CS 481

Artificial Intelligence Language Understanding

February 7, 2023

Announcements / Reminders

- Please follow the Week 04 To Do List instructions
- Quiz #04 due on Sunday (02/12/23) at 11:59 PM CST
- PA #01 due on Monday (02/20/23) at 11:59 PM CST

- Exam dates:
  - Midterm: 03/02/2023 during Thursday lecture time
  - Final: 04/27/2023 during Thursday lecture time

@them.iit.edu | Hackathon 2023

Linktree with **interest form** and **application for executive board**

https://linktr.ee/them.at.iit

Plan for Today

- Spelling: Minimum Edit Distance
- Parts of Speech tagging – introduction (if time permits)

Spelling: Real-world Problems

- **Non-word error detection**
  - *graffe* **instead of** giraffe
- **Isolated-word error correction**
- **Context-dependent error detection and correction**
  - **typos**
    - *three* **instead of** *there*
  - **homophone or near-homophones**
    - *dessert* **instead of** *desert* **or** *piece* **for** *peace*

How Similar are Two Strings?

- **The user typed "*graffe*". Which string is closest?**
  - *graf*
  - *graft*
  - *grail*
  - *giraffe*

- **Why? Spell checking**

How Similar are Two Strings?

- **Why? Computational Biology:**
  - **Align two sequences of nucleotides:**

```
AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC
```

  - **Resulting alignment:**

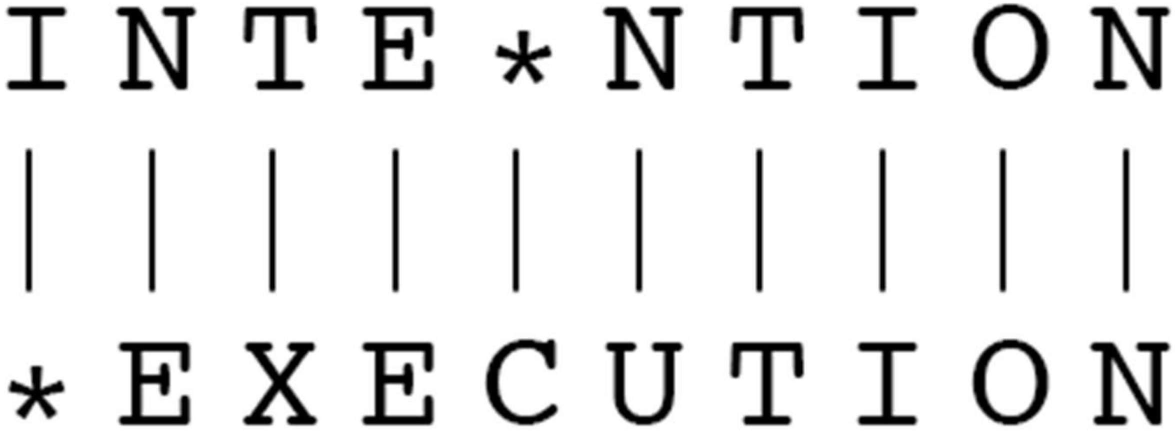
```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC
```

How Similar are Two Strings?

- **The user typed “*graffe*”. Which string is closest?**
  - *graf* **deleted “*i*” deleted “*fe*”**
  - *graft* **deleted “*i*” “*e*” and substituted “*f*”**
  - *grail* **deletion and substitution**
  - *giraffe* **correct form (we need to insert “*i*”)**

Alignment

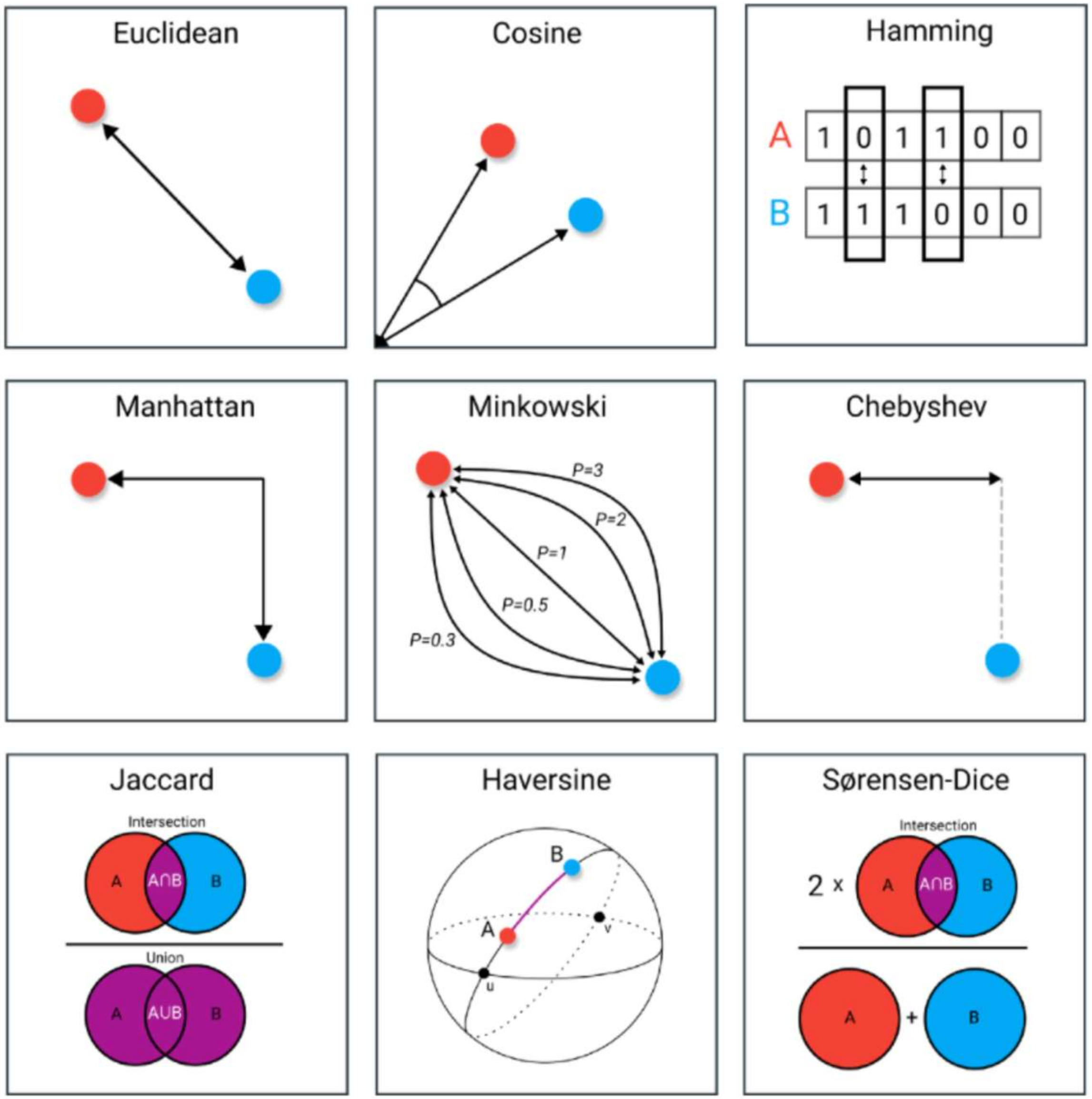
Given two sequences, an alignment is a correspondence between substrings of the two sequences.



```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
```

Alignment is made up of edits.

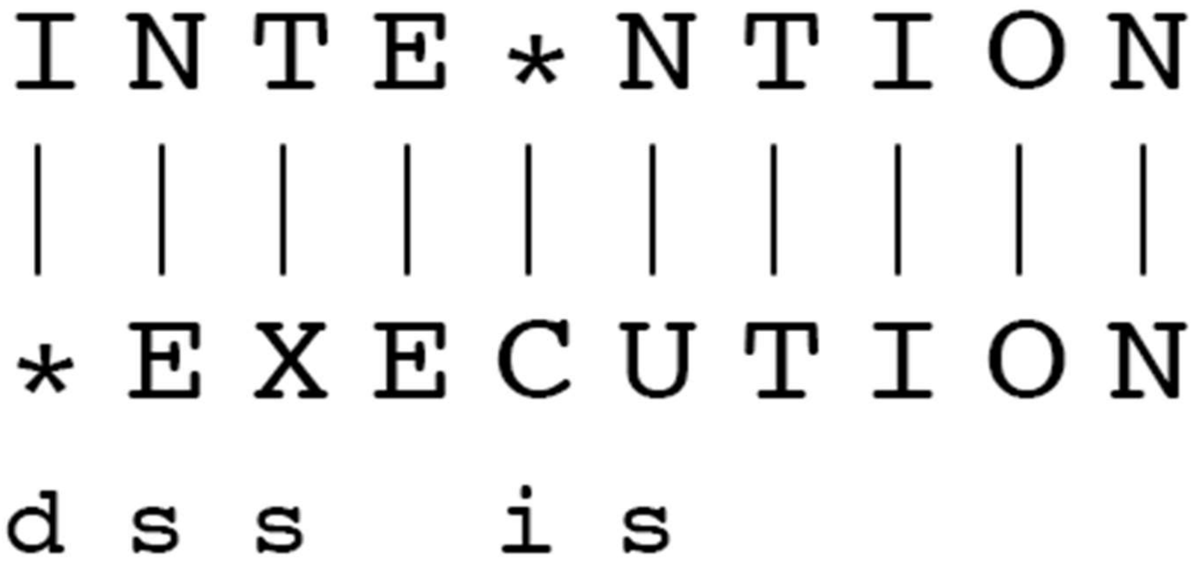
Distance Measures | String Distance?



Source: https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa

Edits

One string can be transformed to another by a sequence of edits (delete, **i**nsert, substitute**).**

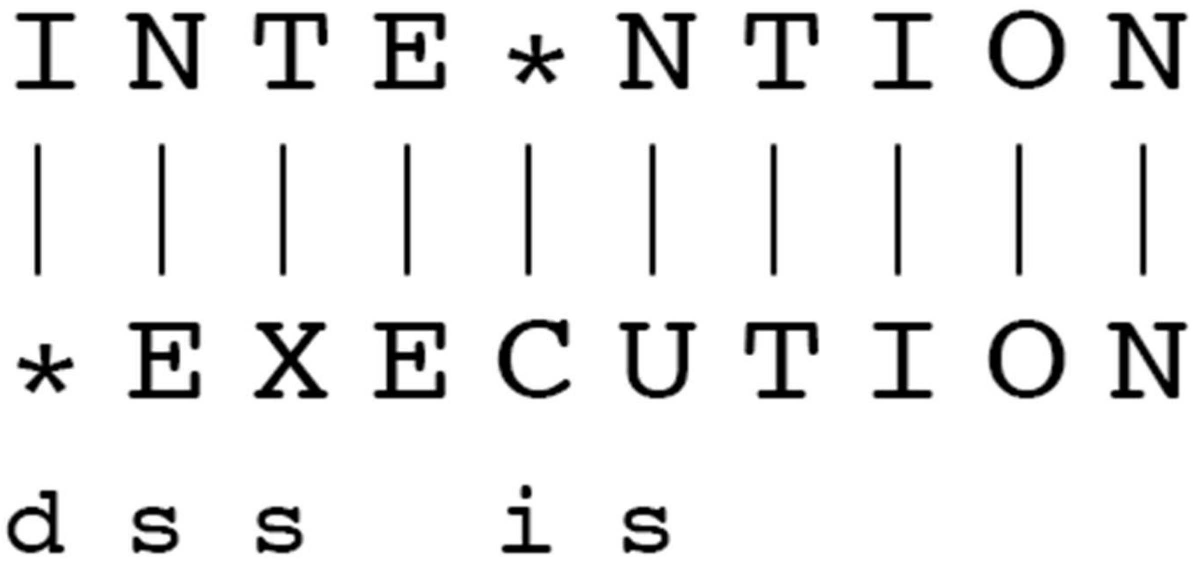


```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

Edits with Costs: Edit Distance

Each edit operation can have its cost:

- **cost(d) = cost(i) = cost(s) = 1**



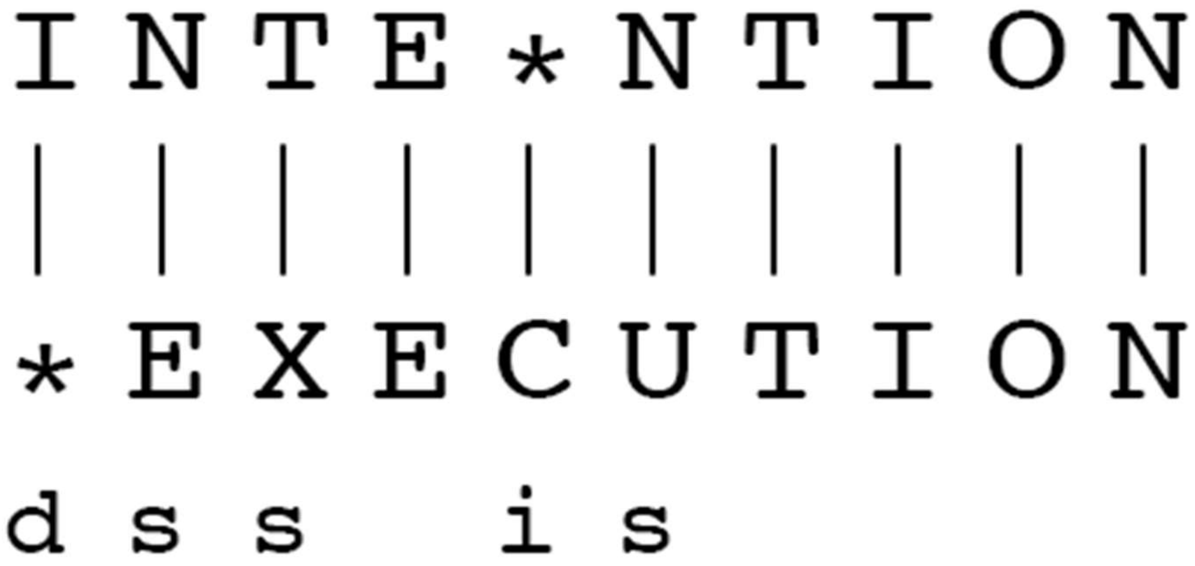
```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

Edit distance = 5

Edits with Costs: Levenshtein Distance

Each edit operation can have its cost:

