Natural Language Processing - 2nd Semester (2024-2025)
1038141

1.13 - Exercises

SAPIENZA
UNIVERSITÀ DI ROMA

Prof. Stefano Faralli
faralli@di.uniroma1.it

Prof. Iacopo Masi
masi@di.uniroma1.it

**credits are reported in the last slide

SAPIENZA
UNIVERSITÀ DI ROMA

1.13 - Exercises

- Regular Expressions, Finite State Automata and Regular Expressions (Parts 1.3 and 1.4)
- Word, Corpora and Text Normalization (Part 1.5)
- Spelling Correction and Minimum Edit Distance (Part 1.6)
- Language Models (Part 1.7)
- Part-of-Speech tagging (Part 1.8)
- Vector Semantics (sparse) (Part 1.11)

SAPIENZA
UNIVERSITÀ DI ROMA

Regular Expressions, Finite State Automata and Regular Expressions (Parts 1.3 and 1.4)

REs, Finite State Automata and REs (Parts 1.3 and 1.4):

Ex: write a RE that matches a string that has an a followed by two to three 'b'.

REs, Finite State Automata and REs (Parts 1.3 and 1.4):

Ex: write a RE that matches a string that has an ‘**a**’ followed by two to three **'b'**.
draw the FSA.

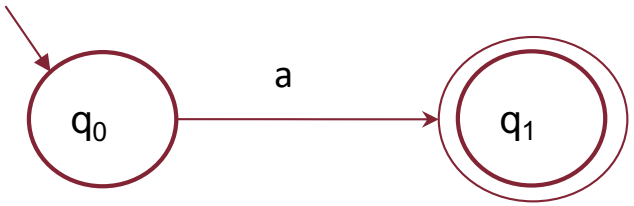
Solution: **ab{2,3}**

REs, Finite State Automata and REs (Parts 1.3 and 1.4):

Ex: write a RE that matches a string that has an **'a'** followed by two to three **'b'**. draw the FSA.

Solution: **ab{2,3}**

RE_1=a

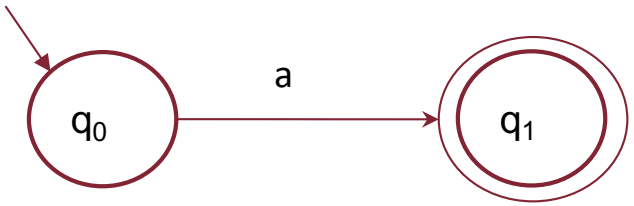


REs, Finite State Automata and REs (Parts 1.3 and 1.4):

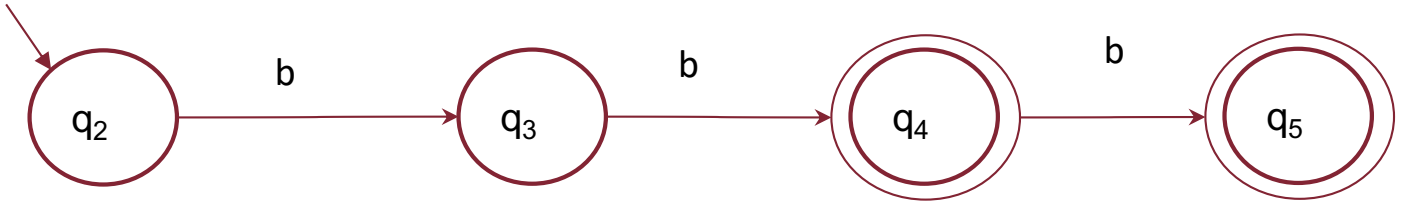
Ex: write a RE that matches a string that has an **'a'** followed by two to three **'b'**. draw the FSA.

Solution: **ab{2,3}**

RE_1=a



RE_2=b{2,3}

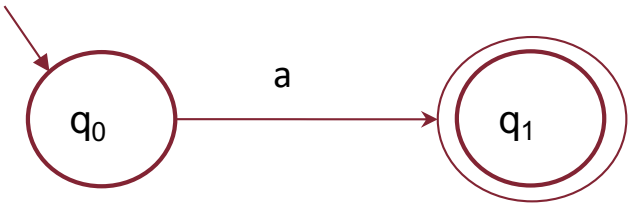


REs, Finite State Automata and REs (Parts 1.3 and 1.4):

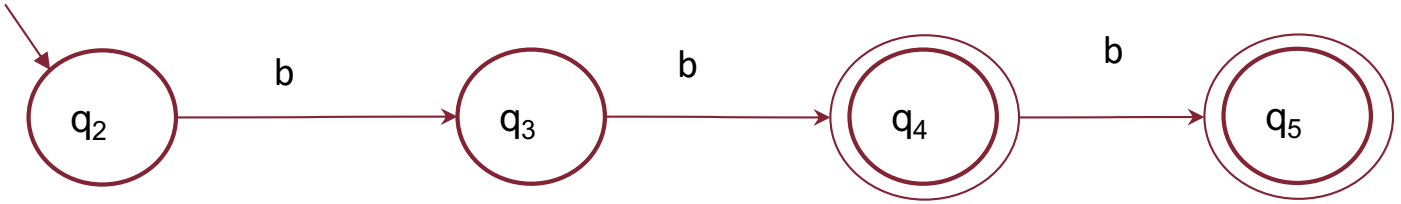
Ex: write a RE that matches a string that has an **'a'** followed by two to three **'b'**. draw the FSA.

