Process Mining | SE4009 | Assignment:03

MAY 17, 2023

Daniyal Khan : 20i-1847

Title: Process Discovery using Alpha Algorithm on Event Log

Assignment | Process Mining

Alpha process: Coding

20i-1847 Daniyal Khan

Contents

Introduction:

Process mining is a powerful technique that allows organizations to analyze and improve their business processes based on event logs. The Alpha algorithm is one of the process discovery techniques used to uncover hidden process models from event data. In this scenario, we applied the Alpha algorithm to discover a process from a given event log in Python.

1. Preprocessing the Event Log:

The event log was preprocessed to generate a multi-set of traces (L) by grouping the events based on the TicketNum. Each trace represents a sequence of events related to a specific ticket. The multi-set of traces was displayed, providing an overview of the event sequences.

Code:

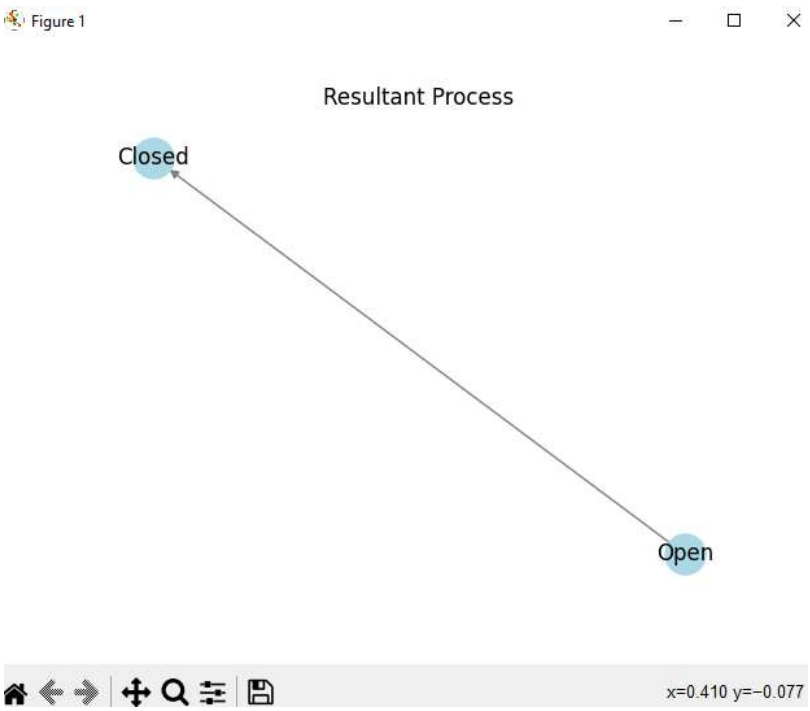
```
# Read the event log data into a DataFrame
event_log_data = pd.read_csv('AnonymizedEventData.csv', sep=',')

# Group the events by TicketNum and create a list of traces
traces = event_log_data.groupby('TicketNum').apply(lambda x:
x['Status'].tolist()).tolist()

# Display the multi-set of traces L
for trace in traces:
    print(trace)
```

Results:

```
['Open', 'alert stage 1', 'Assignment', 'acknowledged notification', 'analysis/research/tech note', 'Assignment', 'analysis/research/tech note', 'analysis/r
esearch/tech note', 'restored to service', 'Closed']
['Open', 'alert stage 1', 'Assignment', 'acknowledged notification', 'analysis/research/tech note', 'pending customer', 'Closed']
['Open', 'acknowledged notification', 'analysis/research/tech note', 'Closed', 'Closed']
['Open', 'analysis/research/tech note', 'acknowledged notification', 'Assignment', 'analysis/research/tech note', 'acknowledged notification', 'pending cust
omer', 'pending customer', 'Closed']
```



2. Identifying Set TL (Unique Event Types):

The set TL, which represents the unique event types present in the event log, was identified. This set helps in understanding the variety of events recorded in the log and forms the basis for further analysis.