- **cost(d) = cost(i) = 1 | cost(s) = cost(d) + cost(i) = 2**

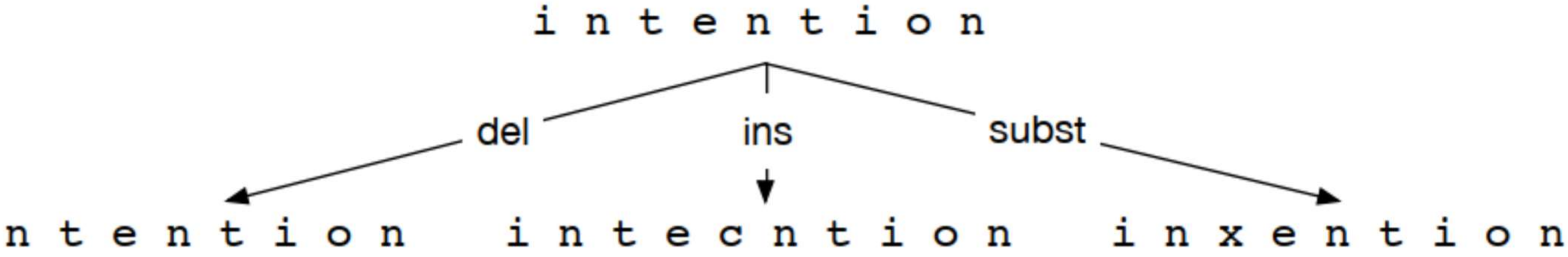


```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

Levenshtein edit distance = 8

Searching for Minimum Edit Path

String transformation (a sequence of edits) can be represented with a tree:

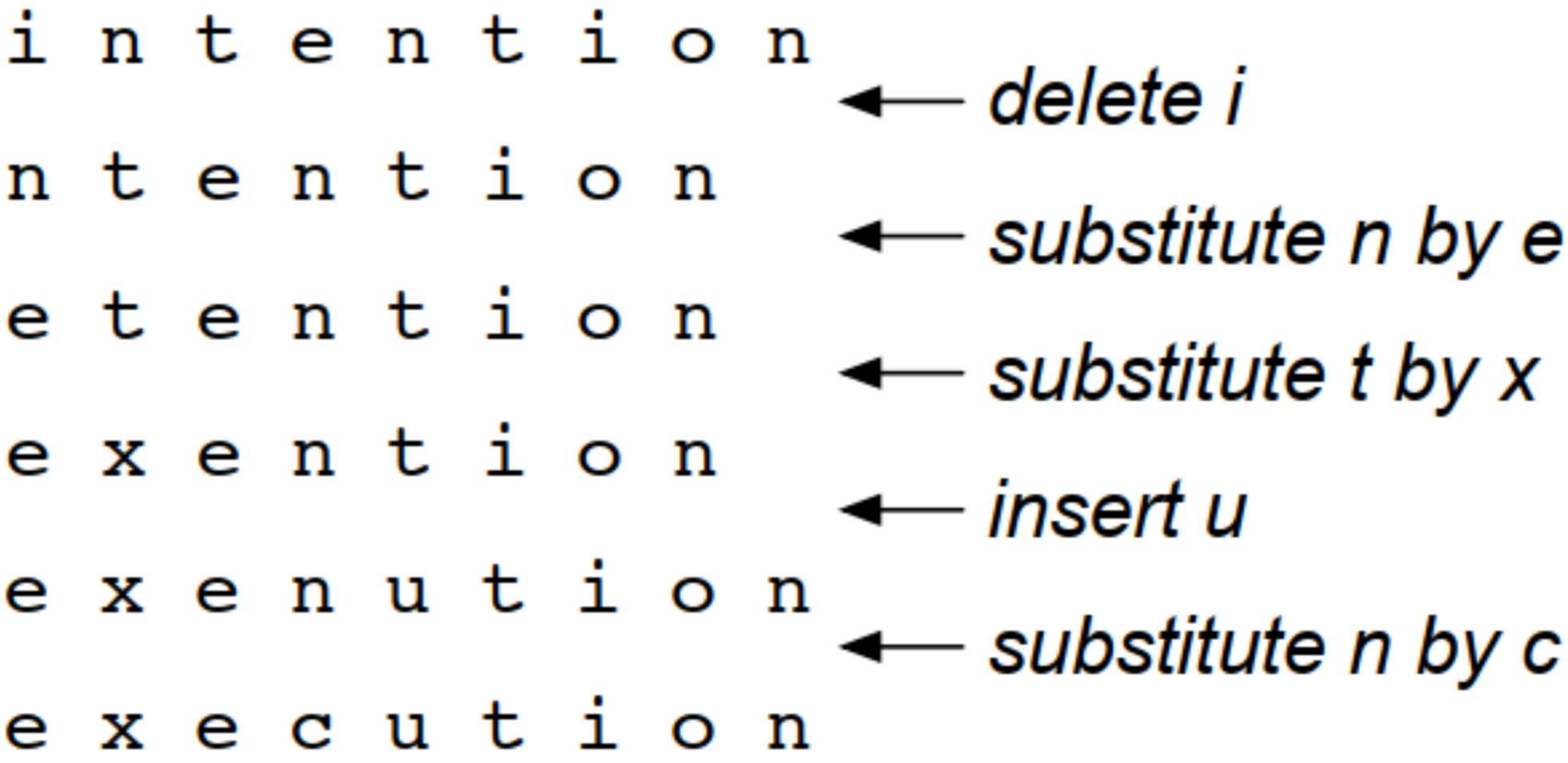


Solution: Minimum Edit Path found via tree search:

- **Initial state (root): the word we're transforming**
- **Operators / actions: insert, delete, substitute**
- **Goal state: the word we're trying to get to**
- **Path cost: what we want to minimize - the number of edits**

Edit Path

One of the edit paths (we want minimum # of edits):



intention

← *delete i*

ntention

← *substitute n by e*

etention

← *substitute t by x*

exention

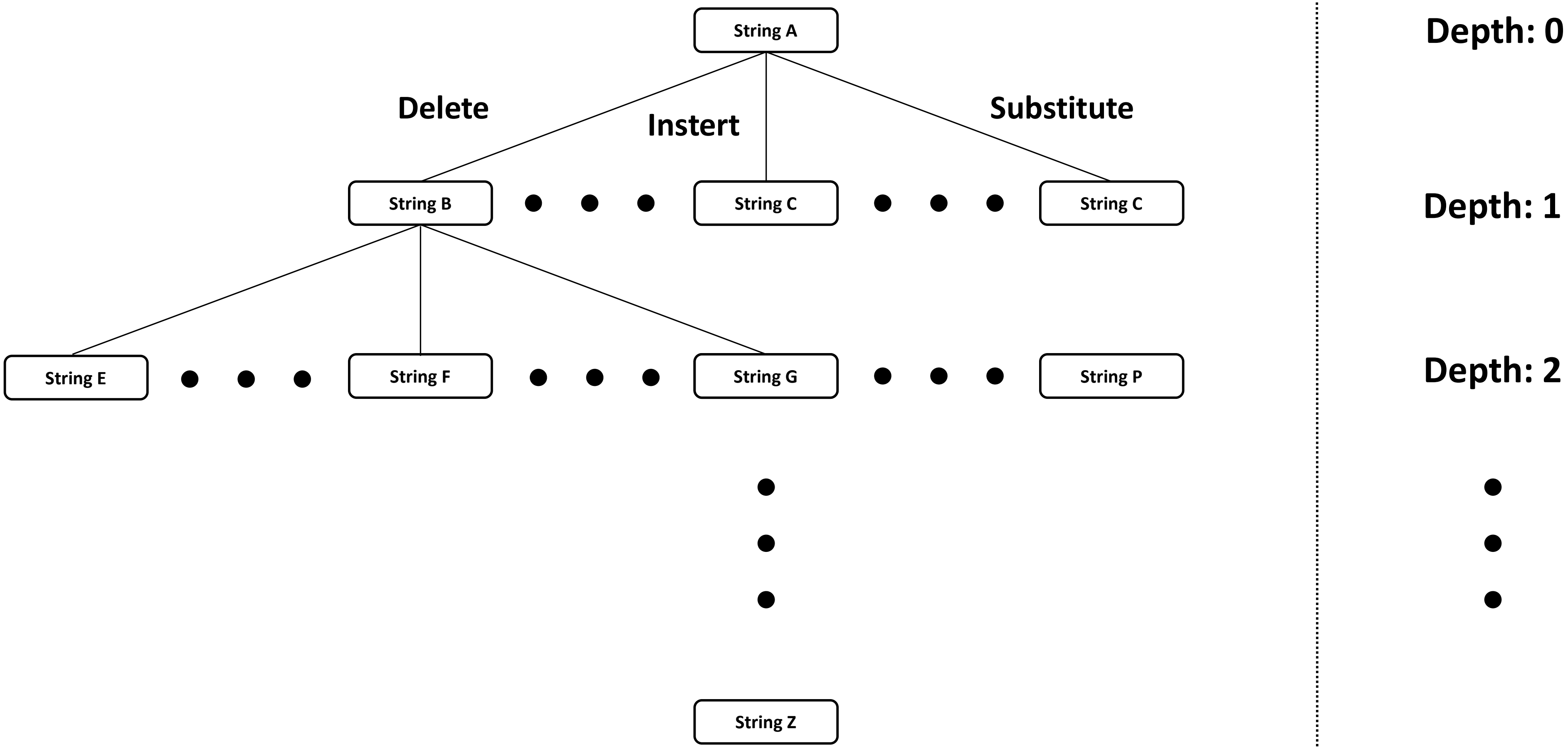
← *insert u*

exenution

← *substitute n by c*

execution

Finding Minimum Edit Path /w Search



Quickly becomes unmanageable and impossible to search with brute force!

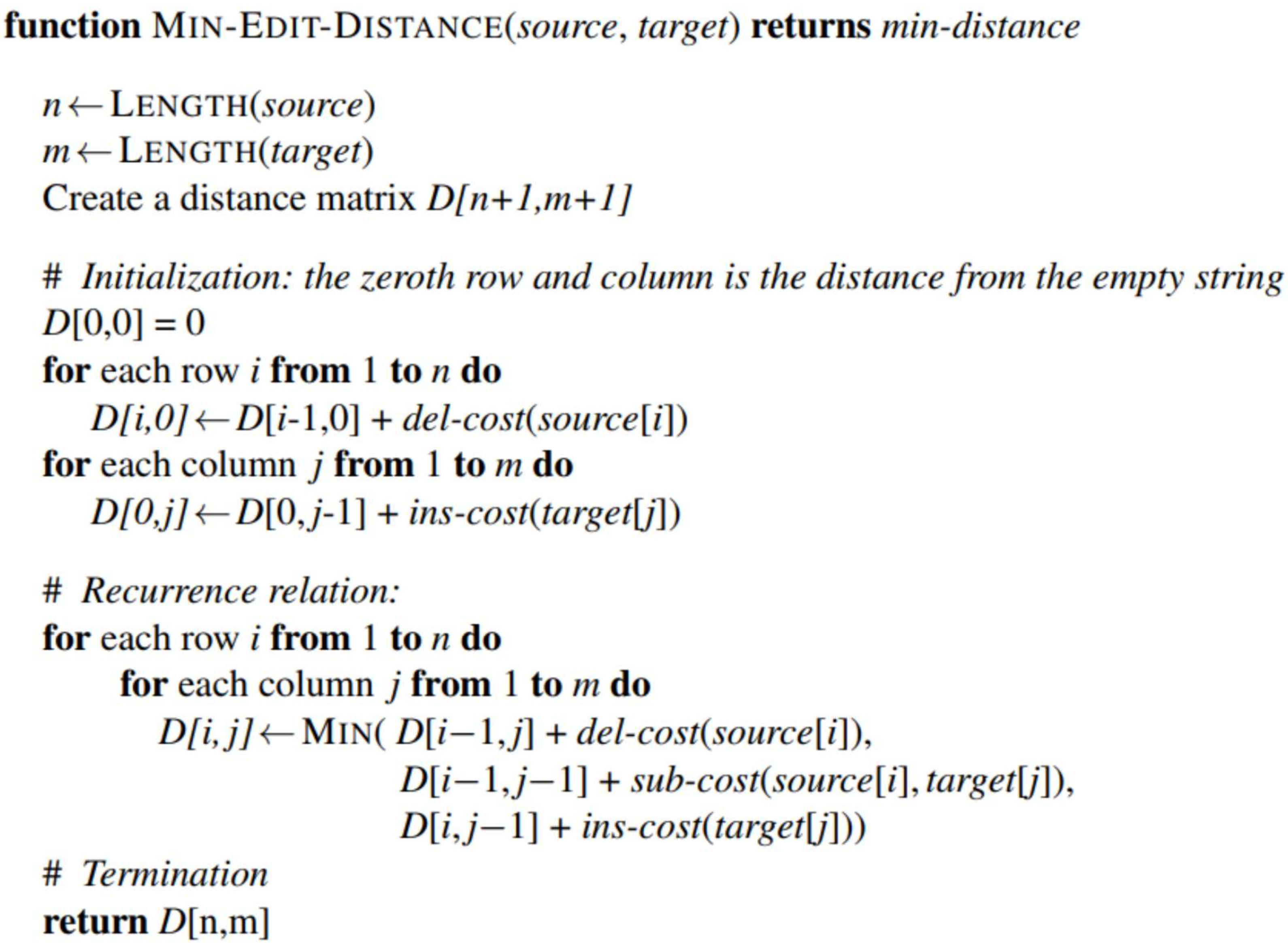
Minimum Edit Distance: Definition

- For two strings:
  - X of length n
  - Y of length m
- We define D(i, j)
  - the edit distance between X[1..i] and Y[1..j]
  - i.e., the first i characters of X and the first j characters of Y
  - The edit distance between X and Y is thus D(n, m)

MED: Dynamic Programming

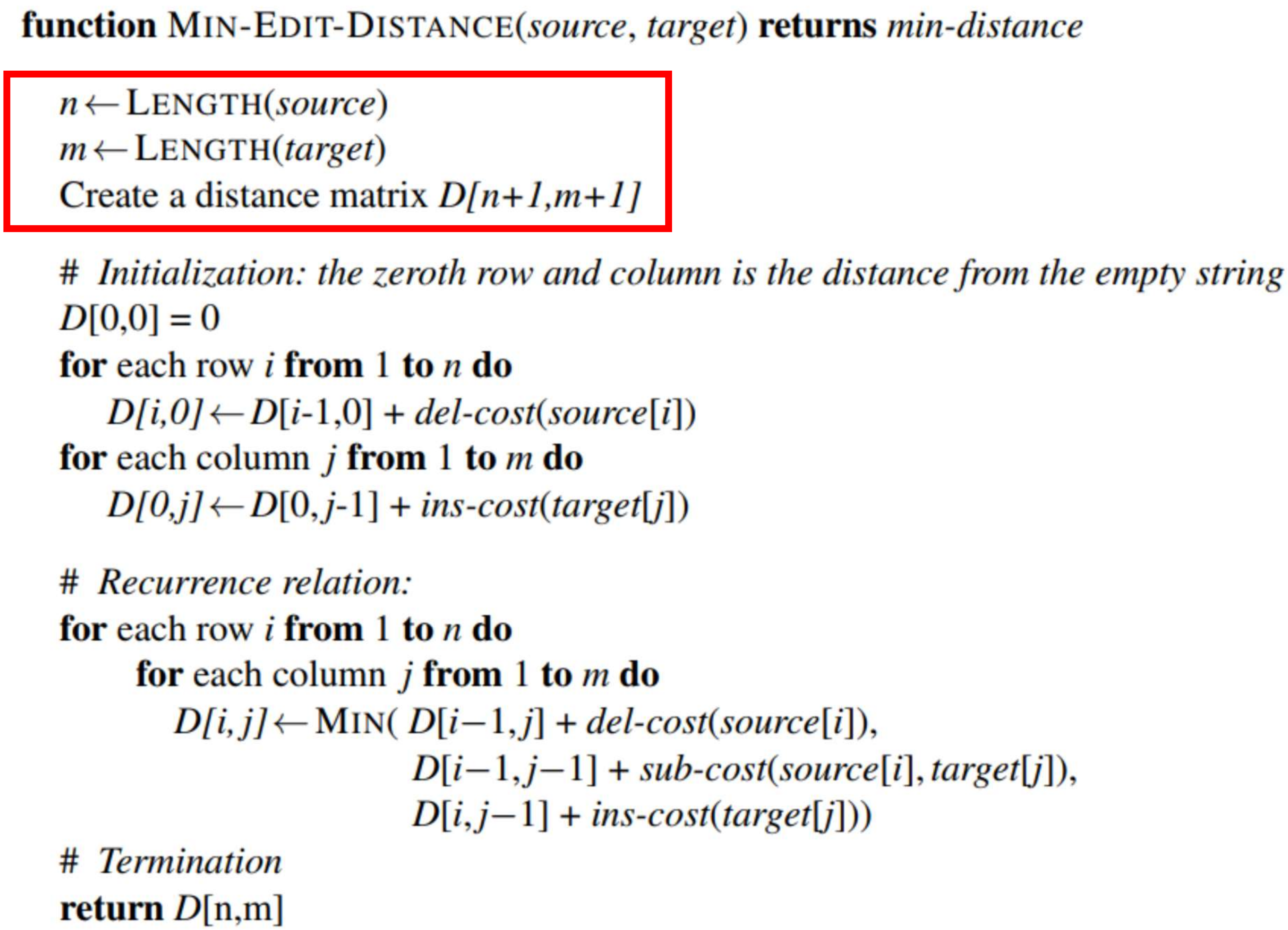
- Dynamic programming: A tabular computation of D(n, m)
  - Solving problems by combining solutions to subproblems.
- Bottom-up approach
  - we compute D(i ,j) for small i, j
  - and then compute larger D(i, j) based on previously computed smaller values
    - i.e., compute D(i, j) for all i (0 < i < n) and j (0 < j < m)

Minimum Edit Distance: Pseudocode



```
function MIN-EDIT-DISTANCE(source, target) returns min-distance

  $n \leftarrow$ LENGTH(source)
  $m \leftarrow$ LENGTH(target)
  Create a distance matrix $D[n+1,m+1]$

  # Initialization: the zeroth row and column is the distance from the empty string
  $D[0,0] = 0$
  for each row i from 1 to n do
     $D[i,0] \leftarrow D[i\text{-}1,0]$ + del-cost(source[i])
  for each column j from 1 to m do
     $D[0,j] \leftarrow D[0,j\text{-}1]$ + ins-cost(target[j])

  # Recurrence relation:
  for each row i from 1 to n do
       for each column j from 1 to m do
          $D[i,j] \leftarrow$ MIN( $D[i-1,j]$ + del-cost(source[i]),
                           $D[i-1,j-1]$ + sub-cost(source[i], target[j]),
                           $D[i,j-1]$ + ins-cost(target[j]))
  # Termination
  return D[n,m]
```

Minimum Edit Distance: Pseudocode



```
function MIN-EDIT-DISTANCE(source, target) returns min-distance

    $n \leftarrow$ LENGTH(source)
    $m \leftarrow$ LENGTH(target)
    Create a distance matrix $D[n+1,m+1]$

    # Initialization: the zeroth row and column is the distance from the empty string
    $D[0,0] = 0$
    for each row i from 1 to n do
        $D[i,0] \leftarrow D[i-1,0]$ + del-cost(source[i])
    for each column j from 1 to m do
        $D[0,j] \leftarrow D[0,j-1]$ + ins-cost(target[j])