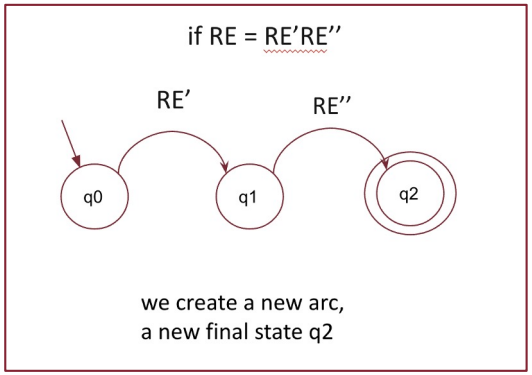
Solution: **ab{2,3}**

RE_1=a



RE_2=b{2,3}



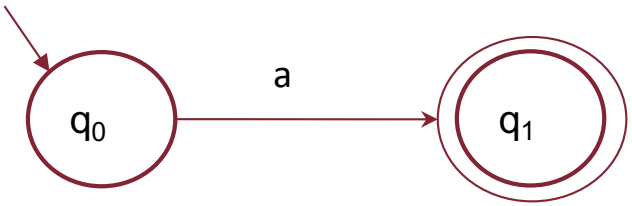


REs, Finite State Automata and REs (Parts 1.3 and 1.4):

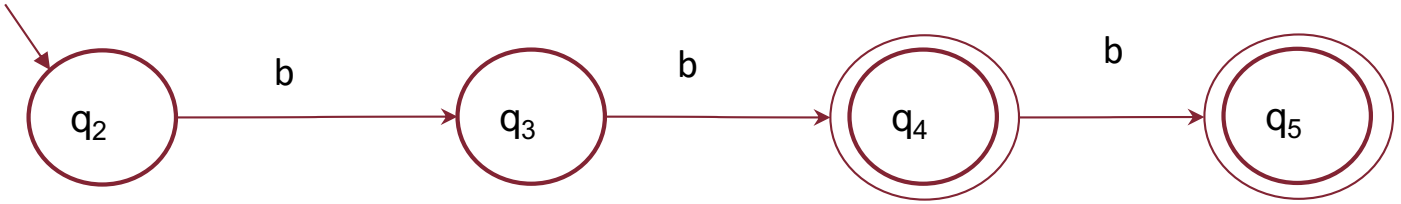
Ex: write a RE that matches a string that has an **'a'** followed by two to three **'b'**. draw the FSA.

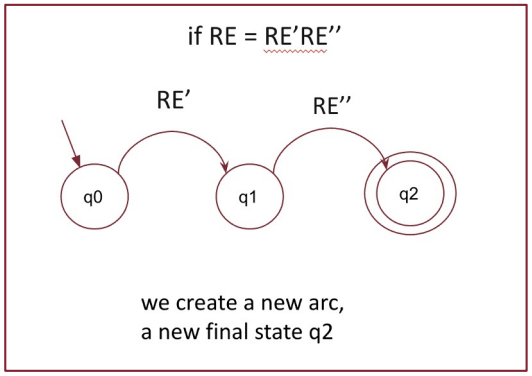
Solution: **ab{2,3}**

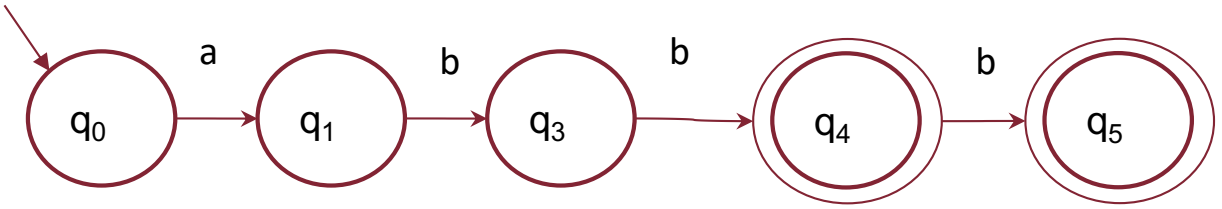
RE_1=a



RE_2=b{2,3}







SAPIENZA
UNIVERSITÀ DI ROMA

Word, Corpora and Text Normalization (Part 1.5)

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):

"we are like the dreamer who dreams, and then lives inside the dream."

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):
"we are like the dreamer who dreams, and then lives inside the dream."

- splits words by punctuations and counts the word token occurrences.

2 t h e _
1 d r e a m _
1 i n s i d e _
1 l i v e s _
1 t h e n _
1 a n d _
1 d r e a m s _
1 w h o _
1 d r e a m e r _
1 l i k e _
1 a r e _
1 w e _

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):
"we are like the dreamer who dreams, and then lives inside the dream."

2 t h e _
1 d r e a m _
1 i n s i d e _
1 l i v e s _
1 t h e n _
1 a n d _
1 d r e a m s _
1 w h o _
1 d r e a m e r _
1 l i k e _
1 a r e _
1 w e _

- create the initial vocabulary

_, a, d, e, h, i, l, m, n, o, r, s, t, v, w

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):
"we are like the dreamer who dreams, and then lives inside the dream."

2 t h e _
1 d r e a m _
1 i n s i d e _
1 l i v e s _
1 t h e n _
1 a n d _
1 d r e a m s _
1 w h o _
1 d r e a m e r _
1 l i k e _
1 a r e _
1 w e _

_, a, d, e, h, i, l, m, n, o, r, s, t, v, w

- **search for the most frequent pair of symbols**

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):
"we are like the dreamer who dreams, and then lives inside the dream."

