Discriminative sequence mining: analysis and alternative applications

Erik Mägi
University of Tartu
erik85@ut.ee

Fabrizio Maria Maggi *
University of Tartu
f.m.maggi@ut.ee

ABSTRACT
This paper gives an overview of a discriminative sequence mining method proposed by Lo et al. [5] and describes experiments to apply the method on medical data.

1. INTRODUCTION
Many processes can be viewed as sequences of events or tokens from a finite set. For example a program execution may look something like this: main() → preprocess(fpPath) → split_classes(data) → collect_stats(classes) → present_stats(stats).

If we have many traces from a program that cover enough of its possible states, condition branches etc., then it may be possible to use that data to find patterns that are related to hard-to-find bugs (i.e. bugs that don’t necessarily result in a crash but create wrong results, for example).

A method for finding such patterns is proposed in [5]. Although the paper is about detecting software bugs, the method itself could be applicable to other types of data. In this paper we try to apply the method to medical data that is similarly structured – each trace is a sequence of events.

2. OBJECTIVES
Main objectives of this paper are:
• to get an overview of discriminative sequence mining
• to get familiar with the toolset provided together with [5]
• to apply discriminative sequence mining on medical data [6]

3. DISCRIMINATIVE SEQUENCE MINING
This section gives a brief overview of the method proposed in [5]. It is not intended as a comprehensive summary of the method proposed by Lo et al. but rather a set of short concept descriptions that help to put this paper in context.

3.1 Terminology
To simplify reading, we briefly explain key terms used in this paper.

Definition 1. Event is a unit of behaviour of interest, e.g. a (logged) method call, statement execution etc.

Definition 2. Trace is a sequence of events $\langle e_1, e_2, \ldots, e_n \rangle$.

The set of traces under consideration is denoted as $TDB$.

Definition 3. (Pattern instance) Given a pattern $P(e_1, e_2, \ldots, e_n)$, a consecutive series of events $SB(sb_1, sb_2, \ldots, sb_m)$ in a sequence $S$ in $TDB$ is an instance of $P$ iff it is of the following QRE expression $e_1; [-e_1, \ldots, e_n]^*; e_2; \ldots; e_1; [-e_1, \ldots, e_n]^*; e_n$ [5]

Definition 4. The number of instances of a pattern $P$ in the $TDB$ is referred as the support of the pattern and denoted as $sup(P)$.

Definition 5. (Frequent iterative pattern) A pattern is frequent if its occurrence in a trace database $TDB$ is above certain threshold $min_sup$.

Definition 6. (Closed iterative pattern) A frequent iterative pattern $P$ is closed if there exists no such supersequence $Q$ that $P$ and $Q$ have the same support and every instance of $P$ corresponds to a unique instance of $Q$.

Definition 7. (Closed unique pattern) A frequent pattern $P$ is a closed unique pattern if $P$ contains no repeated constituent events. As an example consider a sequence $S(A, B, B, B, C, E, D, A, B, B, C, E, D, A, B, B)$. Assuming that $min_sup = 2$, the pattern $\langle A, B \rangle$ is a closed unique pattern. It contains unique elements A and B, and there are no longer unique patterns having the same support as $\langle A, B \rangle$. Pattern $\langle C, D \rangle$ is also unique but it is not closed as there exists a longer pattern $\langle C, E, D \rangle$ which is also unique.

*supervisor, Institute of Computer Science

1Quantified Regular Expression – similar to standard regular expression with ‘;’ as the concatenation operator, ‘[-]’ as the exclusion operator and ‘*’ as the standard Kleene star.
3.2 Method overview
In short, discriminative sequence mining attempts to find highly discriminative sequences between two classes (positive/negative, success/failure etc.) that are then used to train a classifier.

Discriminative sequence mining can be divided into three steps:

1. Mine closed unique iterative patterns from a software trace database TDB using Algorithm 1 [5].
2. Select discriminative iterative patterns from the pattern set in step 1 using Algorithm 2 [5]. Then represent the trace database TDB in the feature space of the selected iterative patterns.
3. Train a classifier from the trace database TDB. [5] uses an SVM with probability estimates the classification model.

3.3 Pattern mining
In this step closed unique patterns are mined. First, frequent single events are determined. Then frequent events are grown into closed unique patterns.

3.4 Pattern and feature selection
Patterns mined in the previous section are considered the initial set of features. Since the previous step is likely to produce a large number of patterns, some selection is necessary.