    # Recurrence relation:
    for each row i from 1 to n do
          for each column j from 1 to m do
             $D[i,j] \leftarrow$ MIN( $D[i-1,j]$ + del-cost(source[i]),
                                $D[i-1,j-1]$ + sub-cost(source[i], target[j]),
                                $D[i,j-1]$ + ins-cost(target[j]))
    # Termination
    return D[n,m]
```

Distance Matrix (m+1 x n+1): Setup

source string (m characters)	m	???	???	???	???	???	???	???	???	???
	m-1	???	???	???	???	???	???	???	???	???
	m-2	???	???	???	???	???	???	???	???	???
	m-3	???	???	???	???	???	???	???	???	???
	…	???	???	???	???	???	???	???	???	???
	…	???	???	???	???	???	???	???	???	???
	3	???	???	???	???	???	???	???	???	???
	2	???	???	???	???	???	???	???	???	???
	1	???	???	???	???	???	???	???	???	???
#	0	1	2	3	…	…	n-3	n-2	n-1	n
	#	target string (n characters)								

# - empty string

Distance Matrix: Levenshtein Distance

source string (m characters)	m	???	???	???	???	???	???	???	???	???	???
	m-1	???	???	???	???	???	???	???	???	???	???
	m-2	???	???	???	???	???	???	???	???	???	???
	m-3	???	???	???	???	???	???	???	???	???	???
	…	???	???	???	???	???	???	???	???	???	???
	…	???	???	???	???	???	???	???	???	???	???
	3	???	???	???	???	???	???	???	???	???	???
	2	???	???	???	???	???	???	???	???	???	???
	1	???	???	???	???	???	???	???	???	???	???
#	0	1	2	3	…	…	n-3	n-2	n-1	n
	#	target string (n characters)									

$$distance[i,j] = min\begin{cases} distance[i-1,j] + insertionCost(target_{i-1}) \\ distance[i-1,j-1] + substitutionCost(source_{j-1}, target_{i-1}) \\ distance[i,j-1] + deletionCost(source_{j-1}) \end{cases}$$

Distance Matrix: Levenshtein Distance

source string (m characters)	m	???	???	???	???	???	???	???	???	???	???
	m-1	???	???	???	???	???	???	???	???	???	???
	m-2	???	???	???	???	???	???	???	???	???	???
	m-3	???	???	???	???	???	???	???	???	???	???
	…	???	???	???	???	???	???	???	???	???	???
	…	???	???	???	???	???	???	???	???	???	???
	3	???	???	???	???	???	???	???	???	???	???
	2	???	???	???	???	???	???	???	???	???	???
	1	???	???	???	???	???	???	???	???	???	???
#	0	1	2	3	…	…	n-3	n-2	n-1	n	
	#	target string (n characters)									

$$distance[col,row] = min\begin{cases} distance[col-1,row] + insertionCost(target_{col-1}) \\ distance[col-1,row-1] + substitutionCost(source_{row-1}, target_{col-1}) \\ distance[col,row-1] + deletionCost(source_{row-1}) \end{cases}$$

Distance Matrix: Levenshtein Distance

source string (m characters)	m	???	???	???	???	???	???	???	???	???	???
	m-1	???	???	???	???	???	???	???	???	???	???
	m-2	???	???	???	???	???	???	???	???	???	???
	m-3	???	???	???	???	???	???	???	???	???	???
	…	???	???	???	???	???	???	???	???	???	???
	…	???	???	???	???	???	???	???	???	???	???
	3	???	???	???	???	???	???	???	???	???	???
	2	???	???	???	???	???	???	???	???	???	???
	1	???	???	???	???	???	???	???	???	???	???
#	0	1	2	3	…	…	n-3	n-2	n-1	n	
	#	target string (n characters)									

$$distance[i,j] = min \begin{cases} distance[i-1,j]+1 \\ distance[i-1,j-1]+2 \\ distance[i,j-1]+1 \end{cases}$$

2 if different characters
0 if same characters

Edit Distance Matrix: Calculations

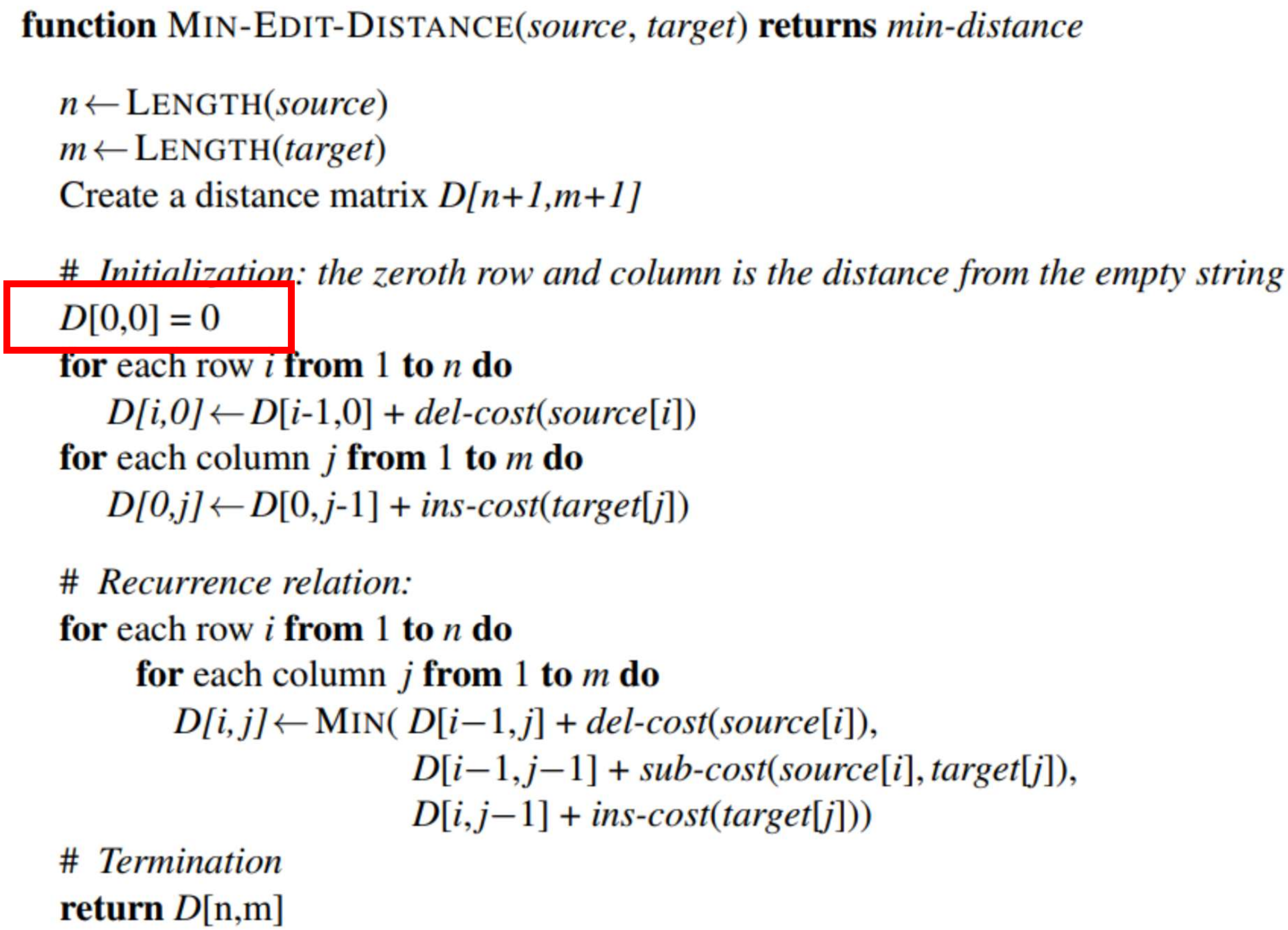
source string (m characters)	m	???	???	???	???	???	???	???	???	???	???
	m-1	???	???	???	???	???	???	???	???	???	???
	m-2	???	???	???	???	???	???	???	???	???	???
	m-3	???	???	???	???	???	???	???	???	???	???
	…	???	???	???	???	???	???	???	???	???	???
	…	???	???	???	???	???	???	???	???	???	???
	3	???	???	???	???	???	???	???	???	???	???
	2	???	???	???	???	???	???	???	???	???	???
	1	???	???	???	???	???	???	???	???	???	???
#	0	1	2	3	…	…	n-3	n-2	n-1	n	
	#	target string (n characters)									

□ ↑ □ - **insertion**

□ ↗ □ - **substitution**

□ ↑ □ - **deletion**

Minimum Edit Distance: Pseudocode

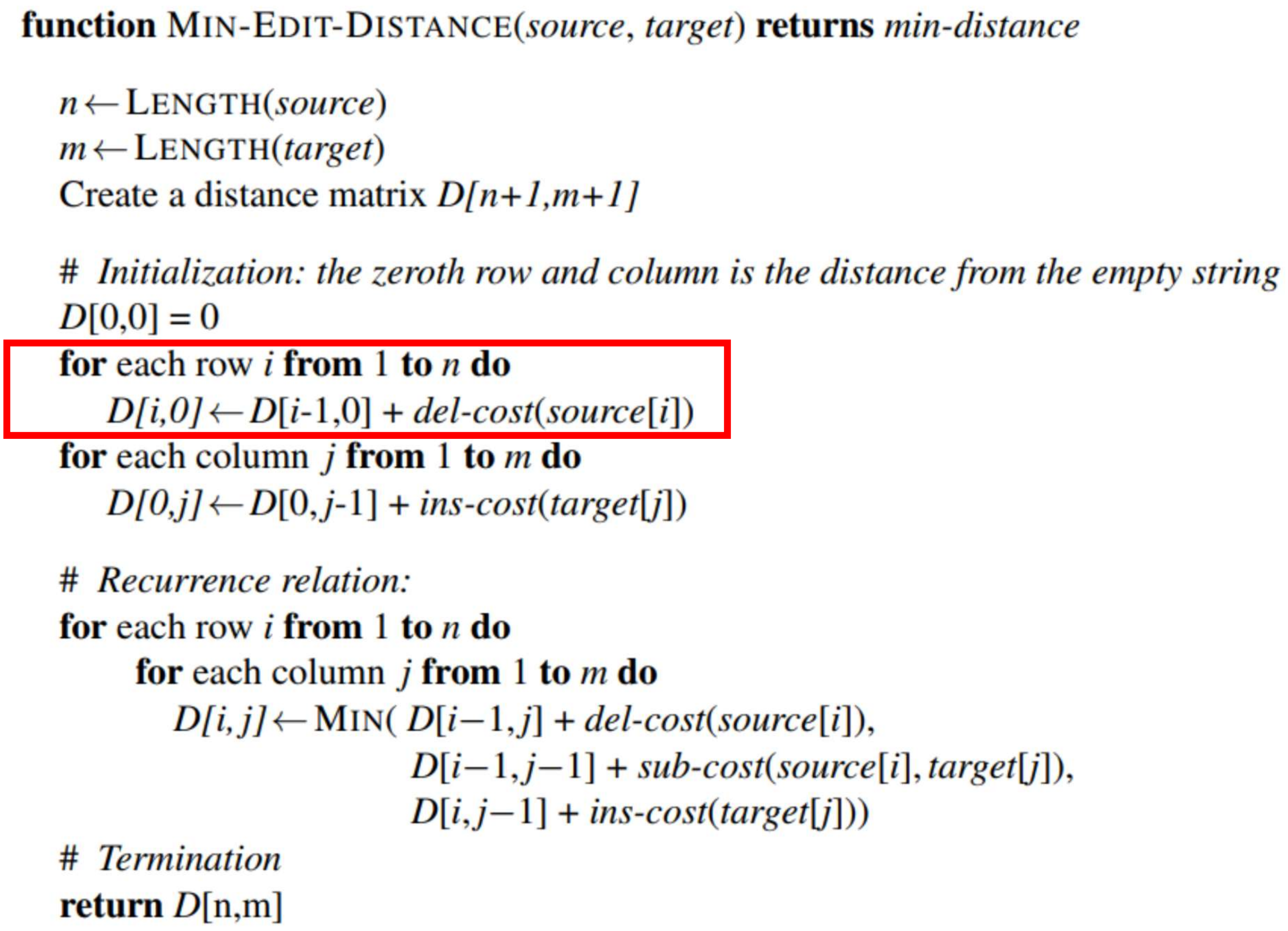


```
function MIN-EDIT-DISTANCE(source, target) returns min-distance

  $n \leftarrow$ LENGTH(source)
  $m \leftarrow$ LENGTH(target)
  Create a distance matrix $D[n+1,m+1]$

  # Initialization: the zeroth row and column is the distance from the empty string
  $D[0,0] = 0$
  for each row $i$ from 1 to $n$ do
      $D[i,0] \leftarrow D[i\text{-}1,0]$ + del-cost(source[$i$])
  for each column $j$ from 1 to $m$ do
      $D[0,j] \leftarrow D[0,j\text{-}1]$ + ins-cost(target[$j$])

  # Recurrence relation:
  for each row $i$ from 1 to $n$ do
        for each column $j$ from 1 to $m$ do
           $D[i,j] \leftarrow$ MIN( $D[i-1,j]$ + del-cost(source[$i$]),
                                $D[i-1,j-1]$ + sub-cost(source[$i$], target[$j$]),
                                $D[i,j-1]$ + ins-cost(target[$j$]))
  # Termination
  return $D$[n,m]
```

Edit Distance Matrix: Initialization 1

n	???	???	???	???	???	???	???	???	???	???
o	???	???	???	???	???	???	???	???	???	???
i	???	???	???	???	???	???	???	???	???	???
t	???	???	???	???	???	???	???	???	???	???
n	???	???	???	???	???	???	???	???	???	???
e	???	???	???	???	???	???	???	???	???	???
t	???	???	???	???	???	???	???	???	???	???
n	???	???	???	???	???	???	???	???	???	???
i	???	???	???	???	???	???	???	???	???	???
#	0	???	???	???	???	???	???	???	???	???
	#	e	x	e	c	u	t	i	o	n

Minimum Edit Distance: Pseudocode

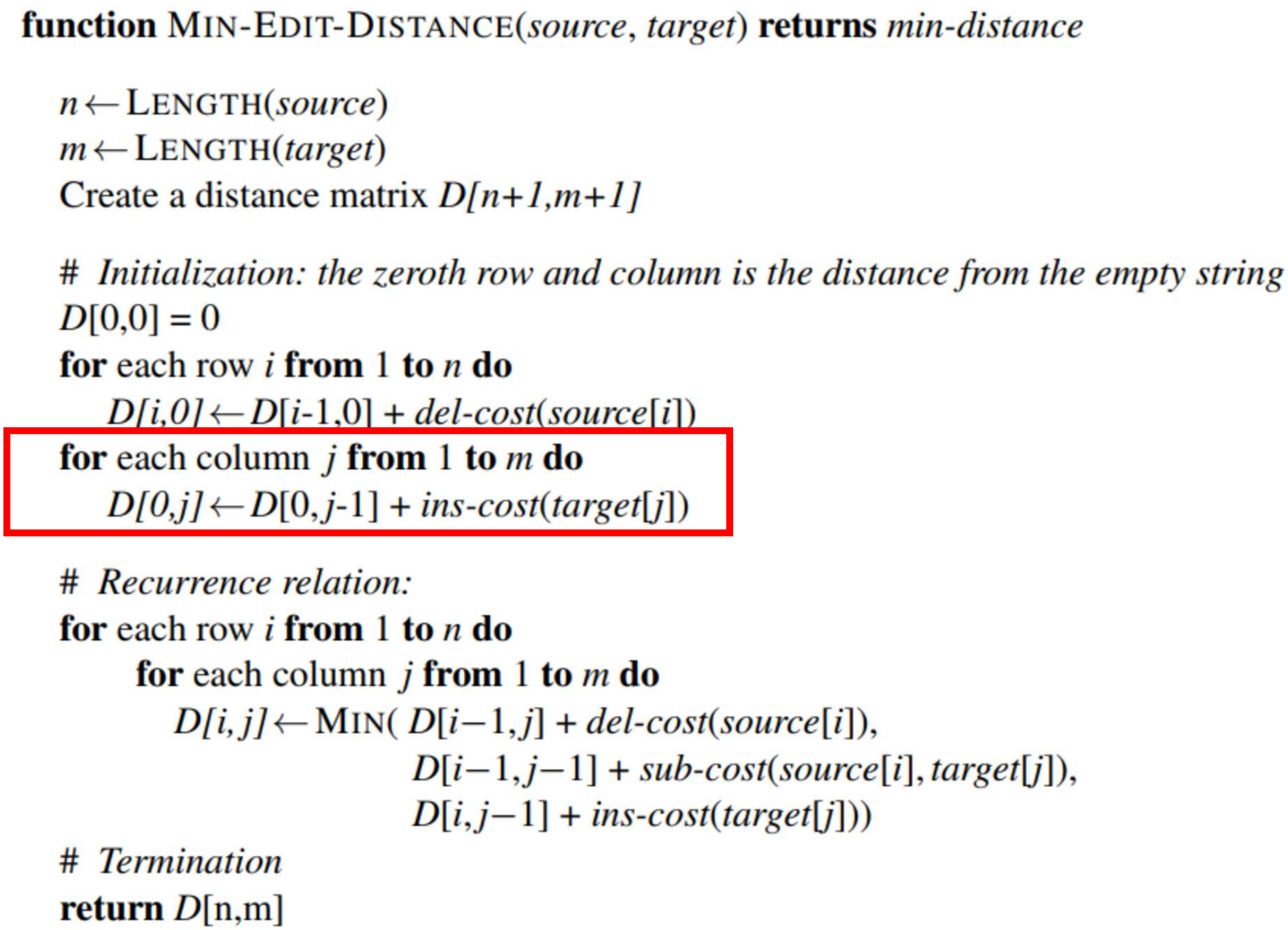


```
function MIN-EDIT-DISTANCE(source, target) returns min-distance

  $n \leftarrow$ LENGTH(source)
  $m \leftarrow$ LENGTH(target)
  Create a distance matrix $D[n+1,m+1]$

  # Initialization: the zeroth row and column is the distance from the empty string
  $D[0,0] = 0$
  for each row i from 1 to n do
     $D[i,0] \leftarrow D[i-1,0]$ + del-cost(source[i])
  for each column j from 1 to m do
     $D[0,j] \leftarrow D[0,j-1]$ + ins-cost(target[j])

