Statistical Learning for Process Data

Session 4: Problem-Solving Strategies and Subtask Analysis

Xueying Tang

Department of Mathematics
University of Arizona

June 4, 2021

Process Data Research

- Existing problems:
  - assessment
  - differential item functioning
  - computerized adaptive testing
  - ...
- New problems:
  - problem-solving strategy analysis
  - ...

Process Data Research

- Existing problems:
  - assessment
  - differential item functioning
  - computerized adaptive testing
  - ...
- New problems:
  - problem-solving strategy analysis
  - ...

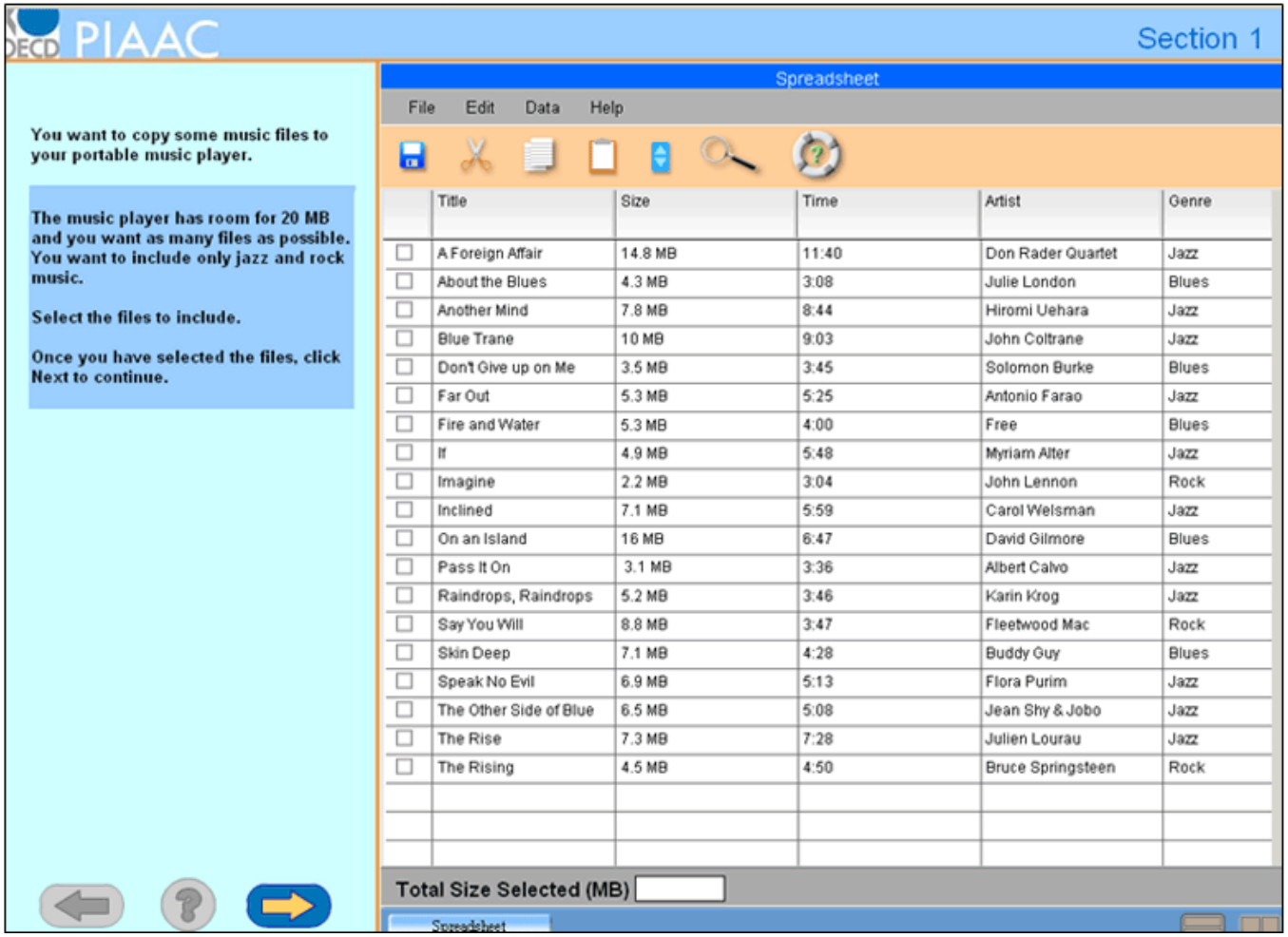
Process Data Research

- Existing problems:
  - assessment
  - differential item functioning
  - computerized adaptive testing
  - ...
- New problems:
  - problem-solving strategy analysis
  - ...

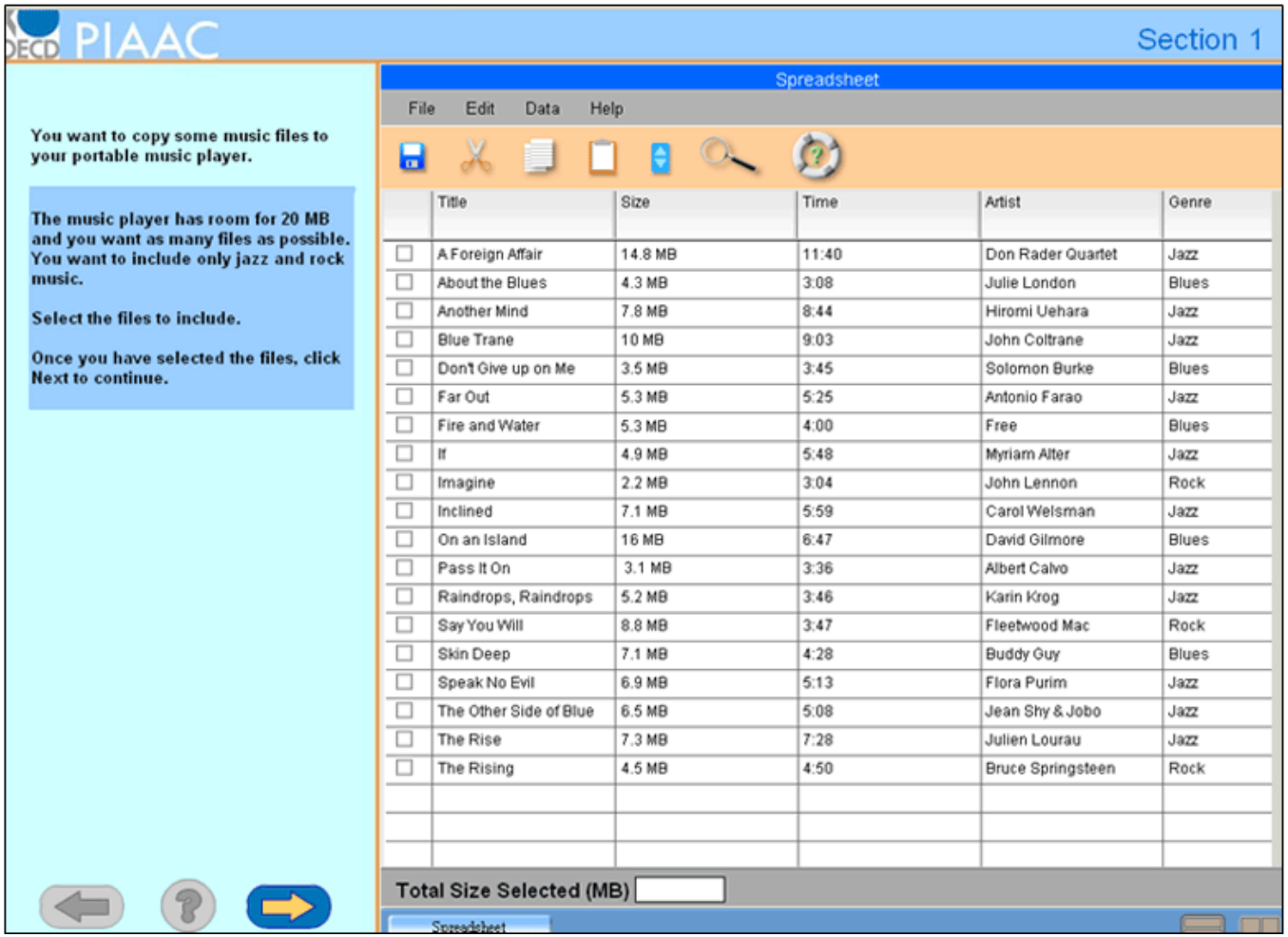
Agenda

- Problem-solving strategy and subtask
- Subtask identification procedure
- Empirical results
- Implementation

Example


PIAAC
Section 1
You want to copy some music files to your portable music player.
The music player has room for 20 MB and you want as many files as possible. You want to include only jazz and rock music.
Select the files to include.
Once you have selected the files, click Next to continue.
Spreadsheet
File Edit Data Help
Title Size Time Artist Genre
A Foreign Affair 14.8 MB 11:40 Don Rader Quartet Jazz
About the Blues 4.3 MB 3:08 Julie London Blues
Another Mind 7.8 MB 8:44 Hiromi Uehara Jazz
Blue Trane 10 MB 9:03 John Coltrane Jazz
Don't Give up on Me 3.5 MB 3:45 Solomon Burke Blues
Far Out 5.3 MB 5:25 Antonio Farao Jazz
Fire and Water 5.3 MB 4:00 Free Blues
If 4.9 MB 5:48 Myriam Alter Jazz
Imagine 2.2 MB 3:04 John Lennon Rock
Inclined 7.1 MB 5:59 Carol Welsman Jazz
On an Island 16 MB 6:47 David Gilmore Blues
Pass It On 3.1 MB 3:36 Albert Calvo Jazz
Raindrops, Raindrops 5.2 MB 3:46 Karin Krog Jazz
Say You Will 8.8 MB 3:47 Fleetwood Mac Rock
Skin Deep 7.1 MB 4:28 Buddy Guy Blues
Speak No Evil 6.9 MB 5:13 Flora Purim Jazz
The Other Side of Blue 6.5 MB 5:08 Jean Shy & Jobo Jazz
The Rise 7.3 MB 7:28 Julien Lourau Jazz
The Rising 4.5 MB 4:50 Bruce Springsteen Rock
Total Size Selected (MB)

Example



Start
Toolbar_Sort
Sort_A_2
Sort_OK
Menu_Edit
Menu_Help
Menu_Data
Toolbar_Help
Toolbar_Save
Tick_9
Tick_12
Tick_19
Tick_13
Tick_14
Next

How to Identify Problem-Solving Strategies?

Start, Toolbar_Sort, Sort_A_2, Sort_OK, Menu_Edit, Menu_Help, Menu_Data, Toolbar_Help, Toolbar_Save, Tick_9, Tick_12, Tick_19, Tick_13, Tick_14, Next

How to Identify Problem-Solving Strategies?

Start, Toolbar_Sort, Sort_A_2, Sort_OK, Menu_Edit, Menu_Help, Menu_Data, Toolbar_Help, Toolbar_Save, Tick_9, Tick_12, Tick_19, Tick_13, Tick_14, Next

- Examining the processes one by one?

How to Identify Problem-Solving Strategies?

Start, Toolbar_Sort, Sort_A_2, Sort_OK, Menu_Edit, Menu_Help, Menu_Data, Toolbar_Help, Toolbar_Save, Tick_9, Tick_12, Tick_19, Tick_13, Tick_14, Next

- Examining the processes one by one?
  - Time-consuming

How to Identify Problem-Solving Strategies?

Start, Toolbar_Sort, Sort_A_2, Sort_OK, Menu_Edit, Menu_Help, Menu_Data, Toolbar_Help, Toolbar_Save, Tick_9, Tick_12, Tick_19, Tick_13, Tick_14, Next

- Examining the processes one by one?
  - Time-consuming
- Checking whether the key action appear or not?

How to Identify Problem-Solving Strategies?

Start, Toolbar_Sort, Sort_A_2, Sort_OK, Menu_Edit, Menu_Help, Menu_Data, Toolbar_Help, Toolbar_Save, Tick_9, Tick_12, Tick_19, Tick_13, Tick_14, Next

- Examining the processes one by one?
  - Time-consuming
- Checking whether the key action appear or not?
  - Most strategies cannot be described by a single action.

How to Identify Problem-Solving Strategies?

Start, Toolbar_Sort, Sort_A_2, Sort_OK, Menu_Edit, Menu_Help, Menu_Data, Toolbar_Help, Toolbar_Save, Tick_9, Tick_12, Tick_19, Tick_13, Tick_14, Next

- Examining the processes one by one?
  - Time-consuming
- Checking whether the key action appear or not?
  - Most strategies cannot be described by a single action.
- Checking whether certain action patterns appear or not?

How to Identify Problem-Solving Strategies?

Start, Toolbar_Sort, Sort_A_2, Sort_OK, Menu_Edit, Menu_Help, Menu_Data, Toolbar_Help, Toolbar_Save, Tick_9, Tick_12, Tick_19, Tick_13, Tick_14, Next

- Examining the processes one by one?
  - Time-consuming
- Checking whether the key action appear or not?
  - Most strategies cannot be described by a single action.
- Checking whether certain action patterns appear or not?
  - A strategy could be indicated by a variety of action patterns.

How to Identify Problem-Solving Strategies?

Start, Toolbar_Sort, Sort_A_2, Sort_OK, Menu_Edit, Menu_Help, Menu_Data, Toolbar_Help, Toolbar_Save, Tick_9, Tick_12, Tick_19, Tick_13, Tick_14, Next

- Examining the processes one by one?
  - Time-consuming
- Checking whether the key action appear or not?
  - Most strategies cannot be described by a single action.
- Checking whether certain action patterns appear or not?
  - A strategy could be indicated by a variety of action patterns.

Problem-Solving Strategy and Subtasks

Start, *Toolbar_Sort*, *Sort_A_2*, *Sort_OK*, *Menu_Edit*, *Menu_Help*, *Menu_Data*, *Toolbar_Help*, *Toolbar_Save*, *Tick_9*, *Tick_12*, *Tick_19*, *Tick_13*, *Tick_14*, *Next*

- A complex task can often be decomposed into subtasks.

Problem-Solving Strategy and Subtasks

Start, *Toolbar_Sort*, *Sort_A_2*, *Sort_OK*, *Menu_Edit*, *Menu_Help*, *Menu_Data*, *Toolbar_Help*, *Toolbar_Save*, *Tick_9*, *Tick_12*, *Tick_19*, *Tick_13*, *Tick_14*, *Next*

- A complex task can often be decomposed into subtasks.

Problem-Solving Strategy and Subtasks

Start, *Toolbar_Sort*, *Sort_A_2*, *Sort_OK*, *Menu_Edit*, *Menu_Help*, *Menu_Data*, *Toolbar_Help*, *Toolbar_Save*, *Tick_9*, *Tick_12*, *Tick_19*, *Tick_13*, *Tick_14*, *Next*

SORT, *EXPLORE*, *ANSWER*

- A complex task can often be decomposed into subtasks.
- A response process has a corresponding subtask process.

Problem-Solving Strategy and Subtasks

Start, *Toolbar_Sort*, *Sort_A_2*, *Sort_OK*, *Menu_Edit*, *Menu_Help*, *Menu_Data*, *Toolbar_Help*, *Toolbar_Save*, *Tick_9*, *Tick_12*, *Tick_19*, *Tick_13*, *Tick_14*, *Next*

SORT, *EXPLORE*, *ANSWER*

- A complex task can often be decomposed into subtasks.
- A response process has a corresponding subtask process.
- The subtasks used and their order reflect respondents' problem-solving strategies.

