Using Bayesian networks to predict changes

Sarah Nadi SNADI@UWATERLOO.CA
University of Waterloo, Waterloo, ON, Canada

Abstract

STILLLL

1. Introduction

Change is inevitable in any Information Technology(IT) system. New features are added, different configurations are used, upgrades are introduced and new software and hardware are added. In a large system, such changes can cause other parts of the system to malfunction without the ability of the analyst to foresee this side effect. Accordingly, we need to ensure that we consider the impact of all changes before they are implemented, and make sure we adjust the potentially affected components accordingly while implementing the change. This leads to the process of change set detection which is finding all the components of the system that need to be included in your change set where the change set is the set of components that need to be changed to avoid any side effects.

Many of the large organizations with large and sophisticated IT systems employ a Configuration Management Database (CMDB) to help them manage these systems. The CMDB keeps track of all the components in a system, how they are related, as well as all the changes that have occurred to them. Any component in the CMDB is called a Configuration Item (CI). In previous work (Nadi et al., 2010), we mined the CMDB repository for historical co-changes, and used the mined correlations to predict change sets. That is, given an initial CI that is going to change, we predict what other CIs might need to be changed as well based on its change history with other CIs. We obtained really good results in terms of recall and precision (69.8% and 88.5% respectively).

In this paper, we wish to examine the same problem, but from a Bayesian perspective. Instead of looking at pairwise historical co-changes of CI, we would like to consider all the changed CIs in the past to deduce the causality relations between them. That is, what is the probability that if A changes, B changes as well. In that case, A would be a parent to B in a Bayesian network. Accordingly, we explore the different ways a Bayesian network can be constructed from the data we have, and then test the predictions produced by querying this network to examine the obtained recall and precision. We hope that we can obtain better results using Bayesian networks since they are not limited to pairwise comparisons.

The rest of this paper is organized as follows. BLA BLA BLA

2. Background

2.1. CMDBs and Change Sets

A Configuration Management Database (CMDB) (Office of Government Commerce , OGC) is useful in Enterprise IT Management (EITM) since it provides information about the various critical components in a system including hardware, software, and services provided by the company. It records the configuration of these items, their change history, their incident history, as well as the relationships between them. Each item stored in the CMDB is referred to as a Configuration Item (CI). Figure 1 shows an example CMDB to illustrate the concepts of CIs and relationships.

A CMDB provides a basis for decision making processes such as Incident Management, Change Management, etc. In this paper, we focus on the process of Change Management, and in particular, on the problem of change set detection. A *change* is the addition, modification, or removal of anything that could affect on IT services. A poorly planned change may lead to a fault in the system. Accordingly, when one wants to change one CI is the system, other CIs that might need to be changed as well must be correctly identified. The set of CIs that will need to be modified for the change to be complete is called a change set.

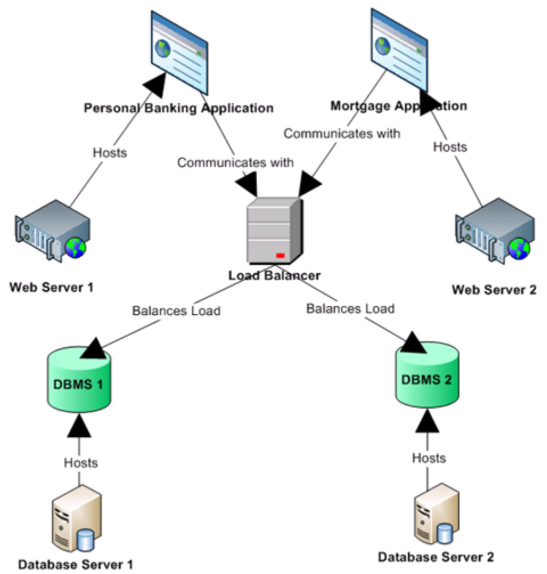

Figure 1. An IT System Stored in a CMDB. Each item shown is a CI, and CIs are tied through different relationship types

2.2. Bayesian Networks

2.3. Bayesian Network Tools

2.3.1. Weka

WEKA (Hall et al., 2009) is a toolbox for machine learning. It provides different algorithms for classification and clustering as well as other machine learning problems. It also provides Bayesian learning techniques for classification problems. Additionally, it provides explorations tools for Bayesian Networks, and allows the user to learn the structure and CPTs of a Bayesian Network. We will mainly be using WEKA for learning the CPTs in a fixed structure network using the SimpleEstimator algorithm (**?**) implemented there. This simple estimates the probability values on edges based on the frequency values of each variable given its parents in the training set.