  # Recurrence relation:
  for each row i from 1 to n do
        for each column j from 1 to m do
           $D[i,j] \leftarrow$ MIN( $D[i-1,j]$ + del-cost(source[i]),
                             $D[i-1,j-1]$ + sub-cost(source[i], target[j]),
                             $D[i,j-1]$ + ins-cost(target[j]))
  # Termination
  return D[n,m]
```

Edit Distance Matrix: Initialization 2

n	9	???	???	???	???	???	???	???	???	???
o	8	???	???	???	???	???	???	???	???	???
i	7	???	???	???	???	???	???	???	???	???
t	6	???	???	???	???	???	???	???	???	???
n	5	???	???	???	???	???	???	???	???	???
e	4	???	???	???	???	???	???	???	???	???
t	3	???	???	???	???	???	???	???	???	???
n	2	???	???	???	???	???	???	???	???	???
i	1	???	???	???	???	???	???	???	???	???
#	0	???	???	???	???	???	???	???	???	???
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Minimum Edit Distance: Pseudocode

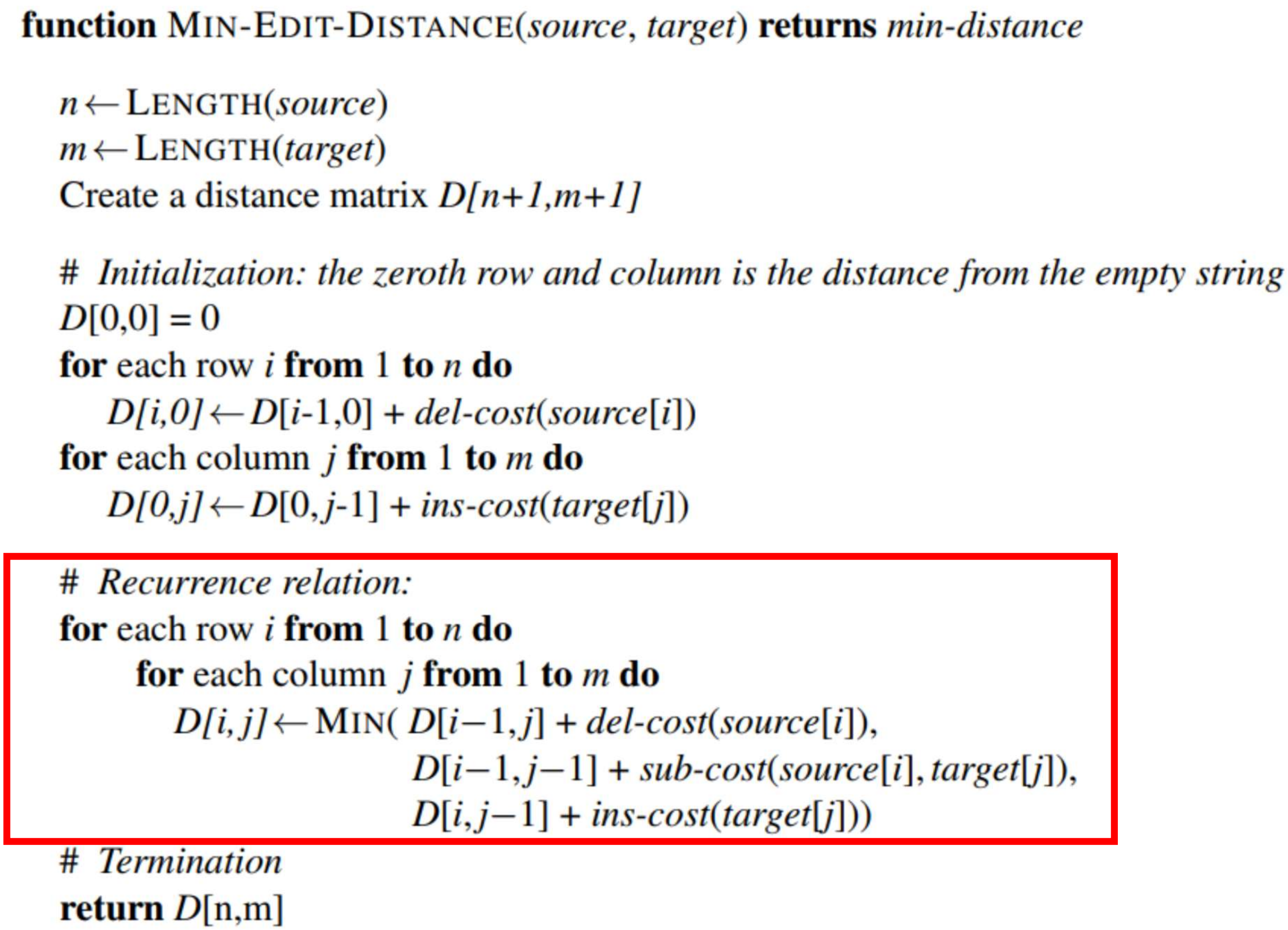


```
function MIN-EDIT-DISTANCE(source, target) returns min-distance

  $n \leftarrow$ LENGTH(source)
  $m \leftarrow$ LENGTH(target)
  Create a distance matrix $D[n+1,m+1]$

  # Initialization: the zeroth row and column is the distance from the empty string
  $D[0,0] = 0$
  for each row $i$ from 1 to $n$ do
      $D[i,0] \leftarrow D[i\text{-}1,0] + \textit{del-cost}(\textit{source}[i])$
  for each column $j$ from 1 to $m$ do
      $D[0,j] \leftarrow D[0,j\text{-}1] + \textit{ins-cost}(\textit{target}[j])$

  # Recurrence relation:
  for each row $i$ from 1 to $n$ do
        for each column $j$ from 1 to $m$ do
           $D[i,j] \leftarrow \text{MIN}(\ D[i-1,j] + \textit{del-cost}(\textit{source}[i]),$
                                  $D[i-1,j-1] + \textit{sub-cost}(\textit{source}[i],\textit{target}[j]),$
                                  $D[i,j-1] + \textit{ins-cost}(\textit{target}[j]))$
  # Termination
  return $D$[n,m]
```

Edit Distance Matrix: Initialization 3

n	9	???	???	???	???	???	???	???	???	???
o	8	???	???	???	???	???	???	???	???	???
i	7	???	???	???	???	???	???	???	???	???
t	6	???	???	???	???	???	???	???	???	???
n	5	???	???	???	???	???	???	???	???	???
e	4	???	???	???	???	???	???	???	???	???
t	3	???	???	???	???	???	???	???	???	???
n	2	???	???	???	???	???	???	???	???	???
i	1	???	???	???	???	???	???	???	???	???
#	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Minimum Edit Distance: Pseudocode



```
function MIN-EDIT-DISTANCE(source, target) returns min-distance

  $n \leftarrow$ LENGTH(source)
  $m \leftarrow$ LENGTH(target)
  Create a distance matrix $D[n+1,m+1]$

  # Initialization: the zeroth row and column is the distance from the empty string
  $D[0,0] = 0$
  for each row i from 1 to n do
      $D[i,0] \leftarrow D[i\text{-}1,0]$ + del-cost(source[i])
  for each column j from 1 to m do
      $D[0,j] \leftarrow D[0,j\text{-}1]$ + ins-cost(target[j])