2 t h e _
1 d r e a m _
1 i n s i d e _
1 l i v e s _
1 t h e n _
1 a n d _
1 d r e a m s _
1 w h o _
1 d r e a m e r _
1 l i k e _
1 a r e _
1 w e _

_, a, d, e, h, i, l, m, n, o, r, s, t, v, w

1/7

- **search for the most frequent pair of symbols**
  “e _” occurs 6 times -> merge(e, _)

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):
"we are like the dreamer who dreams, and then lives inside the dream."

2 t h **e_**
1 d r e a m _
1 i n s i d **e_**
1 l i v e s _
1 t h e n _
1 a n d _
1 d r e a m s _
1 w h o _
1 d r e a m e r _
1 l i k **e_**
1 a r **e_**
1 w **e_**

_, a, d, e, h, i, l, m, n, o, r, s, t, v, w, **e_**

1/7

- **search for the most frequent pair of symbols "e _" occurs 6 times -> merge(e, _)**

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):
"we are like the dreamer who dreams, and then lives inside the dream."

2 t h e_
1 d **re** a m _
1 i n s i d e_
1 l i v e s _
1 t h e n _
1 a n d _
1 d **re** a m s _
1 w h o _
1 d **re** a m e r _
1 l i k e_
1 a r e_
1 w e_

_, a, d, e, h, i, l, m, n, o, r, s, t, v, w, e_, **re**

2/7

- **search for the most frequent pair of symbols**
  “r e” occurs 3 times -> merge(r, e)

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):
"we are like the dreamer who dreams, and then lives inside the dream."

2 t h e_
1 **dre** a m _
1 i n s i d e_
1 l i v e s _
1 t h e n _
1 a n d _
1 **dre** a m s _
1 w h o _
1 **dre** a m e r _
1 l i k e_
1 a r e_
1 w e_

_, a, d, e, h, i, l, m, n, o, r, s, t, v, w, e_, re,
dre

3/7

- **search for the most frequent pair of symbols**
  “d re” occurs 3 times -> merge(d, re)

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):
"we are like the dreamer who dreams, and then lives inside the dream."

2 t h e_
1 **dre** a m _
1 i n s i d e_
1 l i v e s _
1 t h e n _
1 a n d _
1 **dre** a m s _
1 w h o _
1 **dre** a m e r _
1 l i k e_
1 a r e_
1 w e_

_, a, d, e, h, i, l, m, n, o, r, s, t, v, w, e_, re,
dre

4/7

- **search for the most frequent pair of symbols**
  “d re” occurs 3 times -> merge(d, re)

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):
"we are like the dreamer who dreams, and then lives inside the dream."

2 **th** e_
1 dre a m _
1 i n s i d e_
1 l i v e s _
1 **th** e n _
1 a n d _
1 dre a m s _
1 w h o _
1 dre a m e r _
1 l i k e_
1 a r e_
1 w e_

_, a, d, e, h, i, l, m, n, o, r, s, t, v, w, e_, re,
dre, **th,**

5/7

- **search for the most frequent pair of symbols**
  “t h” occurs 3 times -> merge(t, h)

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):
"we are like the dreamer who dreams, and then lives inside the dream."

2 th e_
1 **drea** m _
1 i n s i d e_
1 l i v e s _
1 th e n _
1 a n d _
1 **drea** m s _
1 w h o _
1 **drea** m e r _
1 l i k e_
1 a r e_
1 w e_

_, a, d, e, h, i, l, m, n, o, r, s, t, v, w, e_, re,
dre, th**, drea**

6/7

- **search for the most frequent pair of symbols**
  “dre a” occurs 3 times -> merge(dre, a)

Word, Corpora and Text Normalization (Part 1.5):

EX: Perform tokens learning with the byte-pair encoding, given the following corpus (only 7 iterations):
"we are like the dreamer who dreams, and then lives inside the dream."

2 th e_
1 **dream** _
1 i n s i d e_
1 l i v e s _
1 th e n _
1 a n d _
1 **dream** s _
1 w h o _
1 **dream** e r _
1 l i k e_
1 a r e_
1 w e_

_, a, d, e, h, i, l, m, n, o, r, s, t, v, w, e_, re, dre, th, drea, **dream**

7/7

- **search for the most frequent pair of symbols "drea m" occurs 3 times -> merge(drea, m)**

SAPIENZA
UNIVERSITÀ DI ROMA

Spelling Correction and Minimum Edit Distance (Part 1.6)

Spelling Correction and Minimum Edit Distance (Part 1.6)

EX: Compute the Minimum Edit Distance between “home” and “house”. Assume the Levenshtein cost function.

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	o	u	s	e
i=0	#						
i=1	h						
i=2	o						
i=3	m						
i=4	e						

How to find the Minimum Edit Distance: Dynamic programming

Exercise:

Compute the $D_{X,Y}$ for, X=hey and Y=hello and provide an optimal alignment. assume the Levenshtein costs for editing operations.

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#						
i=1	h						
i=2	e						
i=e	y						

Initialization

create a matrix $D_{X,Y}$ with n+1 rows (n=|X|) and m+1 columns (m=|Y|)

$D_{X,Y}(i,0) = i$

$D_{X,Y}(0,j) = j$

How to find the Minimum Edit Distance: Dynamic programming

Exercise:

Compute the $D_{X,Y}$ for, X=hey and Y=hello. and provide an optimal alignment. assume the Levenshtein costs for editing operations.