Problem-Solving Strategy and Subtasks

Start, *Toolbar_Sort*, *Sort_A_2*, *Sort_OK*, *Menu_Edit*, *Menu_Help*, *Menu_Data*, *Toolbar_Help*, *Toolbar_Save*, *Tick_9*, *Tick_12*, *Tick_19*, *Tick_13*, *Tick_14*, *Next*

SORT, *EXPLORE*, *ANSWER*

- A complex task can often be decomposed into subtasks.
- A response process has a corresponding subtask process.
- The subtasks used and their order reflect respondents' problem-solving strategies.
- **Subtask Identification Procedure**: a data-driven method to identify subtasks in response processes.

Notation

- a generic response process $\mathbf{s} = (s_1, \ldots, s_L)$
- action set $\mathcal{A} = \{a_1, \ldots, a_M\}$ (all possible actions in an item)
- $s_t \in \mathcal{A}$, for $t = 1, \ldots, L$.

Main Idea

- Subprocesses performing the same subtask are often similar.

Main Idea

- Subprocesses performing the same subtask are often similar.
- Consider predicting s_{t+1} based on $s_1, \ldots, s_t$.

Main Idea

- Subprocesses performing the same subtask are often similar.
- Consider predicting s_{t+1} based on $s_1, \ldots, s_t$.
- In which case the prediction problem is easier?
  - s_t and s_{t+1} is in the middle of a subtask.
  - s_t and s_{t+1} belongs to different subtasks.

Main Idea

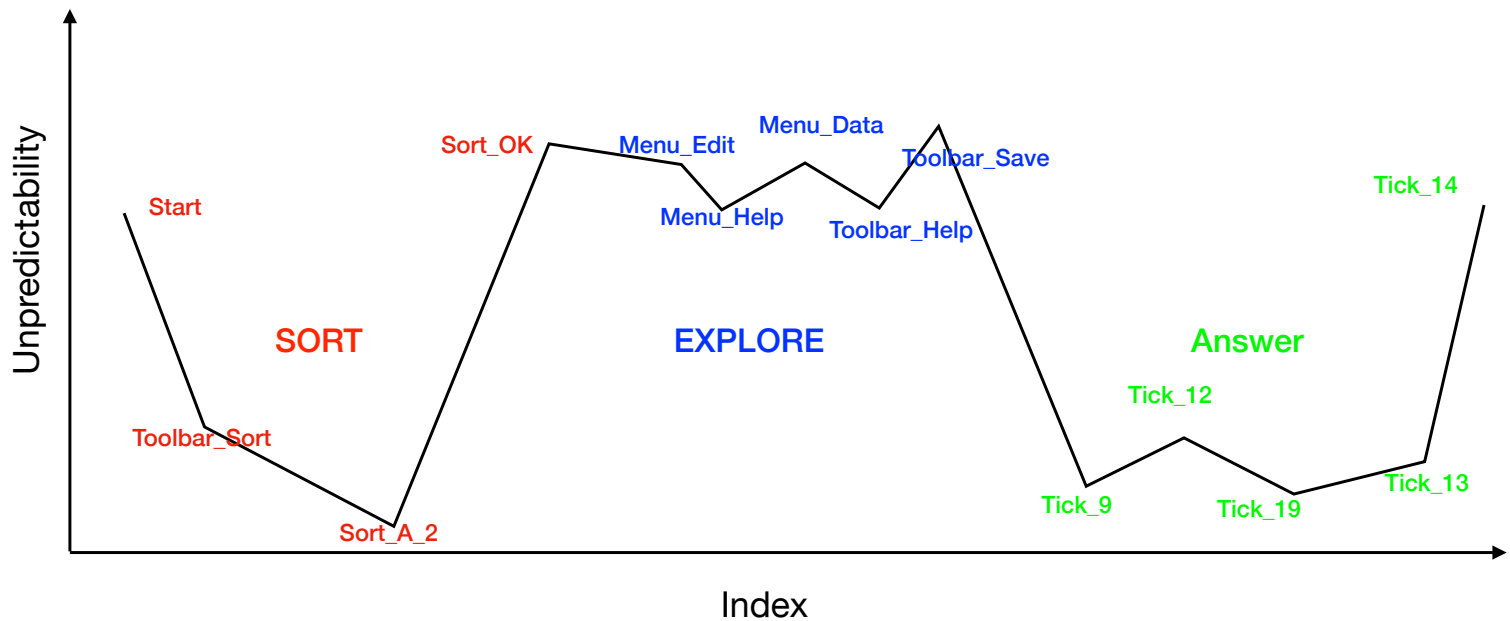
- Subprocesses performing the same subtask are often similar.
- Consider predicting s_{t+1} based on $s_1, \ldots, s_t$.
- In which case the prediction problem is easier?
  - s_t and s_{t+1} is in the middle of a subtask.
  - s_t and s_{t+1} belongs to different subtasks.

Main Idea

- Subprocesses performing the same subtask are often similar.
- Consider predicting s_{t+1} based on $s_1, \ldots, s_t$.
- In which case the prediction problem is easier?
  - s_t and s_{t+1} is in the middle of a subtask. more predictable
  - s_t and s_{t+1} belongs to different subtasks.

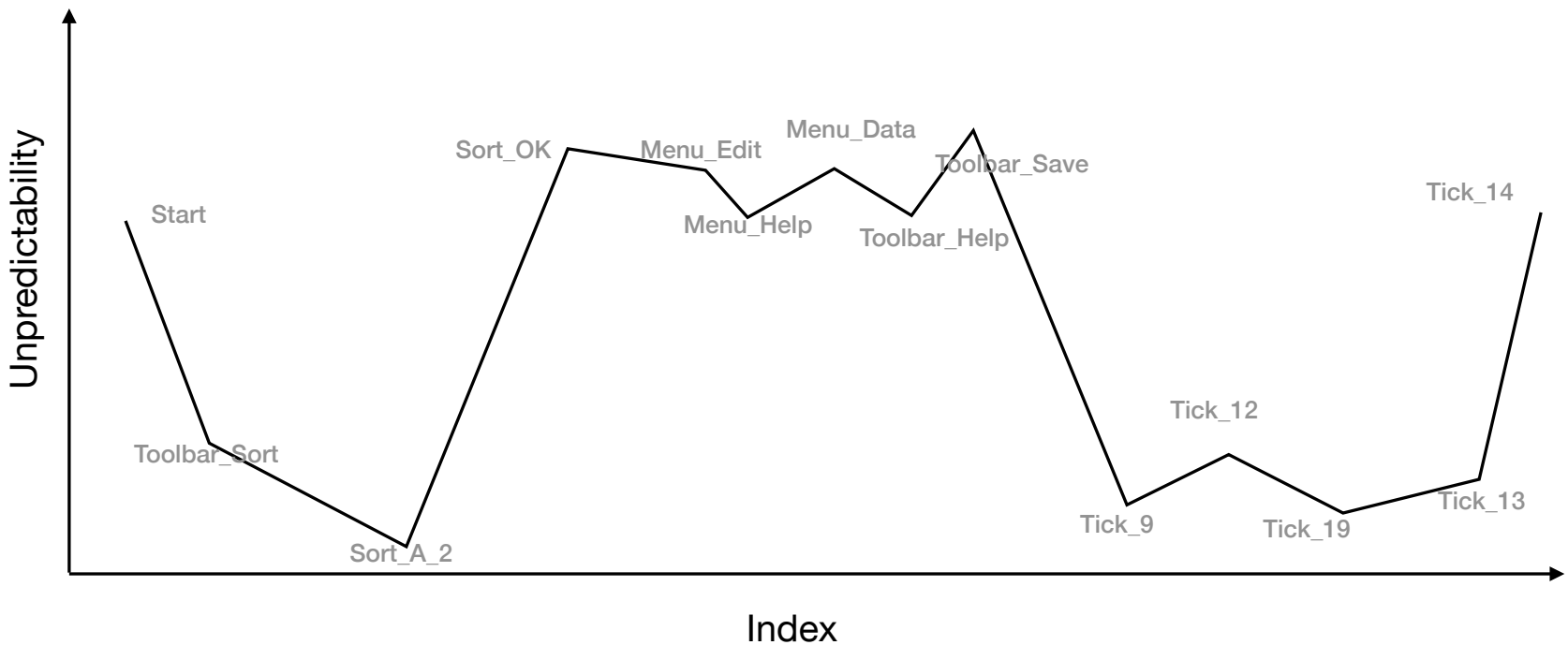
Main Idea

- Subprocesses performing the same subtask are often similar.
- Consider predicting s_{t+1} based on $s_1, \ldots, s_t$.
- In which case the prediction problem is easier?
  - s_t and s_{t+1} is in the middle of a subtask. more predictable
  - s_t and s_{t+1} belongs to different subtasks.
- A typical curve of unpredictability



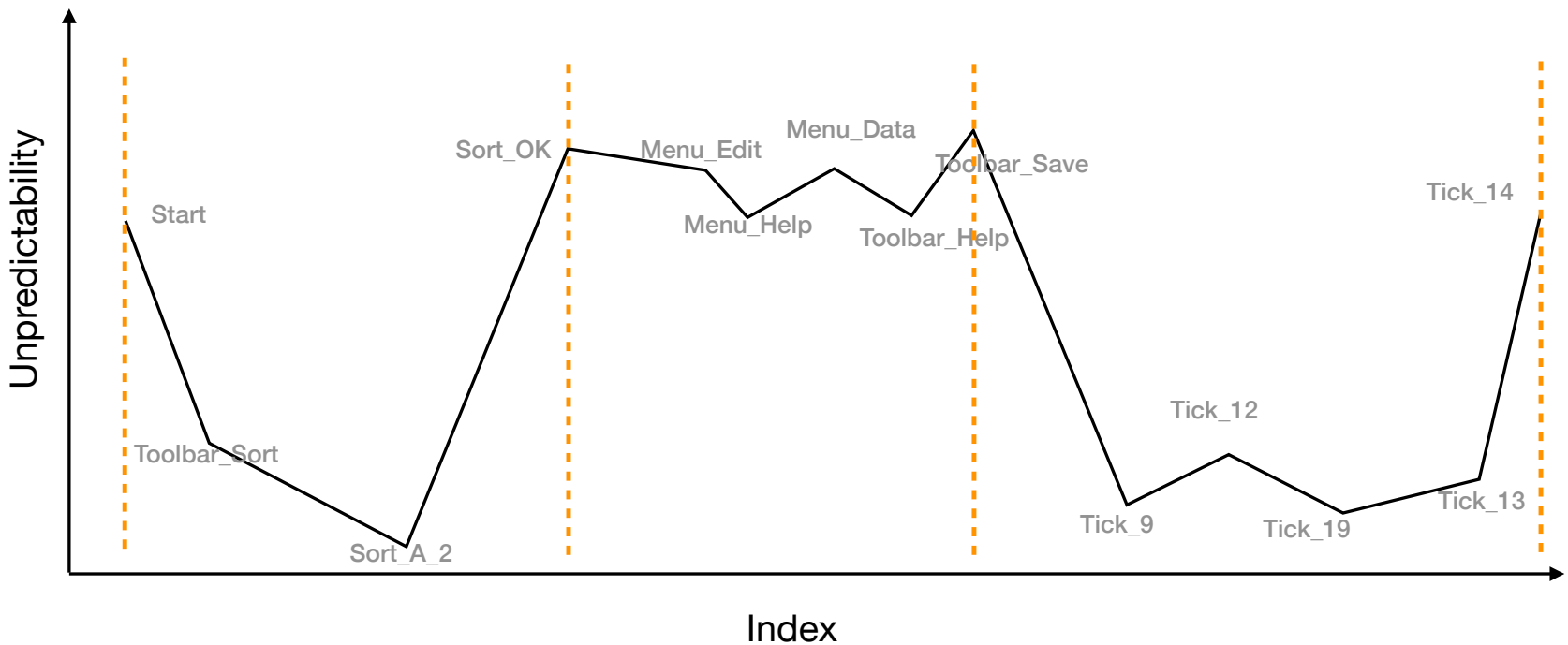
Subtask Identification Procedure

1. Prediction. Fit an action prediction model and quantify the unpredictability of the next action at each step.



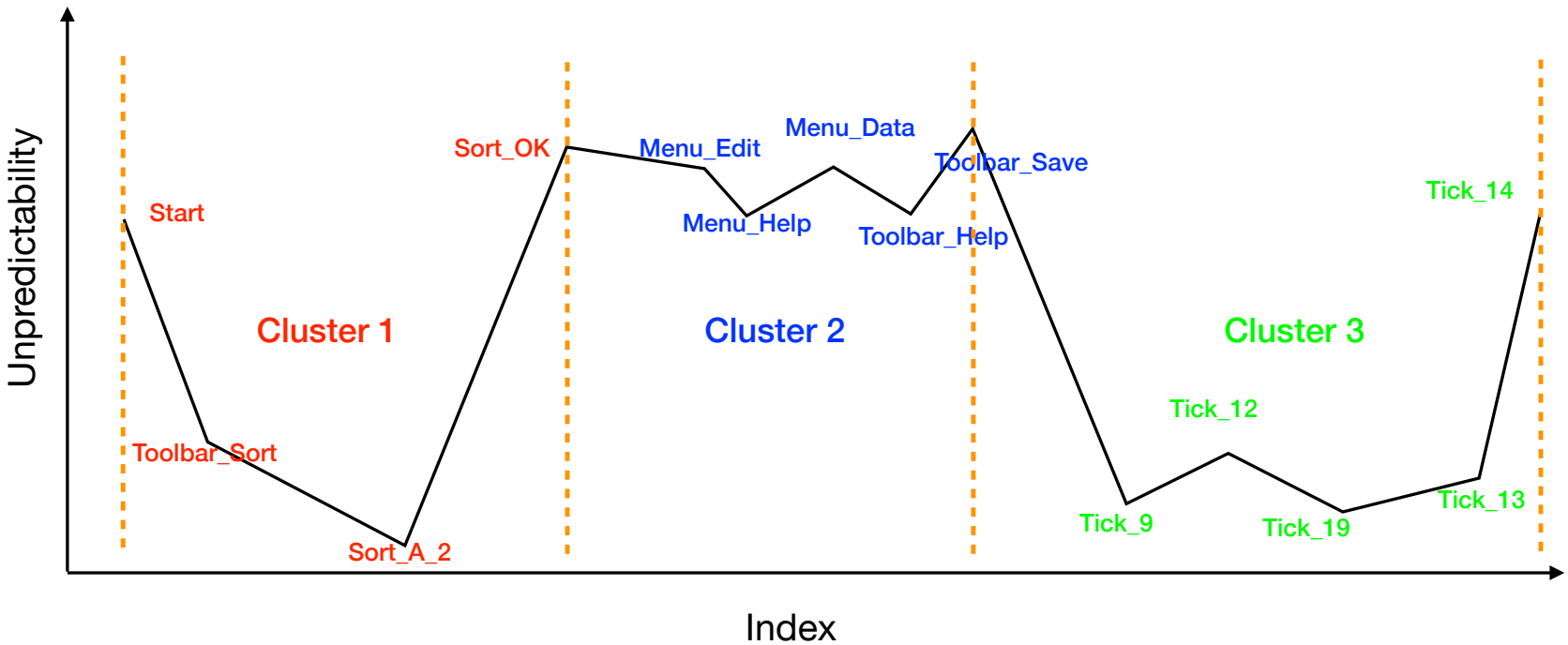
Subtask Identification Procedure

1. Prediction. Fit an action prediction model and quantify the unpredictability of the next action at each step.
2. Segmentation. Partition the response process into multiple subprocesses based on the unpredictability curve.



