Lecture 9

Sequence Segmentation

Part 1: Definition and applications

Machine Learning for Structured Data

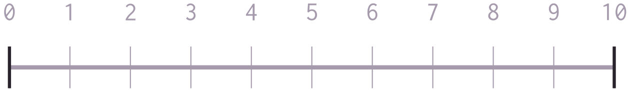
Vlad Niculae · LTL, UvA · https://vene.ro/mlsd

Sequence Segmentation

1. **Definition and applications**
2. Representation and scoring
3. Algorithm
4. Evaluation
5. Extensions

Sequence Segmentation

The rod cutting problem: We have a rod of length n units, and we can cut it at every marker. What cuts to make to maximize the total value of the resulting pieces?

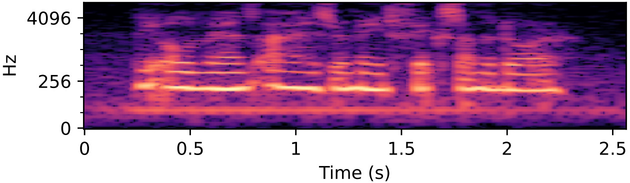


DNA/RNA: A C A G A T T A C C

Word segmentation: 私 は 日 本 語 を 学 習

Entity Extraction: Mayor Halsema to visit the University of Amsterdam next Friday

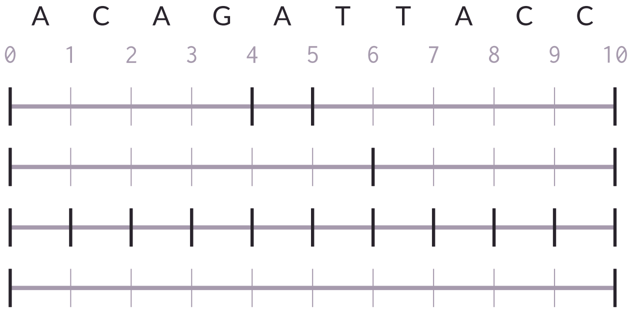
Speech:



Machine Learning for Structured Data
Vlad Niculae · LTL, UvA · `https://vene.ro/mlsd`

Sequence Segmentation

Representing and scoring segmentations

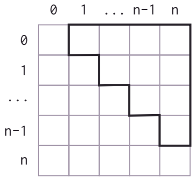


cuts	score
[4,5]	$a_{0,4} + a_{4,5} + a_{5,10}$
[6]	$a_{0,6} + a_{6,10}$
[1,2,...,9]	$a_{0,1} + a_{1,2} + \ldots + a_{9,10}$
[]	$a_{0,10}$

How many possible segmentations? How many possible segments?

Assign a score for every possible segment $(i, j) : 0 \le i < j \le n$.

Easiest is to store in the "upper triangle" of a $(n+1) \times (n+1)$ matrix:



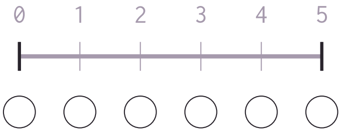
Machine Learning for Structured Data

Vlad Niculae · LTL, UvA · https://vene.ro/mlsd

Sequence Segmentation

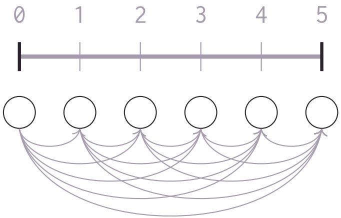
1. Definition and applications
2. Representation and scoring
3. **Algorithm**
4. Evaluation
5. Extensions

Dynamic programming: DAG formulation



Nodes: one per fencepost. $V = \{0, 1, \ldots, n\}$.

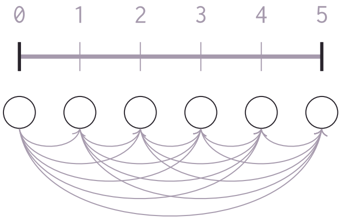
Dynamic programming: DAG formulation



Nodes: one per fencepost. $V = \{0, 1, \ldots, n\}$.