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0					
i=1	h	↑ 1					
i=2	e	↑ 2					
i=e	y	↑ 3					

Initialization

create a matrix $D_{X,Y}$ with n+1 rows (n=|X|) and m+1 columns (m=|Y|)

$D_{X,Y}(i,0) = i$ ↑

$D_{X,Y}(0,j) = j$ ←

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

Exercise:

Compute the $D_{X,Y}$ for, X=hey and Y=hello. and provide an optimal alignment. assume the Levenshtein costs for editing operations.

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1					
i=2	e	↑ 2					
i=e	y	↑ 3					

Initialization

create a matrix $D_{X,Y}$ with n+1 rows (n=|X|) and m+1 columns (m=|Y|)

$D_{X,Y}(i,0) = i$

$D_{X,Y}(0,j) = j$

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0				
i=2	e	↑ 2					
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=1

j=1

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1			
i=2	e	↑ 2					
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=1

j=2

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); \text{ if } X[i] \neq Y[j] \\ 0; \text{ if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2		
i=2	e	↑ 2					
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=1

j=3

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	
i=2	e	↑ 2					
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=1

j=4

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2					
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=1

j=5

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1				
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=2

j=1

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0			
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=2

j=2

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} & \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1		
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=2

j=3

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=2

j=4

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=e	y	↑ 3					

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=2

j=5

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=e	y	↑ 3	↑ 2				

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=3

j=1

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1			

Recurrence Relation:

```
For each  i = 1…n                    i=3
    For each  j = 1…m                j=2
```

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2		

Recurrence Relation:

For each i = 1…n

For each j = 1…m

i=3

j=3

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	

Recurrence Relation:

```
For each  i = 1…n
    For each  j = 1…m
```

i=3

j=4

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

Recurrence Relation:

For each i = 1…n

For each j = 1…m

i=3

j=5

$$D_{X,Y}(i,j) = \min \begin{cases} D_{X,Y}(i-1,j) + \text{cost}(\mathbf{d}) & \uparrow \\ D_{X,Y}(i,j-1) + \text{cost}(\mathbf{i}) & \leftarrow \\ D_{X,Y}(i-1,j-1) + \begin{cases} \text{cost}(\mathbf{s}); & \text{if } X[i] \neq Y[j] \\ 0; & \text{if } X[i] = Y[j] \end{cases} \end{cases}$$

when computing the minimum keep track of the cells with the minimum value with backtrack pointers (↑, ←, ↖)

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

Termination:

$D_{X,Y}(n,m)$ is the minimum edit distance;

[Jurafsk

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

X : # h e y
Y : # _ _ _
op: _ _ _ _

↑ deletion
↖ substitution
← insertion

Optimal alignment:
start form $D_{X,Y}(n,m)$ and follow the backtrack pointers;

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

X : # h e y
Y : # _ _ _ **o**
op: _ _ _ _ **i**

↑ deletion
↖ substitution
← insertion

Optimal alignment:

start form $D_{X,Y}(n,m)$ and follow the backtrack pointers;

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

X : # h e y
Y : # _ _ _ l o
op: _ _ _ _ **i** i

↑ deletion
↖ substitution
← insertion

Optimal alignment:

start form $D_{X,Y}(n,m)$ and follow the backtrack pointers;

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

X : # h e y
Y : # _ _ **l** l o
op: _ _ _ **s** i i

↑ deletion
↖ substitution
← insertion

Optimal alignment:

start form $D_{X,Y}(n,m)$ and follow the backtrack pointers;

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

X : # h e y
Y : # _ **e** l l o
op: _ _ _ s i i

↑ deletion
↖ substitution
← insertion

Optimal alignment:

start form $D_{X,Y}(n,m)$ and follow the backtrack pointers;

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

X : # h e y
Y : # **h** e l l o
op: _ _ _ s i i

↑ deletion
↖ substitution
← insertion

Optimal alignment:

start form $D_{X,Y}(n,m)$ and follow the backtrack poi[illegible]s;

[Jurafsky&Martin, 2022]

How to find the Minimum Edit Distance: Dynamic programming

		j=0	j=1	j=2	j=3	j=4	j=5
		#	h	e	l	l	o
i=0	#	0	← 1	← 2	← 3	← 4	← 5
i=1	h	↑ 1	↖ 0	← 1	← 2	← 3	← 4
i=2	e	↑ 2	↑ 1	↖ 0	← 1	← 2	← 3
i=3	y	↑ 3	↑ 2	↑ 1	←↖↑ 2	←↖↑ 3	←↖↑ 4

X : # h e y
Y : # h e l l o
op: _ _ _ s i i

↑ deletion
↖ substitution
← insertion

Other paths are also possible !

Optimal alignment:

start form $D_{X,Y}(n,m)$ and follow the backtrack pointers;

SAPIENZA
UNIVERSITÀ DI ROMA

Language Models (Part 1.7)

Language Models (Part 1.7)

Given the following corpus C, with the following 5 sentences:

1. I am Sam
2. Sam I am
3. Sam I like
4. Sam I do like
5. do I like Sam

1) **Define the 2-gram (bigram) model on the corpus C.**
2) **What are the most probable words to follow the sequences:**
   a) **<s> Sam . . .**
   b) **<s> Sam I do . . .**
   c) **<s> Sam I am Sam . . .**
   d) **<s> do I like . . .**
3) **Compute the probability of the following sequences:**
   <s> Sam do like </s>
   <s> Sam I am </s>
   <s> I am Sam </s>

Language Models (Part 1.7)

1) Define the 2-gram (bigram) model on the corpus C.