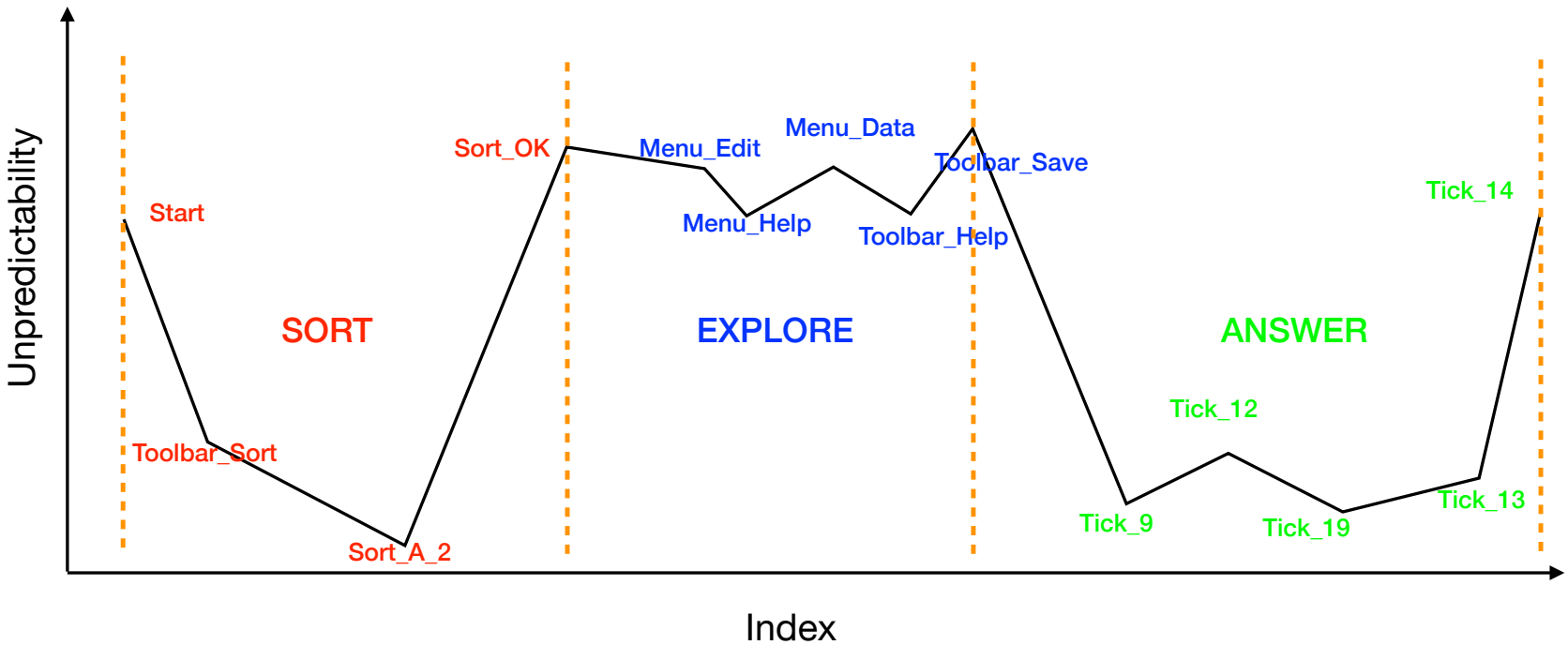
Subtask Identification Procedure

1. Prediction. Fit an action prediction model and quantify the unpredictability of the next action at each step.
2. Segmentation. Partition the response process into multiple subprocesses based on the unpredictability curve.
3. Clustering. Cluster the subprocesses according to their action frequency and label the clusters as subtasks.



Subtask Identification Procedure

1. Prediction. Fit an action prediction model and quantify the unpredictability of the next action at each step.
2. Segmentation. Partition the response process into multiple subprocesses based on the unpredictability curve.
3. Clustering. Cluster the subprocesses according to their action frequency and label the clusters as subtasks.

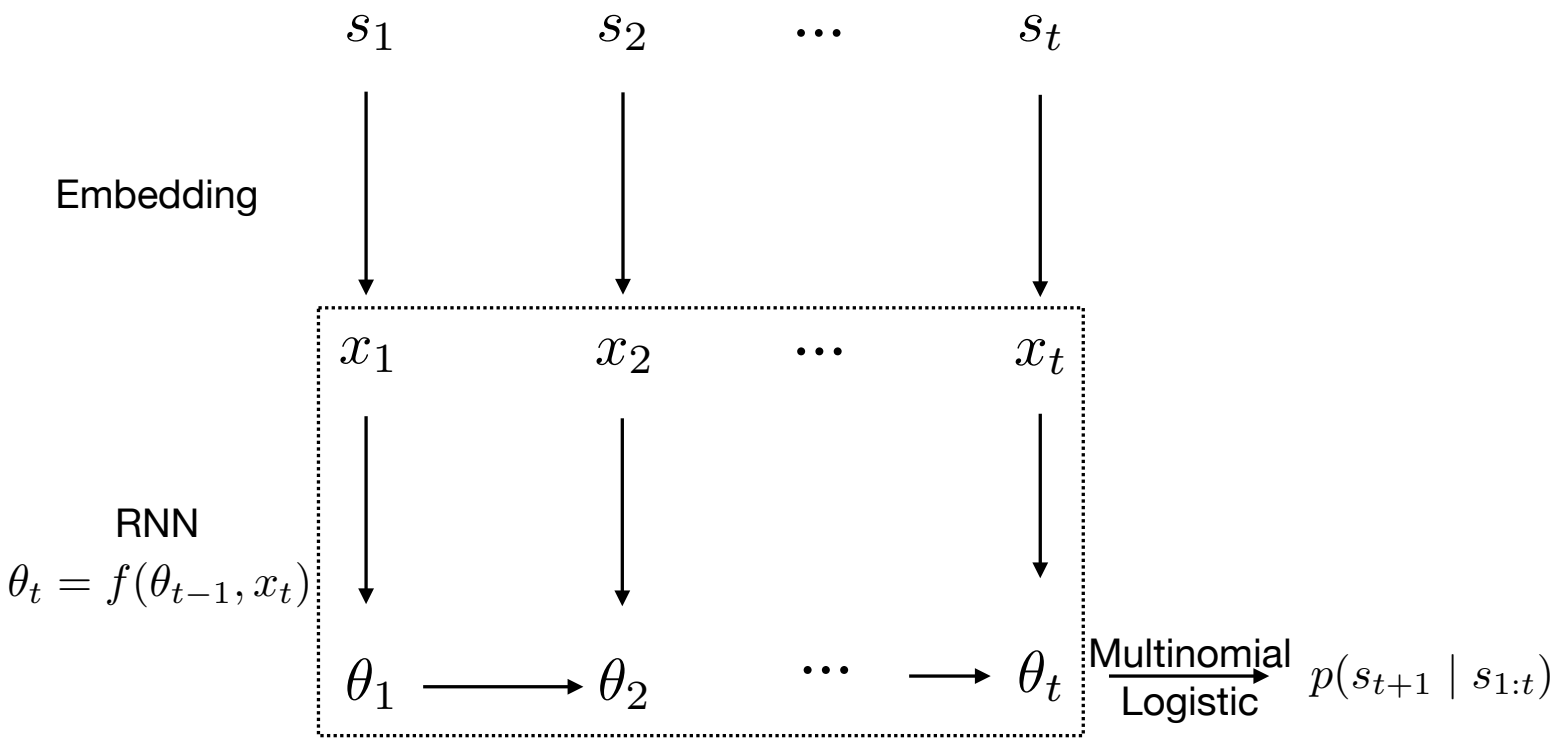


Step 1: Prediction – Action Prediction Model

$$P(\mathbf{s}) = \prod_{t=1}^{L-1} P(s_{t+1} \mid s_{1:t})$$

- We need to characterize $p(s_{t+1}|s_{1:t})$ for all possible $s_{1:t}$ and t.
- The model should
  - account for the long term dependence in response processes
  - be flexible to describe complex problem-solving behaviors.
- Three key components:
  - embedding: $a_j \in \mathcal{A} \leftrightarrow \mathbf{e}_j \in \mathbb{R}^K$
  - recurrent neural network
  - multinomial logistic model

Step 1: Prediction – Action Prediction Model



Step 1: Prediction – Entropy

- Unpredictability of the next action is characterized by the entropy of its predictive distribution:

$$h_t = -\sum_{j=1}^{M} p_{tj} \log(p_{tj}),$$

where $p_{tj} = p(s_{t+1} = a_j \mid s_{1:t})$.

  - s_{t+1} is completely unpredictable if $p(s_{t+1} \mid s_{1:t})$ is distributed evenly on $\mathcal{A}$.
  - s_{t+1} is completely predictable if $p(s_{t+1} \mid s_{1:t})$ is concentrated on a single element in $\mathcal{A}$.

- Response process $\mathbf{s} = (s_1, \ldots, s_L)$ is transformed into an entropy process $\mathbf{h} = (h_1, \ldots, h_{L-1})$.

Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).

Step 2: Segmentation

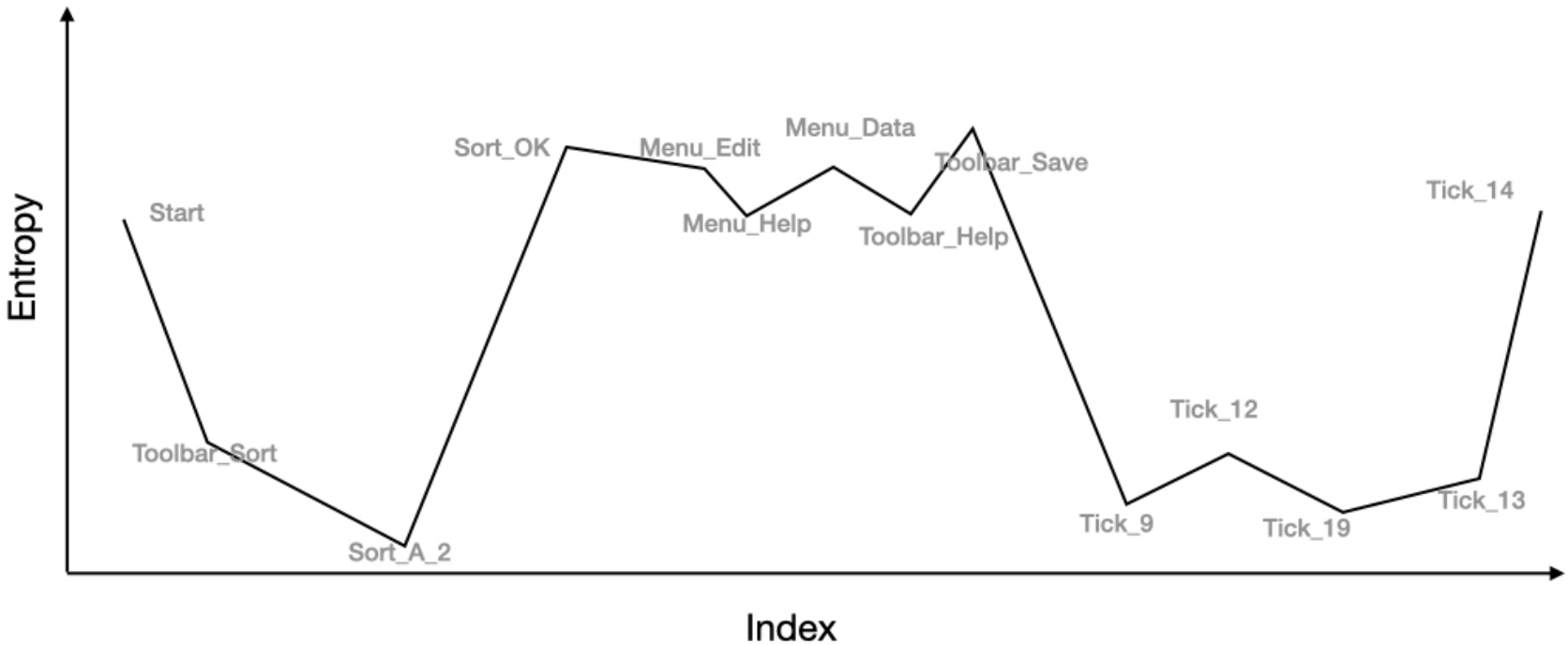
- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$

Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

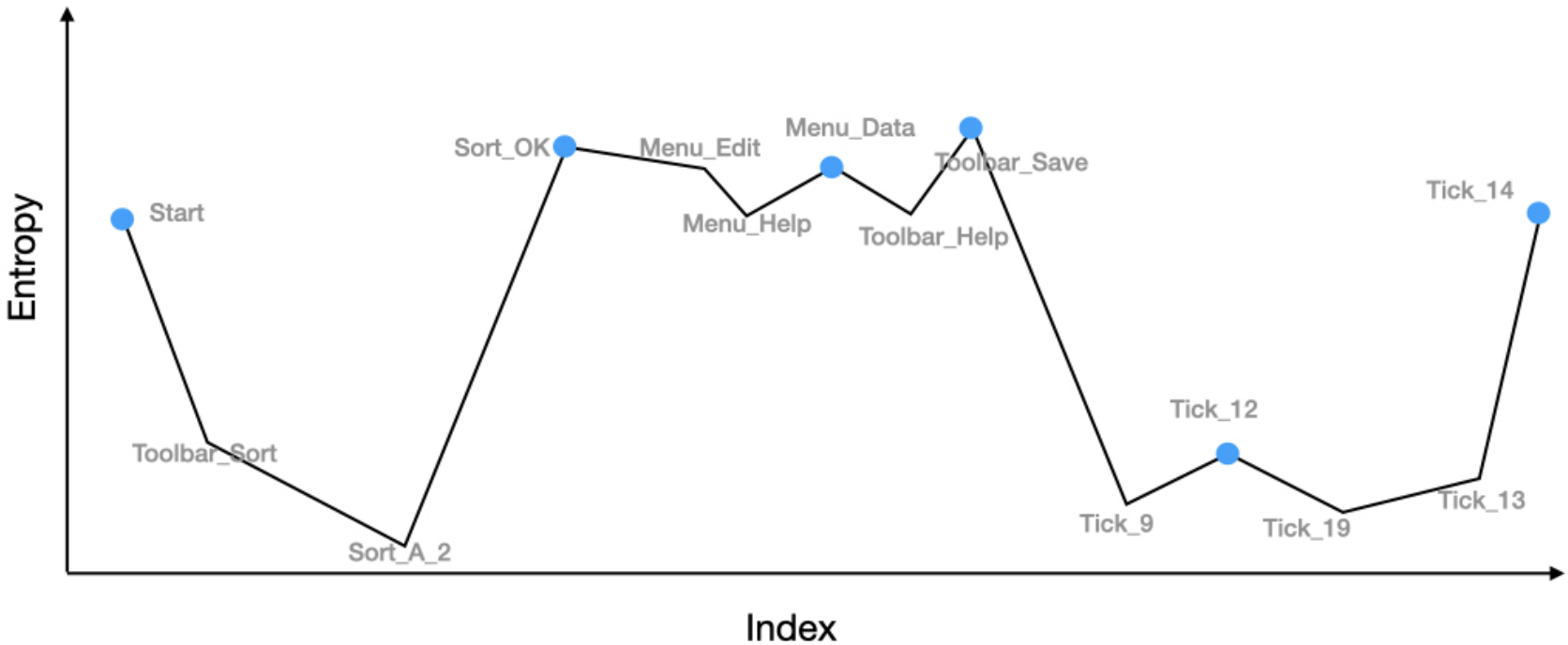
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

