PENS Data Augmentation Algorithms

Algorithm 1 Find Valid Segment

```
1: procedure FindSegment($docs, acts, len$) // docs: List of document IDs to process // acts: List of corresponding actions (gen_summ, summ_gen) // len: Desired length of the segment to extract
2:     if $|docs| < len$ then return null // Return null if trajectory too short
3:     end if
4:     $valid \leftarrow \emptyset$ // Store segments that match our pattern
5:     for $i \leftarrow 0$ to $|docs| - len$ do
6:         $segDocs \leftarrow docs[i : i + len]$ // Get window of documents
7:         $segActs \leftarrow acts[i : i + len]$ // Get corresponding actions
8:         for $j \leftarrow 0$ to $|segActs| - 1$ do // Check if we have a valid pattern: // 1. Must find gen_summ followed by summ_gen // 2. Cannot start with summ_gen (needs previous context) // 3. Cannot end with gen_summ (needs completion)
9:             if $segActs[j : j + 2] = [\text{'gen\_summ'}, \text{'summ\_gen'}]$ and
10:     $segActs[0] \neq \text{'summ\_gen'}$ and
11:     $segActs[-1] \neq \text{'gen\_summ'}$ then
12:                 $valid \leftarrow valid \cup \{(segDocs, segActs)\}$
13:                 break // Found valid pattern, check next window
14:             end if
15:         end for
16:     end for
17:     if $valid \neq \emptyset$ then return Random($valid$) // Return random segment for diversity
18:     elsereturn null // No valid segments found
19:     end if
20: end procedure
```

Parameter Descriptions

Key Parameters and Their Significance:

- **docs, actions**: The document IDs and their corresponding actions in the trajectory
  - docs: Contains the sequence of document identifiers
  - acts: Contains actions like 'gen_summ' (generate then summarize) and 'summ_gen' (summarize then generate)
- **len**: Length of segment to extract from source trajectory
  - Must be long enough to contain valid patterns
  - Controlled by minLen and maxLen bounds
- β: Maximum length limit for augmented trajectories
  - Prevents trajectories from growing too large
  - Acts as a hard cutoff for final sequence length

Algorithm 2 Augment Data

```
1: procedure AUGMENT($traj, \beta, srcNum, minLen, maxLen, gap$) // traj: Collection of trajectories to augment // : Maximum allowed length of final trajectory // srcNum: Number of source trajectories to sample // minLen, maxLen: Min/Max segment length bounds // gap: Number of original elements to keep between segments
2:     for each $t$ in $traj$ do
3:         $docs, acts \leftarrow t.docs, t.acts$ // Get current trajectory data
4:         $src \leftarrow$ SAMPLE($traj \setminus \{t\}, srcNum$) // Get source trajectories
5:         $segs \leftarrow \emptyset$ // Store valid segments we find
6:         for each $s$ in $src$ do
7:             $len \leftarrow$ RANDOM($minLen, maxLen$) // Random segment length
8:             $seg \leftarrow$ FINDSEGMENT($s.docs, s.acts, len$)
9:             if $seg \neq null$ then
10:                $segs \leftarrow segs \cup \{seg\}$ // Store if valid
11:            end if
12:        end for
13:        if $segs = \emptyset$ then
14:            continue // Skip if no valid segments
15:        end if
16:        $pos \leftarrow$ RANDOM($0, |docs|$) // Random start position
17:        $augDocs \leftarrow docs[0 : pos]$ // Keep start of original
18:        $augActs \leftarrow acts[0 : pos]$
19:        $curr \leftarrow pos$ // Track current position
20:        for each $(sDocs, sActs)$ in $segs$ do
21:            if $curr \geq \beta$ then
22:                break // Stop if reached max length
23:            end if
24:            $space \leftarrow \beta - curr$ // Calculate remaining space
25:            $len \leftarrow \min(|sDocs|, space)$ // Limit segment length
26:            $augDocs \leftarrow augDocs \mathbin{+\!\!+} sDocs[0 : len]$ // Add segment
27:            $augActs \leftarrow augActs \mathbin{+\!\!+} sActs[0 : len]$
28:            $curr \leftarrow curr + len$
29:            if $curr + gap \leq \beta$ and $curr < |docs|$ then
30:                $gEnd \leftarrow \min(curr + gap, |docs|)$ // Calculate gap end
31:                $augDocs \leftarrow augDocs \mathbin{+\!\!+} docs[curr : gEnd]$ // Add gap
32:                $augActs \leftarrow augActs \mathbin{+\!\!+} acts[curr : gEnd]$
33:                $curr \leftarrow gEnd$
34:            end if
35:        end for
36:        $t.docs \leftarrow augDocs[0 : \beta]$ // Update trajectory
37:        $t.acts \leftarrow augActs[0 : \beta]$
38:        $t.sums \leftarrow$ COUNT($augActs$, 'summ_gen') // Update summary count
39:    end for
40: end procedure
```

- **srcNum**: Number of source trajectories to sample from
  - Controls how many different trajectories contribute segments
  - More sources = more diversity in augmented data
- **minLen, maxLen**: Segment length bounds
  - minLen: Minimum length of extracted segments
  - maxLen: Maximum length of extracted segments
  - Controls size of inserted content
- **gap**: Spacing between inserted segments
  - Number of original elements to keep between insertions
  - Helps maintain coherence in final sequence