1. I am Sam
2. Sam I am
3. Sam I like
4. Sam I do like
5. do I like Sam

We add the <s> and </s> tags

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

Exercise:

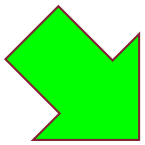
1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

if we adopt the closed vocabulary assumption, the vocabulary is:

<s>
I
am
Sam
</s>
like
do

Exercise:

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>



w	C(w)
<s>	5
I	5
am	2
Sam	5
</s>	5
like	3
do	2

We count $\mathbf{C(w_i)}$

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	**1**	0	0	0	0	0
I	0	0	0	0	0	0	0
am	0	0	0	0	0	0	0
Sam	0	0	0	0	0	0	0
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I)

Exercise:

	<s>	I	**am**	Sam	like	do	</s>
<s>	0	1	0	0	0	0	0
I	0	0	**1**	0	0	0	0
am	0	0	0	0	0	0	0
Sam	0	0	0	0	0	0	0
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am)

Exercise:

	<s>	I	am	**Sam**	like	do	</s>
<s>	0	1	0	0	0	0	0
I	0	0	1	0	0	0	0
am	0	0	0	**1**	0	0	0
Sam	0	0	0	0	0	0	0
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam)

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	0	0	0	0
I	0	0	1	0	0	0	0
am	0	0	0	1	0	0	0
Sam	0	0	0	0	0	0	**1**
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam </s>)

Exercise:

	<s>	I	am	**Sam**	like	do	</s>
<s>	0	1	0	**1**	0	0	0
I	0	0	1	0	0	0	0
am	0	0	0	1	0	0	0
Sam	0	0	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	1	0	0	0
I	0	0	1	0	0	0	0
am	0	0	0	1	0	0	0
Sam	0	**1**	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I)

...

Exercise:

	<s>	I	**am**	Sam	like	do	</s>
<s>	0	1	0	1	0	0	0
I	0	0	**2**	0	0	0	0
am	0	0	0	1	0	0	0
Sam	0	1	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	1	0	0	0
I	0	0	2	0	0	0	0
am	0	0	0	1	0	0	1
Sam	0	1	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)

...

Exercise:

	<s>	I	am	**Sam**	like	do	</s>
<s>	0	1	0	2	0	0	0
I	0	0	2	0	0	0	0
am	0	0	0	1	0	0	1
Sam	0	1	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>

2 <s> Sam I am </s>

3 <s> Sam I like </s>

4 <s> Sam I do like </s>

5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam,</s>) 2 (<s>,Sam)(Sam,I) (I,am) (am,</s>)

3 (<s>,Sam)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	2	0	0	0
I	0	0	2	0	0	0	0
am	0	0	0	1	0	0	1
Sam	0	**2**	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I)

...

Exercise:

	<s>	I	am	Sam	**like**	do	</s>
<s>	0	1	0	2	0	0	0
I	0	0	2	0	1	0	0
am	0	0	0	1	0	0	1
Sam	0	2	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam,</s>) 2 (<s>,Sam)(Sam,I) (I,am) (am,</s>)

3 (<s>,Sam) (Sam,I) (I,like)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	2	0	0	0
I	0	0	2	0	1	0	0
am	0	0	0	1	0	0	1
Sam	0	2	0	0	0	0	1
like	0	0	0	0	0	0	**1**
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)

...

Exercise:

	<s>	I	am	**Sam**	like	do	</s>
<s>	0	1	0	**3**	0	0	0
I	0	0	2	0	1	0	0
am	0	0	0	1	0	0	1
Sam	0	2	0	0	0	0	1
like	0	0	0	0	0	0	**1**
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	3	0	0	0
I	0	0	2	0	1	0	0
am	0	0	0	1	0	0	1
Sam	0	**3**	0	0	0	0	1
like	0	0	0	0	0	0	**1**
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I)

...

Exercise:

	<s>	I	am	Sam	like	**do**	</s>
<s>	0	1	0	3	0	0	0
I	0	0	2	0	1	**1**	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	1
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I) (I,do)

...

Exercise:

	<s>	I	am	Sam	**like**	do	</s>
<s>	0	1	0	3	0	0	0
I	0	0	2	0	1	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	1
do	0	0	0	0	**1**	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I) (I,do) (do, like)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	3	0	0	0
I	0	0	2	0	1	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	2
do	0	0	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)

...

Exercise:

	<s>	I	am	Sam	like	**do**	</s>
<s>	0	1	0	3	0	1	0
I	0	0	2	0	1	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	2
do	0	0	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>

2 <s> Sam I am </s>

3 <s> Sam I like </s>

4 <s> Sam I do like </s>

5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam,</s>)

2 (<s>,Sam)(Sam,I) (I,am) (am,</s>)

3 (<s>,Sam) (Sam,I) (I,like) (like, </s>)

4 (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)

5 (<s>,do)

…

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	3	0	1	0
I	0	0	2	0	1	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	2
do	0	1	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)
5. (<s>,do) (do, I)

...

Exercise:

	<s>	I	am	Sam	**like**	do	</s>
<s>	0	1	0	3	0	1	0
I	0	0	2	0	**2**	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	2
do	0	1	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)
5. (<s>,do) (do, I) (I, like)

...

