Raport do ćwiczenia realizującego część I z przedmiotu MiAPB

24.03.2022

Autorzy: Roman Dembrovskyi, Filim Żmijewski

Ćwiczenia

1. **Zmodyfikuj etykiety zadań w modelu, tak by dla każdego zadania oprócz nazwy wyświetlana była także liczba wystąpień odpowiadającego mu zdarzenia w logu. Dodaj opcję filtrowania (po zdarzeniach lub przepływach) w zależności od określonych progów, aby pokazać lub ukryć zadania lub przepływy zgodnie z wybranym progiem. Zaprezentuj modele dla wybranych progów np. 420 dla przepływów, 700 dla zdarzeń (osobno i razem).**

In [ ]:

```
#modyfikacja etykiet
 for event, succesors in w_net.items():
  value = ev_counter[event]

  if float(color_min-color_max) == 0:
    color = int(80)
  else:
    color = int(float(color_min-value)/float(color_min-color_max)*100.00)

  my_color = "#ff9933"+str(hex(color))[2:]
  G.add_node(event, style="rounded,filled", fillcolor=my_color, label=f'{event} - {value
}')
  for succesor, cnt in succesors.items():
    if float(color_min-color_max) == 0:
      G.add_edge(event, succesor, penwidth=4*cnt/10000, label=cnt) #dodanie wyświetlenia
liczby wystąpień w postaci cnt
    else:
      G.add_edge(event, succesor, penwidth=4*cnt/(trace_max-trace_min)+0.1, label=cnt) #
dodanie wyświetlenia liczby wystąpień w postaci cnt
 G.add_node("end", shape="circle", label="", penwidth='3')
 for ev_end in ev_end_set:
   G.add_edge(ev_end, "end")
```

In [ ]:

```
#Event filtering
ev_counter = dfs.Activity.value_counts()

def threshold_filter(threshold, counter_dict):
  for event, count in counter_dict.items():
    if count < threshold:
      print(event)
      print(count)
      counter_dict.pop(event)
      dfs.drop(dfs[dfs['Activity'] == event].index, inplace=True)


threshold_filter(filter1, ev_counter)
ev_counter = dfs.Activity.value_counts()
dfs.Activity.value_counts()

#Trace filtering
def trace_filter(traceNO, trace_counter):
  for count in trace_counter:
    if count < traceNO:
      dfs.drop(dfs[dfs['count'] == count].index, inplace=True)

trace_filter(filter2, dfs['count'])
```

```
dfs
```

In [1]:

```
from IPython.display import Image
print('filtering events; threshold = 700')
Image(url="filter_700.png")
```

```
filtering events; threshold = 700
```

Out[1]:

In [2]:

```
from IPython.display import Image
print('filtering events and traces; threshold = 700;420 respectively')
Image(url="filter_700_420.png")
```

```
filtering events and traces; threshold = 700;420 respectively
```

Out[2]:

1. **Dodaj możliwość ustawiania progów przez użytkownika (poprzez podanie liczby lub użycie suwaka z `ipywidgets`) i wyświetlania przefiltrowanego modelu. Przetestuj jego działanie na różnych progach, aby określić, czy model wygląda prawidłowo, w szczególności, czy jakieś zdania nie zostają odczepione od modelu lub nie są prawidłowo połączone z modelem, np. jeśli wcześniej zadanie występowało pomiędzy innymi zadaniami nie należy filtrować wszystkich przepływów, nawet jeśli są poniżej progu.**

In [ ]:

```
#sterowanie progami poprzez UI suwaków
slider1 = widgets.IntSlider(
    value=1000,
    min=350,
    max=1400,
    step=1,
    description='Filter by tasks:',
    readout_format='d'
)

slider2 = widgets.IntSlider(
    value=300,
    min=0,
    max=500,
    step=1,
    description='Filter by trace:',
    readout_format='d'
)
```

In [ ]:

```
display(slider1)
print(slider1.value)
filter1 = slider1.value
```

In [ ]:

```
display(slider2)
print(slider2.value)
filter2 = slider2.value
```

In [3]:

```
print('filtering events and traces; threshold = 1129;0 respectively')
Image(url="filter_1129_0.png")
```

```
filtering events and traces; threshold = 1129;0 respectively
```

Out[3]:

1. **Zmodyfikuj progowanie, tak aby w takim wypadku zachować najlepszy przepływ, aby zadanie było połączone z pozostałymi co najmniej jednym wchodzącym i co najmniej jednym wychodzącym przepływem. Zaprezentuj modele po prostej filtracji i porównaj je z modelami po poprawionej filtracji.**

In [ ]:

```
#kod, odpowiadający za zachowanie najlepszego przepływu
for key, value in w_net.items():
    cnt_dict = dict(value)
    if len(cnt_dict) > 1:
        order = list(cnt_dict.keys())
        order.pop(0)
        for key2 in order:
            if cnt_dict[key2] < filter2:
                cnt_dict.pop(key2)
    value = Counter(cnt_dict)
    print(key,":",value)

    w_net[key] = value
```

