

# Mining Logs Files for Computing System Management

Wei Peng, Tao Li  
School of Computer Science  
Florida International University  
Miami, FL 33199  
{wpeng002,taoli}@cs.fiu.edu

Sheng Ma  
Machine Learning for Systems  
IBM T.J. Watson Research Center  
Hawthorne, NY 10532  
shengma@us.ibm.com

## 1. Introduction

With advancement in science and technology, computing systems become increasingly more difficult to monitor, manage and maintain. Traditional approaches to system management have been largely based on domain experts through a knowledge acquisition process to translate domain knowledge into operating rules and policies. This has been experienced as a cumbersome, labor intensive, and error prone process. There is thus a pressing need for automatic and efficient approaches to monitor and manage complex computing systems. A popular approach to system management is based on analyzing system log files. However, several new aspects of the system log data have been less emphasized in existing analysis methods and posed several challenges. The aspects include disparate formats and relatively short text messages in data reporting, asynchronous data collection, and temporal characteristics in data representation. First, a typical computing system contains different devices with different software components, possibly from different providers. These various components have multiple ways to report events, conditions, errors and alerts. The heterogeneity and inconsistency of log formats make it difficult to automate problem determination. To perform automated analysis, we need to categorize the text messages with disparate formats into common situations. Second, text messages in the log files are relatively short with a large vocabulary size [3]. Third, each text message usually contains a timestamp. The temporal characteristics provide additional context information of the messages and can be used to facilitate data analysis. In this paper, we apply text mining to automatically categorize the messages into a set of common categories, and propose two approaches of incorporating temporal information to improve the categorization performance.

## 2. System Log Categorization

The disparate logging mechanisms impede problem investigation. Two components experiencing the same problem may use different formats to report the same informa-

tion. For instance, when component A starts running, it reports “A has started” in log file. However, component B may report “B has begun execution” when it starts running. This makes it difficult to correlate logs from across different components when problems occur. To solve this problem, we first manually determine a set of categories. The set of categories is based on the CBE (Common Base Event) format established by IBM initiative [4]. CBE provides a finite set of canonical situations after analyzing thousands of log entries across multiple IBM and non-IBM products. The set of categories includes *start*, *stop*, *dependency*, *create*, *connection*, *report*, *request*, *configuration*, and *other*. Given the set of common categories, we can then categorize the messages reported by different components into the prescribed categories. In our work, we use naive Bayes as our classification approach. Naive Bayes is a simple practical generative classification algorithm based on Bayes Theorem with the assumption of class conditional independence. Given a set of training samples  $S$ , the naive Bayes classifier uses  $S$  to estimate  $P(d_i|C_j)$  and  $P(C_j)$ . To classify a new sample, it uses Bayes rule to compute class posterior  $P(C_j|d_i) = \frac{P(C_j)P(d_i|C_j)}{\sum_{j=1}^L P(C_j)P(d_i|C_j)}$ . The predicted class for the document  $d_i$  is then just  $\text{argmax}_j P(C_j|d_i)$ . More details on Naive Bayes can be found in [2].

## 3. Incorporating The Temporal Information

As we discussed in Section 1, log messages generated usually contain timestamps. This structural constraint can be naturally represented using naive Bayes algorithm and hidden Markov models. In this section, we describe two approaches of utilizing the temporal information to improve classification performance.

**Modified Naive Bayes algorithm:** If the sequence of log messages is considered, the accuracy of categorization for each message can be improved as the structure relationships among the messages can be used for analysis. For example, in our experiment the component will report *dependency* messages if it cannot find some features or components. This

Time	Mod Naive	Ori Naive	Component	Msg
1101243861	configuration	configuration	10	Product: Microsoft SQL Server Setup Support Files – Configuration completed successfully.< br >< br >
1101243864	start	configuration	10	Product: Microsoft SQL Server Setup 2005 Beta 2 – Install started.