Exercise:

	<s>	I	am	**Sam**	like	do	</s>
<s>	0	1	0	3	0	1	0
I	0	0	2	0	2	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	**1**	0	0	2
do	0	1	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)
5. (<s>,do) (do, I) (I, like) (like, Sam)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	3	0	1	0
I	0	0	2	0	2	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	**2**
like	0	0	0	1	0	0	2
do	0	1	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>

2 <s> Sam I am </s>

3 <s> Sam I like </s>

4 <s> Sam I do like </s>

5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam,</s>)

2 (<s>,Sam)(Sam,I) (I,am) (am,</s>)

3 (<s>,Sam) (Sam,I) (I,like) (like, </s>)

4 (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)

5 (<s>,do) (do, I) (I, like) (like, Sam) (Sam, </s>)

…

Exercise:

w	C(w)
<s>	5
I	5
am	2
Sam	5
</s>	5
like	3
do	2

	<s>	I	am	Sam	like	do	</s>
<s>	0/5	1/5	0/5	3/5	0/5	1/5	0/5
I	0/5	0/5	2/5	0/5	2/5	1/5	0/5
am	0/2	0/2	0/2	1/2	0/2	0/2	1/2
Sam	0/5	3/5	0/5	0/5	0/5	0/5	2/5
like	0/3	0/3	0/3	1/3	0/3	0/3	2/3
do	0/2	1/2	0/2	0/2	1/2	0/2	0/2
</s>	0/5	0/5	0/5	0/5	0/5	0/5	0/5

we divide each cell (u,w) by the c(u)

...

Language Models (Part 1.7)

w	C(w)
<s>	5
I	5
am	2
Sam	5
</s>	5
like	3
do	2

	<s>	I	am	Sam	like	do	</s>
<s>	**0**	**1/5**	**0**	**3/5**	**0**	**1/5**	**0**
I	**0**	**0**	**2/5**	**0**	**2/5**	**1/5**	**0**
am	**0**	**0**	**0**	**1/2**	**0**	**0**	**1/2**
Sam	**0**	**3/5**	**0**	**0**	**0**	**0**	**2/5**
like	**0**	**0**	**0**	**1/3**	**0**	**0**	**2/3**
do	**0**	**1/2**	**0**	**0**	**1/2**	**0**	**0**
</s>	**0**	**0**	**0**	**0**	**0**	**0**	**0**

now each cell (u,w) = P(w|u)

...

Language Models (Part 1.7)

2. What are the most probable words to follow the sequences:

a. **<s> Sam . . .**
b. **<s> Sam I do . . .**
c. **<s> Sam I am Sam . . .**
d. **<s> do I like . . .**

Language Models (Part 1.7)

2. What are the most probable words to follow the sequences:

- **a. <s> Sam . . .**
- **b. <s> Sam I do . . .**
- **c. <s> Sam I am Sam . . .**
- **d. <s> do I like . . .**

Markov assumption with a bigram model

	<s>	I	am	Sam	like	do	</s>
<s>	0	1/5	0	3/5	0	1/5	0
I	0	0	2/5	0	2/5	1/5	0
am	0	0	0	1/2	0	0	1/2
Sam	0	**3/5**	0	0	0	0	2/5
like	0	0	0	1/3	0	0	2/3
do	0	1/2	0	0	1/2	0	0
</s>	0	0	0	0	0	0	0

a) **<s> Sam …**
we must search **w** with the highest P(**w** |Sam)

we have only one w= I
P(I | Sam) = ⅗

Language Models (Part 1.7)

2. What are the most probable words to follow the sequences:

a. <s> Sam . . .
b. <s> Sam I do . . .
c. <s> Sam I am Sam . . .
d. <s> do I like . . .

Markov assumption with a bigram model

	<s>	I	am	Sam	like	do	</s>
<s>	0	1/5	0	3/5	0	1/5	0
I	0	0	2/5	0	2/5	1/5	0
am	0	0	0	1/2	0	0	1/2
Sam	0	3/5	0	0	0	0	2/5
like	0	0	0	1/3	0	0	2/3
do	0	**1/2**	0	0	**1/2**	0	0
</s>	0	0	0	0	0	0	0

b) **<s> Sam I do …**

we must search **w** with the highest P(**w** | do)

we have two **w**
P(I | do) = P(like | do) = ½

Language Models (Part 1.7)

2. What are the most probable words to follow the sequences:

- **a. <s> Sam . . .**
- **b. <s> Sam I do . . .**
- **c. <s> Sam I am Sam . . .**
- **d. <s> do I like . . .**

Markov assumption with a bigram model

	<s>	I	am	Sam	like	do	</s>
<s>	0	1/5	0	3/5	0	1/5	0
I	0	0	2/5	0	2/5	1/5	0
am	0	0	0	1/2	0	0	1/2
Sam	0	3/5	0	0	0	0	2/5
like	0	0	0	1/3	0	0	2/3
do	0	1/2	0	0	1/2	0	0
</s>	0	0	0	0	0	0	0

c. <s> Sam I am Sam . . .
d. <s> do I like . . .

do it yourself

Language Models (Part 1.7)

3) Compute the probability of the following sequences:

<s> Sam do like </s>
<s> Sam I am </s>
<s> I am Sam </s>

P(<s> Sam like </s>)=
=P(</s>|like)*P(like|Sam)*P(Sam|<s>) =
= ⅔ * ⅓ * ⅗ = 0,134

P(<s> Sam I am </s>)=

P(<s> I am Sam </s>)=

do it yourself

Language Models (Part 1.7)

4) Apply the Laplace smoothing on the language model
5) What are the new probabilities P* for:

<s> Sam do like </s>
<s> Sam I am </s>
<s> I am Sam </s>

do it yourself

SAPIENZA
UNIVERSITÀ DI ROMA

Part-of-Speech tagging (Part 1.8)

Hidden Markov Models: PoS tagger - Decoding

The Viterbi Algorithm

observations are words

states are PoS tags

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi*[N,T]
for each state s **from** 1 **to** N **do** ; initialization step
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
$bestpath \leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Figure 8.10 Viterbi algorithm for finding the optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A,B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence.