The patterns are evaluated based on their discriminative power which is calculated using the Fisher algorithm. Patterns are then sorted in the decreasing order of their Fischer score.

The algorithm then iterates over the ordered patterns and selects those patterns that are found in at least one sequence $S$ in the TDB. If a sequence $S$ has been covered (it includes a given pattern) at least $\delta$ times, it is removed from the TDB; $\delta$ is the coverage threshold. The algorithm terminates if all sequences are covered by the patterns or the TDB becomes empty.

3.5 Pattern-based classification
For classification Lo et al. use an SVM. The entire pipeline looks like this:

1. mine closed unique patterns
2. select features
3. convert selected features for SVM
4. train classifier using a TDB

3.6 A note on skewed distributions
Lo et al. use a five fold cross-validation scheme for pattern mining, feature selection and model training. Each training set is then rebalanced by duplicating the minority class until the ratio of positive/negative classes in a training set is 1.0.

3.7 Overview of the original test data and results
Lo et al. use three datasets for experiments: Siemens Test Suite [4], their own dataset based on a known MySQL bug [1] and a synthetic dataset generated based on models of the X11 windowing system and a particular CVS system.

The results are presented by comparing the proposed pattern-based approach to a baseline method that uses single events as features rather than patterns (loss of temporal information). The results show that the proposed method has significantly better results in cases where there are few traces where a bug manifests and the order of events matters.

Each dataset consisted of five sub-datasets (five folds) and in each fold there were four files: $train-m.txt$, $train-c.txt$, $test-m.txt$ and $test-c.txt$. $*-m.files$ contained only event sequences while $*-c.files$ contained event sequences together with class info they belonged to.

3.8 Overview of the provided tools
A set of tools is also provided that implements the proposed algorithm(s). TPMiner.exe for pattern mining and patclassify.exe for feature selection.

4. MINING BPI CHALLENGE 2011 DATA
The dataset [6] is a real-life log, taken from a Dutch Academic Hospital. It consist of more than 150k individual events that are divided between 1143 sequences. It is provided in several formats, we used XES.

![Figure 1: Main characteristics of the BPI Challenge 2011 dataset](image)

As can be seen from from Figure 1, the events are not evenly distributed across individual sequences and the number of unique events/classes is quite high at $>600$.

The original dataset contains class names that are in Dutch, we decided to use a version translated into English provided by one of the challenge contestants [2].

4.1 Defining classes
To apply discriminative sequence we need to split the data into two classes – one conforming to some criteria that we are interested in and the other not. We will call these classes positive and negative.
We used the LTL Checker plugin in ProM to create the classes, the used LTL rules are provided in Table 1. Rules \( \varphi_1 \) and \( \varphi_2 \) resulted in well-balanced classes where positive and negative instances were divided nearly equally, thus requiring minimal rebalancing. Rules \( \varphi_2 \) and \( \varphi_3 \) resulted in heavily skewed distributions where positive instances were < 5 %.

\[
\begin{align*}
\varphi_1 & : F(\text{"tumor marker CA} - 19.9") \\
\varphi_2 & : F(\text{"ca} - 125 \text{ using meia"}) \vee F(\text{"ultrasound} - \text{internal genitals"}) \\
\varphi_3 & : F(\text{"ovarian} - \text{ovarian re} \text{-} \text{debulking CarCine"}) \\
\varphi_4 & : F(\text{"thorax"}) \\
\end{align*}
\]

Table 1: Used LTL rules

To test how much does class rebalancing affect the results we created two datasets for each rule – one with rebalancing enabled and the in other one disabled.

4.2 Preprocessing

Since the tools provided use their own format for input files, we needed to convert our XES files. We created a small converter application that uses the OpenXES java library.

Although there is no detailed explanation of the file format used by TPMiner.exe, we could deduce that each line in the file represented a single sequence and a sequence was comprised of event ids separated with spaces. The ids are positive integers.

Listing 2: Structure of *-c.txt files

id1 id2 ... idn cls (0/1)

Listing 4: Example of a *-c.txt file

9 1 68 70 72 75 78 88 0
56 56 60 68 70 72 75 78 88 1
56 78 88 0
60 68 70 72 75 78 88 1

Note that the last line in a *-m.txt is also suffixed with a -2. We assume this indicates TPMiner.exe that the last sequence has been read.