The example of effectiveness of modified Naive Bayes

happens especially just right after it generates the *create* message– “cannot create \* ” (\* represents certain components or features). In order to take the relationships between adjacent messages into account, we make some modifications to the naive Bayes algorithm. Suppose we are given a sequence of adjacent messages  $D = (d_1, d_2, \dots, d_T)$ . let  $C_i$  be the category labels for message  $d_{i-1}$ . Now we want to classify  $d_i$ . We assign  $d_i$  to the category which is  $\text{argmax}_j(P(C_j|d_i) \times P(C_j|C_i))$ . In other words, we aim at maximizing the multiplication of text classification probability  $P(C_j|d_i)$  and state transition probability  $P(C_j|C_i)$ . Our experiments show that the modified algorithm enhances the classification accuracy.

**Hidden Markov Model:** Hidden Markov Model(HMM) is another approach to incorporate the temporal information for message categorization. The temporal relations among messages are naturally captured in HMM [1]. In our experiment, the category labels we specified above are regarded as states. The emitting observation symbols are the log messages corresponding to their state labels. HMM can be used to compare all state paths from the start state to the destination state, and then choose the best state sequence that emits a certain observation sequence. When one log message has been assigned several competitive state labels by text classification, HMM helps define a certain label for it by traversing the best state sequence. The probability of emitting messages can be estimated as the ratio of the occurrence probabilities of log messages to the occurrence of their possible states in the test log files. In our experiment, the HMM categorization also improves the classification accuracy.

## 4. Experiment

We use logdump2td (NT data collection tool) developed by Event Mining Team at IBM T.J. Watson research center to collect log files from several different machines with different operating systems. To preprocess text messages, we remove stop words and perform stemming operations. We use accuracy to evaluate the classification performance and use the 30-70 split for training and test. Our first experiment is on using naive Bayes classifier to categorize log messages. The classification tool is built on the Bow toolkit available at <http://www-2.cs.cmu.edu/~mccallum/bow/>. The accuracy of log categorization is about 0.717. The keywords which have high occurrence probabilities in the corresponding classes and their probabilities can be thus extracted. For example, key words “stopped”, “restarted”, and “failed” have occur-

rence probabilities of 0.0595, 0.0405, and 0.0324 respectively in the category *Stop*. In order to improve the classification accuracy of log messages, we consider time stamps of log messages as discussed in Section 3. The state transition probability can be derived from training data. After applying modified naive Bayes which considering the state transition probability, the categorization accuracy is improved up to 0.8232. Table 1 gives an example taken from our experiments illustrating the effectiveness of the modified naive Bayes. The columns in the tables are, from left to right, time stamps, category labels obtained by incorporating transition probability, category label without considering the transition probability, the reporting component, and log message. The first row represents the previous message, and the second row represents the current message. We can observe the label obtained by original naive Bayes classifier is not correct, while the modified approach does well. Knowing from results of our experiment, the transition probability from *configuration* to *start* is 0.1869, and the probability from *configuration* to *configuration* is 0.1111. Original naive Bayes classifier assigns this message to *start* with probability of 0.443. It assigns this message to *configuration* category with probability 0.498. By incorporating temporal information, modified naive Bayes algorithm assigns this message to *start* as  $0.1869 \times 0.443 > 0.1111 \times 0.498$ . Our implementation of HMM tool is based on the package UMDHMM version 1.02 from University of Maryland. Viterbi algorithm is used to discover the best state sequence [1]. HMM categorization accuracy is around 0.85 in our experiment. The detailed report and the tools for mining log files can be found at <http://www.cs.fiu.edu/~wpeng002/log>.

## References

- [1] C. Li and G. Biswas. Temporal pattern generation using hidden Markov model based unsupervised classification. In *In Proc. of IDA-99*, pages 245–256, 1999.
- [2] Tom M. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., 1997.
- [3] Jon Stearley. Towards informatic analysis of syslogs. In *Proceedings of IEEE International Conference on Cluster Computing*, 2004.
- [4] Brad Topol, David Ogle, Donna Pierson, Jim Thoensen, John Sweitzer, Marie Chow, Mary Ann Hoffmann, Pamela Durham, Ric Telford, Sulabha Sheth, and Thomas Studwell. Automating problem determination: A first step toward self-healing computing systems. IBM White Paper, 2003.