Edges: one per segment.
$E = \{(i, j) : 0 \leq i < j \leq n\}$.

Dynamic programming: DAG formulation

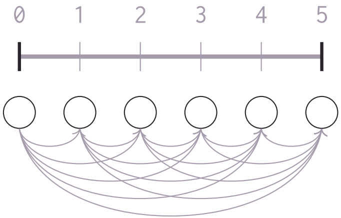


Nodes: one per fencepost. $V = \{0, 1, \ldots, n\}$.

Edges: one per segment.
$E = \{(i, j) : 0 \leq i < j \leq n\}$.

Topologic order?

Dynamic programming: DAG formulation



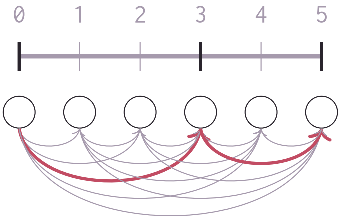
Nodes: one per fencepost. $V = \{0, 1, \ldots, n\}$.

Edges: one per segment.
$E = \{(i, j) : 0 \leq i < j \leq n\}$.

Topologic order?

Any path from 0 to n corresponds to a segmentation of the sequence.

Dynamic programming: DAG formulation



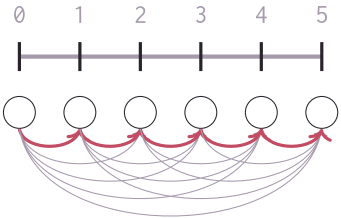
Nodes: one per fencepost. $V = \{0, 1, \ldots, n\}$.

Edges: one per segment.
$E = \{(i, j) : 0 \leq i < j \leq n\}$.

Topologic order?

Any path from 0 to n corresponds
to a segmentation of the sequence.

Dynamic programming: DAG formulation



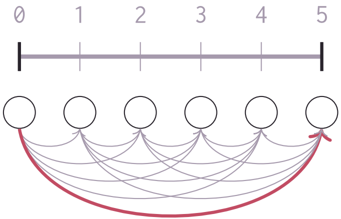
Nodes: one per fencepost. $V = \{0, 1, \ldots, n\}$.

Edges: one per segment.
$E = \{(i, j) : 0 \leq i < j \leq n\}$.

Topologic order?

Any path from 0 to n corresponds
to a segmentation of the sequence.

Dynamic programming: DAG formulation



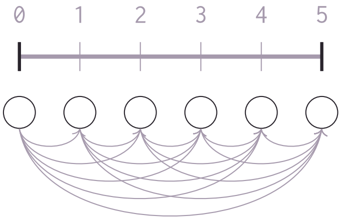
Nodes: one per fencepost. $V = \{0, 1, \ldots, n\}$.

Edges: one per segment.
$E = \{(i, j) : 0 \le i < j \le n\}$.

Topologic order?

Any path from 0 to n corresponds
to a segmentation of the sequence.

Dynamic programming: DAG formulation



Nodes: one per fencepost. $V = \{0, 1, \ldots, n\}$.

Edges: one per segment.
$E = \{(i, j) : 0 \le i < j \le n\}$.

Topologic order?

Any path from 0 to n corresponds to a segmentation of the sequence.

Viterbi for segmentation

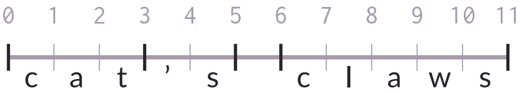
input: segment scores $\boldsymbol{A} \in \mathbb{R}^{n \times n}$

Forward: compute recursively
$m_1 = a_{01}; \pi_1 = 0$
for $j = 2$ to n **do**
$\quad m_j \leftarrow \max_{0 \le i < j} m_i + a_{ij}$
$\quad \pi_j \leftarrow \arg\max_{0 \le i < j} m_i + a_{ij}$
$f^\star = m_n$
Backward: follow backpointers
$\boldsymbol{y}^\star = []; j \leftarrow n$
while $j > 0$ **do**
$\quad \boldsymbol{y}^\star = [(\pi_j, j)] + \boldsymbol{y}^\star$
$\quad j = \pi_j$

Analogously, we can obtain a *Forward* algorithm for $\log Z$: exercise for you.