2.3.2. Banjo

Banjo (Banjo) is a tool written in Java which infers the structure of a Bayesian network given training data. It has two different search algorithms: Greedy and Simulated Annealing. For either of these search algorithms, it has two methods of proposing a new edge. The first is the "proposeRandomLocalMove" which basically proposes a random addition or removal of an edge. The other is "proposeAllLocalMoves" which proposes all possible moves, and only keeps the best one (?). Banjo uses the BDe metric to compute a network's score.

2.3.3. JavaBayes

3. Related Work

Mirarab et al. (Mirarab et al., 2007) investigate the same problem as our work. However, their work is on the level of source code changes. They build three different Bayesian Networks, one that is based on package and class dependency information (static relationships), one which is dependent on historical co-changes, and one which uses both. For the first graph, the initial structure is essentially "given" according to the static dependencies, and then the CPTs are learnt using the importance sampling algorithm proposed by Changhe and Marek (Yuan & Druzdzel, 2003). The way static dependencies are defined in their case is specific to Java. The third one is essentially the first graph, but updated using the historic change information according to the Expectation Maximization (EM) algorithm (Dempster et al., 1977). The second was solely based on historic information where the network is build using a greedy structure learning algorithm (Friedman & Goldzsmidt, 1996). They did some preprocessing to their data such as filtering out large changes (with more than 30 elements changed at once) since this was probably an insignificant change.

Zhou et al. (Zhou et al., 2008) try to answer a slightly different problem. They do not only look at the probability of other elements changing given a specific element, they also add features such as authors, change significance levels etc. and try to predict if two elements are co-changes or not accordingly. Thus, their problem is more of a classification problem where given two elements, and some observed features they try to determine the class as co-changes or not. They use the K2 algorithm proposed by Cooper et. al (Cooper & Herskovits, 1992) to estimate the structure of the Bayesian network, and use the SimpleEstimator algorithm built in WEKA (Witten & Frank, 2005).

4. Constructing the Bayesian Network

Before constructing the Bayesian Network, we first had to process and prepare the data we have. Then, in order to construct a Bayesian network to use for predictions there are two steps involved. First, determining the structure of the actual network, and then estimating the Conditional Probability Tables (CPT). Section 4.1 first explains the data set available to us, and the data preprocessing involved before building the network. Section 4.2 then explains the different techniques we experimented with to build the network

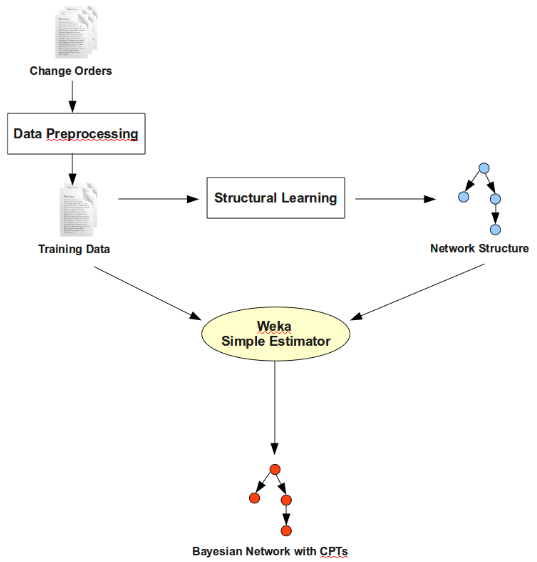

Figure 2. Building the Bayesian Network

structure. Section 4.3 shows the last step to build the network which is estimating the CPTs. Figure 2 shows this overall process.

4.1. Data Preprocessing

Data Size

The original data set is from three years, and has 7,999 distinct CIs, and 27,305 change orders. This amount of data was infeasible to work with as no tool could handle such a large amount of variables in a Bayesian network. For the purposes of this project, which is mainly to experiment with Bayesian techniques for change set prediction, it is sufficient to choose a small representative subset of the data. Accordingly, in order to be able to test things properly, we used observations from three months data from January 1, 2008 to March 31, 2008 to build the model. However, even in such a short period, there were already 2,841 distinct CIs appearing and 2,229 observations (i.e. change orders). Therefore, we needed to perform further data preprocessing.