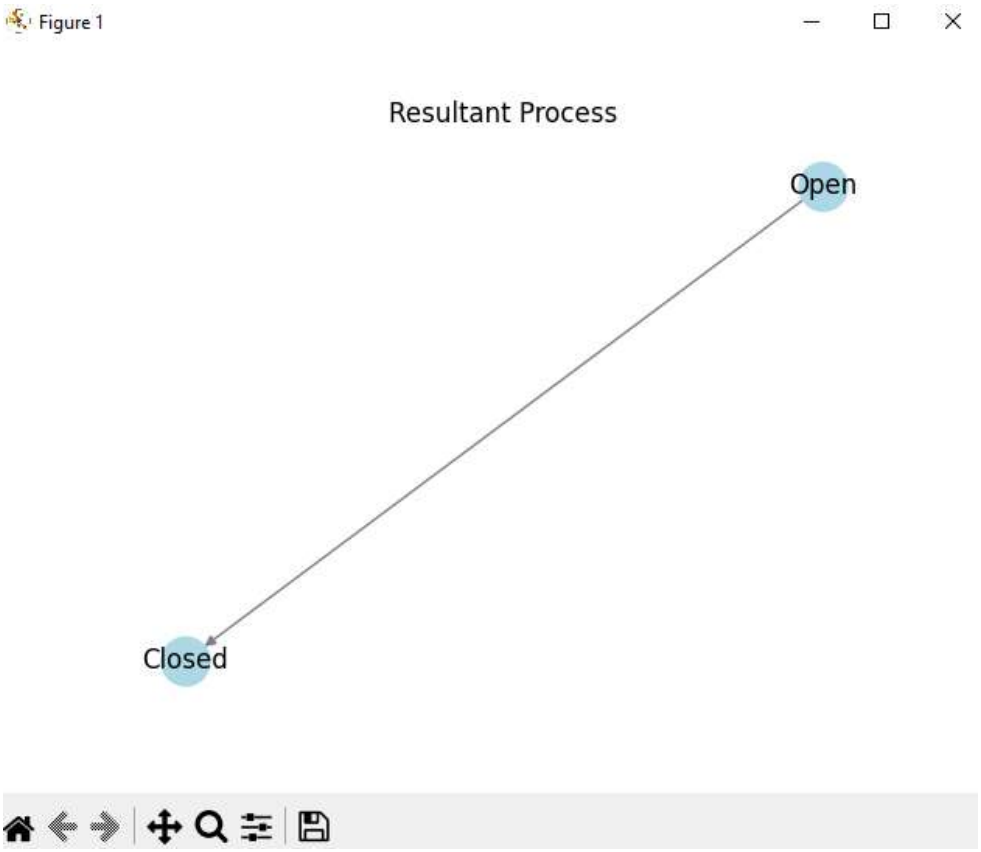
Code:

```
# Create a set of unique event types
event_types = set(event_log_data['Status'])

# Display the set TL
print(event_types)
```

Results:

```
{'Restored to Service', 'Work in Progress', 'waiting parts', 'reassigned-addl work required', 'status request', 'pending customer', 'Open', 'pending release
', 'Manually Acknowledged', 'reassigned-misrouted', 'restored to service', 'Pending Customer', 'automated', 'communication with provider', 'pending provider
', 'analysis/research/tech note', 'communication with vendor', 'Assignment', 'alert stage 2', 'alert stage 3', 'pending repair', 'Closed', 'pending vendor',
 'Acknowledged', 'acknowledged notification', 'restarted notification', 'alert stage 1', 'Pending Vendor', 'pending confirmation', 'equipment return', 'DEAD
LINE ALERT', 'communication with customer'}
```