  # Recurrence relation:
  for each row i from 1 to n do
       for each column j from 1 to m do
          $D[i,j] \leftarrow$ MIN( $D[i-1,j]$ + del-cost(source[i]),
                               $D[i-1,j-1]$ + sub-cost(source[i], target[j]),
                               $D[i,j-1]$ + ins-cost(target[j]))
  # Termination
  return D[n,m]
```

Edit Distance Matrix: Populate

n	9	???	???	???	???	???	???	???	???	???
o	8	???	???	???	???	???	???	???	???	???
i	7	???	???	???	???	???	???	???	???	???
t	6	???	???	???	???	???	???	???	???	???
n	5	???	???	???	???	???	???	???	???	???
e	4	???	???	???	???	???	???	???	???	???
t	3	???	???	???	???	???	???	???	???	???
n	2	???	???	???	???	???	???	???	???	???
i	1	**0 + 2 = 2**	???	???	???	???	???	???	???	???
#	0	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

$$distance[i,j] = min\begin{cases} distance[i-1,j] + insertionCost(target_{i-1}) \\ distance[i-1,j-1] + substitutionCost(source_{j-1}, target_{i-1}) \\ distance[i,j-1] + deletionCost(source_{j-1}) \end{cases}$$

Edit Distance Matrix: Populate

n	9	8	9	10	11	12	11	10	9	8
o	8	7	8	9	10	11	10	9	8	9
i	7	6	7	8	9	10	9	8	9	10
t	6	5	6	7	8	9	8	9	10	11
n	5	4	5	6	7	8	9	10	11	10
e	4	3	4	5	6	7	8	9	10	9
t	3	4	5	6	7	8	7	8	9	8
n	2	3	4	5	6	7	8	7	8	7
i	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

$$distance[i,j] = min \begin{cases} distance[i-1,j] + insertionCost(target_{i-1}) \\ distance[i-1,j-1] + substitutionCost(source_{j-1}, target_{i-1}) \\ distance[i,j-1] + deletionCost(source_{j-1}) \end{cases}$$

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Idea: while populating, add “pointers” (↓←↙) to indicate which cell did we come from. Use pointers to “backtrace” by following the minimum edit path.

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

↓←↙ - which cell did we come from?

red - minimum edit cost

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

↓←↙ - which cell did we come from?

red - minimum edit cost

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

↓←↙ - which cell did we come from?

red - minimum edit cost

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

↓←↙ - which cell did we come from?

red - minimum edit cost

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

↓←↙ - which cell did we come from?

red - minimum edit cost

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

↓←↙ - which cell did we come from?

red - minimum edit cost

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

↓←↙ - which cell did we come from?

red - minimum edit cost

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

↓←↙ - which cell did we come from?

red - minimum edit cost

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

↓←↙ - which cell did we come from?

red - minimum edit cost

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

↓←↙ - which cell did we come from?

red - minimum edit cost

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	↙8
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	↙8	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	↙8	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙8	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	↙←5	←6	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	1	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	0	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

↓←↙ - which cell did we come from?

red - minimum edit cost

Minimum Edit Path with Backtrace

n	9	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	**↙8**
o	8	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	**↙8**	←9
i	7	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	**↙8**	←9	←10
t	6	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	**↙8**	←9	←10	←↓11
n	5	↓4	↙←↓5	↙←↓6	↙←↓7	**↙←↓8**	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	4	↙3	←4	**↙←5**	**←6**	←7	←↓8	↙←↓9	↙←↓10	↓9
t	3	↙←↓4	**↙←↓5**	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	2	**↙←↓3**	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	**1**	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
#	**0**	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Final minimum edit path.

Time and Space Complexity

- Time:

$$O(n * m)$$

- Space:

$$O(n * m)$$

- Backtrace time complexity:

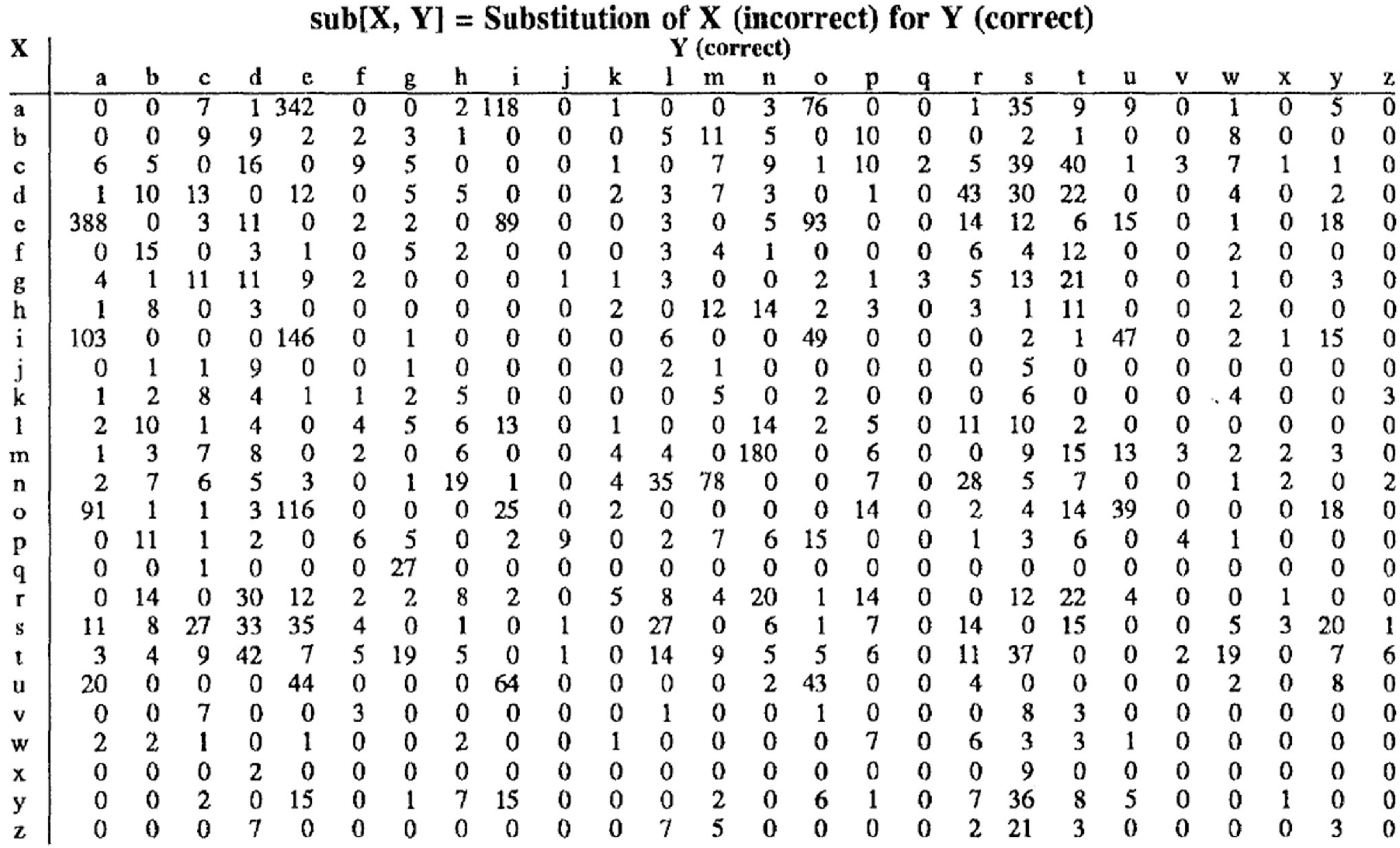
$$O(n + m)$$

Weighted Edit Distance

- Why would we add weights to the computation?
- Spell Correction:
  - some letters are more likely to be mistyped than others
- Biology:
  - certain kinds of deletions or insertions are more likely than others

Weighted Edit Distance

Confusion matrix for spelling errors:



sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X	Y (correct)																									
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

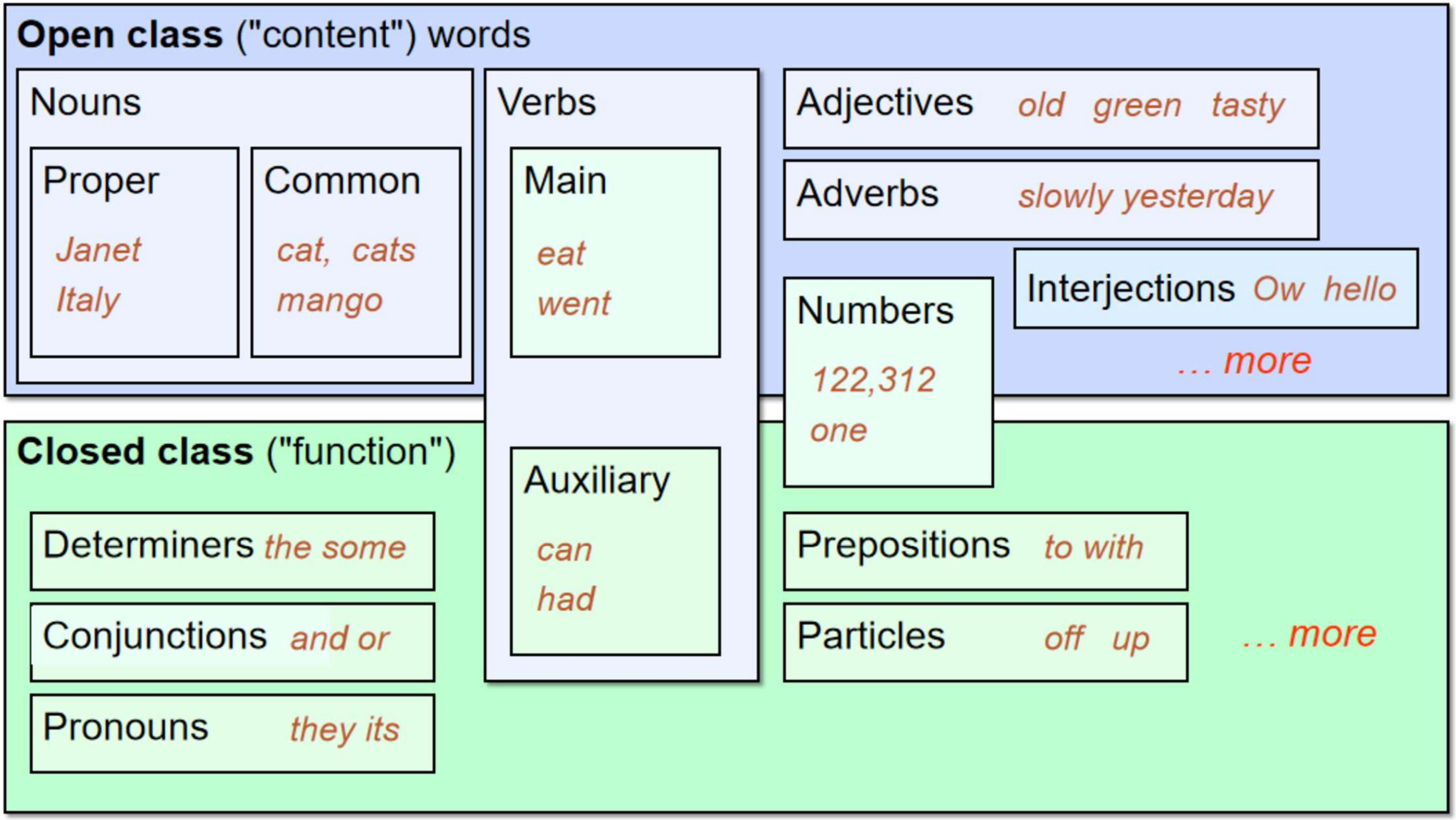
Parts of Speech

- Idea:
  - classify words according to their grammatical categories
- Categories = part of speech, word classes, POS, POS tags
- Basic categories / tags:
  - noun, verb, pronoun, preposition, adverb, conjunction, participle, article

Parts of Speech: Closed vs. Open

- Closed class:
  - relatively fixed set - new members rarely added
  - usually function words: short, frequent words with grammatical function:
    - determiners: *a*, *an*, *the*
    - pronouns: *she*, *he*, *I*
    - prepositions: *on*, *under*, *over*, *near*, ***by***
- Open class:
  - word sets where new members are constantly created
  - usually content words: nouns, verbs, adjectives, adverbs
  - new words | examples: nouns (*iPhone*), verbs (*to google*)

Parts of Speech: Closed vs. Open



Open class ("content") words

Nouns

Proper *Janet Italy*

Common *cat, cats mango*

Verbs

Main *eat went*

Auxiliary *can had*

Adjectives *old green tasty*

Adverbs *slowly yesterday*

Numbers *122,312 one*

Interjections *Ow hello*

… more

Closed class ("function")

Determiners *the some*

Conjunctions *and or*