First, we removed all CIs that have changed less than 12 times within this time frame (i.e were associated with 12 different change orders in our training observations). There is no particular reason for choosing 12 as a cut off. It simply gave a feasible data set to deal with. This yielded 120 distinct CIs. Then, in order to slightly increase our variable space to include other related CIs, we found all the parent CIs related to these 120 CIs from the CMDB perspective. However, we ignored three common, and not extremely meaningful relations, which are “supports”, “is location for”, and “backs up”. Adding the related parent CIs, we now had a set of 241 CIs. However, some of the added parent CIs may not be in the original set of 2,841 CIs. Accordingly, we just kept CIs tht appeared more than 12 times or were in the set of related CIs. This provided us our final data set of 170 CIs which we use throughout our models for fair comparison. Additionally, filtering out CIs meant filtering out some of the observations that did not have any CIs satisfying our criteria. This led to us having 1,305 observations instead of 2,229 which was a more manegeable set. Accordingly, our training data set consisted of 170 variables (CIs), and 1,305 observations (change sets).

FIGURE FOR DATA PROCESSING

Data Format

In the CMDB, a Change Order has several fields including the requester, the assignee, the start date of the change, the description and the change set field (called the ‘Configuration items’ fields). Of the fields in a change order, we use only the change set field. The advantage of only using this one field is that change sets are easy to extract and easy to understand. In terms of an observation, CIs were either marked as “true” or “false” indicating whether they have changed or not. Each observation initially consisted of all the in our data set marked as “false”. For each change order in the training set, we would mark each CI appearing in the “Configuration Items” field as true, while all those not appearing were left as “false”. Therefore, the set of training data consisted of an observation entry for each change order with the CIs marked as “true” or “false” accordingly.

4.2. Network Structure

There were different ways in which we could estimate the structure of the network, and so we built a Bayesian Network using each technique.

4.2.1. Model 1 (Banjo)

For the first model, we used Banjo to learn the structure of the Bayesian network from the training data. We use the Greedy searching algorithm implemented there. Banjo produced five different top-scoring networks (having the same score), and we simply chose one of them as the network to be used.

4.2.2. Model 2 (CMDB Relationships)

For the second model, we used the relations existing in the CMDB to infer the structure of the network (that is place the edges between the nodes). We used two variations for this. The first was to follow the direction of the relationship edges in the CMDB. That is if there is an edge from A to B, we will place an edge in the Bayesian network from A to B (we will call this model 2A). The second was to reverse the direction of the relationship edges in the CMDB. That is, if there is an edge from A to B, we will place an edge in the Bayesian network from B to A (we will call this model 2B). In both cases, however, we chose to ignore some relationship edges (WHY, TRY WITHOUT). There was one problem, however, with the generated network. Two of the CIs had more than 15 parents to them which means that their CPTs will be intractable to compute. For those two CIs, we simply removed all their parents to make the computation tractable.

4.2.3. Model 3 (Missing Data)

This model is the same as the second model, except that we experimented with the missing data feature in Weka. Weka allows the user to specify missing values for any variable by simply putting a '?' instead of its value. Therefore, instead of putting the CIs that did not appear in a change order as false, we put a '?' in their place. This seemed closer to practice, because in reality, we are not sure whether this CI actually changed or not. Therefore, this model also has two parts: model 3A and model 3B where the first uses the CMDB relationships in their same direction, and the second uses the reverse direction.

4.3. Estimating the CPTs

However, for all models, we used the SimpleEstimator algorithm (**?**) built in Weka to calculate the CPTs. After setting the data set to be the observations in our training set, Weka learned the CPTs for the network produced by Banjo. This complete network was then saved as an BIF XML file to be used by JavaBayes for general inference.

Algorithm 1 Generating Predictions using the Bayesian Network

Input: changeOrder
Input: BayesianNetwork
Input: threshold
Set initialCI = first CI in changeOrder
Set occurredSet = changeOrder - initialCI
Initialize observedSet = firstCI
Initialize predictedSet =
repeat
  Initialize newPredictions =
  for $node$ **to** $BayesianNetwork$ **do**
    posterior = perform inference using Banjo
    if $posterior > threshold$ **then**
      Add node to newPredictions
    end if
  end for
  for $prediction$ **to** $newPredictions$ **do**
    if $predictionoccurredSet$ **then**
      Add prediction to observedSet
    end if
  end for
  Add newPredictions to predictedSet
until $newPredictions$ is *empty*

5. Experiment Setup

5.1. Change Set Detection Process Simulation

The change set detection process provided is an iterative, collaborative process between the tool and the analyst. When the tool suggests CIs, the analyst can accept or reject these CIs (i.e add them to the change set or not). Based on the CIs added to the change set, the analyst can ask the tool for more suggestions. This is along the lines of "Now, that I am also going to change these CIs, what else do I need to change?".