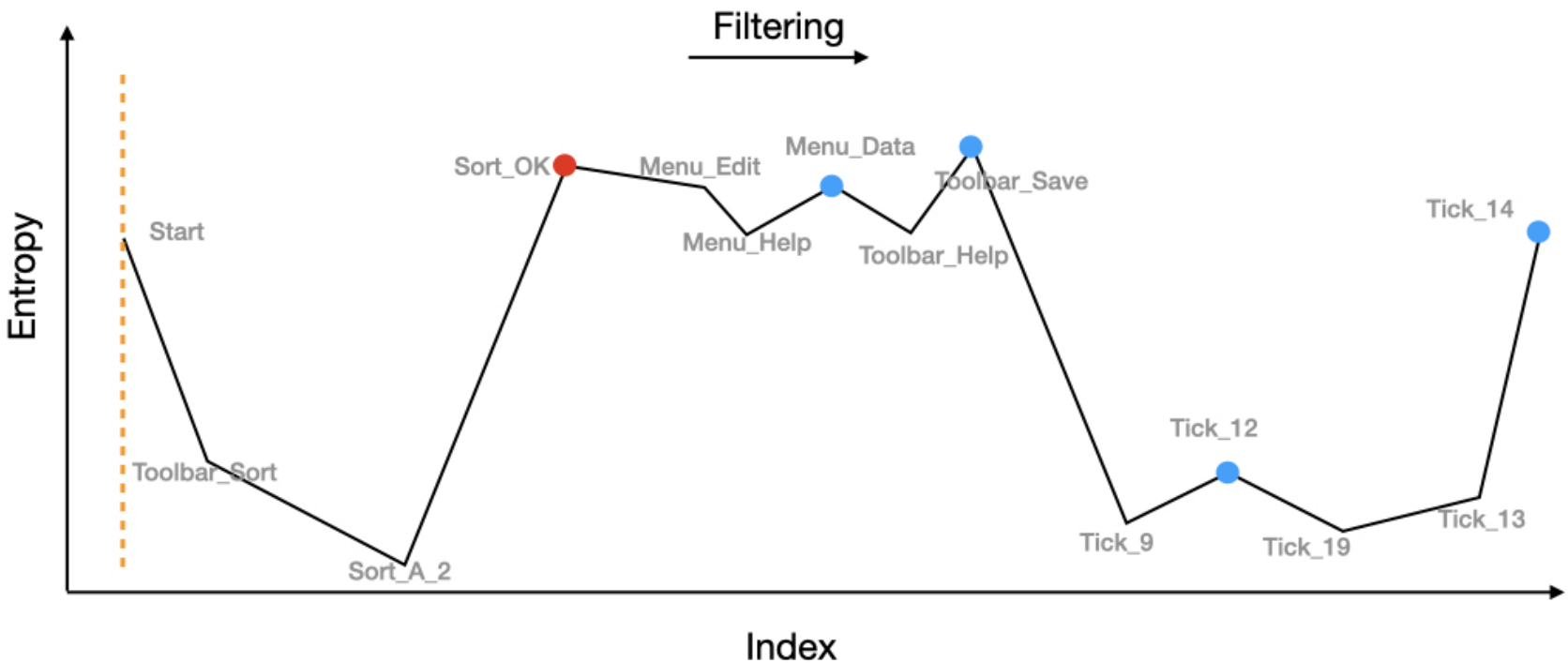
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

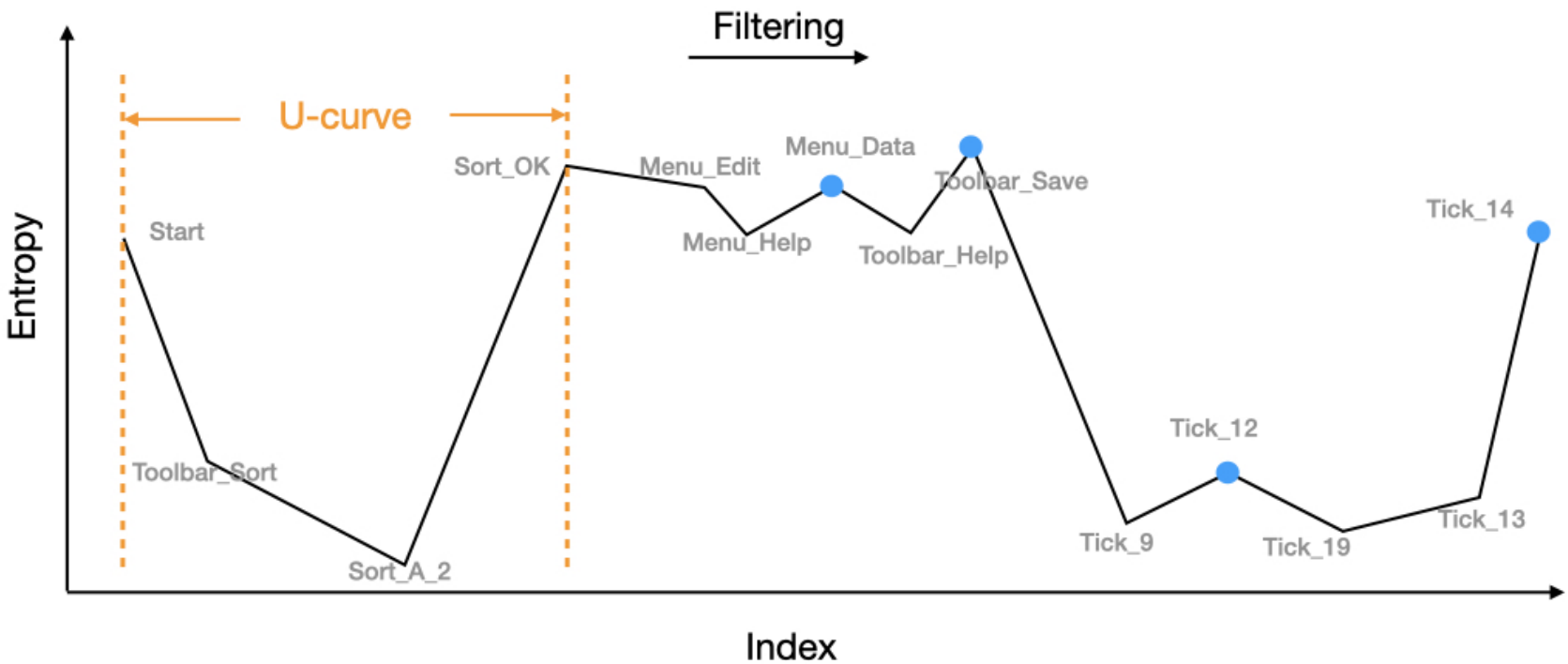
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

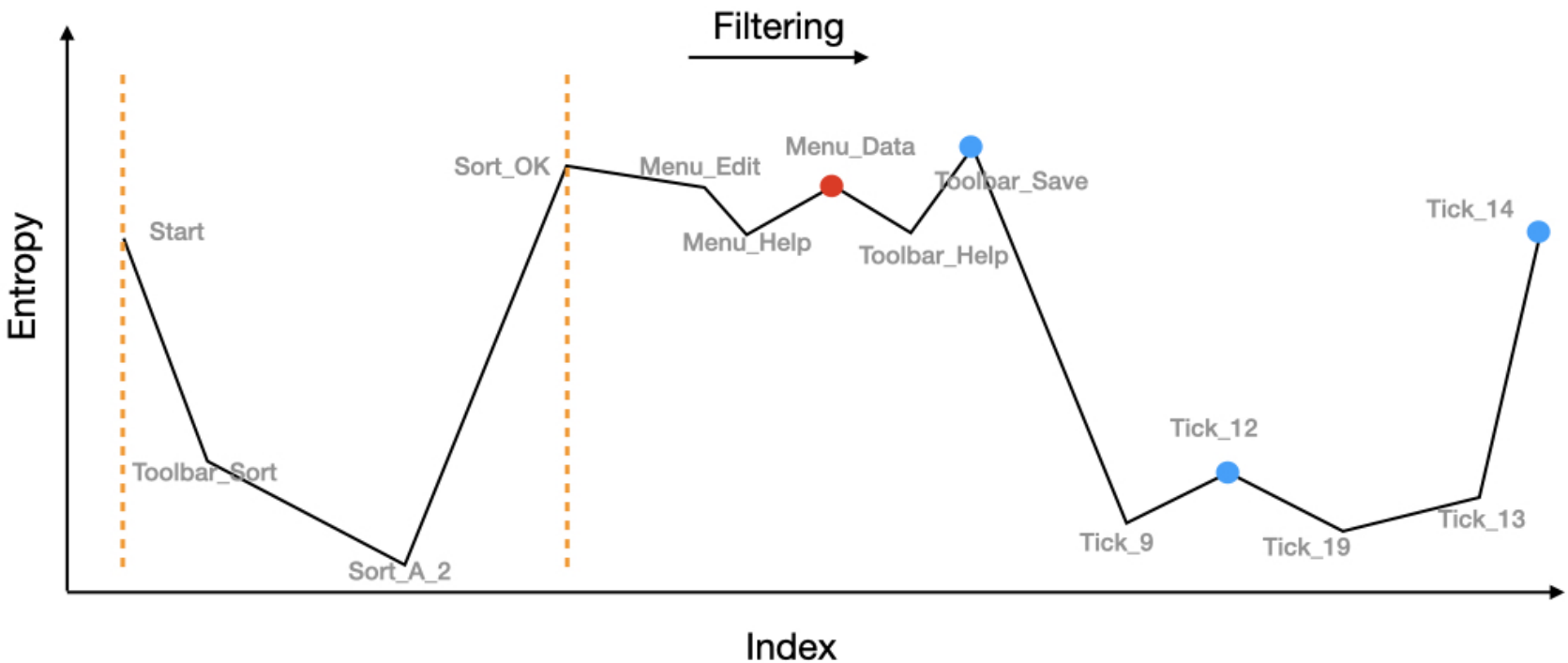
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

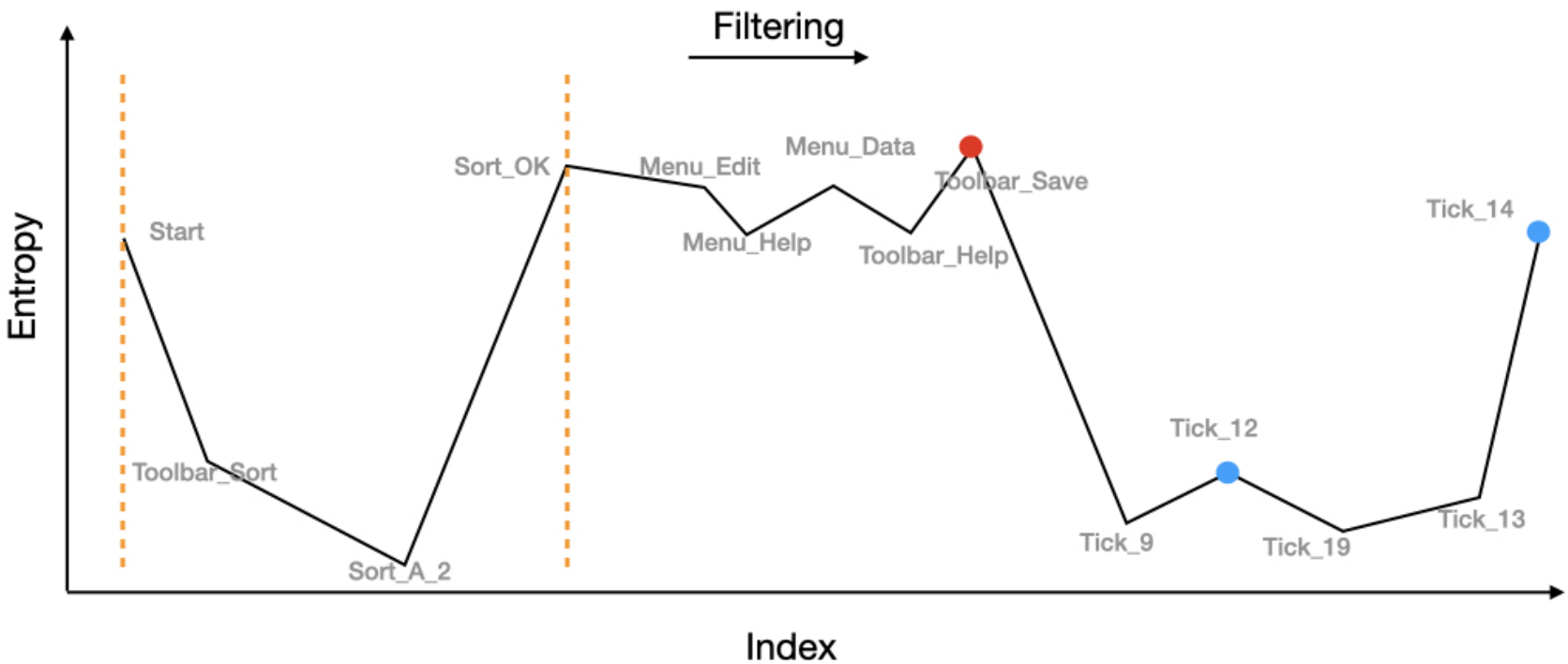
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

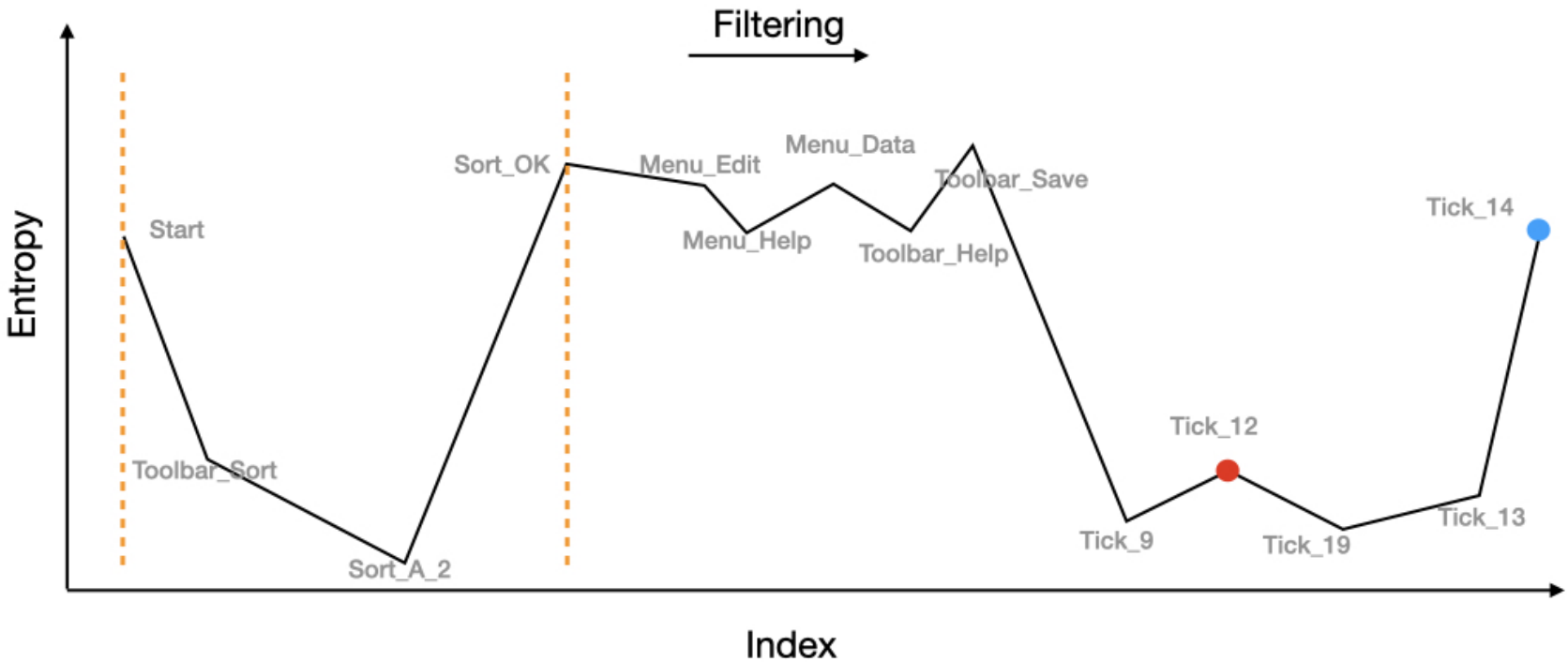
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