Pronouns *they its*

Prepositions *to with*

Particles *off up*

… more

Parts of Speech Tagging

- **Assigning a part-of-speech (POS) to each word in a text.**
- **Words often have more than one POS.**
  - **example:** *book*
    - **VERB:** ***Book*** *that flight*
    - **NOUN:** *Hand me that* ***book***

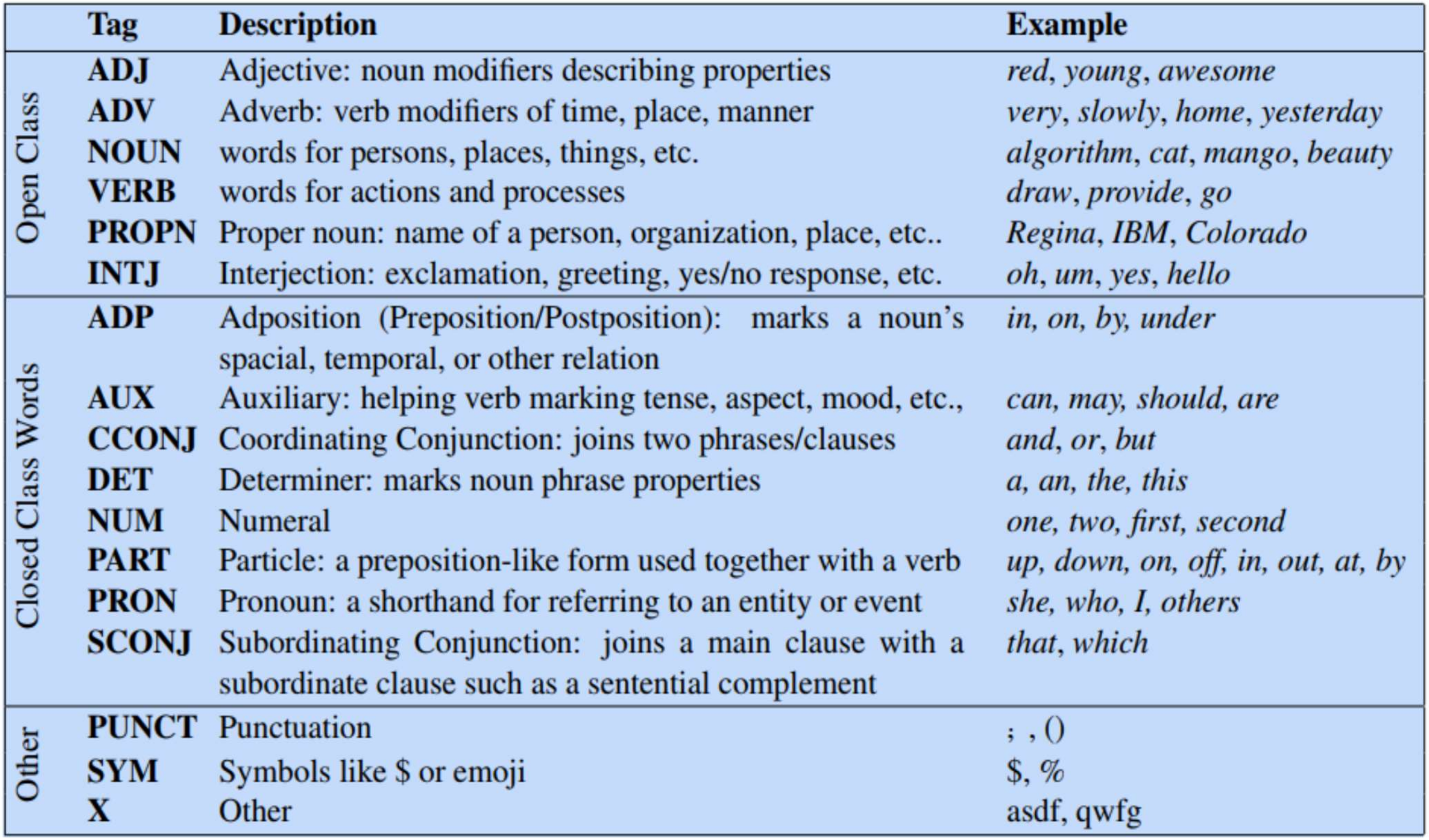
Sample Tagged Sentence

There/PRO were/VERB 70/NUM children/NOUN there/ADV ./PUNC

Preliminary/ADJ findings/NOUN were/AUX reported/VERB in/ADP today/NOUN ’s/PART New/PROPN England/PROPN Journal/PROPN of/ADP Medicine/PROPN

Parts of Speech: Tagset Example

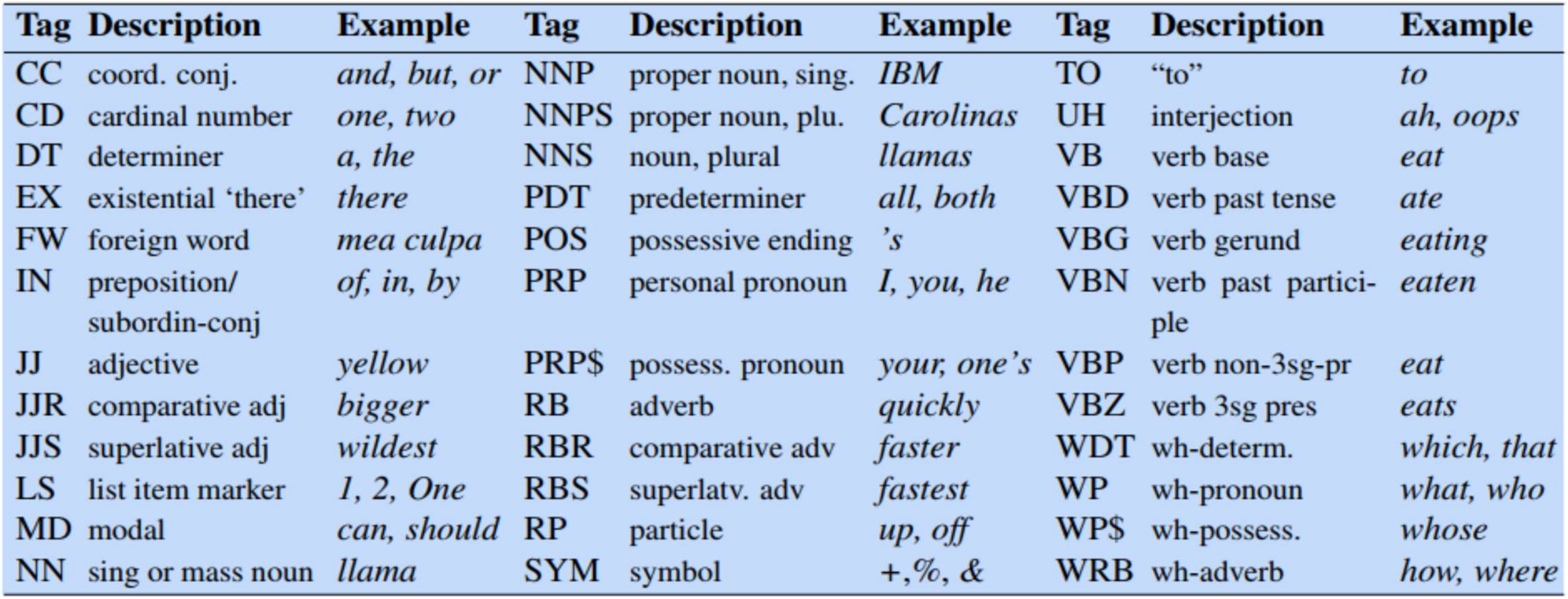
Parts of Speech in the Universal Dependencies tagset



	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red, young, awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very, slowly, home, yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm, cat, mango, beauty*
	VERB	words for actions and processes	*draw, provide, go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina, IBM, Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh, um, yes, hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by, under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and, or, but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that, which*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

Parts of Speech: Tagset Example

Penn Treebank Parts-of-speech tags:



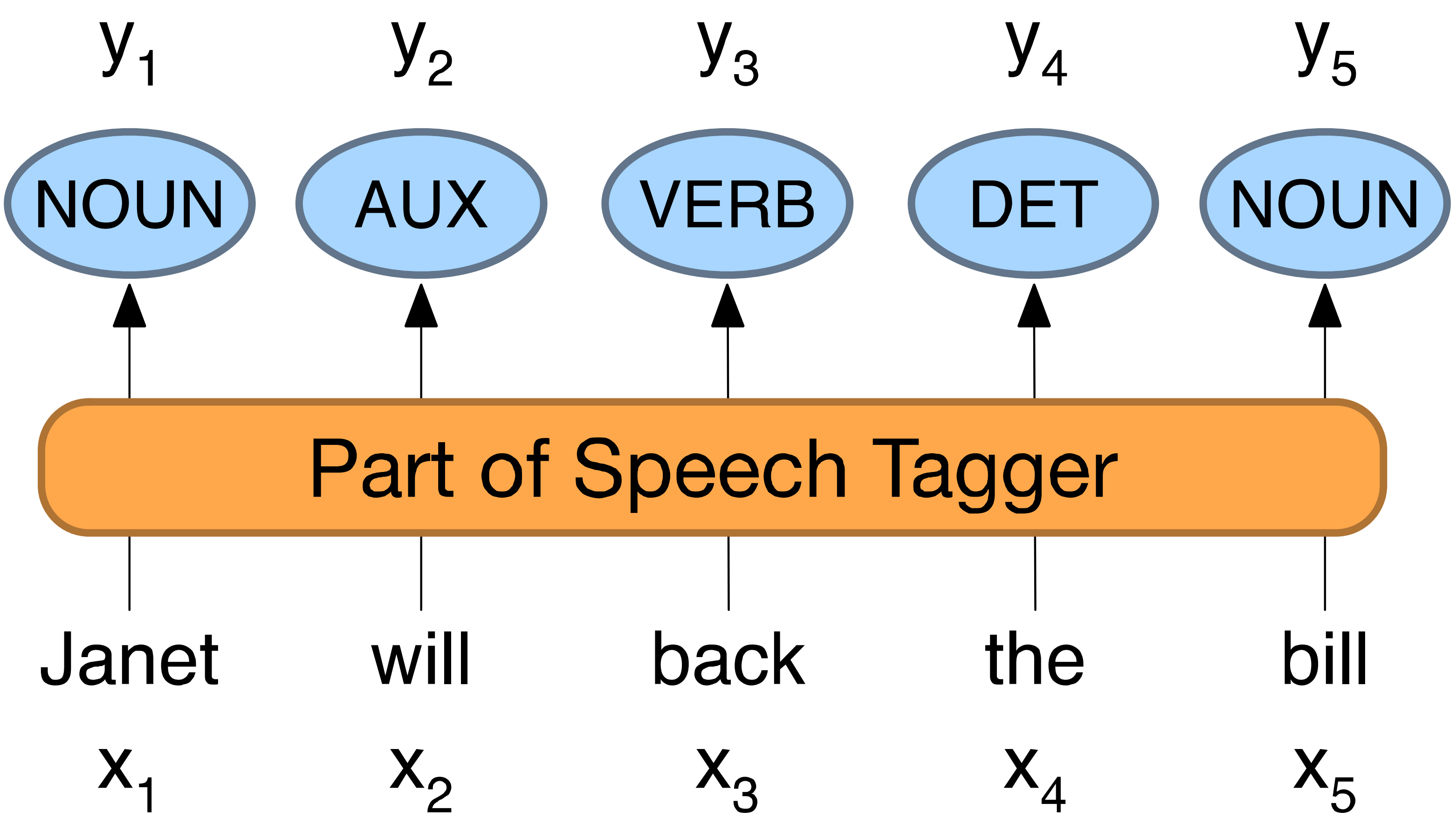
Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	“to”	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential ‘there’	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*’s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past participle	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one’s*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	*+,%, &*	WRB	wh-adverb	*how, where*

Parts of Speech Tagging: Motivation

- Can be useful for other NLP tasks
  - Parsing: POS tagging can improve syntactic parsing
  - MT: reordering of adjectives and nouns (say from Spanish to English)
  - Sentiment or affective tasks: may want to distinguish adjectives or other POS
  - Text-to-speech (how do we pronounce “*lead*” or "***object***"?)
- Or linguistic or language-analytic computational tasks
  - Need to control for POS when studying linguistic change like creation of new words, or meaning shift
  - Or control for POS in measuring meaning similarity or difference

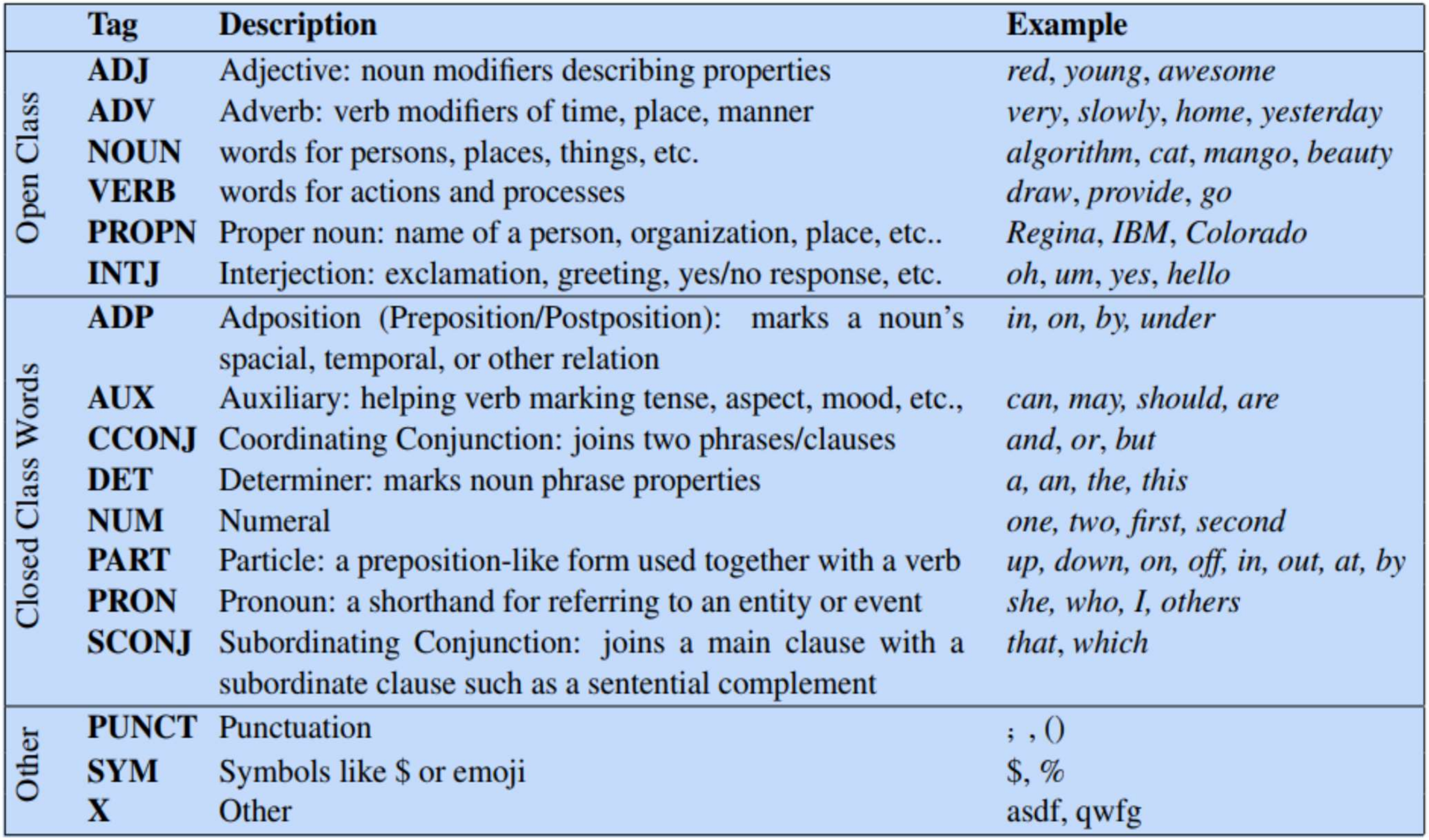
Part of Speech Tagging

- **Task:**
  - **Map sequence $x_1,\ldots,x_n$ of words to $y_1,\ldots,y_n$ of POS tags**



Parts of Speech: Tagset Example

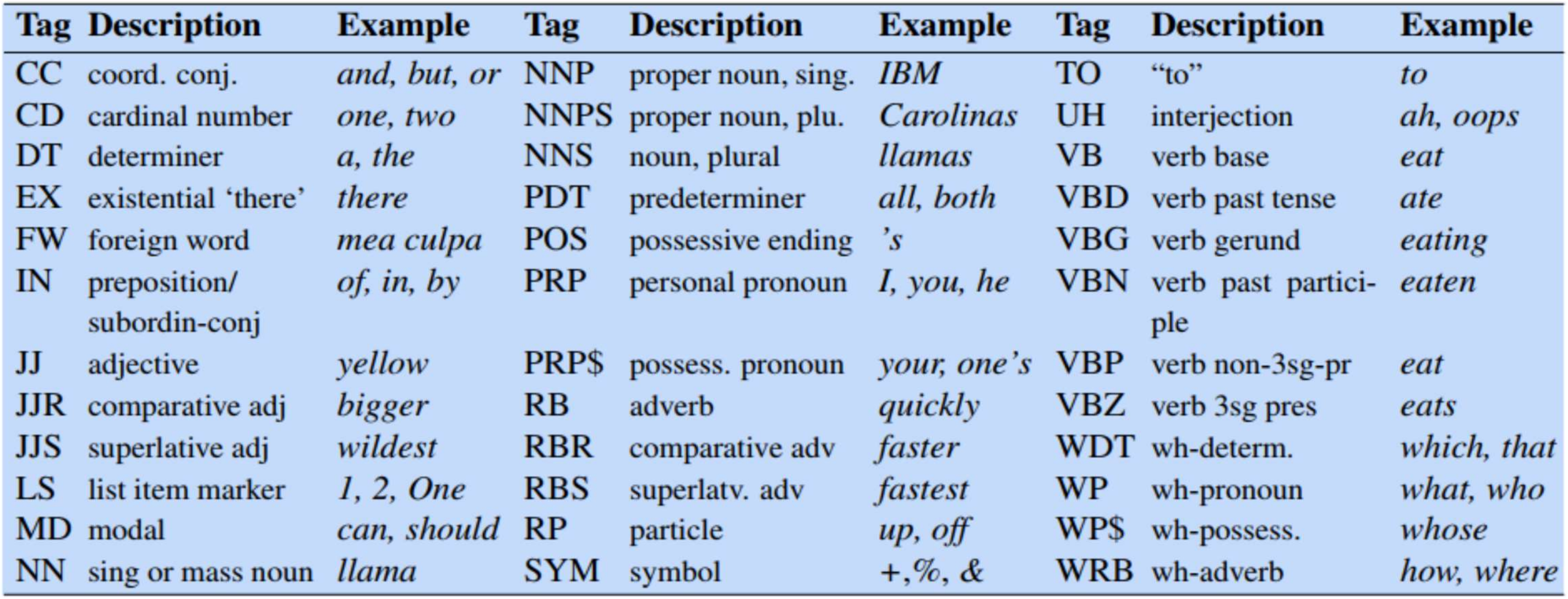
Parts of Speech in the Universal Dependencies tagset



	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red, young, awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very, slowly, home, yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm, cat, mango, beauty*
	VERB	words for actions and processes	*draw, provide, go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina, IBM, Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh, um, yes, hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by, under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and, or, but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that, which*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

Parts of Speech: Tagset Example

Penn Treebank Parts-of-speech tags:



Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	"to"	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past participle	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one's*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	*+,%, &*	WRB	wh-adverb	*how, where*

Sample Tagged Sentence

There/PRO were/VERB 70/NUM children/NOUN there/ADV ./PUNC

Preliminary/ADJ findings/NOUN were/AUX reported/VERB in/ADP today/NOUN ’s/PART New/PROPN England/PROPN Journal/PROPN of/ADP Medicine/PROPN

POS Tagging: Challenges

- **Roughly 15% of word types are ambiguous**
  - **Hence 85% of word types are unambiguous**
  - ***Janet* is always PROPN, *hesitantly* is always ADV**
- **But those 15% tend to be very common.**

- **Effectively around ~60% of word tokens are ambiguous**
  - **For example: *back***
    - **earnings growth took a *back*/ADJ seat**
    - **a small building in the *back*/NOUN**
    - **a clear majority of senators *back*/VERB the bill**
    - **enable the country to buy *back*/PART debt**
    - **I was twenty-one *back*/ADV then**

POS Tagging: Challenges

- How many tags are correct? (Tag accuracy)
  - about 97%
  - different taggers perform similarly, human accuracy about the same
- But baseline is 92%!
- Baseline is performance of stupidest possible method
  - "Most frequent class baseline" is an important baseline for many tasks
    - tag every word with its most frequent tag
    - (and tag unknown words as nouns)
  - Partly easy because many words are unambiguous

Sources of Tagging Information

Janet will back the bill

AUX/NOUN/VERB? **NOUN/VERB?**

- **Prior probabilities of word/tag**
  - **"*will*" is usually an AUX**
- **Identity of neighboring words**
  - **"*the*" means the next word is probably not a verb**
- **Morphology and wordshape:**

Prefixes	*unable*:	*un-* ⍰	**ADJ**
Suffixes	*importantly*:	*-ly* ⍰	**ADJ**
Capitalization	*Janet*:	**CAP**	**PROPN**

Standard POS Tagging Models

- Supervised Machine Learning Algorithms:
  - Hidden Markov Models
  - Conditional Random Fields (CRF) / Maximum Entropy Markov Models (MEMM)
  - Neural sequence models (RNNs or Transformers)
  - Large Language Models (like BERT), finetuned
- All required a hand-labeled training set, all about equal performance (97% on English)
  - All make use of information sources we discussed
  - Via human created features: HMMs and CRFs
  - Via representation learning: Neural LMs

