

String Manipulation in Python

Kristoffer Nielbo

Center for Humanities Computing Aarhus|chcaa.io
aarhus university, denmark



CENTER FOR HUMANITIES
COMPUTING AARHUS



STRINGS

Index from rear:	-6	-5	-4	-3	-2	-1	
Index from front:	0	1	2	3	4	5	
	+---+---+---+---+---+---+						
	a	b	c	d	e	f	
	+---+---+---+---+---+---+						
Slice from front:	:	1	2	3	4	5	:
Slice from rear:	:	-5	-4	-3	-2	-1	:

- immutable sequence data type
- sequence of Unicode characters wrapped inside single, double, or triple quotes
- an ordered collection of characters

WORDS FREQUENCIES

I am Daniel 'I' 'am' 'Daniel'
 I am Sam 'I' 'am' 'Sam'
 Sam I am 'Sam' 'I' 'am'
 That Sam-I-am 'That' 'Sam' 'I'
 That Sam-I-am! 'am' 'That' 'Sam'
 I do not like 'I' 'am' 'I' 'do'
 that Sam-I-am 'not' 'like' 'that'
 ... 'Sam' 'I' 'am' ...

a	1	59	0.073
am	1	16	0.02
and	1	24	0.03
anyhwhere	1	1	0.001
anywhere	1	7	0.009
...			
you	1	34	0.042
<i>total</i>	55	804	1.0

WORD IMPORTANCE

raw term frequency:

$$f_{t,d} = N(t, d)$$

normalized term frequency (prevents bias towards longer documents)

$$f_{t,d} = \frac{N(t, d)}{N(d)}$$

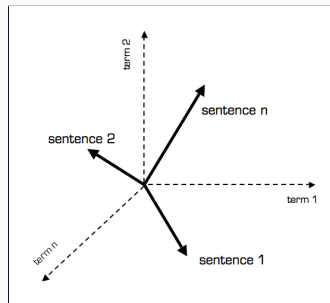
IDF weighted term frequency (removes non-informative words):

$$tfidf(t, d, D) = f_{t,d} \cdot \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

VECTOR SPACE MODEL (VSM)

Any collection of m documents can be represented in the vector space model by a document-term matrix of m documents and n terms

- a document vector with only one word is collinear to the vocabulary word axis
- a document vector that does not contain a specific word is orthogonal/perpendicular to the word axis
- two documents are identical if they contain the same words in a different order (BOW assumption)



Document space		t_1	t_2	t_3	...	t_n	← Term vector space
D_1		a_{11}	a_{12}	a_{13}	...	a_{1n}	
D_2		a_{21}	a_{22}	a_{23}	...	a_{2n}	
D_3		a_{31}	a_{32}	a_{33}	...	a_{3n}	
...							
D_m		a_{m1}	a_{m2}	a_{m3}	...	a_{mn}	
Q		b_1	b_2	b_3	...	b_n	

DOCUMENT COMPARISON

```
1 texts = ['hello world', 'hello foobar']  
2 lexicon = ['foobar', 'hello', 'world']  
3 dtm = [[0, 1, 1], [1, 1, 0]]
```

$$\text{document similarity} = \frac{\mathbf{D}_1 \cdot \mathbf{D}_2}{\|\mathbf{D}_1\| \|\mathbf{D}_2\|}$$

```
1 from scipy import spatial  
2 print('[INFO] Cosine similarity for pairwise document comparison')  
3 print('[INFO] Document 1 and 2:')  
4 print(1 - spatial.distance.cosine(dtm[0],dtm[1]))
```

On this implementation, similarity ranges from 1 meaning exactly the same, with 0 indicating orthogonality or decorrelation.

```
1 [INFO] Cosine similarity for pairwise document comparison  
2 [INFO] Document 1 and 2:  
3 0.5
```

```
1 if questions:
2     try:
3         answer()
4     except RuntimeError:
5         pass
6     else:
7         print('break')
```