4.3 Reducing the number of unique classes

The number of unique event classes is over 600 in the BPI dataset, this means significantly increased search space for pattern mining compared to the original datasets where the maximum number of unique events was over 300. The pattern searching algorithm essentially tries to generate all possible patterns based from the unique events so the search space will explode quickly.

To keep run times manageable, we decided to filter out all events that had occurrence rate of < 1 %. This left us with only 23 event classes. While this inevitably will affect the results in that more nuanced features will be lost, the temporal order will still remain the same so the captured patterns will be 'broader'.

For example a pattern \( P(A, B, C, C, B, D, E, A) \) becomes \((A, B, B, E, A)\) if events C and D are infrequent.

4.4 Generating training sets

As Lo et al., we also used 5-fold cross validation. Each fold had the same distribution of positive and negative instances. In case of rebalancing the minority class was duplicated with random samples from the minority class instances in a particular fold. Minority class is the class that has the least number of instances in a dataset.

4.5 Mining patterns

TPMiner.exe takes as input the directory of the cross validation data. In each fold’s directory it then creates two files: pattrain.txt and pattest.txt, based on the training and testing data respectively. The format of the files is described in Listing 7

Listing 5: Format of the pattrain.txt and pattest.txt files

\(<\text{pattern}> : <\text{total #}> \ (\text{line #, # in line})\)

Listing 8: Example of a pattrain/pattest.txt file

0 : 1981 (1,20) (2,29) (3,36) ... 
0 15 : 932 (1,12) (2,16) (3,18) ... 
0 15 17 : 449 (1,8) (2,9) (3,11) ... 
0 15 17 19 : 68 (2,3) (3,2) (6,5) ... 
0 15 17 18 : 76 (2,3) (3,4) (6,6) ... 
0 15 17 18 19 : 66 (2,3) (3,2) (6,5) ...

4.6 Training and testing the classifier

We use LIBSVM [3] to create a classifier. To convert the found pattern data into a format that LIBSVM can use, a tool is provided – patclassify.exe. The format itself is described in [3].
For each fold patclassify.exe creates files crosstrain<fold#>.txt and crosstest<fold#>.txt.

Listing 7: Command line used to start patclassify.exe

```
./patclassify.exe train-c.txt test-c.txt 
pattrain.txt 5 3 1 \
crosstrain0.txt \
crosstest0.txt 
pattest.txt 0
```

Since the original paper only mentions that LIBSVM [3] is used with probability estimates as the classification model, we assume that other parameters of LIBSVM are left at the default values. For training we use the `svm-train` tool from the package.

For testing we use the `svm-test` tool. The accuracy of each model is determined by averaging all test results (all folds).

4.7 Process pipeline summary

Steps necessary to mine the BPI challenge and create a pattern-based classifier:

1. Split the dataset into two classes: positive and negative.
2. Convert the resulting XES files to the necessary format, remove infrequent classes.
3. Split into smaller datasets for cross validation, create test and training sets, rebalance training sets.
4. Mine the folds for closed unique patterns patterns using the provided tool TPMiner.exe.
5. Run patclassify.exe on each fold and convert the patterns for LIBSVM.
6. Run `svm-train` for each fold, this creates a model.
7. Run `svm-predict` for each fold, average the results.

5. RESULTS

Table 5 shows classification accuracy for all models that were trained based on the results from each respective rule. It is interesting to note that in most cases the accuracy increases if balancing is disabled. A possible explanation for this is that balancing duplicates existing data then it causes the model to overfit.

<table>
<thead>
<tr>
<th>Rule</th>
<th>accuracy (balancing enabled)</th>
<th>accuracy (balancing disabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \varphi_1 )</td>
<td>( 72.72% ) (balancing enabled)</td>
<td>( 70.96% ) (balancing disabled)</td>
</tr>
<tr>
<td></td>
<td>( \sigma = 10.65 )</td>
<td>( \sigma = 6.20 )</td>
</tr>
<tr>
<td>( \varphi_2 )</td>
<td>( 91.35% ) (balancing enabled)</td>
<td>( 92.08% ) (balancing disabled)</td>
</tr>
<tr>
<td></td>
<td>( \sigma = 2.89 )</td>
<td>( \sigma = 0.05 )</td>
</tr>
<tr>
<td>( \varphi_3 )</td>
<td>( 96.54% ) (balancing enabled)</td>
<td>( 96.88% ) (balancing disabled)</td>
</tr>
<tr>
<td></td>
<td>( \sigma = 1.07 )</td>
<td>( \sigma = 0.67 )</td>
</tr>
<tr>
<td>( \varphi_4 )</td>
<td>( 78.91% ) (balancing enabled)</td>
<td>( 82.01% ) (balancing disabled)</td>
</tr>
<tr>
<td></td>
<td>( \sigma = 4.74 )</td>
<td>( \sigma = 7.18 )</td>
</tr>
</tbody>
</table>

Table 2: Classification accuracies for each rule, \( \sigma \) denotes standard deviation across fold test results.