Conditional Probability

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

where $\mathrm{P}(\mathrm{B}) > 0$

Part of Speech: Conditional Probability

$$P(Category = NOUN \mid word\ =\ flies) = \frac{P(word\ =\ flies, Category = NOUN)}{P(Word\ = flies)}$$

where $\mathrm{P}(Word\ =\ flies) > 0$

Part of Speech: Conditional Probability

$$P(C = NOUN \mid w = flies) = \frac{P(w = flies, C = NOUN)}{P(w = flies)}$$

where $\mathrm{P}(w = flies) > 0$

Most Frequent Class Tagging

- Basic idea: calculate conditional probabilities for all possible categories and compare.
- For simplicity, let's assume only two lexical categories: NOUN and VERB.

$$P(C = NOUN \mid w = flies) = \frac{P(w = flies, C = NOUN)}{P(w = flies)}$$

vs.

$$P(C = VERB \mid w = flies) = \frac{P(w = flies, C = VERB)}{P(w = flies)}$$

Most Frequent Class Tagging

Say we have some corpus with 1 273 000 words.

The word *flies* appears 1000 times in the corpus:

- 400 times as a NOUN
- 600 times as a VERB

So:

$$\boldsymbol{P(w\ =\ flies) = 1000/1\,273\,000\ =\ 0.0008}$$
$$\boldsymbol{P(w\ = flies, C = NOUN) = 400/1\,273\,000 = 0.0003}$$
$$\boldsymbol{P(w\ = flies, C = VERB) = 600/1\,273\,000 = 0.0005}$$

Most Frequent Class Tagging

With:

$$P(w = flies) = 1000/1\,273\,000 = 0.0008$$

$$P(w = flies, C = NOUN) = 400/1\,273\,000 = 0.0003$$

$$P(w = flies, C = VERB) = 600/1\,273\,000 = 0.0005$$

$$P(C = NOUN \mid w = flies) = \frac{P(w = flies, C = NOUN)}{P(w = flies)} = \frac{0.0003}{0.0008}$$

vs.

$$P(C = VERB \mid w = flies) = \frac{P(w = flies, C = VERB)}{P(w = flies)} = \frac{0.0005}{0.0008}$$

With this approach *flies* will ALWAYS be tagged as a VERB.

Bayes' Rule

$$\boldsymbol{P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B)}}$$

POS Tagging: General Approach

Given a sequence of **words** (a “sentence”):

$$w_1, w_2, w_3, \ldots, w_T$$

there is going to be a corresponding sequence of lexical categories:

$$C_1, C_2, C_3, \ldots, C_T$$

What is most likely sequence of categories?

POS Tagging: General Approach

To answer this we would want to find a conditional probability:

$$P(C_1, C_2, C_3, \ldots, C_T \mid w_1, w_2, w_3, \ldots, w_T)$$

In other words: what is the probability of having a sequence of lexical categories

$$C_1, C_2, C_3, \ldots, C_T$$

GIVEN that the sequence of words is

$$w_1, w_2, w_3, \ldots, w_T\ ?$$

POS Tagging: General Approach

The probability we are looking for

$$P(C_1, C_2, C_3, ..., C_T \mid w_1, w_2, w_3, ..., w_T)$$

will require a lot of data, which we most likely won't have.

We can use Bayes' Theorem:

$$P(C_1, C_2, C_3, ..., C_T \mid w_1, w_2, w_3, ..., w_T) =$$

$$= \frac{P(w_1, w_2, w_3, ..., w_T \mid C_1, C_2, C_3, ..., C_T) * P(C_1, C_2, C_3, ..., C_T)}{P(w_1, w_2, w_3, ..., w_T)}$$

POS Tagging: General Approach

In order to find the most likely sequence:

$$C_1, C_2, C_3, \ldots, C_T$$

we need to maximize (most likely sequence!):

$$P(C_1, C_2, C_3, \ldots, C_T \mid w_1, w_2, w_3, \ldots, w_T) =$$

$$= \frac{P(w_1, w_2, w_3, \ldots, w_T \mid C_1, C_2, C_3, \ldots, C_T) * (C_1, C_2, C_3, \ldots, C_T)}{P(w_1, w_2, w_3, \ldots, w_T)}$$

POS Tagging: General Approach

Maximizing :

$$P(C_1, C_2, C_3, \ldots, C_T \mid w_1, w_2, w_3, \ldots, w_T)$$

in practice means maximizing the numerator:

$$\frac{P(w_1, w_2, w_3, \ldots, w_T \mid C_1, C_2, C_3, \ldots, C_T) * P(C_1, C_2, C_3, \ldots, C_T)}{P(w_1, w_2, w_3, \ldots, w_T)}$$

as denominator $P(w_1, w_2, w_3, \ldots, w_T)$ will not change:

POS Tagging: General Approach

Estimating:

$$P(w_1, w_2, w_3, ..., w_T \mid C_1, C_2, C_3, ..., C_T) * P(C_1, C_2, C_3, ..., C_T)$$

using counts once again requires a lot of data that we will likely not have.

Alternative: approximate it with N-grams (here bigrams):

$$P(C_1, C_2, C_3, ..., C_T) = \prod_{i=1}^{T} P(C_i \mid all\ categories\ preceding\ C_i)$$

$$P(C_1, C_2, C_3, ..., C_T) \cong \prod_{i=1}^{T} P(C_i \mid C_{i-})$$

POS Tagging: General Approach

Estimating:

$$P(w_1, w_2, w_3, \ldots, w_T \mid C_1, C_2, C_3, \ldots, C_T) * P(C_1, C_2, C_3, \ldots, C_T)$$

Approximate it with N-grams (here bigrams):

$$P(C_1, C_2, C_3, \ldots, C_T) = \prod_{i=1}^{T} P(C_i \mid all\ categories\ preceding\ C_i)$$

$$P(C_1, C_2, C_3, \ldots, C_T) \cong \prod_{i=1}^{T} P(C_i \mid C_{i-1})$$

$$P(w_1, w_2, w_3, \ldots, w_T \mid C_1, C_2, C_3, \ldots, C_T) \cong \prod_{i=1}^{T} P(w_i \mid C_i)$$

POS Tagging: General Approach

With approximations:

$$P(w_1, w_2, w_3, ..., w_T \mid C_1, C_2, C_3, ..., C_T) * P(C_1, C_2, C_3, ..., C_T) \cong$$

$$\cong \prod_{i=1}^{T} P(w_i \mid C_i) * P(C_i \mid C_{i-1})$$

and we want to maximize:

$$\prod_{i=1}^{T} P(w_i \mid C_i) * P(C_i \mid C_{i-1})$$

Individual probabilities can now be estimated using corpus counts!

POS Tagging: Simple Tagset

Let’s assume we have a simple tagset:

- N - NOUN
- V - VERB
- ART - ARTICLE
- P - PREPOSITION

and a some synthetic corpus.

POS Tagging: General Approach

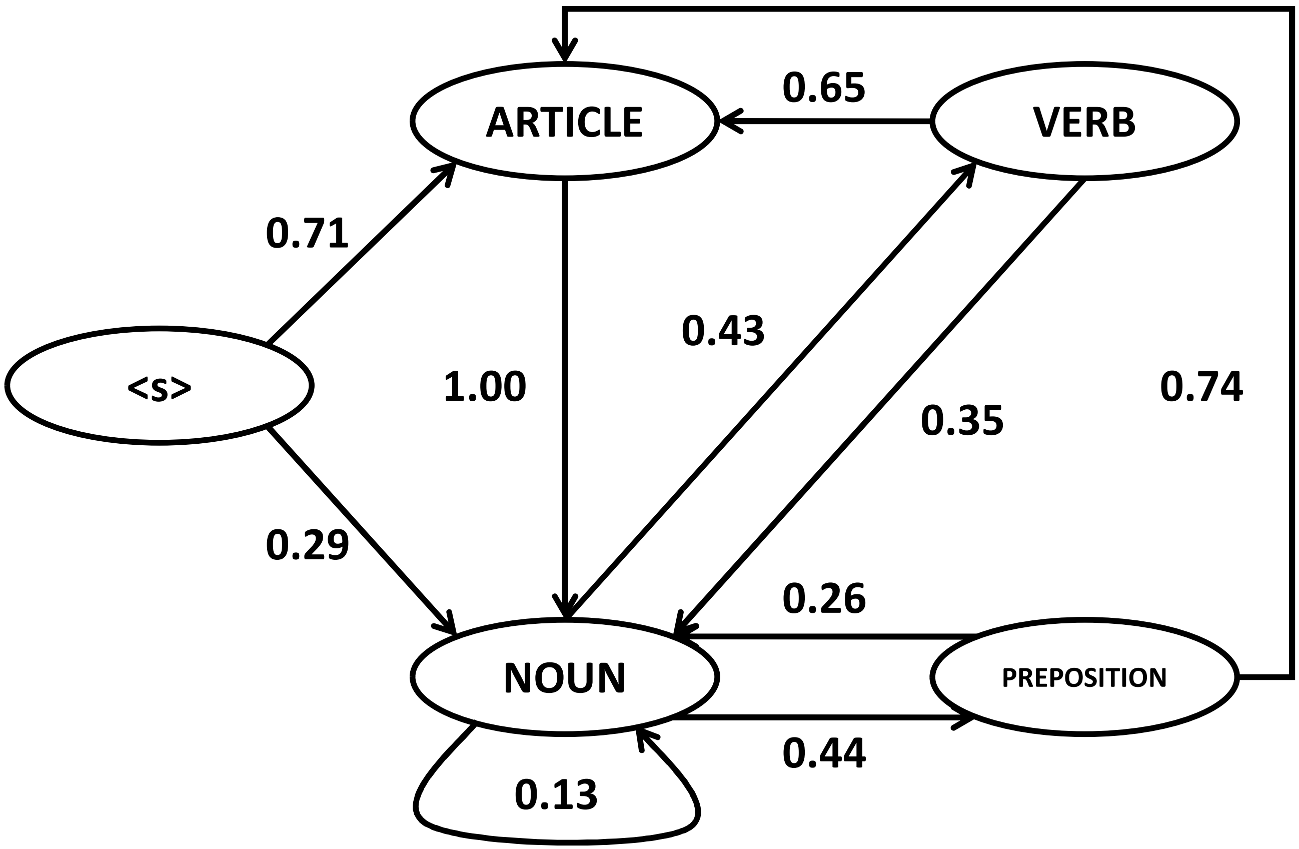
Estimations with corpus counts:

$$P(C_i \ = VERB \mid \ C_{i-1} = NOUN) = \frac{Count\ (NOUN\ at\ position\ i -\ \ and\ VERB\ at\ i)}{Count(NOUN\ at\ position\ i-)}$$

Sample bigram probabilities from our synthetic corpus:

Category	Count at i	Pair	Count at i,i+1	P(Bigram)	Estimate
<s>	300	<s>, ARTICLE	213	P(ARTICLE|<S>)	0.71
<s>	300	<s>, NOUN	87	P(NOUN|<S>)	0.29
ARTICLE	558	ARTICLE, NOUN	558	P(NOUN|ARTICLE)	1.00
NOUN	833	NOUN, VERB	358	P(VERB|NOUN)	0.43
NOUN	833	NOUN, NOUN	108	P(NOUN|NOUN)	0.13
NOUN	833	NOUN, PREPOSITION	366	P(PREPOSITION|NOUN)	0.44
VERB	300	VERB, NOUN	75	P(NOUN|VERB)	0.35
VERB	300	VERB, ARTICLE	194	P(ARTICLE|VERB)	0.65
PREPOSITION	307	PREPOSITION, ARTICLE	226	P(ARTICLE|PREPOSITION)	0.74
PREPOSITION	307	PREPOSITION, NOUN	81	P(NOUN|PREPOSITION)	0.26

Hidden Markov Model



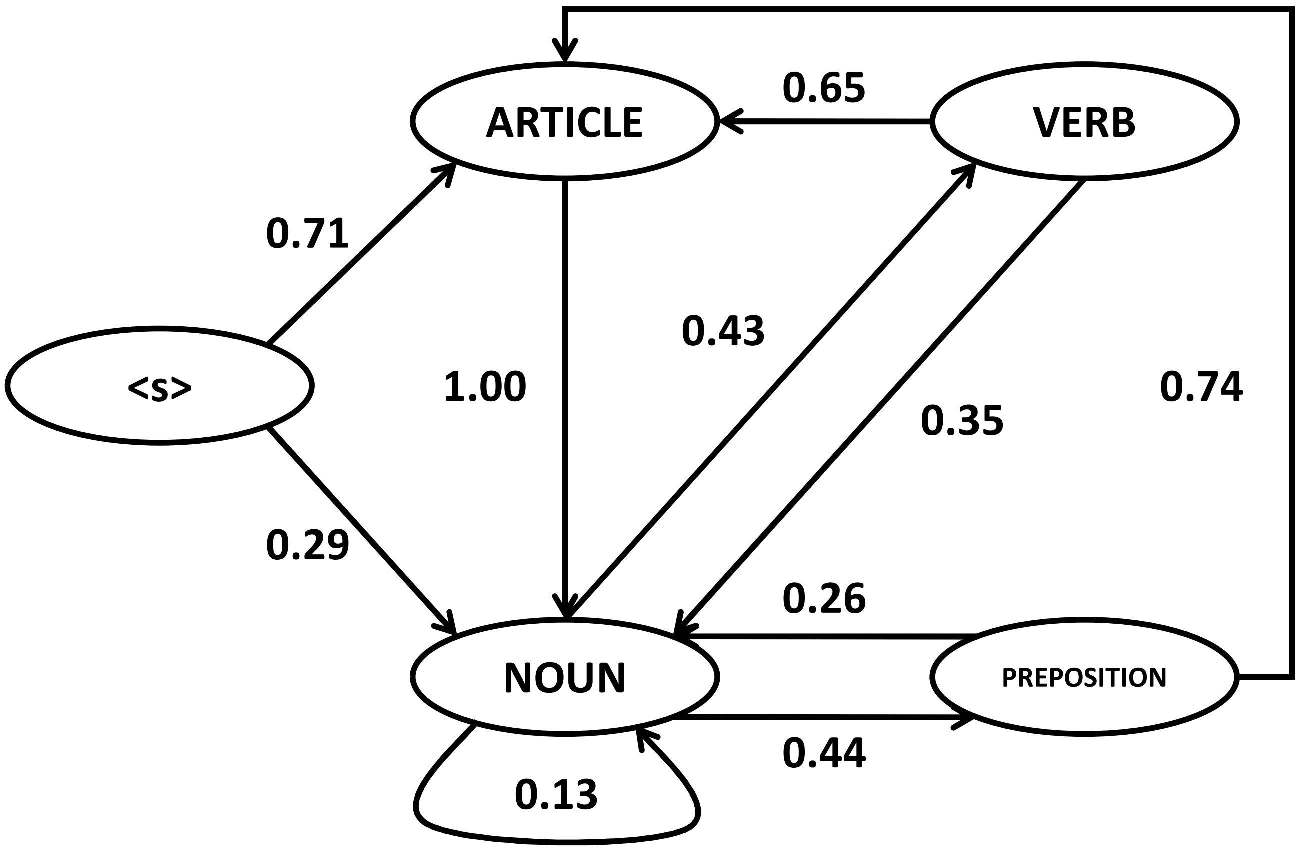
P(Bigram)	Estimate
P(ARTICLE|<S>)	0.71