Hidden Markov Models: PoS tagger - Decoding

The Viterbi Algorithm

Given an hidden markov model HHM=(A, B):

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Figure 8.12 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

$Q=\{q_1,q_2, \dots , q_N\} = \{NNP,MD,VB,JJ,NN,RB,DT\}$

$a_{i,j}$ = transition probability = $P(q_j | q_i)$

$a_{0,j}= \pi_j = P(q_j | <s>)$

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 8.13 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

$O=\{o_1, o_2, \dots , o_T\} =\{Janet, will, back, the, bill\}$

$b_{i,j} = b_i(o_j)$ = emission probability $P(o_j | q_i)$

Hidden Markov Models: PoS tagger - Dec

The Viterbi Algorithm

Initialization:

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
viterbi[*s*,1] $\leftarrow \pi_s * b_s(o_1)$ **s=1**
backpointer[s,1] $\leftarrow 0$

	NNP	MD	VB	JJ	NN	RB	DT
<*s*>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

q_7=DT					
q_6=RB					
q_5=NN					
q_4=JJ					
q_3=VB					
q_2=MD					
q_1=NNP					
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Dec

The Viterbi Algorithm

Initialization:

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
viterbi[*s*,1] $\leftarrow \pi_s * b_s(o_1)$ s=1
backpointer[s,1] $\leftarrow 0$

	NNP	MD	VB	JJ	NN	RB	DT
<*s*>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

q_7=DT					
q_6=RB					
q_5=NN					
q_4=JJ					
q_3=VB					
q_2=MD					
q_1=NNP	**$\pi_1 * b_1(o_1)$ =.28*.000032**				
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Dec

The Viterbi Algorithm

Initialization:

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
viterbi[*s*,1] $\leftarrow \pi_s * b_s(o_1)$
backpointer[s,1] $\leftarrow 0$

s=1

	NNP	MD	VB	JJ	NN	RB	DT
<*s*>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

q_7=DT					
q_6=RB					
q_5=NN					
q_4=JJ					
q_3=VB					
q_2=MD					
q_1=NNP	**$\pi_1 * b_1(o_1)$ =.000009**				
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Dec

The Viterbi Algorithm

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Initialization:

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
$backpointer[s,1] \leftarrow 0$

s=2

q_7=DT					
q_6=RB					
q_5=NN					
q_4=JJ					
q_3=VB					
q_2=MD	**$\pi_2 * b_2(o_1)$**				
q_1=NNP	**$\pi_1 * b_1(o_1)$ =.000009**				
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Dec

The Viterbi Algorithm

Initialization:

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
viterbi[*s*,1] $\leftarrow \pi_s * b_s(o_1)$
backpointer[s,1] $\leftarrow 0$

s=7

	NNP	MD	VB	JJ	NN	RB	DT
<*s*>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

q_7=DT	$\pi_7 * b_7(o_1)$				
q_6=RB	$\pi_6 * b_6(o_1)$				
q_5=NN	$\pi_5 * b_5(o_1)$				
q_4=JJ	$\pi_4 * b_4(o_1)$				
q_3=VB	$\pi_3 * b_3(o_1)$				
q_2=MD	$\pi_2 * b_2(o_1)$				
q_1=NNP	$\pi_1 * b_1(o_1)$ =.000009				
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Dec

The Viterbi Algorithm

Recursion step:

for each time step t **from** 2 **to** T **do** ; recursion step

  for each state s **from** 1 **to** N **do**

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

q_7=DT	.0				
q_6=RB	.0				
q_5=NN	.0				
q_4=JJ	.0				
q_3=VB	.0				
q_2=MD	.0				
q_1=NNP	.000009				
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Dec

The Viterbi Algorithm

Recursion step:

for each time step t **from** 2 **to** T **do** ; recursion step

t=2
s=1

for each state s **from** 1 **to** N **do**

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

q_7=DT	.0 *$a_{7,1}$*$b_1(o_2)$				
q_6=RB	.0				
q_5=NN	.0				
q_4=JJ	.0				
q_3=VB	.0				
q_2=MD	.0 *$a_{2,1}$*$b_1(o_2)$				
q_1=NNP	.000009 *$a_{1,1}$*$b_1(o_2)$				
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Dec

The Viterbi Algorithm

Recursion step:

for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

t=2
s=1

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

q_7=DT	.0	0			
q_6=RB	.0				
q_5=NN	.0				
q_4=JJ	.0				
q_3=VB	.0				
q_2=MD	.0	0			
q_1=NNP	.000009	*0.3777*0	.0		
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Dec

The Viterbi Algorithm

Recursion step:

for each time step t **from** 2 **to** T **do**
  for each state s **from** 1 **to** N **do**

t=2
s=1

; recursion step

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

q_7=DT	.0				
q_6=RB	.0				
q_5=NN	.0				
q_4=JJ	.0				
q_3=VB	.0				
q_2=MD	.0				
q_1=NNP	.000009	.0			
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Dec

The Viterbi Algorithm

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Recursion step:

t=2
s=2

for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

q_7=DT	.0				
q_6=RB	.0				
q_5=NN	.0				
q_4=JJ	.0				
q_3=VB	.0				
q_2=MD	.0	2.772e-8			
q_1=NNP	.000009	.0			
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Dec