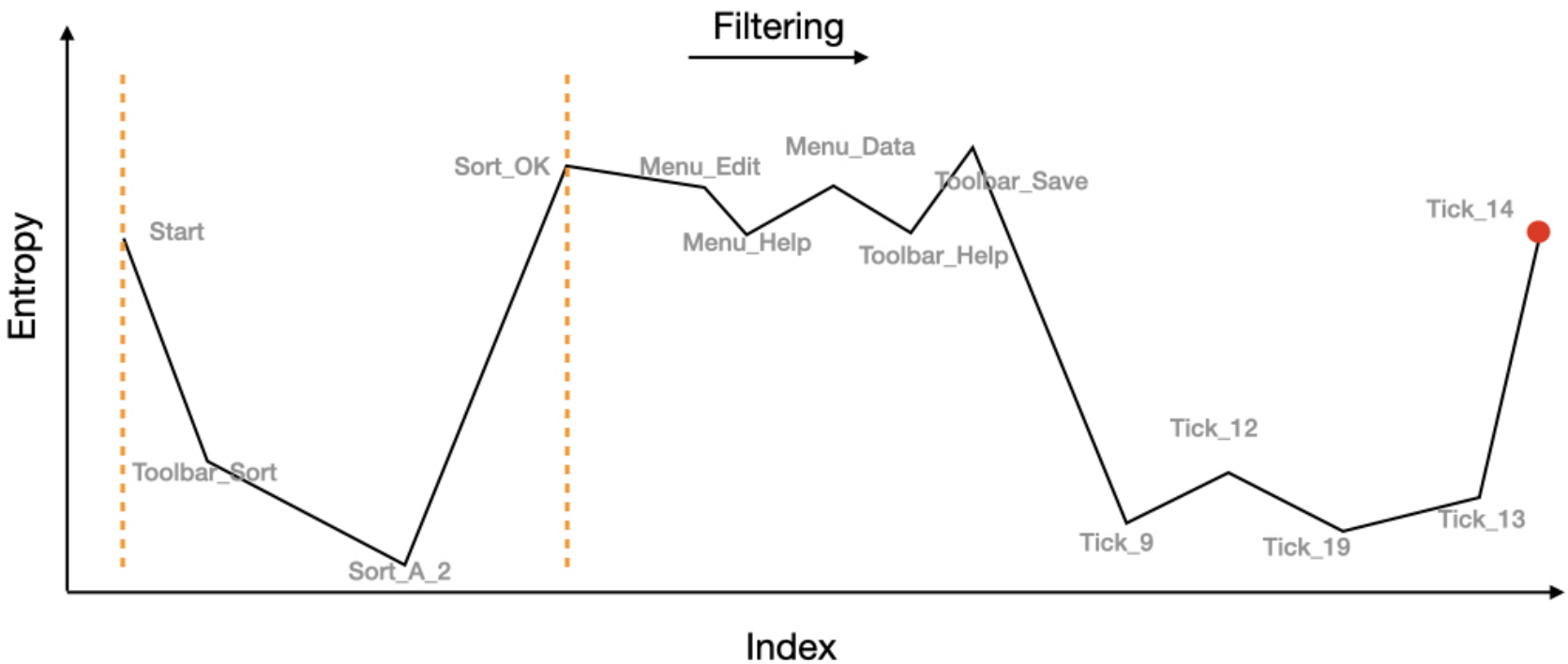
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

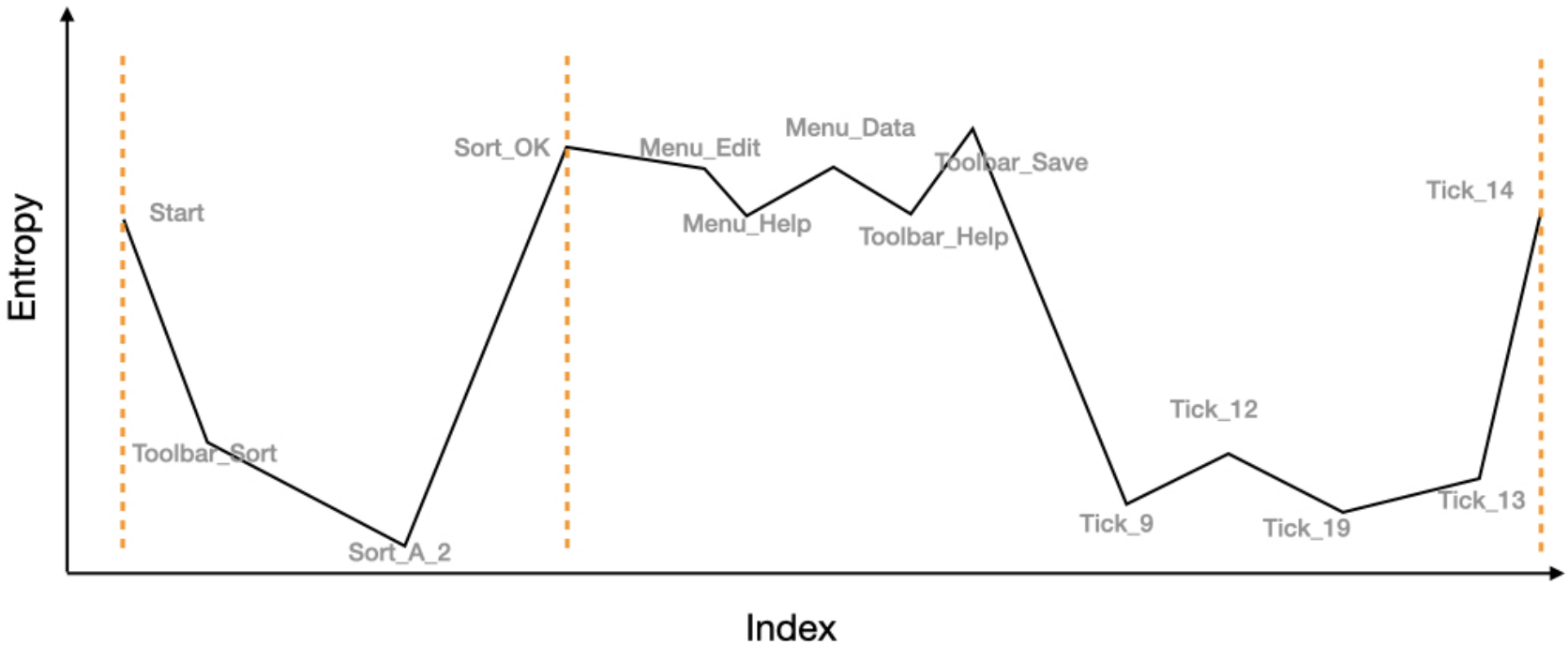
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

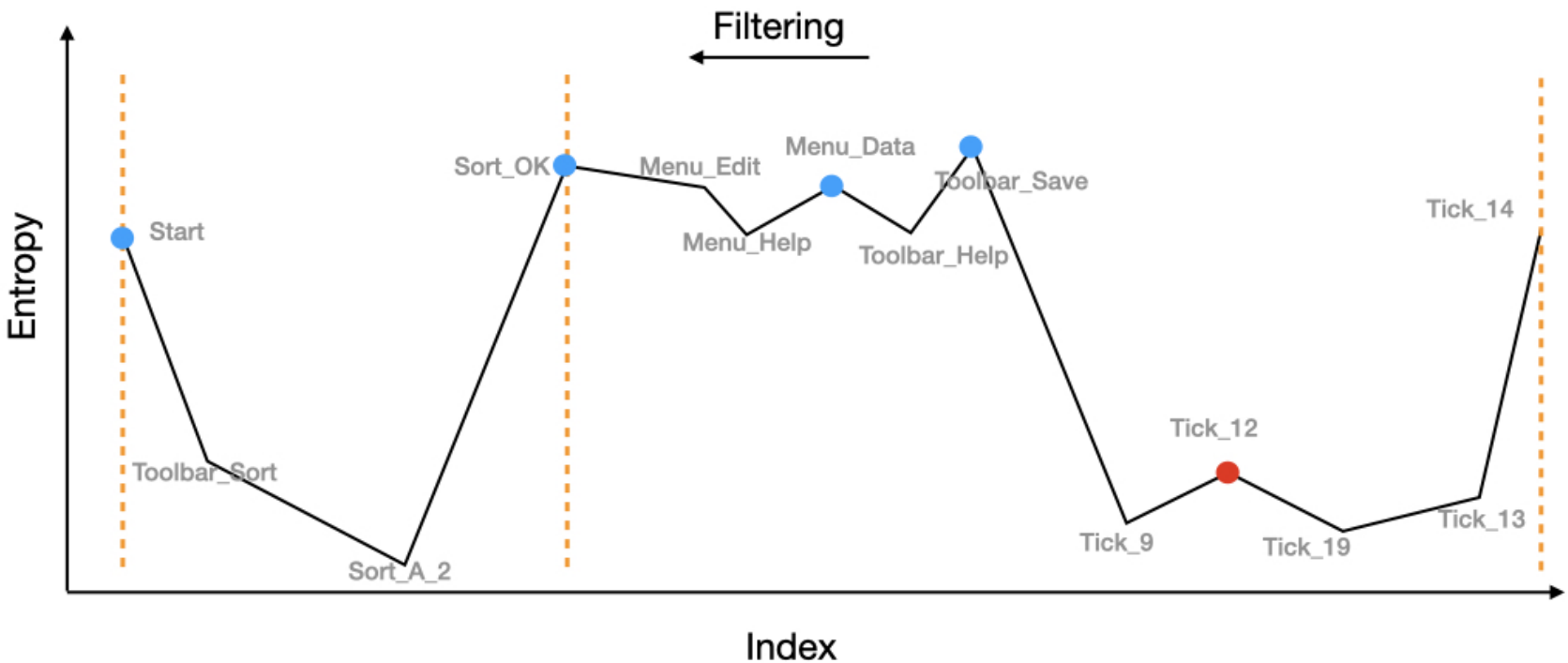
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

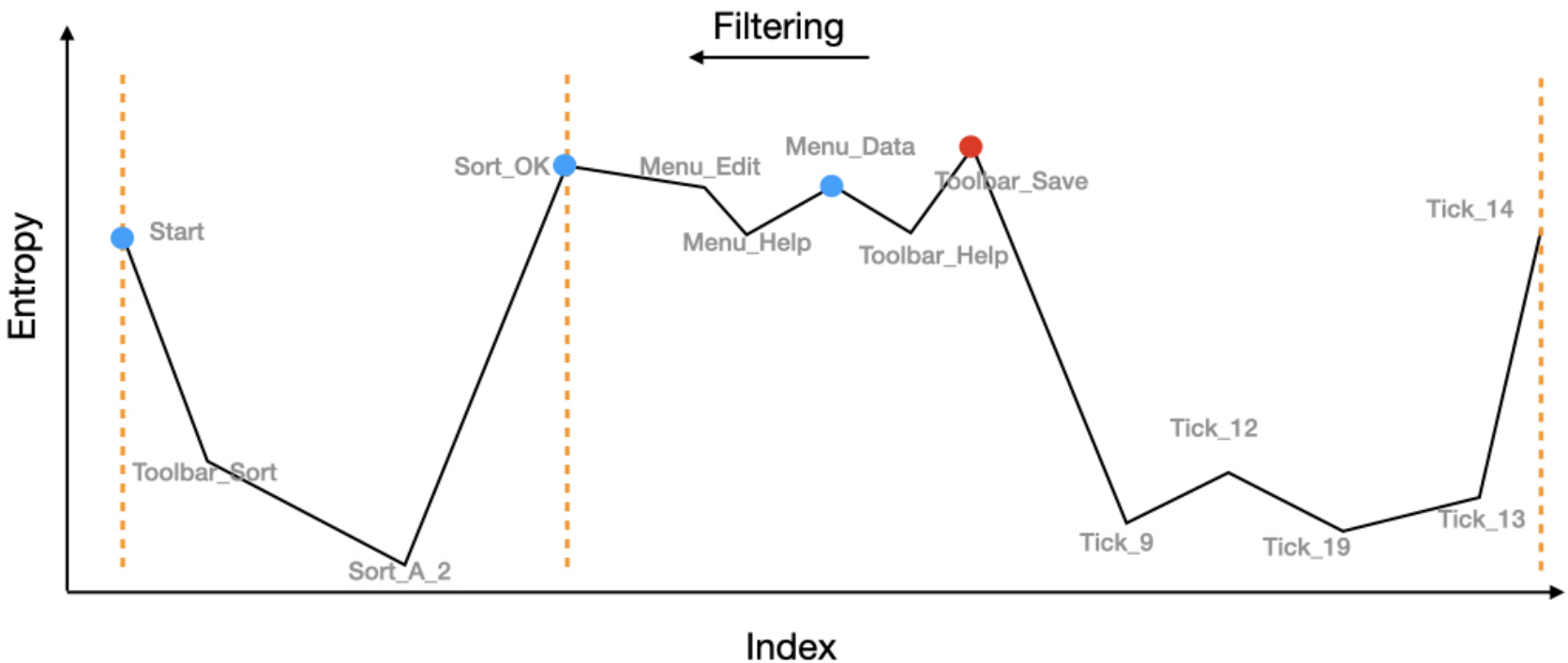
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