P(NOUN|<S>)	0.29
P(NOUN|ARTICLE)	1.00
P(VERB|NOUN)	0.43
P(NOUN|NOUN)	0.13
P(PREPOSITION|NOUN)	0.44
P(NOUN|VERB)	0.35
P(ARTICLE|VERB)	0.65
P(ARTICLE|PREPOSITION)	0.74
P(NOUN|PREPOSITION)	0.26

Consider a following sequence of categories (tags):

<s>, ARTICLE, NOUN, VERB, NOUN

What's the probability of its occurence in our synthetic corpus?

Hidden Markov Model (HMM)

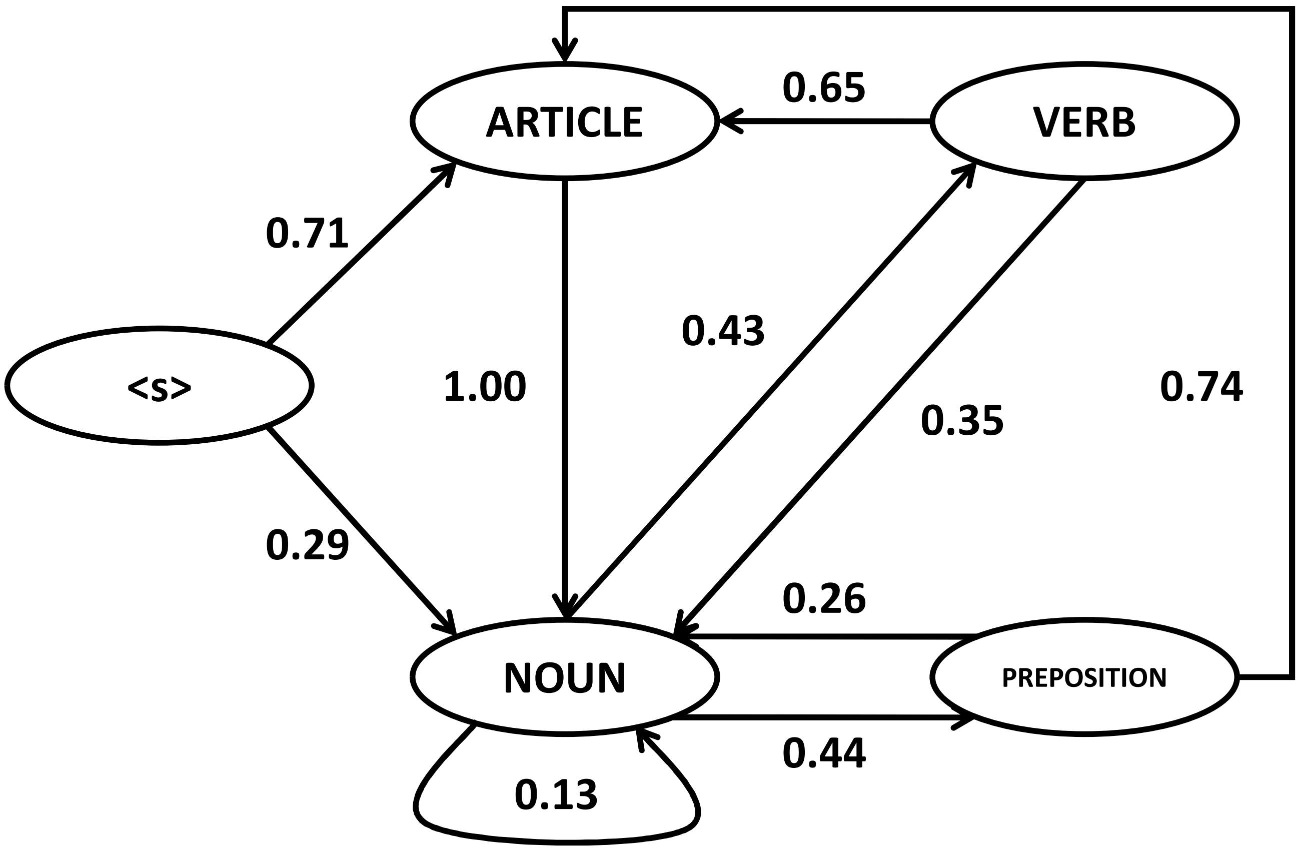


P(Bigram)	Estimate
P(ARTICLE|<S>)	0.71
P(NOUN|<S>)	0.29
P(NOUN|ARTICLE)	1.00
P(VERB|NOUN)	0.43
P(NOUN|NOUN)	0.13
P(PREPOSITION|NOUN)	0.44
P(NOUN|VERB)	0.35
P(ARTICLE|VERB)	0.65
P(ARTICLE|PREPOSITION)	0.74
P(NOUN|PREPOSITION)	0.26

The word "Hidden" in Hidden Markov Model means that for a specific sequence (of words) it is unclear what state the model is in.

The word *flies* could be generated from state NOUN and state VERB.

Hidden Markov Model

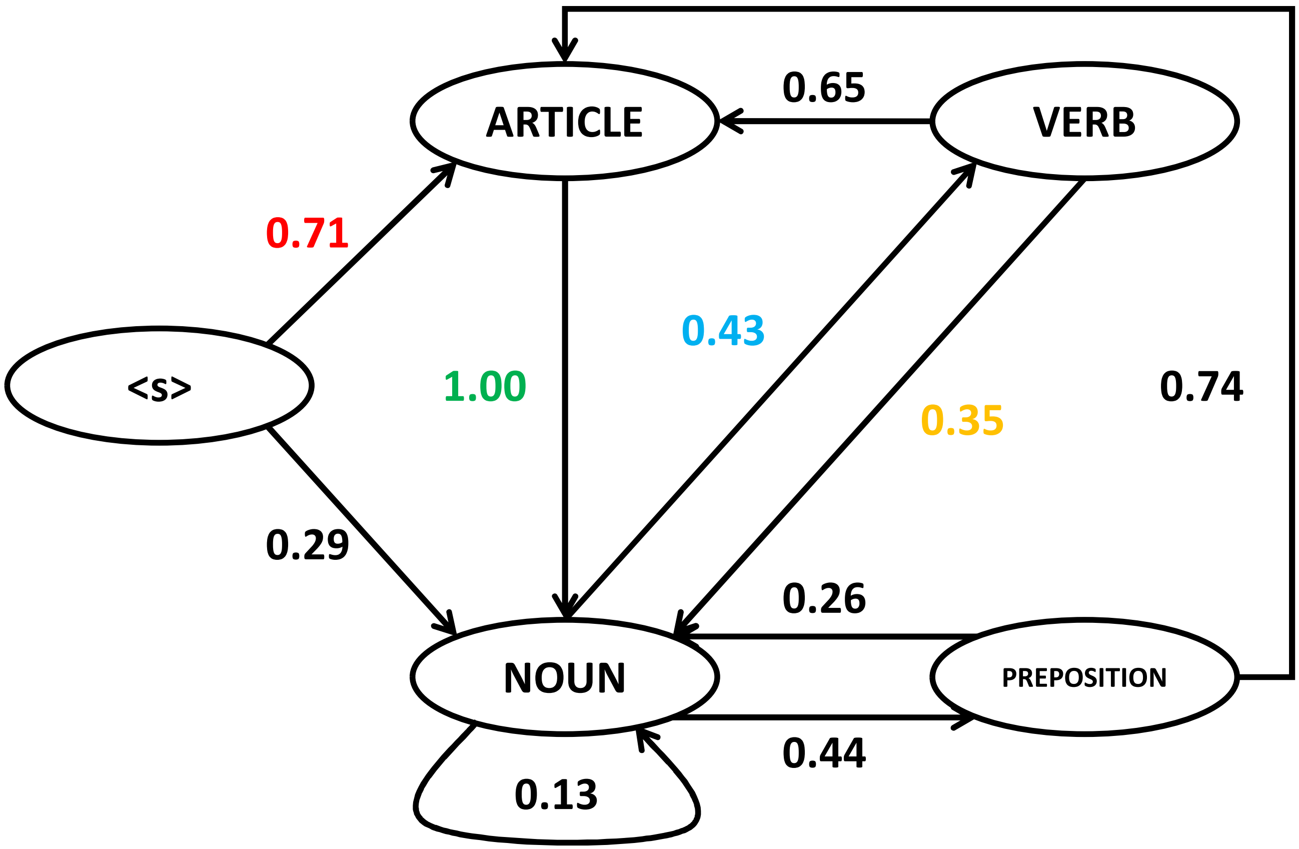


P(Bigram)	Estimate
P(ARTICLE|<S>)	0.71
P(NOUN|<S>)	0.29
P(NOUN|ARTICLE)	1.00
P(VERB|NOUN)	0.43
P(NOUN|NOUN)	0.13
P(PREPOSITION|NOUN)	0.44
P(NOUN|VERB)	0.35
P(ARTICLE|VERB)	0.65
P(ARTICLE|PREPOSITION)	0.74
P(NOUN|PREPOSITION)	0.26

Probability of occurrence of a sequence of categories (tags):

$$P(C_1, C_2, C_3, \ldots, C_T) \cong \prod_{i=1}^{T} P(C_i \mid C_{i-1})$$

Hidden Markov Model



P(Bigram)	Estimate
P(ARTICLE|<S>)	0.71
P(NOUN|<S>)	0.29
P(NOUN|ARTICLE)	1.00
P(VERB|NOUN)	0.43
P(NOUN|NOUN)	0.13
P(PREPOSITION|NOUN)	0.44
P(NOUN|VERB)	0.35
P(ARTICLE|VERB)	0.65
P(ARTICLE|PREPOSITION)	0.74
P(NOUN|PREPOSITION)	0.26

Probability of occurrence of a sequence of categories (tags):

$$P(<s>, \text{ARTICLE}, \text{NOUN}, \text{VERB}, \text{NOUN}) =$$

$$\cong P(ART|<s>) * P(N|ART) * P(V|N) * P(N|V) = 0.71 * 1.00 * 0.43 * 0.35 = 0.107$$

Synthetic Corpus: Word/Tag Counts

Summary of selected word counts in the synthetic corpus:

Word/Tag	N	V	ART	P	TOTAL
flies	21	23	0	0	44
fruit	49	5	1	0	55
like	10	30	0	21	61
a	1	0	201	0	202
the	1	0	300	2	303
flower	53	15	0	0	68
flowers	42	16	0	0	58
birds	64	1	0	0	65
others	592	210	56	284	1142
TOTAL	833	300	558	307	1998

From the table we can calculate lexical generation probabilities $P(w|C)$ estimates:

P(*the*|ART) = 300/558 = 0.54

P(*flies*|N) = 21/833 = 0.025

P(*flies*|V) = 23/300 = 0.076

P(*like*|V) = 30/300 = 0.1

P(*like*|P) = 21/307 = 0.068

P(*a*|ART) = 201/558 = 0.36

P(*a*|N) = 1/833 = 0.001

P(*flower*|N) = 53/833 = 0.063

P(*flower*|V) = 15/300 = 0.05

P(*like*|N) = 10/833 = 0.012

Example

Given our synthetic corpus, what is the most like sequence of categories (tags) corresponding to a sentence:

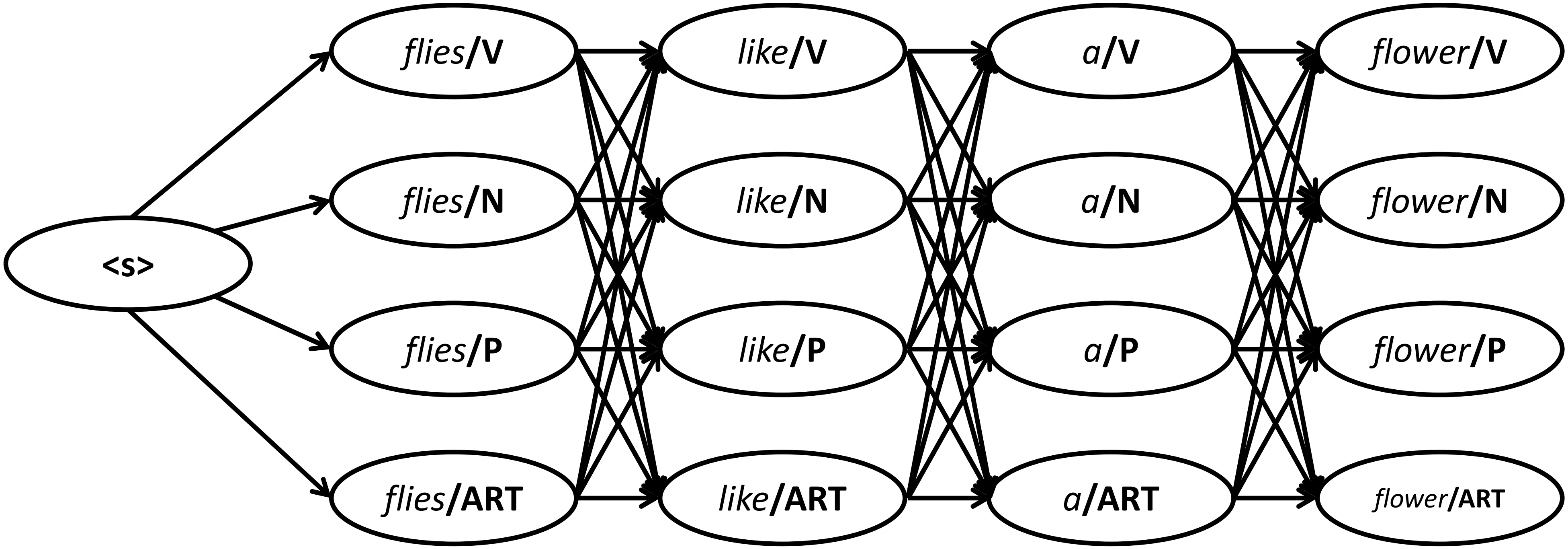
Flies like a flower

We need to maximize:

$$P(w_1, w_2, w_3, \ldots, w_T \mid C_1, C_2, C_3, \ldots, C_T) * P(C_1, C_2, C_3, \ldots, C_T) \cong$$

$$\cong \prod_{i=1}^{T} P(w_i \mid C_i) * P(C_i \mid C_{i-1})$$

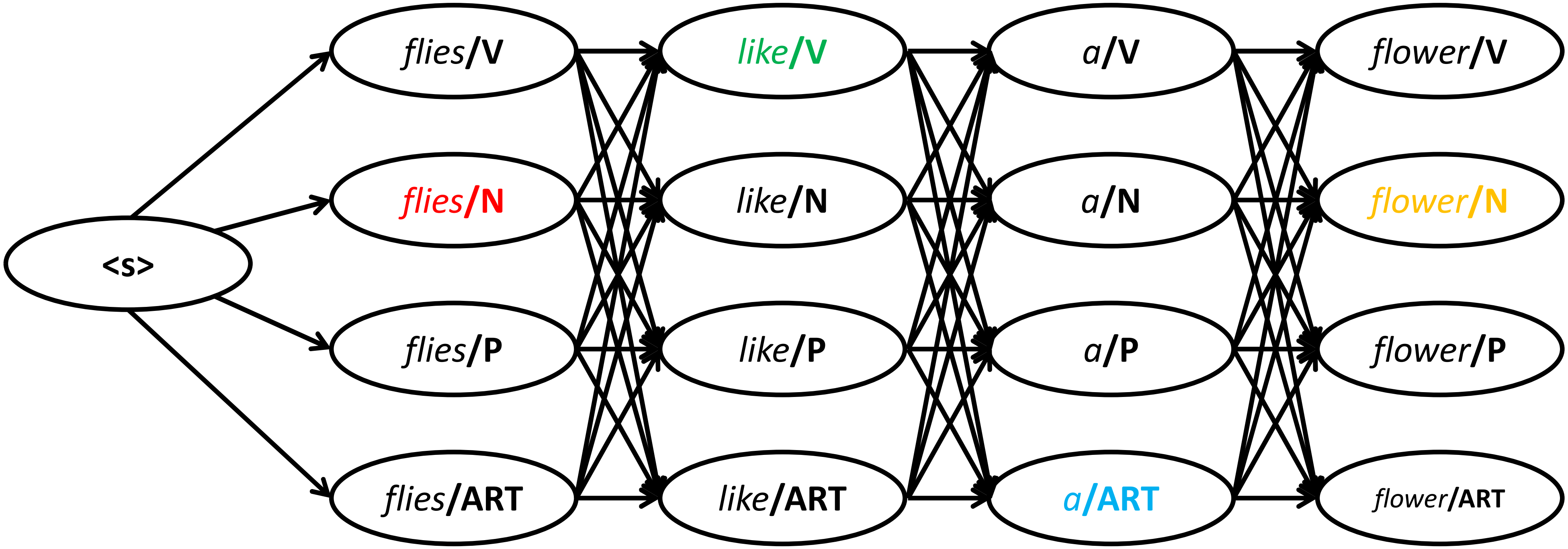
Example: All Possible Sequences



Every sequence can be assigned a probability:

$$P(w_1, w_2, w_3, \ldots, w_T \mid C_1, C_2, C_3, \ldots, C_T) \cong \prod_{i=1}^{T} P(w_i \mid C_i)$$

Example: All Possible Sequences



Every sequence can be assigned a probability:

$$\prod_{i=1}^{T} P(w_i \mid C_i) = P(flies|N) * P(like|V) * P(a|ART) * P(flower|N)$$

Synthetic Corpus: Word/Tag Counts

Summary of selected word counts in the synthetic corpus:

Word/Tag	N	V	ART	P	TOTAL
flies	21	23	0	0	44
fruit	49	5	1	0	55
like	10	30	0	21	61
a	1	0	201	0	202
the	1	0	300	2	303
flower	53	15	0	0	68
flowers	42	16	0	0	58
birds	64	1	0	0	65
others	592	210	56	284	1142
TOTAL	833	300	558	307	1998

From the table we can calculate lexical generation probabilities P(w|C) estimates:

P(*the*|ART) = 300/558 = 0.54

P(*flies*|N) = 21/833 = 0.025

P(*flies*|V) = 23/300 = 0.076

P(*like*|V) = 30/300 = 0.1

P(*like*|P) = 21/307 = 0.068

P(*a*|ART) = 201/558 = 0.36

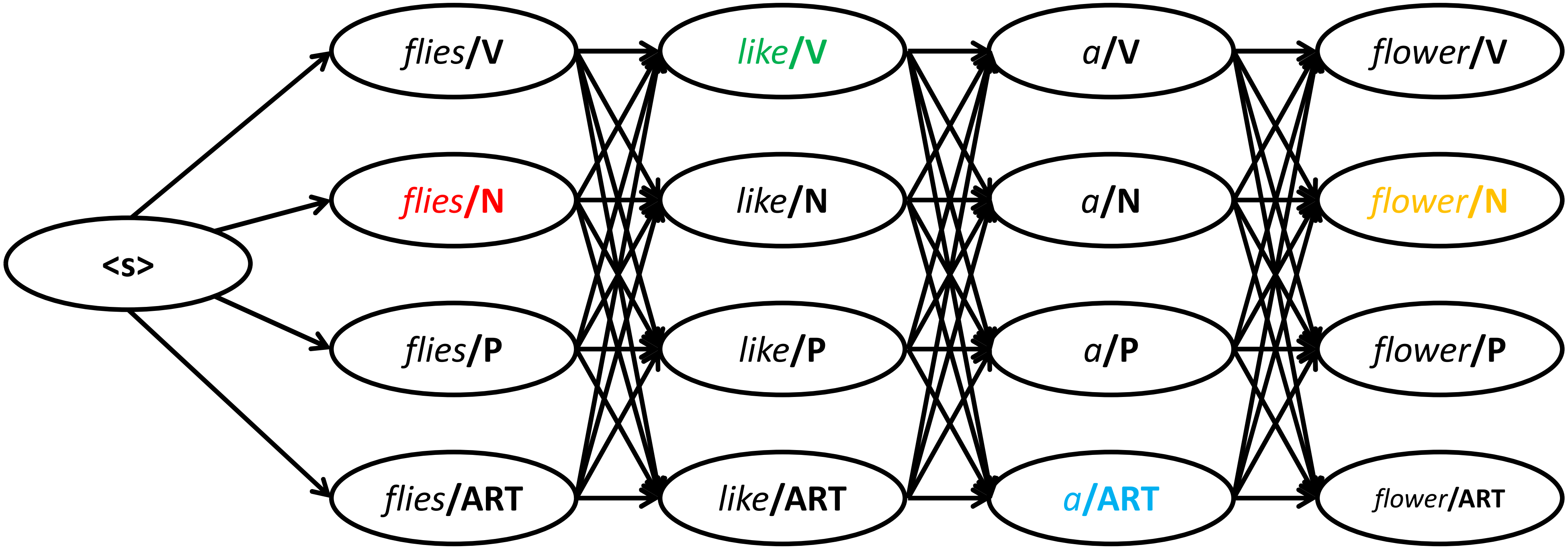
P(*a*|N) = 1/833 = 0.001

P(*flower*|N) = 53/833 = 0.063

P(*flower*|V) = 15/300 = 0.05

P(*like*|N) = 10/833 = 0.012

Example: All Possible Sequences



<s>

flies/V

flies/N

flies/P

flies/ART

like/V

like/N

like/P

like/ART

a/V

a/N

a/P

a/ART

flower/V

flower/N

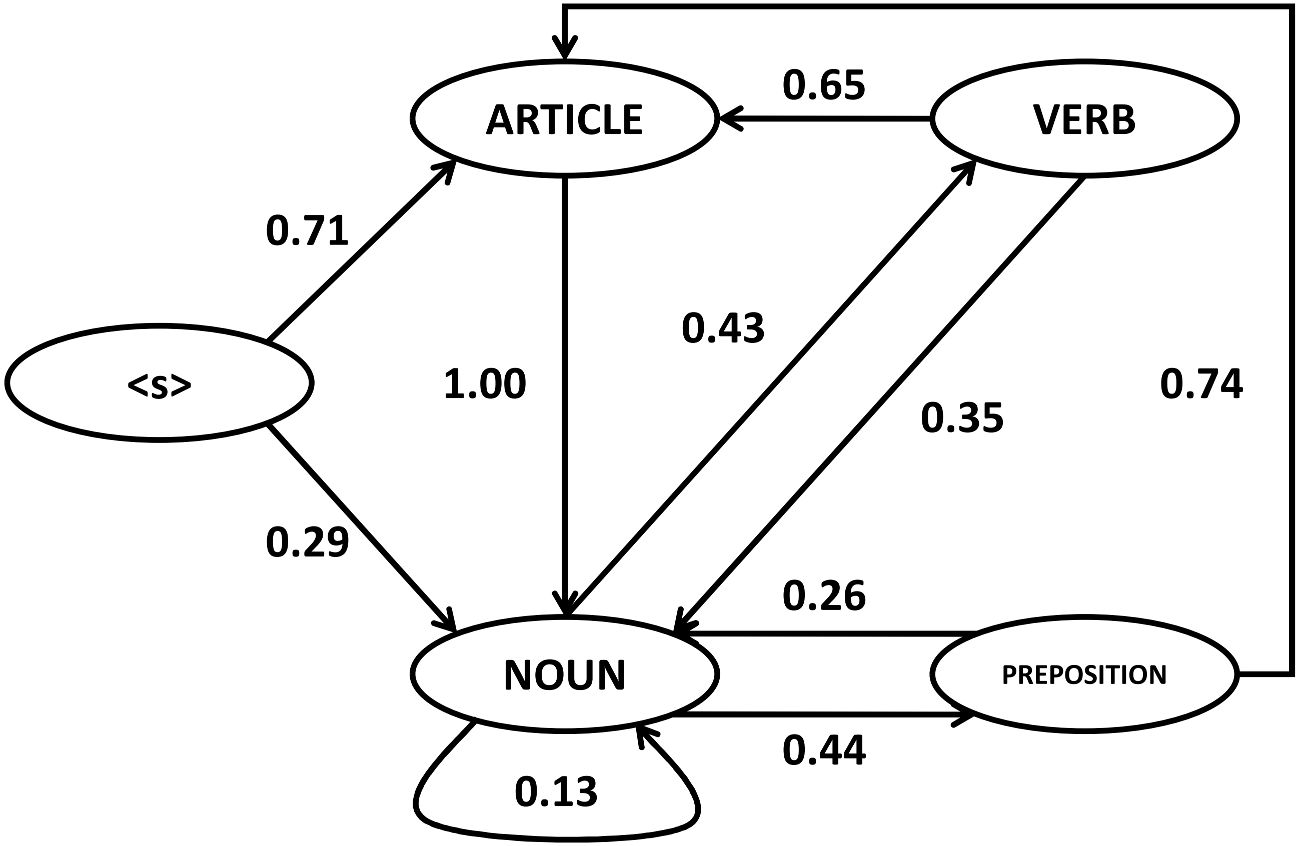
flower/P

flower/ART

Every sequence can be assigned a probability:

$$\prod_{i=1}^{T} P(w_i \mid C_i) = 0.025 * 0.1 * 0.36 * 0.063 = 5.4 * 0^{-}$$

Hidden Markov Model

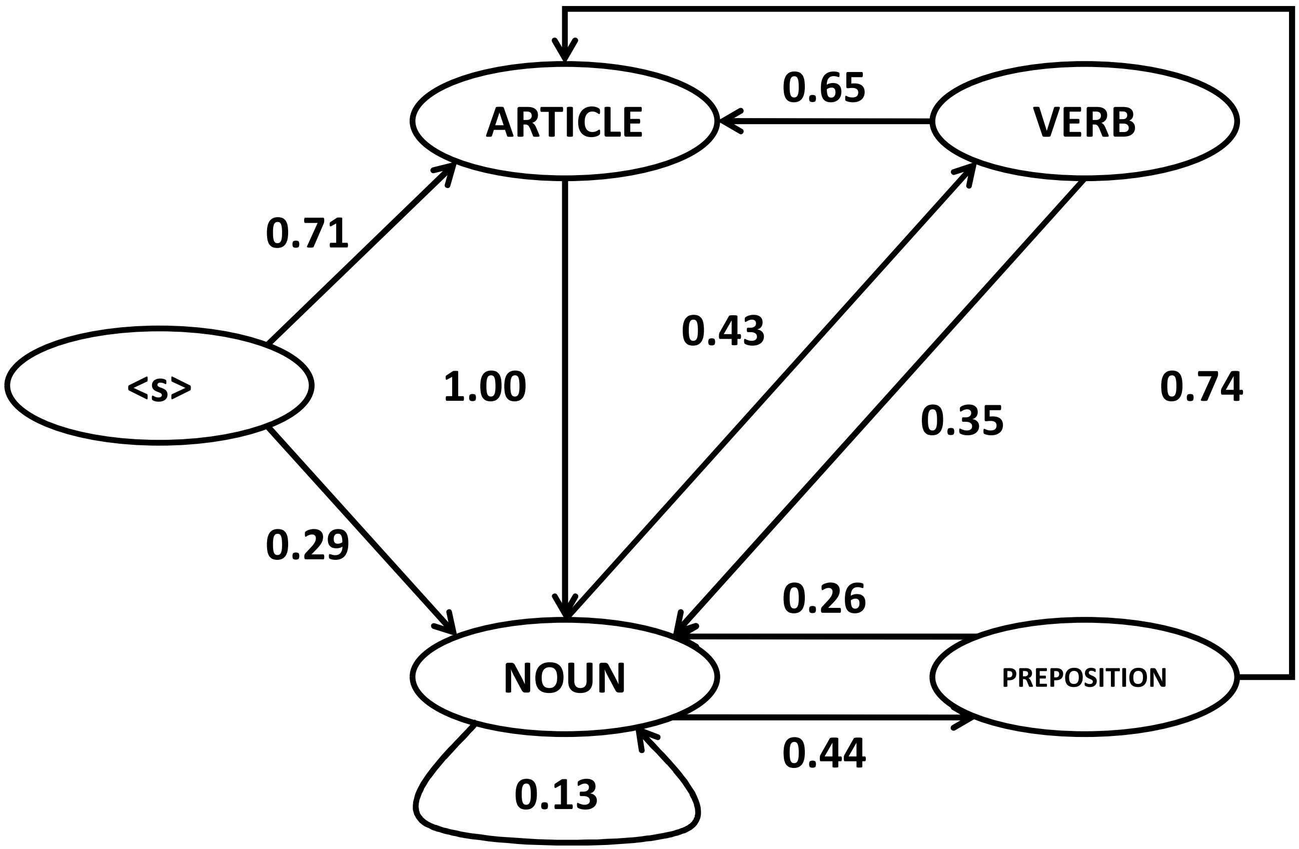


P(Bigram)	Estimate
P(ARTICLE|<S>)	0.71
P(NOUN|<S>)	0.29
P(NOUN|ARTICLE)	1.00
P(VERB|NOUN)	0.43
P(NOUN|NOUN)	0.13
P(PREPOSITION|NOUN)	0.44
P(NOUN|VERB)	0.35
P(ARTICLE|VERB)	0.65
P(ARTICLE|PREPOSITION)	0.74
P(NOUN|PREPOSITION)	0.26

For any sequence of categories (tags), their probability is:

$$P(C_1, C_2, C_3, \ldots, C_T) \cong \prod_{i=1}^{T} P(C_i \mid C_{i-1})$$

Hidden Markov Model

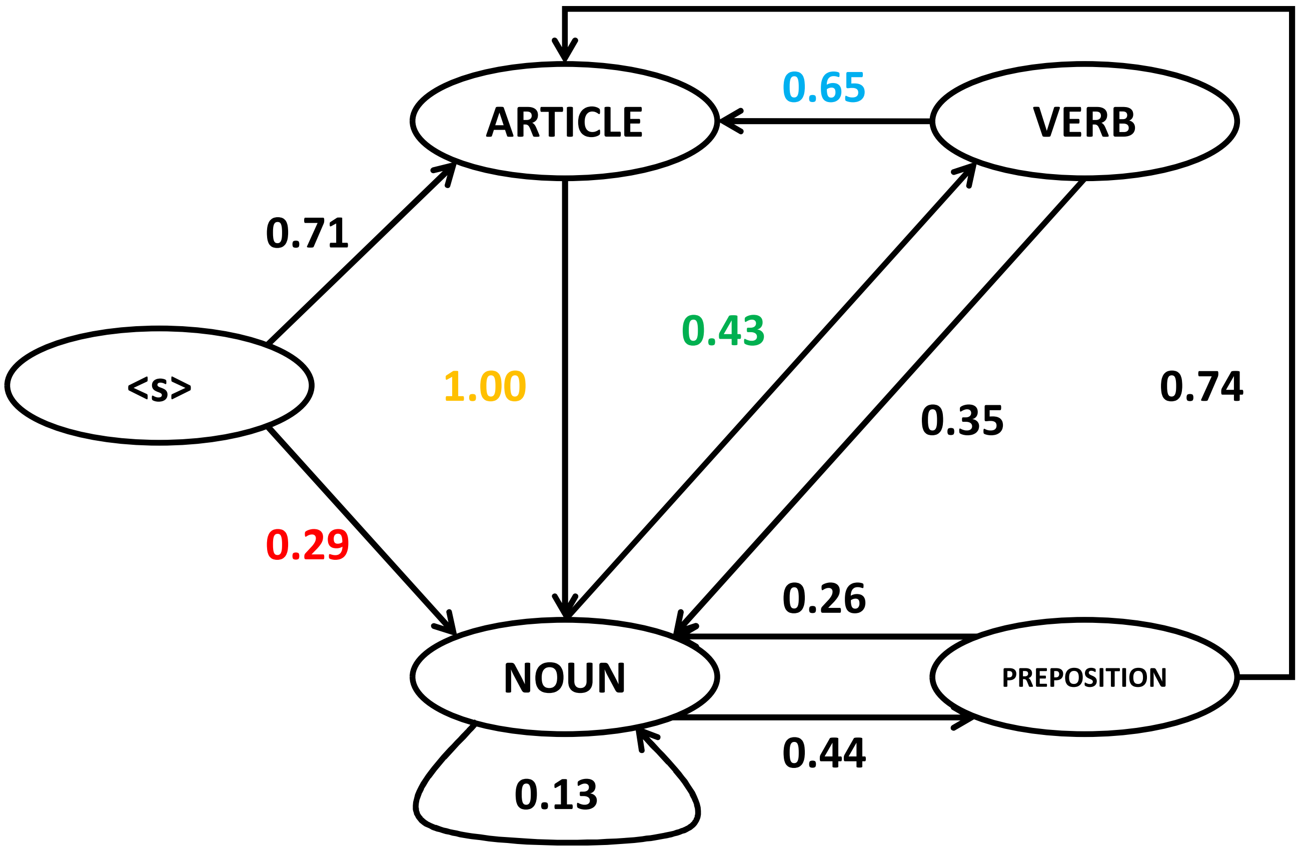


P(Bigram)	Estimate
P(ARTICLE|<S>)	0.71
P(NOUN|<S>)	0.29
P(NOUN|ARTICLE)	1.00
P(VERB|NOUN)	0.43
P(NOUN|NOUN)	0.13
P(PREPOSITION|NOUN)	0.44
P(NOUN|VERB)	0.35
P(ARTICLE|VERB)	0.65
P(ARTICLE|PREPOSITION)	0.74
P(NOUN|PREPOSITION)	0.26

For any sequence of categories (tags), their probability is:

$$\prod_{i=1}^{T} P(C_i \mid C_{i-1}) = P(N \mid < s >) * (V \mid N) * (ART \mid V) * (N \mid ART)$$

Hidden Markov Model



P(Bigram)	Estimate
P(ARTICLE|<S>)	0.71
P(NOUN|<S>)	0.29
P(NOUN|ARTICLE)	1.00
P(VERB|NOUN)	0.43
P(NOUN|NOUN)	0.13
P(PREPOSITION|NOUN)	0.44
P(NOUN|VERB)	0.35
P(ARTICLE|VERB)	0.65
P(ARTICLE|PREPOSITION)	0.74
P(NOUN|PREPOSITION)	0.26

For any sequence of categories (tags), their probability is:

$$\prod_{i=1}^{T} P(C_i \mid C_{i-1}) = 0.29 * 0.43 * 0.65 * 1.00 = 0.081$$

Example

Given our synthetic corpus, what is the most like sequence of categories (tags) corresponding to a sentence:

Flies like a flower

For example:

$$\boldsymbol{P(Flies, like, a, flower \mid N, V, ART, N) * P(N, V, ART, N)}$$

$$\boldsymbol{\cong 5.4 * 10^{-5} * 0.081}$$