Thesis Meeting 9

kense, for the thesis

Outline - Timeline

- Are we on track?
    - Feb 12, Exams End
    - Feb 22, Next Semester
- Projected Dates?
    - Dec?
    - Jan, Paper writing

November

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5
6	7	8	9	10	11	12

December

S	M	T	W	T	F	S
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

January

S	M	T	W	T	F	S
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

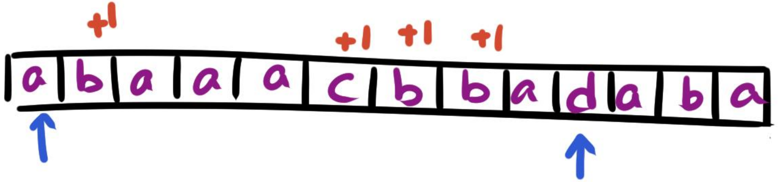
February

S	M	T	W	T	F	S
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	1	2	3	4	5	6
7	8	9	10	11	12	13

Refreshers

- Until-N function ($\boldsymbol{\varphi}\ \boldsymbol{U}^{\leq N}\ \boldsymbol{\omega}$), a modified LTL expression.
- We utilize to find sub-conversations with variables ***c*** and ***l***, which are the number of N violations, and the length of the sequence found.
- Finding good values, and tie-ins to similarity measure from before.
- Side-note: Full Abstraction.

$$\{a\} \cup^{\leq N} \{d\}$$



s : 0

e : 9

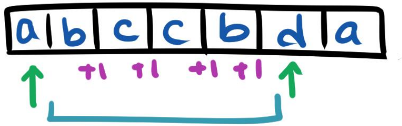
c : 4

l : 10

LTL-*N* Sequence Discovery

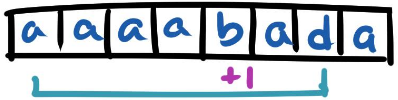
- Starting with a *one -> one* label set, omitting labels such as *x, misc* (leaving nine total labels).
- Looking for properties:
  - 10 < length < 50
  - c < (length*0.5)
- Generates a preliminary list for further consideration
  - Length 30?
  - c < 20-40%?

$\{a\}\ U^{\leq N}\ \{d\} \qquad N = 4$



$c = 4$
$l = 6$

Violation count too high
66% of length



$c = 1$
$l = 7$

Violation count good
14% of length

LTL-*N* Sequence Discovery

- The *one -> many* label set will generate a lot of redundant entries, most likely dismissable as a method to find key values.
- Pick reasonable average length ***l*** for discovered sequences
- Pick lower percentage of ***c*** value compared to length ***l***
- Can it improve clustering?

Sub-Conversation Weighting

- Frequency based analysis, using the sequences of sub-conversations.
- Similar to “bag of activities” approach, but bypass the drawbacks (order doesn’t matter, context doesn’t matter).
- Use weighting on sub-conversations to create a metric, that we can use to modify levenshtein distance (adds on to similarity measure).

Sub-Conversation Weighting

For a given trace $\boldsymbol{T_i}$ from the event log $\boldsymbol{E}$, where $\boldsymbol{i}$ is the number of traces:

$\boldsymbol{T_i}$ is a sequence of labels $\boldsymbol{a, \ldots, x}$ where $\boldsymbol{a,x \in \ell}$

Where each label $\boldsymbol{x}$ can be assigned a local weight $\boldsymbol{w_{id}}$ where $\boldsymbol{id}$ is the position of the label $\boldsymbol{x}$ in the sequence $\boldsymbol{T_i}$.

Each label $\boldsymbol{x}$ can be part of a *sub-conversation* **Sc**, which is a sub-sequence of labels in $\boldsymbol{T_i}$.

Sub-Conversation Weighting

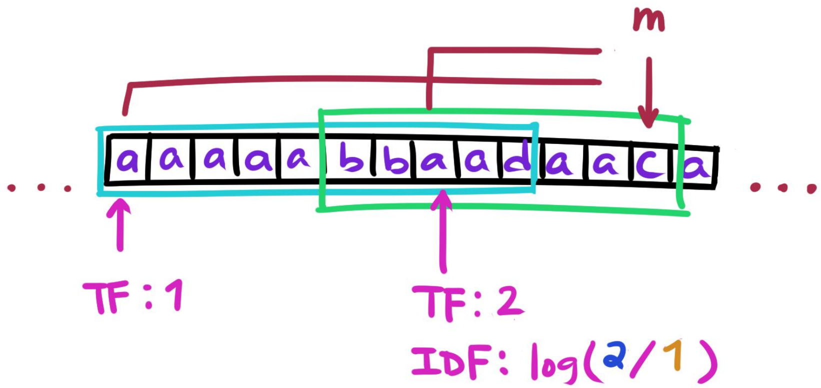
Each label ***a*** in $\boldsymbol{T_i}$ is assigned a Total Weight $\boldsymbol{Tw_{id}}$, where ***id*** is the position of the label in the sequence $\boldsymbol{T_i}$.

$$Tw_{id} = w_{id} + TF\text{-}IDF(a_{id})$$

TF-IDF is a variant based on the normal TF-IDF calculation such that:

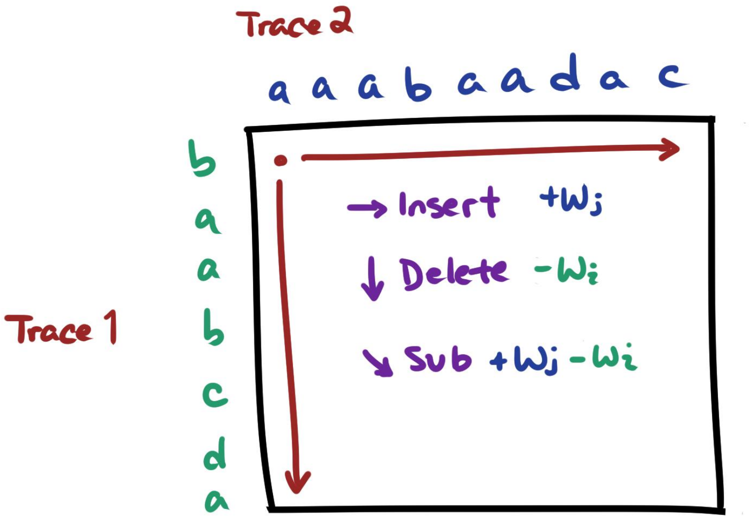
TF corresponds to the number of sub-conversations in $\boldsymbol{T_i}$ that $\boldsymbol{a_{id}}$ belongs to, with the condition that $\boldsymbol{a \in \varphi}$ ***or*** $\boldsymbol{a \in \omega}$.

IDF = log(# of sub-convos/d), where ***d*** corresponds to the number of sub-conversations in $\boldsymbol{T_i}$ that $\boldsymbol{a_{id}}$ belongs to, where $\boldsymbol{a \notin \varphi}$ ***and*** $\boldsymbol{a \notin \omega}$.


m
a a a a a b b a a d a a c a
TF: 1
TF: 2
IDF: log(2/1)

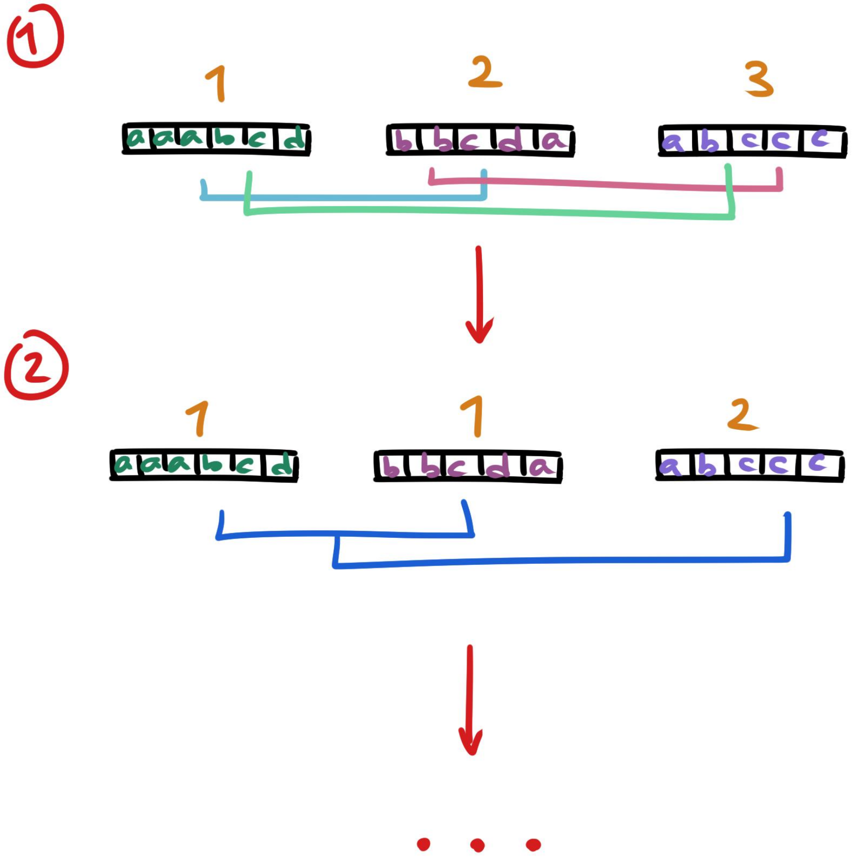
Sub-Conversation Weighting

- Applied as a variant of Levenshtein distance, we can modify the costs based on the weight $\boldsymbol{Tw_{id}}$ of each label $\boldsymbol{a_{id}}$.
- Insertion/Deletion cost is the weight of the added label, $\boldsymbol{Tw_{id+1}}$
- Substitution cost is the weight of both adding and removing, $\boldsymbol{Tw_{id+1} - Tw_{id}}$
- This results in another variant of the Levenshtein distance that focuses exclusively on weights with contributing factors from position in the conversation and contribution to sub-conversations.


Trace 2
a a a b a a d a c
b a a b c d a
Trace 1
→ Insert +Wj
↓ Delete -Wi
↘ Sub +Wj -Wi

Clustering - Big Picture

- The context-trace clustering works as follows:
    a. Given an event log ***K*** of ***M*** traces, and a clustering algorithm.
    b. Cluster the ***M*** traces into ***N*** clusters.
    c. Discover a process model for each cluster.
    d. Evaluate the quality of the process model.
- For our uses, we have less interest (currently) on discovering a process model.
- More focused on how similarity measure/weighting can be used to produce “better” clusters.
- In this case, “better” clusters may require further consideration.


1
1
2
3
aaabcd
bbcda
abccc
2
1
1
2
aaabcd
bbcda
abccc
. . .

Clustering - Weighting + Similarity

- Implement Clustering algorithm
  - with similarity numbers alone (Case A)
  - with weighting alone (Case B)
  - with similarity and weighting (Case C)
- Compare Clustering results with expected results (i.e: traces from the same show get into the same cluster, etc)

Clustering - Results and Discussion

- Case A and Case B clustering both result in expected preliminary outputs.
  - Case A has more distance amongst comparisons, probably due to algorithm
  - Case B has closer distances, weights are more or less uniform
- Case C results are also expected, values without normalizing.
- Since we're only running on three trace event logs, the results are to be expected.
- Move onto automatic labeling for larger event log clustering?