Tables 3 - 6 show the most discriminative patterns found from for each rule in the decreasing order of their discriminative power. 0 or 1 on front of each pattern indicates whether the pattern is mostly found in positive or negative cases. The discriminative power of a pattern is determined by its Fischer score which is defined as:

\[
Fr = \frac{\sum_{i=1}^{c} n_i (\mu_i - \mu)^2}{\sum_{i=1}^{c} n_i \sigma_i^2}
\]

where \( n_i \) is the number of samples in class \( i \), \( \mu_i \) is the average occurrence of the feature in class \( i \), \( \sigma_i \) is the standard deviation of the feature in class \( i \) and \( \mu \) is the average occurrence of the feature in the whole dataset.

<table>
<thead>
<tr>
<th>( \varphi_1 ) (balancing enabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 [assumption laboratory]</td>
</tr>
<tr>
<td>2 1 [order rate]</td>
</tr>
<tr>
<td>3 1 [order rate] -&gt; [assumption laboratory]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \varphi_1 ) (balancing disabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 [demurrage - all spec.beh.kinderg.-Reval.]</td>
</tr>
<tr>
<td>2 0 [demurrage - all spec.beh.kinderg.-Reval.] -&gt; [190205 Class 3b A205]</td>
</tr>
<tr>
<td>3 0 [190205 Class 3b A205] -&gt; [demurrage - all spec.beh.kinderg.-Reval.]</td>
</tr>
</tbody>
</table>

Table 3: Top three patterns for rule \( \varphi_1 \)

<table>
<thead>
<tr>
<th>( \varphi_2 ) (balancing enabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 [glucose] -&gt; [creatinine] -&gt; [190205 Class 3b A205] -&gt; [190101 reg.toesl above. A101] -&gt; [assumption laboratory]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \varphi_2 ) (balancing disabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 [assumption laboratory]</td>
</tr>
<tr>
<td>2 1 [order rate]</td>
</tr>
<tr>
<td>3 1 [order rate]</td>
</tr>
</tbody>
</table>

Table 4: Top three patterns for rule \( \varphi_2 \)

It can be seen that for rules with heavily skewed distributions \( \varphi_2 \) and \( \varphi_4 \) the top patterns are shorter if the balancing is disabled. It may indicate that with balancing, the model is overfitted and very specific and long patterns start to dominate.

The results above show that applying this method on severely skewed distributions may result in overfitted models. Although this is not conclusive as the number of tested rules is small, it does highlight the potential problems that may occur if rebalancing is used.

6. PROBLEMS ENCOUNTERED

Since the toolset provided with [5] came with minimal documentation and no source code, it might be of some use to someone if the experienced issues are documented.
Run-time of the tool is not easily predictable. Although we filtered out all infrequent events so that we had only 23 events, the run times for rules varied significantly from 5 - 15 minutes to over 10 hours for $\varphi_2$. We suspect that this may be caused by the big variation in sequence lengths but it requires more analysis. Because of the performance issues the number of rules we could experiment with was limited.

The BPI challenge dataset event names are of poor quality making it hard to visually inspect the results and try to put them in context.

### 7. CONCLUSION

Lo et al. demonstrated that the proposed method performs well in cases where temporal information is important – namely program execution logs. For example in cases where the order in which methods are called determines whether an error occurs or not (a possible case might be inadequate syncing between threads).

As medical data is also of similar structure, at least in principle, this method could be used to mine medical logs to find patterns that with great probability would detect a particular medical condition or predict its onset in the future. Another possible outcome would be finding some interesting patterns in the managerial area – maybe some stages of treatment/diagnosis could be organized more efficiently etc.

However, due to performance issues and the BPI challenge dataset attributes – relatively small size and very specific data (events only from gynaecology clinic) we cannot yet fully assess how fitting this method would be for analysing medical data and additional analysis is needed.

### References