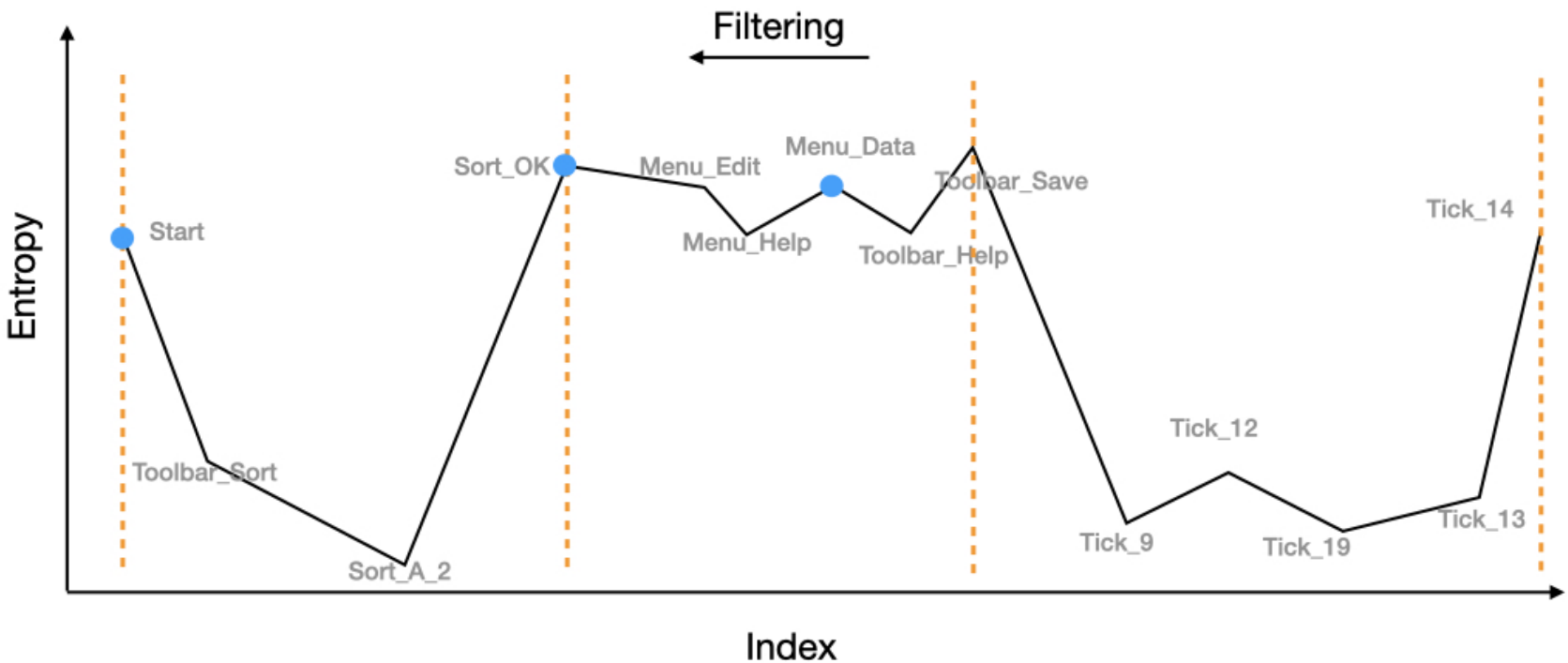
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

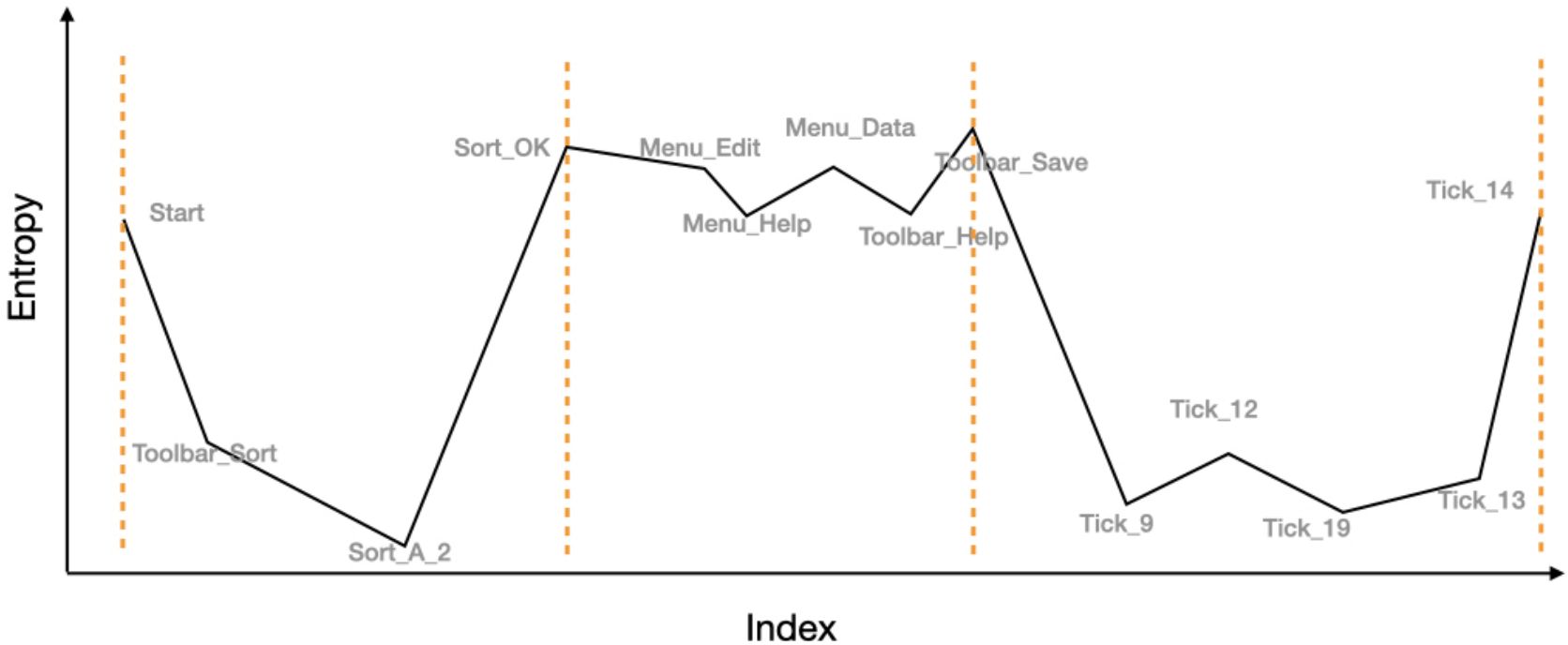
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



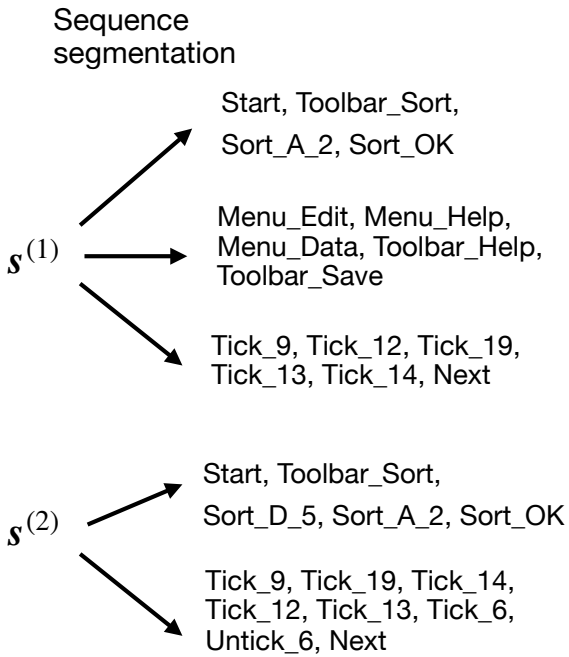
Step 2: Segmentation

- The segmentation step partitions an entropy sequence by identifying U-curves (deep enough U-shaped curves).
- A subset $h_{i:j}$ of the entropy process $\mathbf{h}$ forms a U-curve if

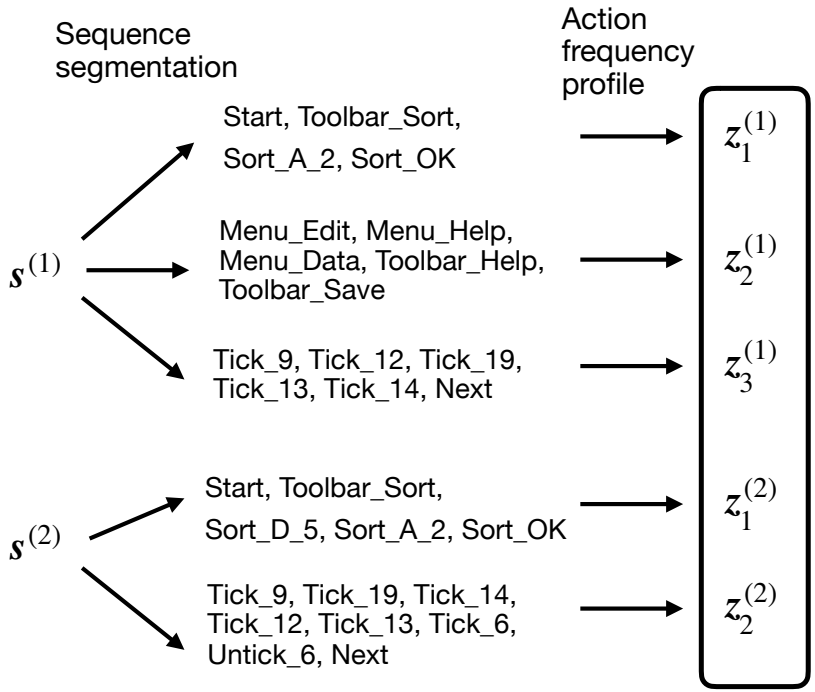
$$\min\{h_i, h_j\} - \min_{i \le t \le j} h_t \ge \lambda \left(\max_{1 \le t \le L-1} h_t - \min_{1 \le t \le L-1} h_t \right)$$



Step 3: Clustering

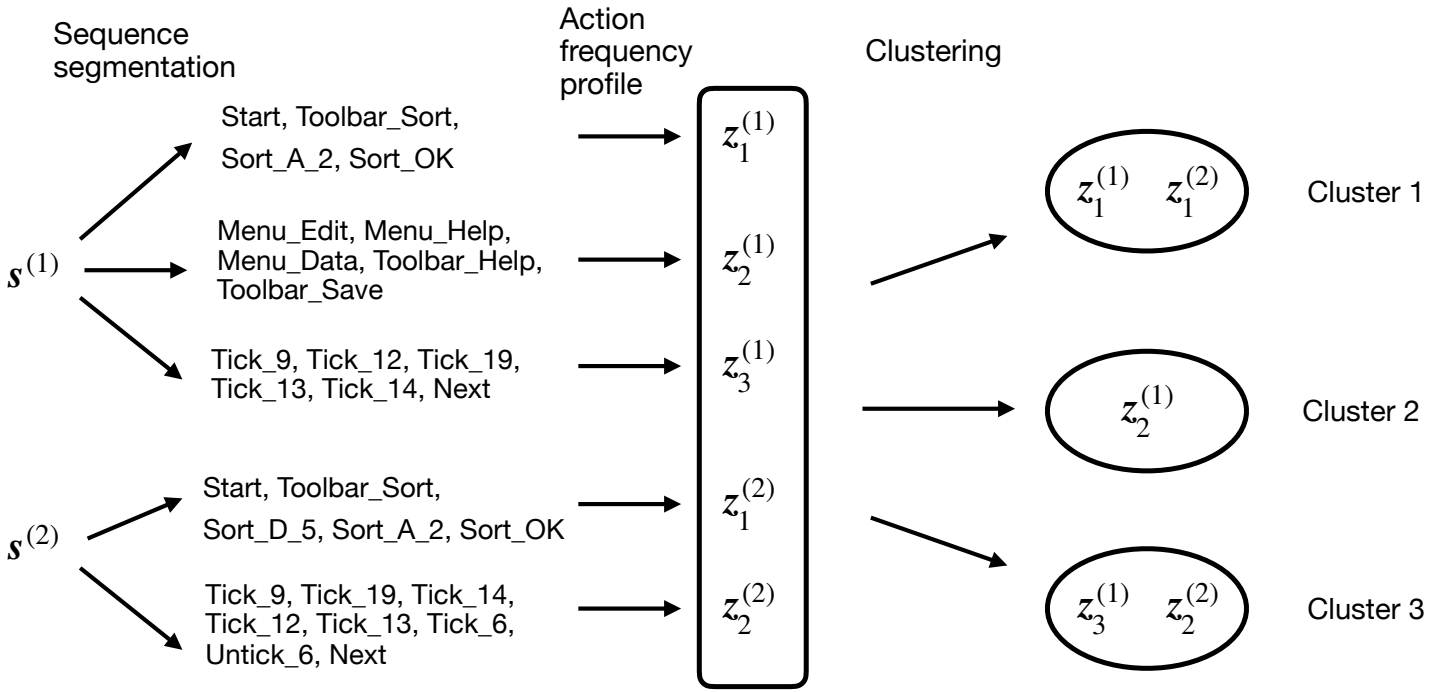


Step 3: Clustering



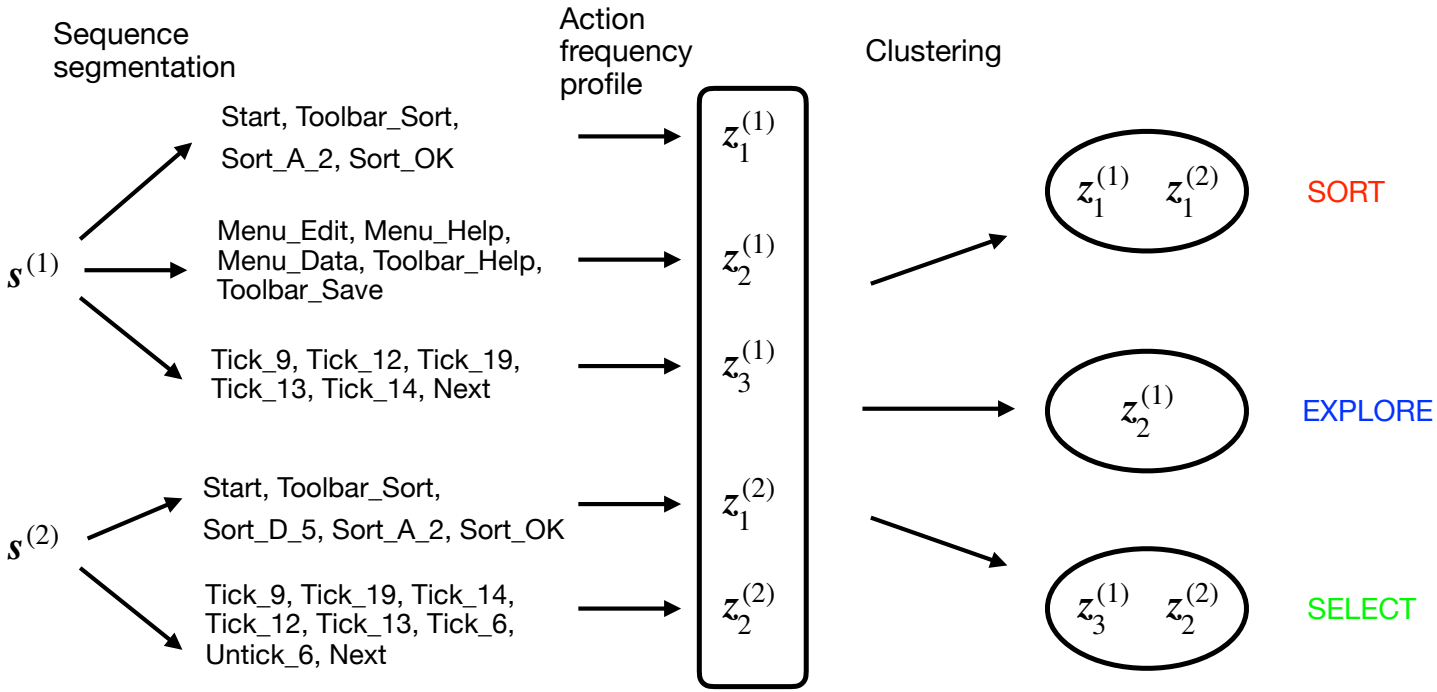
- Each segment is represented by its action frequency profile $\mathbf{z} = (z_1, \ldots, z_M)$
  - z_j is the proportion of action a_j in the segment.