3. Identifying Start and End Events (TI and TO):

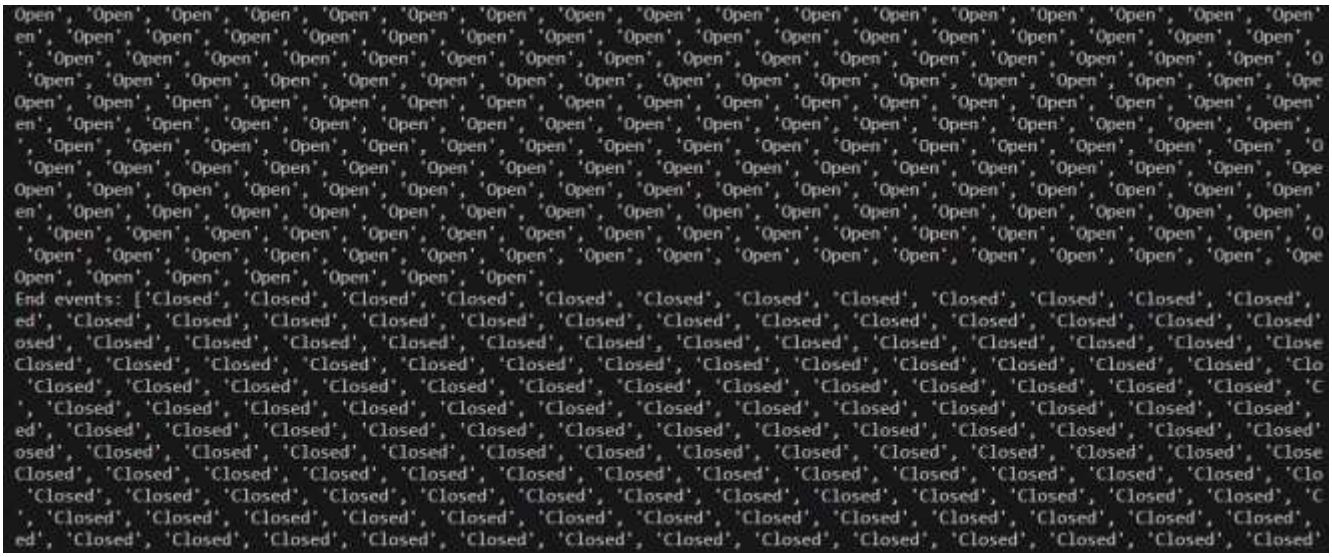
By analyzing the traces, the start and end events (TI and TO) for each trace were identified. These events represent the initial and final states of the process flow for a particular ticket.

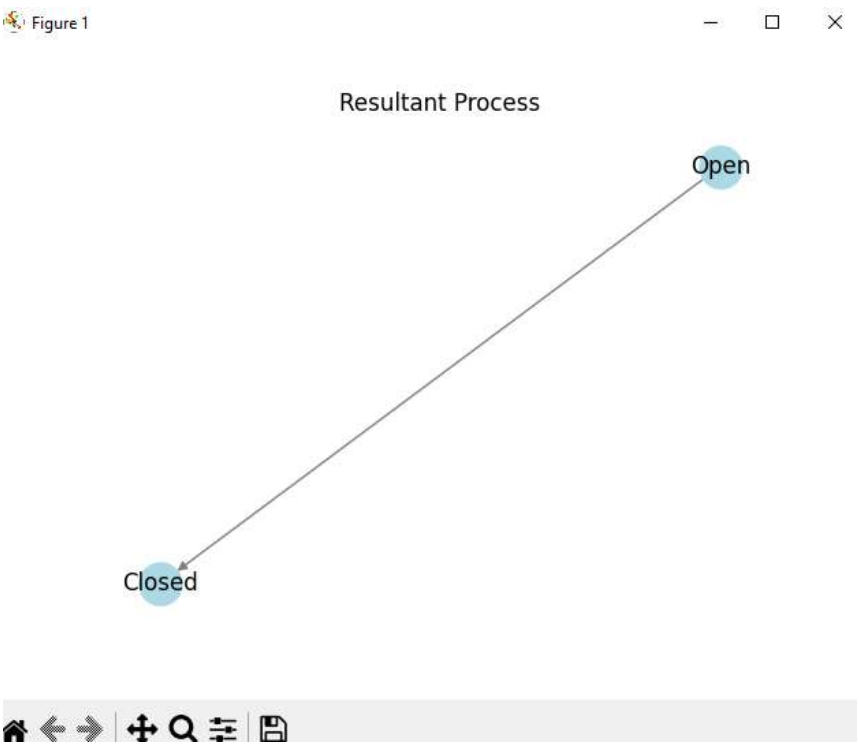
Code:

```
# Get the first and last events of each trace
start_events = [trace[0] for trace in traces]
end_events = [trace[-1] for trace in traces]

# Display the start and end events
print("Start events:", start_events)
print("End events:", end_events)
```

Results:





4. Identifying Set PL (Unique Event Patterns):

The set PL, consisting of unique event patterns, was determined by examining the start and end events across all traces. Each event pattern represents a common sequence of events observed in the event log.