In [4]:

```
print('old trace filtering; threshold = 1000;300 respectively')
Image(url="filter_100_300.png")
```

```
old trace filtering; threshold = 1000;300 respectively
```

Out[4]:

In [8]:

```
print('new trace filtering; threshold = 1000;300 respectively')
Image(url="new_1000_300.png")
```

```
new trace filtering; threshold = 1000;300 respectively
```

Out[8]:

In [10]:

```
print('new trace filtering for comparising; threshold = 1000;560 respectively')
Image(url="new_1000_560.png")
```

```
new trace filtering for comparising; threshold = 1000;560 respectively
```

Out[10]:

Przy prostym filtorowaniu w przypadku usunięcia wszystkich przejść jednego ze zdarzeń występował błąd indeksacji.

1. **Dodaj perspektywę wydajności, tzn. oblicz średni czas trwania i pokoloruj zadania zgodnie z czasem ich trwania, a przepływy odpowiedniej grubości.**

In [ ]:

```
trace_counts = sorted(chain(*[c.values() for c in w_net.values()]))
trace_min = trace_counts[0]
trace_max = trace_counts[-1]
color_min = avgTime.min()
color_max = avgTime.max()

G = pgv.AGraph(strict= False, directed=True)
 G.graph_attr['rankdir'] = 'LR'
```

```
 G.node_attr['shape'] = 'Mrecord'

 G.add_node("start", shape="circle", label="")
 for ev_start in ev_start_set:
   G.add_edge("start", ev_start)

 for event, succesors in w_net.items():
  value = avgTime[event]

  if float(color_min-color_max) == 0:
    color = int(50)
  else:
    color = int(float(color_min-value)/float(color_min-color_max)*100.00)

  my_color = "#ff9933"+str(hex(color))[2:]
  G.add_node(event, style="rounded,filled", fillcolor=my_color, label=f'{event} - {value
}')
  for succesor, cnt in succesors.items():
    if float(trace_max-trace_min) == 0:
      G.add_edge(event, succesor, penwidth=4*cnt/10000, label=cnt)
    else:
      G.add_edge(event, succesor, penwidth=4*cnt/(trace_max-trace_min)+0.1, label=cnt)
 G.add_node("end", shape="circle", label="", penwidth='3')
 for ev_end in ev_end_set:
   G.add_edge(ev_end, "end")

 G.draw('simple_heuristic_net_with_events.png', prog='dot')
 display(Image('simple_heuristic_net_with_events.png'))
```

In [11]:

```
print('productivity coloring; threshold = 700;420 respectively')
Image(url="time_700_420.png")
```

```
productivity coloring; threshold = 700;420 respectively
```

Out[11]:

1. **Na podstawie obserwacji otrzymanego modelu, jakie wnioski można wyciągnąć z odkrytego procesu naprawy telefonów? Pomyśl o kilku wnioskach wynikających z obserwacji modelu i dotyczących odkrytego procesu, najlepiej takich, które nie są oczywiste, czy ogólne (dotyczące dowolnych modeli procesów).**

Wnioski:

Na podstawie generowanych modeli warto jest zauważyć, iż filtrowanie pozwala na dokładną obserwację wystepujących zjawisk z pominięciem kolejnych przepływów lub zdarzeń. Wraz ze zmniejszeniem ilości zdarzeń zmienia się ilość i wygląd przepływów, pozwala to na analize potencjalnych sytuacji. Dla różnych modeli możemy zaobserwować, iż pomimo wprowadzanych modyfikacji każde ze zdarzeń posiada conajmniej jedno wejście oraz wyjście. Analizując poszczególne progi dla posiadanych danych możemy zauważyć, iż w badanym procesie znajdują się zdarzenia, które muszą wystąpić zawsze jeżeli oczekujemy modelu składającego się z więcej niż jednego zdarzenia, pomijając start i stop.

1. **(*nadobowiązkowe*) Istnieją biblioteki do animowania grafów np. `GraphvizAnim`. Przy użyciu tego typu biblioteki można pokusić się o wyświetlenie animacji, jak przebiegał proces wg logu (podświetlając odpowiednie elementy). Jak może wyglądać bardzo zaawansowana animacja takiego modelu można zobaczyć w komercyjnym narzędziu Disco.**

Zadania mogą być realizowane w parach, natomiast w sprawozdaniu należy podać wtedy imiona i nazwiska osób z pary oraz każda osoba powinna wysłać zadanie/sprawozdanie przez system MS Teams osobno.

W sprawozdaniu należy umieścić odpowiednie krótkie opisy z realizacji poszczególnych punktów, zrzuty ekranu oraz kod źródłowy (istotne fragmenty kodu ze zmianami).

Sprawozdanie z wykonanych ćwiczeń należy przesłać przez platformę MS Teams do 24.03.2022 w postaci:

- **pdf z raportem o wykonaniu ćwiczeń,**
- **ipynb z wykonanymi ćwiczeniami (pdf oraz ipynb).**