Step 3: Clustering



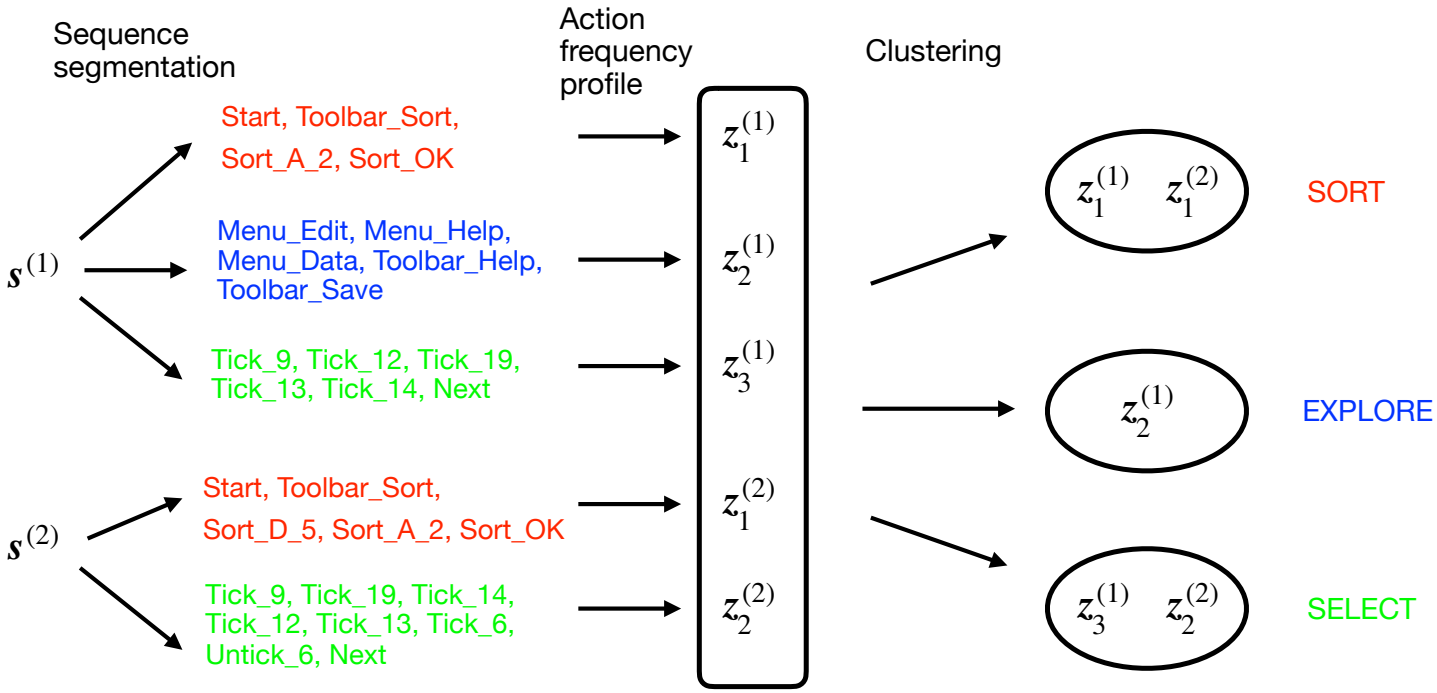
- Similar segments are grouped by clustering the action frequency profiles.
  - We use k-means.

Step 3: Clustering



- Each cluster is interpreted as a subtask.
  - It is often helpful to examine each cluster's action frequency profile relative to the overall action frequency in the dataset.

Step 3: Clustering



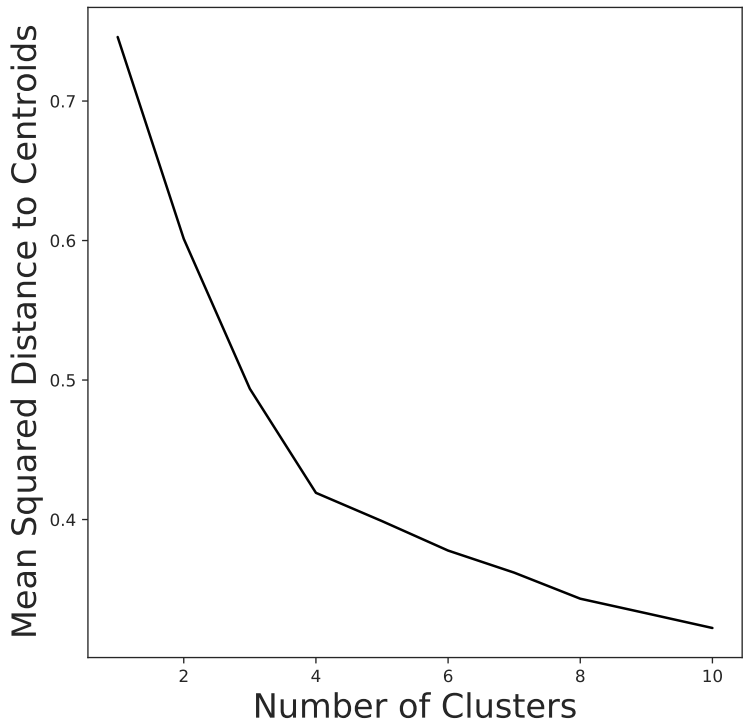
- Each process is transformed into a subtask sequence
  - $s^{(1)} \rightarrow$ (SORT, EXPLORE, SELECT)
  - $s^{(2)} \rightarrow$ (SORT, SELECT)

Step 3: Clustering - Choose Number of Subtasks

- Based on the understanding of the item

Step 3: Clustering - Choose Number of Subtasks

- Based on the understanding of the item
- A data-driven method: elbow method.



Empirical Results – PIAAC U19b

- Item set-up is similar to the example item.

Empirical Results – PIAAC U19b

- Item set-up is similar to the example item.
- M: $\sim$300; L: 10 $\sim$ 1000

Empirical Results – PIAAC U19b

- Item set-up is similar to the example item.
- M: $\sim$300; L: 10 $\sim$ 1000
- Three subtasks (clusters) are identified according to the elbow method.

Empirical Results – PIAAC U19b

- Item set-up is similar to the example item.
- M: $\sim$300; L: 10 $\sim$ 1000
- Three subtasks (clusters) are identified according to the elbow method.
- High relative frequency actions in each cluster:

Empirical Results – PIAAC U19b

- Item set-up is similar to the example item.
- M: $\sim$300; L: 10 $\sim$ 1000
- Three subtasks (clusters) are identified according to the elbow method.
- High relative frequency actions in each cluster:
  - Cluster 1: SS, WP, Menu_WP_Edit, Menu_WP_File, Tick_1, Menu_SS_Edit, Menu_SS_File

Empirical Results – PIAAC U19b

- Item set-up is similar to the example item.
- M: $\sim$300; L: 10 $\sim$ 1000
- Three subtasks (clusters) are identified according to the elbow method.
- High relative frequency actions in each cluster:
  - Cluster 1: SS, WP, Menu_WP_Edit, Menu_WP_File, Tick_1, Menu_SS_Edit, Menu_SS_File
  - Cluster 2: Sort_2A, Sort_2D, Sort_2_D, Sort_OK, Sort_1_D, Sort_1_C, Sort_1D, Sort_2_C, Sort_1A, Menuitem_Sort

Empirical Results – PIAAC U19b

- Item set-up is similar to the example item.
- M: $\sim$300; L: 10 $\sim$ 1000
- Three subtasks (clusters) are identified according to the elbow method.
- High relative frequency actions in each cluster:
  - Cluster 1: SS, WP, Menu_WP_Edit, Menu_WP_File, Tick_1, Menu_SS_Edit, Menu_SS_File
  - Cluster 2: Sort_2A, Sort_2D, Sort_2_D, Sort_OK, Sort_1_D, Sort_1_C, Sort_1D, Sort_2_C, Sort_1A, Menuitem_Sort
  - Cluster 3: Tick_29, Tick_55, Tick_160, Tick_82, Tick_8, Tick_96, Tick_18, Tick_193

Empirical Results – PIAAC U19b

- Item set-up is similar to the example item.
- M: $\sim$300; L: 10 $\sim$ 1000
- Three subtasks (clusters) are identified according to the elbow method.
- High relative frequency actions in each cluster:
  - EXPLORE: SS, WP, Menu_WP_Edit, Menu_WP_File, Tick_1, Menu_SS_Edit, Menu_SS_File
  - Cluster 2: Sort_2A, Sort_2D, Sort_2_D, Sort_OK, Sort_1_D, Sort_1_C, Sort_1D, Sort_2_C, Sort_1A, Menuitem_Sort
  - Cluster 3: Tick_29, Tick_55, Tick_160, Tick_82, Tick_8, Tick_96, Tick_18, Tick_193

Empirical Results – PIAAC U19b

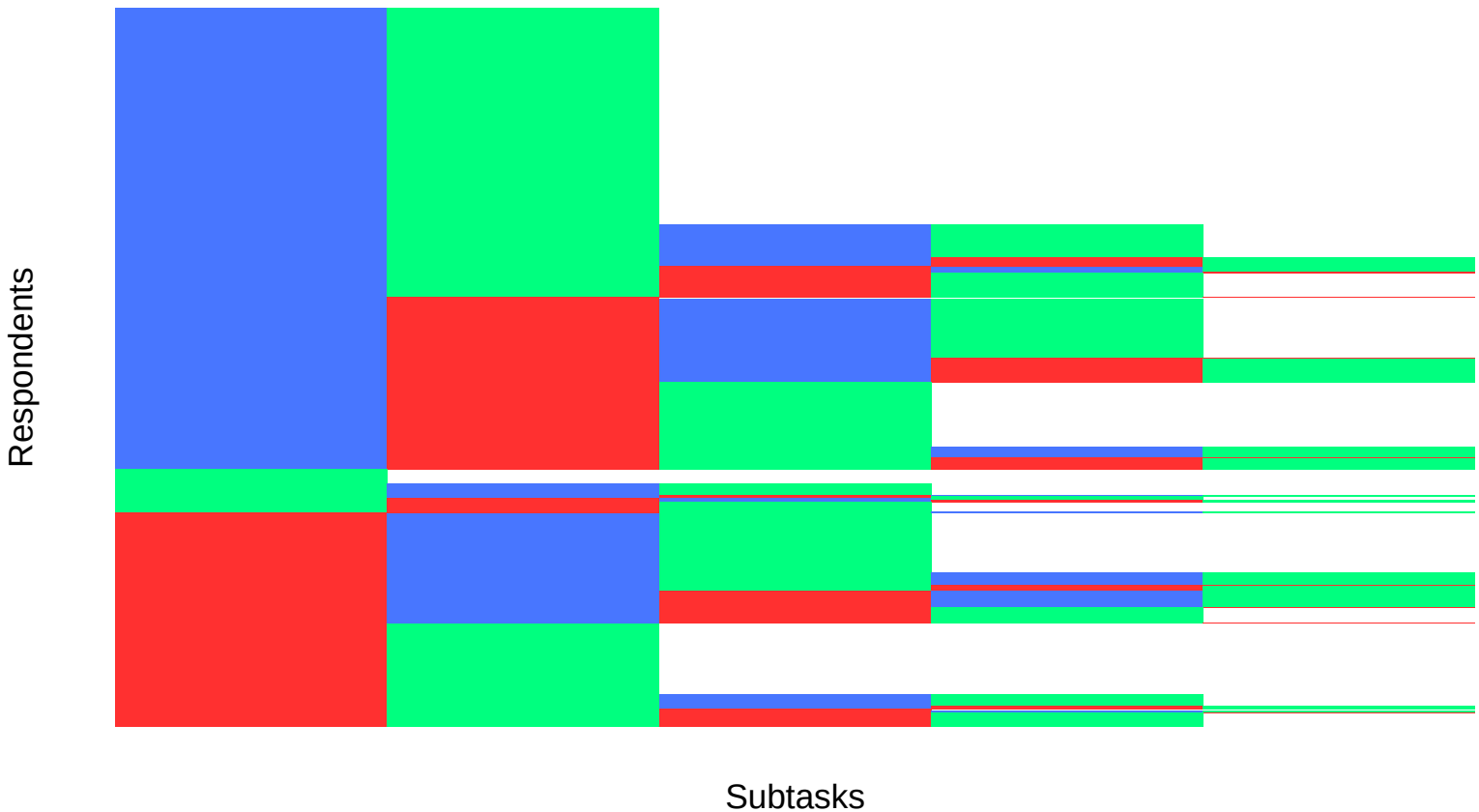
- Item set-up is similar to the example item.
- M: $\sim$300; L: 10 $\sim$ 1000
- Three subtasks (clusters) are identified according to the elbow method.
- High relative frequency actions in each cluster:
  - EXPLORE: SS, WP, Menu_WP_Edit, Menu_WP_File, Tick_1, Menu_SS_Edit, Menu_SS_File
  - SORT: Sort_2A, Sort_2D, Sort_2_D, Sort_OK, Sort_1_D, Sort_1_C, Sort_1D, Sort_2_C, Sort_1A, Menuitem_Sort
  - Cluster 3: Tick_29, Tick_55, Tick_160, Tick_82, Tick_8, Tick_96, Tick_18, Tick_193

Empirical Results – PIAAC U19b

- Item set-up is similar to the example item.
- M: $\sim$300; L: 10 $\sim$ 1000
- Three subtasks (clusters) are identified according to the elbow method.
- High relative frequency actions in each cluster:
  - EXPLORE: SS, WP, Menu_WP_Edit, Menu_WP_File, Tick_1, Menu_SS_Edit, Menu_SS_File
  - SORT: Sort_2A, Sort_2D, Sort_2_D, Sort_OK, Sort_1_D, Sort_1_C, Sort_1D, Sort_2_C, Sort_1A, Menuitem_Sort
  - ANSWER: Tick_29, Tick_55, Tick_160, Tick_82, Tick_8, Tick_96, Tick_18, Tick_193

Empirical Results – PIAAC U19b

Subtask Visualization



Empirical Results – PIAAC U19b

Some descriptive statistics on strategies:

Empirical Results – PIAAC U19b

Some descriptive statistics on strategies:

- 69.2% of the respondents used the sorting strategy.

Empirical Results – PIAAC U19b

Some descriptive statistics on strategies:

- 69.2% of the respondents used the sorting strategy.
- Use SORT: 71.0% correct; No SORT: 61.5% correct.

Empirical Results – PIAAC U19b

Some descriptive statistics on strategies:

- 69.2% of the respondents used the sorting strategy.
- Use SORT: 71.0% correct; No SORT: 61.5% correct.
- 1 - 4 SORT: 72% correct; $>$ 4 SORT: 55% correct

Empirical Results – PIAAC U19b

Some descriptive statistics on strategies:

- 69.2% of the respondents used the sorting strategy.
- Use SORT: 71.0% correct; No SORT: 61.5% correct.
- 1 - 4 SORT: 72% correct; $>$ 4 SORT: 55% correct
- Total response time in seconds for correct answers:
  - Use SORT: 201 (211);
  - No SORT: 213 (705).