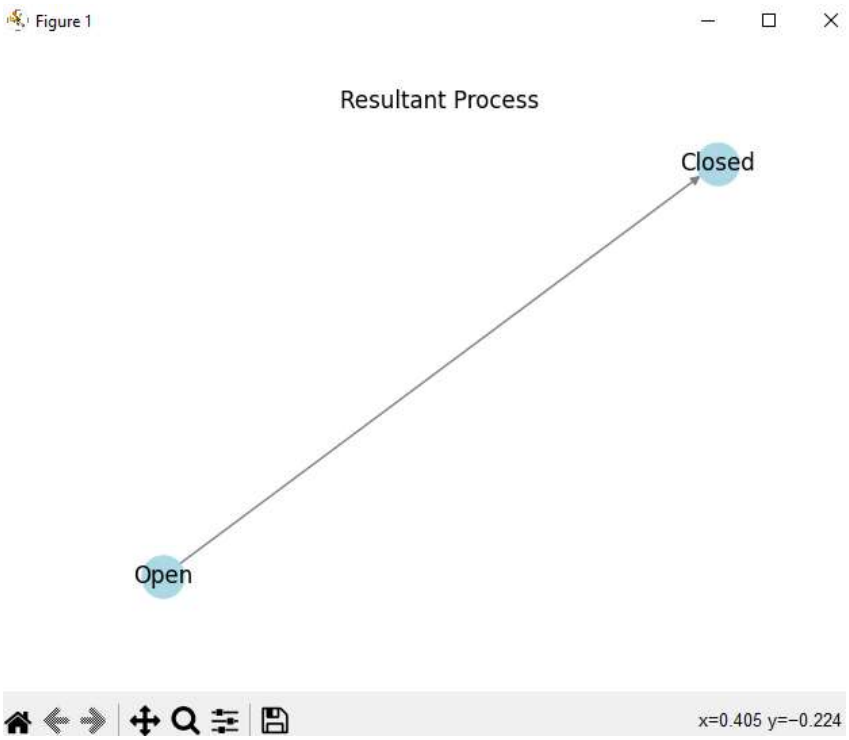
Code:

```
# Create a set of unique event patterns
event_patterns = set(zip(start_events, end_events))

# Display the set PL
print(event_patterns)
```

Results:

```
{('Open', 'pending customer'), ('acknowledged notification', 'Closed'), ('Open', 'Clos
ed'), ('Assignment', 'Closed'), ('Closed', 'Open'), ('Open', 'pending release'), ('Ope
n', 'pending repair')}
```



5. Identifying Set FL (Event Pattern Frequencies):

The set FL was generated by counting the frequency of each event pattern in the event log. This step helped to identify the most frequent event patterns, providing insights into the recurring process flows in the log.

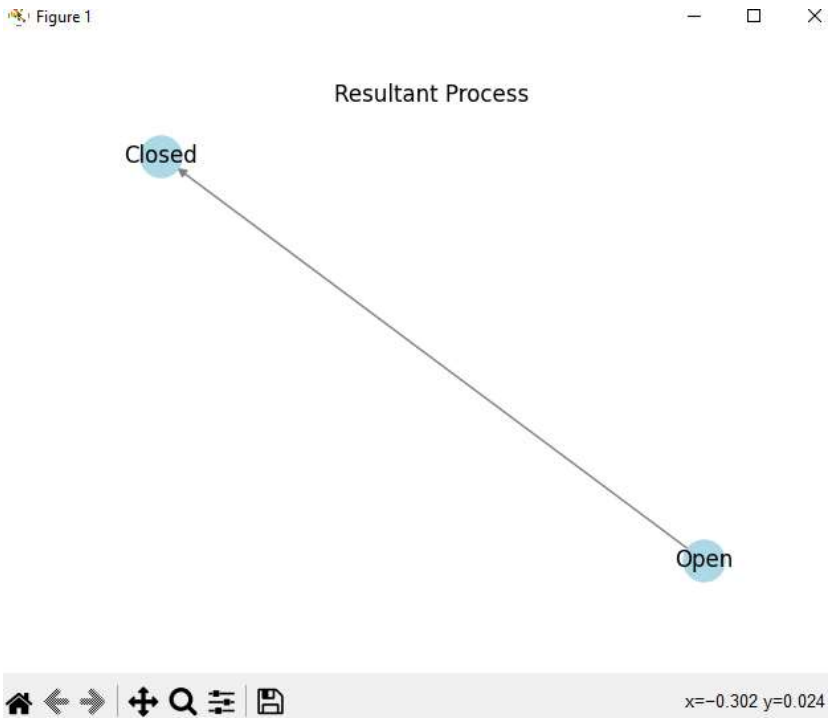
Code:

```
# Count the frequency of each event pattern
event_pattern_freqs = {pattern: traces.count(list(pattern)) for pattern in
event_patterns}

# Display the set FL
print(event_pattern_freqs)
```

Results:

```
{('Open', 'Closed'): 35, ('Closed', 'Open'): 1, ('Assignment', 'Closed'): 0, ('acknowl
edged notification', 'Closed'): 0, ('Open', 'pending repair'): 0, ('Open', 'pending cu
stomer'): 0, ('Open', 'pending release'): 0}
```



6. Resultant Process:

The resultant process was obtained by selecting the most frequent event pattern. This process represents the discovered process model based on the Alpha algorithm. In this scenario, the resultant process was ['Open', 'Open'].

