libraries, architectures, or programming languages, which is a promising improvement as compared to manual categorization and existing techniques.

State of art in software categorization

To facilitate easier browsing and searching of open-source repositories such as SourceForge.net, software systems are placed into categories (e.g., text editors, anti-virus, databases, etc).

An existing research prototype, *MUDABlue* [3], has successfully used Latent Semantic Indexing (LSI) [2], to automatically categorize software systems in open-source software repositories.

Latent Dirichlet Allocation (LDA) [1] is an IR approach in which documents can be viewed as mixtures of topics, which may make it more amenable to software categorization than LSI.

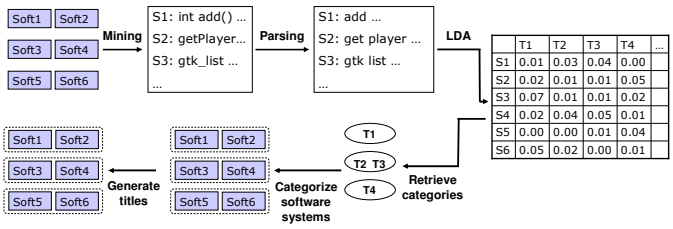
We propose a technique called LACT to automatically categorize software systems in a repository using LDA.

Software categorization with LACT

LACT is a four step approach to automatically categorize software systems using LDA.

1. **Parse software systems**. Each software system in a repository is parsed and represents a document in the corpus.
2. **Index corpus**. Obtain a topic-document matrix in which each document (system) is probabilistically associated with a set of topics.
3. **Retrieve categories**. Group topics around categories using cosine similarity. If a cosine similarity between two topics is greater than *0.8*, cluster them into the same category.
4. **Categorize software systems**. Assign systems to categories if one of the category topics belongs to a system with a probability above a distribution threshold.

LACT requires two parameters: the number of topics to generate and a distribution threshold to determine to which categories topics belong.



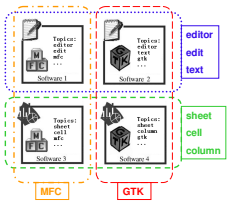
	T1	T2	T3	T4	...
S1	0.01	0.03	0.04	0.00	
S2	0.02	0.01	0.01	0.05	
S3	0.07	0.01	0.01	0.02	
S4	0.02	0.04	0.05	0.01	
S5	0.00	0.00	0.01	0.04	
S6	0.05	0.02	0.00	0.01	

Summary

- *LACT* uses Latent Dirichlet Allocation to automatically categorize software systems from open-source repositories.
- *LACT*'s performance is comparable to *MUDABlue*'s performance on systems written in C.
- Additionally, *LACT* can effectively categorize systems implemented in different programming languages.
- *LACT* finds new, useful categories that were not manually assigned.
- *LACT* automatically generates category names that are more comprehensible than existing software categorization systems.
- Future work includes eliminating the generation of categories whose meanings are undeterminable.

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [1] is a probabilistic topic model. In LDA, a document is viewed as a mixture of topics, and each topic is characterized by a distribution over a set of words.



To apply LDA to software, we use the mapping shown in Table 1 from elements of the LDA model to source code entities.

Table 1. Mapping the LDA model to source code.

LDA model	Source code entities
word	Identifiers and comments. Exclude programming language keywords, stop words, and punctuation. Compound identifiers are split based on observed naming conventions. $V=\{w_1, w_2, ..., w_v\}$
document	A software system, which can be expressed as n identifies and comments from a vocabulary. $s_i=(w_1, w_2, ..., w_n)$
corpus	A set of software systems from a repository. $C = \{S_1, S_2, ..., S_Z\}$

Tuning LACT

Using *LACT* requires setting two parameters: the number of topics to generate and a distribution threshold.

Different configurations of the number of topics and distribution thresholds were investigated to find LACT's optimal setting.

The same 41 systems used to evaluate *MUDABlue* [3] were used.

The SourceForge categories were used as a gold standard. Table 2 shows the precision and recall results. LACT's best performance was with 40 topics and a threshold of 0.02.

Table 2. Precision (top value) and recall (bottom value) for various numbers of topics and distribution threshold.

# topics	Distribution Threshold					
	0.001	**0.005**	**0.01**	**0.02**	**0.05**	**0.1**
10	0.54, 0.52	0.56, 0.53	0.57, 0.52	0.59, 0.54	0.58, 0.56	0.56, 0.54
20	0.57, 0.55	0.58, 0.56	0.61, 0.58	0.64, 0.63	0.65, 0.63	0.62, 0.59
30	0.62, 0.61	0.62, 0.61	0.64, 0.63	0.69, 0.70	0.68, 0.65	0.68, 0.64
40	0.66, 0.65	0.69, 0.66	0.71, 0.70	**0.74, 0.72**	0.73, 0.73	0.69, 0.69
50	0.63, 0.61	0.62, 0.62	0.65, 0.64	0.68, 0.70	0.68, 0.69	0.66, 0.65
60	0.64, 0.63	0.66, 0.65	0.69, 0.68	0.69, 0.70	0.64, 0.63	0.69, 0.68
70	0.68, 0.67	0.71, 0.70	0.74, 0.73	0.73, 0.73	0.69, 0.68	0.62, 0.64
80	0.56, 0.57	0.73, 0.72	0.64, 0.63	0.70, 0.69	0.64, 0.63	0.68, 0.67

LACT vs. MUDABlue

LACT's performance was compared head-to-head with *MUDABlue* using the same 41 software systems from *MUDABlue*'s evaluation.

LACT was configured with 40 topics and a distribution threshold of 0.02.

Table 2 summarizes the results. Overall, *LACT*'s performance is comparable to *MUDABlue*'s.

Table 2. Comparison of the categories produced by *MUDABlue* and *LACT*.

	MUDABlue	*LACT*
# Categories	40	32
Same as SourceForge	18	19
New, meaning	11	7
Indeterminate	11	6
Avg. # per category	2.6	3.125

Language Independence

LACT also evaluated using 43 open-source software systems written in C/C++, Java, PHP, Perl, and C#.

The systems were selected from six categories on SourceForge: *Game*, *Editor*, *Database*, *Terminal*, *E-mail*, and *Chat*.

LACT was configured with 45 topics and a distribution threshold of 0.05.

- 34 categories found
- 9 categories the same as SourceForge
- 25 new categories
- 15 of new categories are meaningful

References

[1] Blei, D. M., Ng, A. Y., and Jordan, M. I., "Latent Dirichlet Allocation", *Journal of Machine Learning Research*, vol. 3, 2003.

[2] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R., "Indexing by Latent Semantic Analysis", *Journal of the American Society for Information Science*, vol. 41, 1990.

[3] Kawaguchi, S., Garg, P. K., Matsushita, M., and Inoue, K., "MUDABlue: An automatic categorization system for Open Source repositories", *Journal of Systems and Software*, vol. 79, no. 7, 2006.

Acknowledgments

We acknowledge Dr. Shinji Kawaguchi and Dr. Katsuro Inoue for providing us with *MUDABlue*'s categorization results. We thank anonymous reviewers for detailed comments and helpful suggestions. This research was supported in part by the United States Air Force Office of Scientific Research under grant number FA9550-07-1-0030.

Further information

For further information, please contact Denys Poshyvanyk, *denys@cs.wm.edu*.

An online appendix with all of our data can be found at:
http://www.cs.wm.edu/~denys/data/msr09/msr09-appendix.htm

Information on other projects from our research group can be found at:
http://www.cs.wm.edu/semeru