Sequence Segmentation

Evaluation



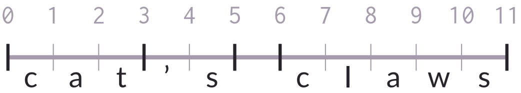
True segments: $\boldsymbol{y} = [(0,3),(3,5),(5,6),(6,11)]$

A few possible predictions:
$\hat{\boldsymbol{y}}_a = [(0,11)]$

$\hat{\boldsymbol{y}}_b = [(0,1),(1,2),\ldots,(10,11)]$

$\hat{\boldsymbol{y}}_c = [(0,3),(3,5),(5,11)]$

Evaluation



True segments: $\boldsymbol{y} = [(0,3),(3,5),(5,6),(6,11)]$

A few possible predictions:
$\hat{\boldsymbol{y}}_a = [(0,11)]$

$\hat{\boldsymbol{y}}_b = [(0,1),(1,2),\dots,(10,11)]$

$\hat{\boldsymbol{y}}_c = [(0,3),(3,5),(5,11)]$

The number of predicted and true segments differ.

A common way to evaluate in this scenario is:

$$\text{precision} = \frac{\text{n. correctly predicted segments}}{\text{n. predicted segments}}$$

$$\text{recall} = \frac{\text{n. correctly predicted segments}}{\text{n. true segments}}$$

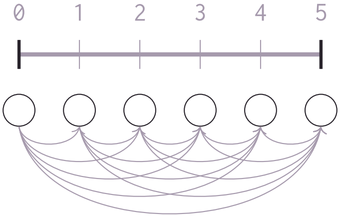
$$F_1 = \frac{2PR}{P+R}$$

More advanced metrics can partially reward overlaps.

Machine Learning for Structured Data
Vlad Niculae · LTL, UvA · `https://vene.ro/mlsd`

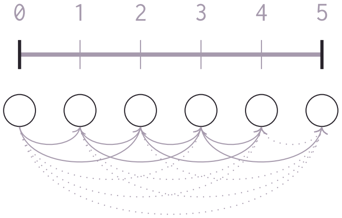
Sequence Segmentation

Extension 1: bounded segment length



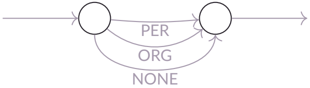
- can be much faster if we limit segment lengts to $L \ll n$.
- in terms of the DAG: discard edges ij where $j - i > L$

Extension 1: bounded segment length



- can be much faster if we limit segment lengts to $L \ll n$.
- in terms of the DAG: discard edges ij where $j - i > L$

Extension 2: labeled segments



- each segment also receives a label (e.g., `PERSON`, `ORGANIZATION`, `NONE`...)
- the labels are independent given the cuts: for any two nodes in the DAG, we only need to pick the best edge between them.

Extension 3: labeled + transitions

- drawing inspiration from sequence tagging: what if we want a reward/penalty for consecutive `PERSON`$\rightarrow$`ORGANIZATION` segments?
- labels no longer independent given cuts.
- still solvable via DP, but must keep track of transitions.
- essentially a combination of the sequence tagging DAG and the segmentation DAG.

Summary

- Segmentations of a length-n sequence: $O(2^n)$ possible segmentations, $O(n^2)$ possible segments.
- Dynamic programming gives us polynomial-time complexity.
- Extensions can accommodate maximum lengths, labels, transitions.