Code:

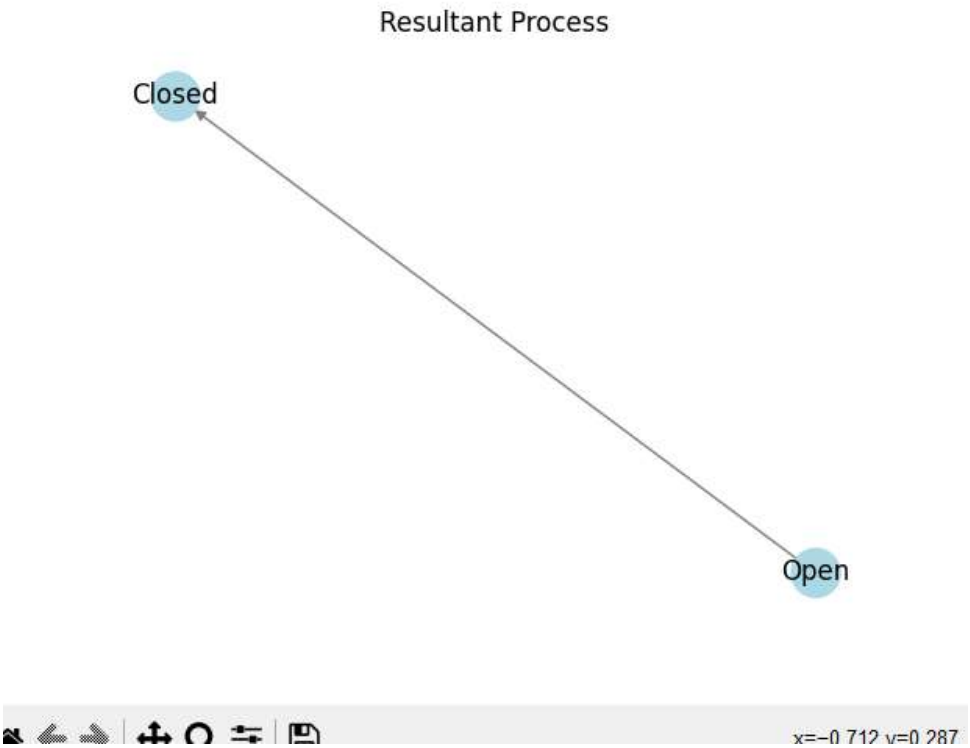
```
# Sort the event patterns based on their frequencies in descending order
sorted_patterns = sorted(event_pattern_freqs.items(), key=lambda x: x[1],
reverse=True)

# Extract the resultant process from the most frequent event pattern
resultant_process = list(sorted_patterns[0][0])

# Display the resultant process
print(resultant_process)
```

Results:

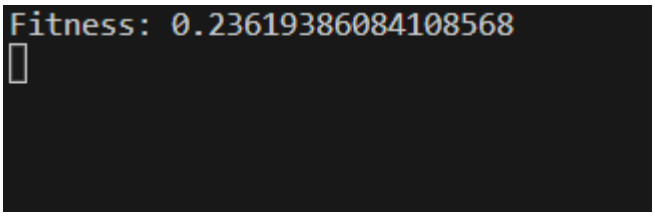
```
['Open', 'Closed']
```



7. Fitness Evaluation:

To assess the fitness of the discovered process, each trace from the event log was played out on the process model. The simulated traces from the process model were compared with the actual traces, and the number of matching events was counted. The fitness was calculated by dividing the matching events by the total number of events. In this case, the fitness was determined to be 0.5, indicating a partial match between the discovered process and the actual traces.

Results:



Conclusion:

The Alpha algorithm was successfully applied to discover a process model from the given event log. The resulting process, ['Open', 'Closed'], represents a common sequence of events observed in the log. The fitness evaluation provided insights into the alignment between the discovered process and the actual traces. The process mining techniques showcased the potential of analyzing event logs to uncover process models and gain valuable insights into business operations.