Subtask Analysis in ProcData[1]

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=20,
                 step_size=0.001, batch_size=1,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE,
                 ...)
```

- `subtask_analysis()` performs the three-step Subtask Identification Procedure (SIP).

[1] The functions mentioned in pages 19–22 is currently available in the version hosted on Github and will be available in the next version on CRAN.

Subtask Analysis in ProcData[1]

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=20,
                 step_size=0.001, batch_size=1,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE,
                 ...)
```

- `subtask_analysis()` performs the three-step Subtask Identification Procedure (SIP).
  - Step 1: `action2entropy()` fits an action prediction model and computes the entropies of predictive distributions.

[1] The functions mentioned in pages 19–22 is currently available in the version hosted on Github and will be available in the next version on CRAN.

Subtask Analysis in ProcData[1]

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=20,
                 step_size=0.001, batch_size=1,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE,
                 ...)
```

- `subtask_analysis()` performs the three-step Subtask Identification Procedure (SIP).
  - Step 1: `action2entropy()` fits an action prediction model and computes the entropies of predictive distributions.
  - Step 2: `entropy2segment()` segments the entropy sequences

[1] The functions mentioned in pages 19–22 is currently available in the version hosted on Github and will be available in the next version on CRAN.

Subtask Analysis in `ProcData`[1]

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=20,
                 step_size=0.001, batch_size=1,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE,
                 ...)
```

- `subtask_analysis()` performs the three-step Subtask Identification Procedure (SIP).
  - Step 1: `action2entropy()` fits an action prediction model and computes the entropies of predictive distributions.
  - Step 2: `entropy2segment()` segments the entropy sequences
  - Step 3: `segment2subtask()` clustering the segments to form subtasks.

[1] The functions mentioned in pages 19–22 is currently available in the version hosted on Github and will be available in the next version on CRAN.

Subtask Analysis in ProcData

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=50,
                 step_size=0.001, batch_size=16,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE, ...)
```

Subtask Analysis in ProcData

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=50,
                 step_size=0.001, batch_size=16,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE, ...)
```

- It takes
  - a list of action sequences,

Subtask Analysis in ProcData

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=50,
                 step_size=0.001, batch_size=16,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE, ...)
```

- It takes
  - a list of action sequences,
  - the desired or candidate values of the number of subtasks,

Subtask Analysis in ProcData

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=50,
                 step_size=0.001, batch_size=16,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE, ...)
```

- It takes
  - a list of action sequences,
  - the desired or candidate values of the number of subtasks,
  - the tuning parameter in segmentation,

Subtask Analysis in ProcData

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=50,
                 step_size=0.001, batch_size=16,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE, ...)
```

- It takes
  - a list of action sequences,
  - the desired or candidate values of the number of subtasks,
  - the tuning parameter in segmentation,
  - the configuration of the RNN-based action prediction model,

Subtask Analysis in `ProcData`

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=50,
                 step_size=0.001, batch_size=16,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE, ...)
```

- It takes
  - a list of action sequences,
  - the desired or candidate values of the number of subtasks,
  - the tuning parameter in segmentation,
  - the configuration of the RNN-based action prediction model,
  - control parameters for fitting the action prediction model.

Subtask Analysis in ProcData

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=50,
                 step_size=0.001, batch_size=16,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE, ...)
```

- It returns an object of class "subtask", a list containing final and intermediate results from the procedure:

Subtask Analysis in ProcData

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=50,
                 step_size=0.001, batch_size=16,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE, ...)
```

- It returns an object of class "subtask", a list containing final and intermediate results from the procedure:
  - entropy sequences, subtask sequences,

Subtask Analysis in `ProcData`

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=50,
                 step_size=0.001, batch_size=16,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE, ...)
```

- It returns an object of class `"subtask"`, a list containing final and intermediate results from the procedure:
  - entropy sequences, subtask sequences,
  - relative action frequency for each subtask,

Subtask Analysis in ProcData

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=50,
                 step_size=0.001, batch_size=16,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE, ...)
```

- It returns an object of class "subtask", a list containing final and intermediate results from the procedure:
  - entropy sequences, subtask sequences,
  - relative action frequency for each subtask,
  - configuration of the fitted action prediction model

Subtask Analysis in `ProcData`

```
subtask_analysis(action_seqs, n_subtask, lambda=0.3,
                 rnn_dim=20, n_epoch=50,
                 step_size=0.001, batch_size=16,
                 optimizer_name="rmsprop",
                 index_valid = 0.2, verbose=FALSE, ...)
```

- It returns an object of class `"subtask"`, a list containing final and intermediate results from the procedure:
  - entropy sequences, subtask sequences,
  - relative action frequency for each subtask,
  - configuration of the fitted action prediction model

Code demonstration: chunk 1 in `subtask_demo.R`

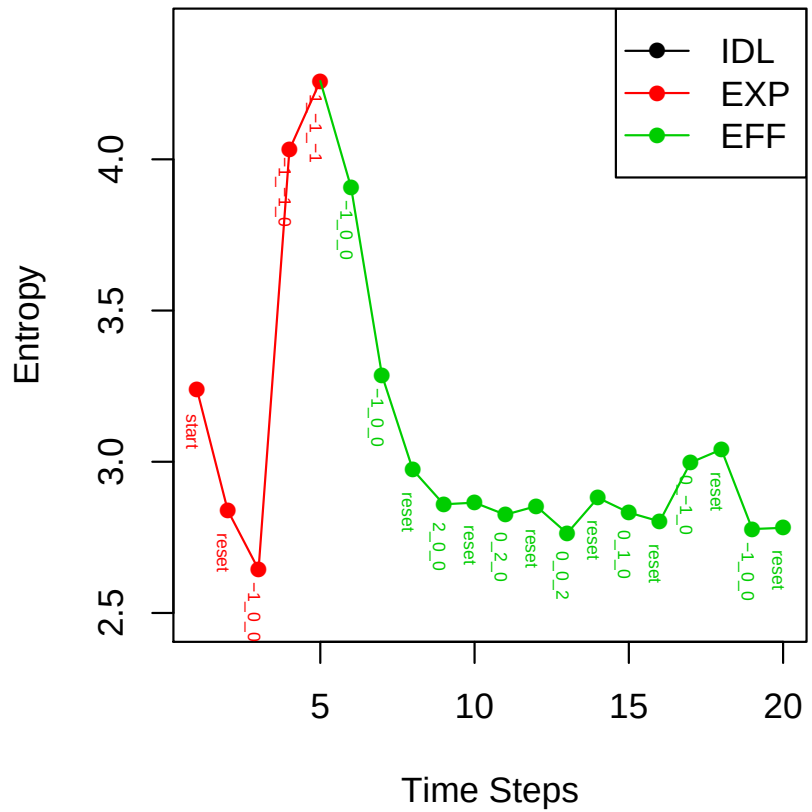
Plotting Results

`plot.subtask()`: `plot` method for objects of class `"subtask"`.

Plotting Results

`plot.subtask()`: `plot` method for objects of class `"subtask"`.

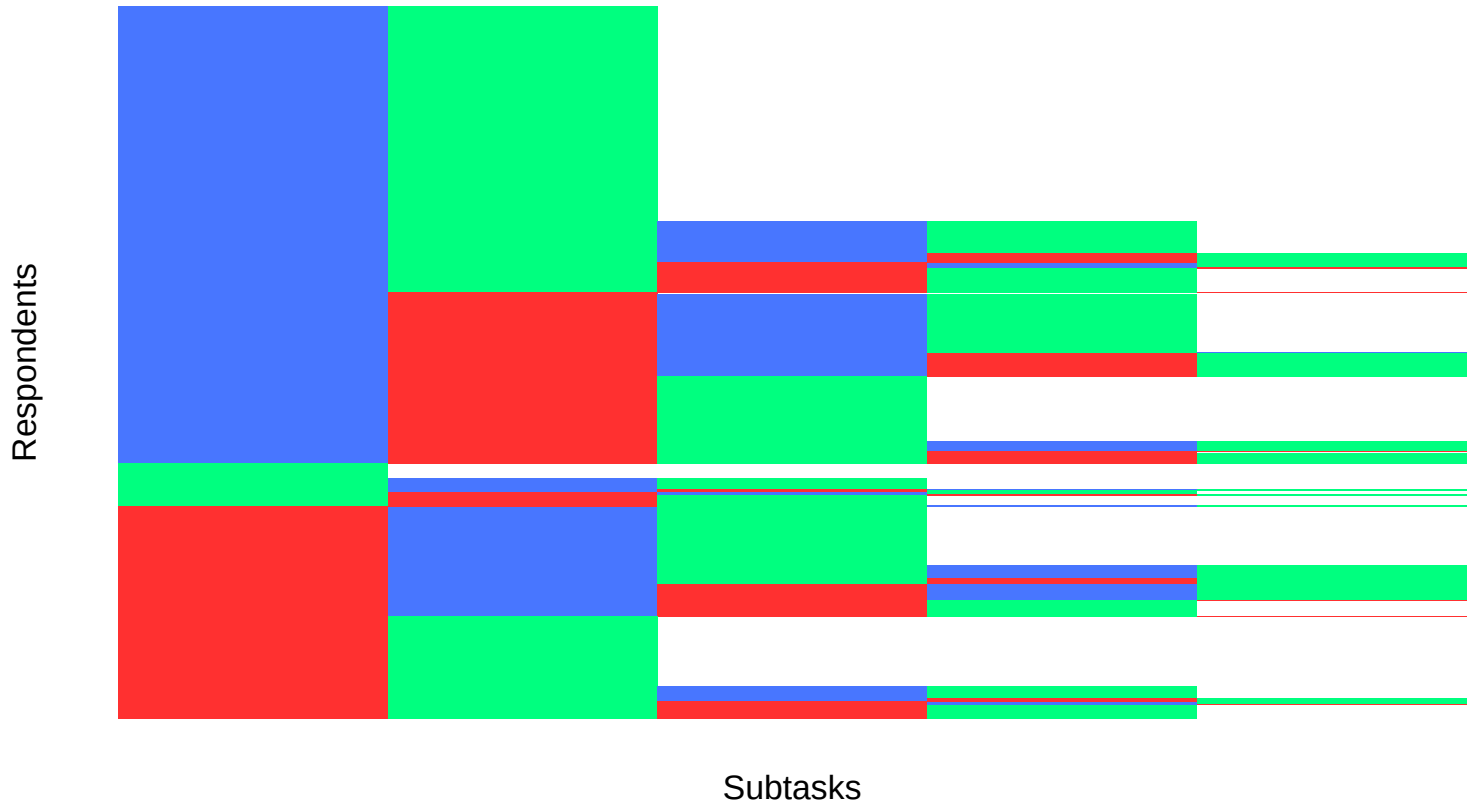
- It plots subtask analysis results for individual sequences or the entire dataset.



Plotting Results

`plot.subtask()`: `plot` method for objects of class `"subtask"`.

- It plots subtask analysis results for individual sequences or the entire dataset.



Plotting Results

```
plot.subtask(object, type, col.subtask, max_len,
             index, cex.action, lty, pch, srt,
             plot_legend, legend_pos, ...)
```

Plotting Results

```
plot.subtask(object, type, col.subtask, max_len,
             index, cex.action, lty, pch, srt,
             plot_legend, legend_pos, ...)
```

- `object` is an object of class `"subtask"`.
- `type` specifies the plot type: `"individual"` or `"all"`.

Plotting Results

```
plot.subtask(object, type, col.subtask, max_len,
             index, cex.action, lty, pch, srt,
             plot_legend, legend_pos, ...)
```

- `object` is an object of class `"subtask"`.
- `type` specifies the plot type: `"individual"` or `"all"`.
- `col.subtask` specifies the colors for the subtasks.

Plotting Results

```
plot.subtask(object, type, col.subtask, max_len,
             index, cex.action, lty, pch, srt,
             plot_legend, legend_pos, ...)
```

- `object` is an object of class `"subtask"`.
- `type` specifies the plot type: `"individual"` or `"all"`.
- `col.subtask` specifies the colors for the subtasks.
- `plot_lengend` specifies whether to plot legends for subtasks. If `TRUE`, `legend_pos` specifies where to position the legend.

Plotting Results

```
plot.subtask(object, type, col.subtask, max_len,
             index, cex.action, lty, pch, srt,
             plot_legend, legend_pos, ...)
```

- `object` is an object of class `"subtask"`.
- `type` specifies the plot type: `"individual"` or `"all"`.
- `col.subtask` specifies the colors for the subtasks.
- `plot_lengend` specifies whether to plot legends for subtasks. If `TRUE`, `legend_pos` specifies where to position the legend.
- If `type="all"`, `max_len` specifies the max length of the plotted subtask sequences.

Plotting Results

```
plot.subtask(object, type, col.subtask, max_len,
             index, cex.action, lty, pch, srt,
             plot_legend, legend_pos, ...)
```

- `object` is an object of class `"subtask"`.
- `type` specifies the plot type: `"individual"` or `"all"`.
- `col.subtask` specifies the colors for the subtasks.
- `plot_lengend` specifies whether to plot legends for subtasks. If `TRUE`, `legend_pos` specifies where to position the legend.
- If `type="all"`, `max_len` specifies the max length of the plotted subtask sequences.
- If `type="individual"`,
  - `index` are the indices of the sequences to be plotted.

Plotting Results

```
plot.subtask(object, type, col.subtask, max_len,
             index, cex.action, lty, pch, srt,
             plot_legend, legend_pos, ...)
```

- `object` is an object of class `"subtask"`.
- `type` specifies the plot type: `"individual"` or `"all"`.
- `col.subtask` specifies the colors for the subtasks.
- `plot_lengend` specifies whether to plot legends for subtasks. If `TRUE`, `legend_pos` specifies where to position the legend.
- If `type="all"`, `max_len` specifies the max length of the plotted subtask sequences.
- If `type="individual"`,
  - `index` are the indices of the sequences to be plotted.
  - `lty` and `pch` specifies line type and point characters.
  - `cex.action` specifies the size of action text labels.
  - `srt` specifies the angle to rotate the text labels.

Plotting Results

```
plot.subtask(object, type, col.subtask, max_len,
             index, cex.action, lty, pch, srt,
             plot_legend, legend_pos, ...)
```

- `object` is an object of class `"subtask"`.
- `type` specifies the plot type: `"individual"` or `"all"`.
- `col.subtask` specifies the colors for the subtasks.
- `plot_lengend` specifies whether to plot legends for subtasks. If `TRUE`, `legend_pos` specifies where to position the legend.
- If `type="all"`, `max_len` specifies the max length of the plotted subtask sequences.
- If `type="individual"`,
  - `index` are the indices of the sequences to be plotted.
  - `lty` and `pch` specifies line type and point characters.
  - `cex.action` specifies the size of action text labels.
  - `srt` specifies the angle to rotate the text labels.

Code demonstration: chunk 2 in `subtask_demo.R`

References

- Wang, Z., Tang, X., Liu, J., and Ying, Z. (2020). Subtask Analysis of Process Data Through a Predictive Model. https://arxiv.org/abs/2009.00717

Thank You!

xytang@math.arizona.edu