To simulate this process for our experiments, we do the following. At this point, we have the Bayesian network ready, and we would like to perform Inference. More formally, given that a CI will change (our observation), we want to infer the probability that the other CIs in the network might change as well. Unfortunately, neither of the two tools previously used provide an Bayesian inference engine. We, therefore, use JavaBayes in this step since it accepts the same ARFF format used by Weka. We using a one month testing set where we try to predict all the change orders in April 2008. There was a total of 883 change orders in that month.

For each change order, we would take the first CI as the initial CI to change, then we would set that as an observed node, and update the beliefs of all the nodes (CIs) in the network using JavaBayes. We would then loop on all the updated CIs, and add those that match our threshold criteria to the predicted change set. To simulate a real life scenario, we then checked which of these predicted CIs actually lies in the target change set we are trying to predict. This is similar to an analyst accepting CIs into their change set. These common CIs would then also be marked as observations so that we can predict now that I'm going to change these accepted CIs as well, what else do I need to change. Again, we would calculate the posterior probability, and continue doing so until there are no more common CIs. All the CIs that match the threshold criteria (whether accepted by the analyst or not) are part of the predicted set.

5.2. Evaluation Techniques

We need a way to evaluate the predicted CIs. The recall and precision measures from the information retrieval field are appropriate for this type of evaluation. Recall measures the proportion of correct CIs retrieved by the system, while precision measures the proportion of suggested CIs that are correct (**?**).

Similar to Hassan et. al (**?**), we define the *Predicted Set* (P) as the set of all CIs DRACA suggests through the full the iteration process (see Figure 2). We define the *Occurred Set* (O) as the CIs remaining in the change set after excluding the Initial CI provided by the analyst (i.e Change Set - Initial CI). The intersection of the predicted set and the occurred set, called PO, is the common CIs in both sets. For each constructed change set, we then calculate the recall and precision values for the predictions according to the following definitions (**?**):

$$Recall = \frac{|PO|}{|O|} \tag{1}$$

$$Precision = \frac{|PO|}{|P|} \tag{2}$$

If no CIs are predicted (i.e., P and thus PO are empty), precision is defined as 1 since there cannot exist any incorrect predictions in an empty set. On the other hand, if the size of the change set is 1, and thus the size of the occurred set is 0, recall is defined as 1 since there are no CIs to predict (**?**).

In order to have a single measure that indicates the effectiveness of our predictions, we use the F-measure which is based on van Rijsbergen's effectiveness measure which combines recall and precision (**?**). The F-measure is calculated according to Equation 3 which gives equal weighting to recall and precision. The ideal F-measure is 1 where both recall and precision are 1.

$$F = 2 * \frac{precision * recall}{precision + recall} \tag{3}$$

Algorithm 1 shows this process, and how the recall and precision are calculated accordingly.

6. Results

7. Conclusion

References

Banjo. Banjo. http://www.cs.duke.edu/~amink/software/banjo/.

Cooper, G.F. and Herskovits, E. A Bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992. ISSN 0885-6125.

Dempster, A.P., Laird, N.M., Rubin, D.B., et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 0035-9246.

Friedman, N. and Goldzsmidt, M. Learning Bayesian Networks with Local Structure. 1996.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I.H. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009. ISSN 1931-0145.

Mirarab, S., Hassouna, A., and Tahvildari, L. Using bayesian belief networks to predict change propagation in software systems. pp. 177 –188, jun. 2007.

Nadi, S., Holt, R., and Mankovskii, S. Does the past say it all? Using history to predict change sets in a CMDB. In *CSMR'10: Proceedings of the 14th European Conference on Software Maintenance and Reengineering*, Madrid, Spain, 2010.

Office of Government Commerce (OGC), ed.: Service Support. Office of government commerce (ogc), ed.: Service support. *IT Infrastructure Library (ITIL)*, 2000.

Witten, I.H. and Frank, E. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann Pub, 2005. ISBN 0120884070.

Yuan, C. and Druzdzel, M.J. An importance sampling algorithm based on evidence pre-propagation. In *Proceedings of the 19th Annual Conference on Uncertainty on Artificial Intelligence*, pp. 624–631. Citeseer, 2003.

Zhou, Yu, Würsch, Michael, Giger, Emanuel, Gall, Harald C., and L?, Jian. A bayesian network based approach for change coupling prediction. *Reverse Engineering, Working Conference on*, 0:27–36, 2008. ISSN 1095-1350. doi: http://doi.ieeecomputersociety.org/10.1109/WCRE.2008.39.