The Viterbi Algorithm

Recursion step:

for each time step t **from** 2 **to** T **do** ; recursion step

  for each state s **from** 1 **to** N **do**

$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

q_7=DT	.0	.0		>0	
q_6=RB	.0	.0			
q_5=NN	.0	.0000000001			>0
q_4=JJ	.0	.0			
q_3=VB	.0	2.5e-13	>0		
q_2=MD	.0	2.772e-8			
q_1=NNP	.000009	.0			
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Decoding

The Viterbi Algorithm

termination step:

$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step

$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step

bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time

return *bestpath*, *bestpathprob*

we search for the max value in the last column e.g., $v_{5,5}$

q_7=DT	.0	.0		>0	
q_6=RB	.0	.0			
q_5=NN	.0	.0000000001			>0
q_4=JJ	.0	.0			
q_3=VB	.0	2.5e-13	>0		
q_2=MD	.0	2.772e-8			
q_1=NNP	.000009	.0			
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Decoding

The Viterbi Algorithm

termination step:

$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step

$bestpathpointer \leftarrow \text{argmax}_{s=1}^{N} viterbi[s,T]$; termination step

bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time

return *bestpath*, *bestpathprob*

we backtrack until we reach π

q_7=DT	.0	.0		>0	
q_6=RB	.0	.0			
q_5=NN	.0	.0000000001			>0
q_4=JJ	.0	.0			
q_3=VB	.0	2.5e-13	>0		
q_2=MD	.0	2.772e-8			
q_1=NNP	.000009	.0			
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Hidden Markov Models: PoS tagger - Decoding

The Viterbi Algorithm

q_7=DT	.0	.0		>0	
q_6=RB	.0	.0			
q_5=NN	.0	.0000000001			>0
q_4=JJ	.0	.0			
q_3=VB	.0	2.5e-13	>0		
q_2=MD	.0	2.772e-8			
q_1=NNP	.000009	.0			
π	o_1=Janet	o_2=will	o_3=back	o_4=the	o_5=bill

Janet/**NNP** will/**MD** back/**VB** the/**DT** bill/**NN**

SAPIENZA
UNIVERSITÀ DI ROMA

Vector Semantics (sparse) (Part 1.11)

tf-idf: exercise

- **Corpus**:

d_1: “Frodo accidentally stabbed Sam and then some orcs”

d_2: “Frodo was stabbing regular orcs but never stabbed super orcs – Uruk-Hais”

d_3: “Sam was having a barbecue with some friendly orcs”

tf-idf: exercise

- **Corpus**:

d_1: "Frodo accidentally stabbed Sam and then some orcs"

d_2: "Frodo was stabbing regular orcs but never stabbed super orcs – Uruk-Hais"

d_3: "Sam was having a barbecue with some friendly orcs"

- **Compute tf-idf for each word in "Frodo stabs orc":**

tf-idf: exercise

- **Corpus**:

d_1: "Frodo accidentally stabbed Sam and then some orcs"

d_2: "Frodo was stabbing regular orcs but never stabbed super orcs – Uruk-Hais"

d_3: "Sam was having a barbecue with some friendly orcs"

- **Compute tf-idf for each word in "Frodo stabs orc":** idf("Frodo") = $\log_{10}(3/2) = .176$

tf("Frodo", d_1) = 1

tf("Frodo", d_2) = 1

tf("Frodo", d_3) = 0

tf-idf("F.", d_1) = tf("Frodo",d_1) x idf("Frodo")=.176

…

tf-idf: exercise

- **Corpus**:

d_1: "Frodo accidentally stabbed Sam and then some orcs"

d_2: "Frodo was stabbing regular orcs but never stabbed super orcs – Uruk-Hais"

d_3: "Sam was having a barbecue with some friendly orcs"

- **Compute tf-idf for each word in "Frodo stabs orc":** idf("Frodo") = $\log_{10}(3/2)$ = .176

tf("Frodo", d_1) = 1

tf("Frodo", d_2) = 1

tf("Frodo", d_3) = 0

tf-idf("F.", d_1) = tf("Frodo",d_1) x idf("Frodo")=.176

…

PPMI: exercise

Consider a **term-context matrix** F with W rows and C columns; f_{ij} is number of times the word w_i occurs in context c_j

or co-occurs with the word c_j

Count(w,context)

	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

PPMI: exercise

Calculate a P matrix with W rows and C columns; $p_{ij} = p(w_i, c_j)$

Count(w,context)

	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$p(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{N}$$

$$N = \sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}$$

$$p(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{N}$$

p(w,context) **p(w)**

	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

PPMI: exercise

Calculate the PPMI matrix with W rows and C columns; $ppmi_{ij}$ =PPMI(w_i, c_j)

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

$$\text{PPMI}(w_1, w_2) = \max\left(\log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}, 0\right)$$

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

PPMI: exercise

Calculate the PPMI matrix with W rows and C columns; $ppmi_{ij}$ =PPMI(w_i, c_j)

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

$$\text{PPMI}(w_1, w_2) = \max\left(\log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}, 0\right)$$

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

Q&A

**Credits

The slides of this part of the course are the result of a personal reworking of the slides and of the course material from different sources:

1. The NLP course of Prof. Roberto Navigli, Sapienza University of Rome
2. The NLP course of Prof. Simone Paolo Ponzetto, University of Mannheim, Germany
3. The NLP course of Prof. Chris Biemann, University of Hamburg, Germany
4. The NLP course of Prof. Dan Jurafsky, Stanford University, USA