Topic modeling

Martin Kroon

Hacking the Humanities
19 November 2018

What is topic modeling?

trying to discover abstract "topics" that occur in a collection of documents

Topics

a topic is a collection of words, each assigned a different probability of existing within a topic

the "topics" produced by topic modeling techniques are clusters of similar words

documents can then be for x% about topic X and y% about topic Y

What we get

		words		
		w_1	w_2	...
	t_1	0.05	0.32	...
topics	t_2	0.041	0.13	...
	...	...	...	...

LDA

Latent Dirichlet Allocation

“uncovers the “latent” structure of a document based on the words therein”

set amount of topics!

clean data!

LDA Buffet

do check out the LDA Buffet anecdote by Matthew Jockers to get a grasp of what it does

gensim

we use the gensim package to run LDA

it talks to MALLET, so we also need that (needs JDK, not JRE)

Code 1

23_gensim1_prep.py

Output 1

[(0, 1), (1, 10), (2, 1), (3, 2), (4, 1), (5, 2), (6, 1), (7, 6), (8, 1), (9, 1)]
[('charing', 1), ('three', 10), ('easy', 1), ('leave', 2), ('calf', 1), ('rich', 2), ('idler', 1), ('side', 6), ('strict', 1), ('edge', 1)]

Code 2

23_gensim2_topicmodel1.py

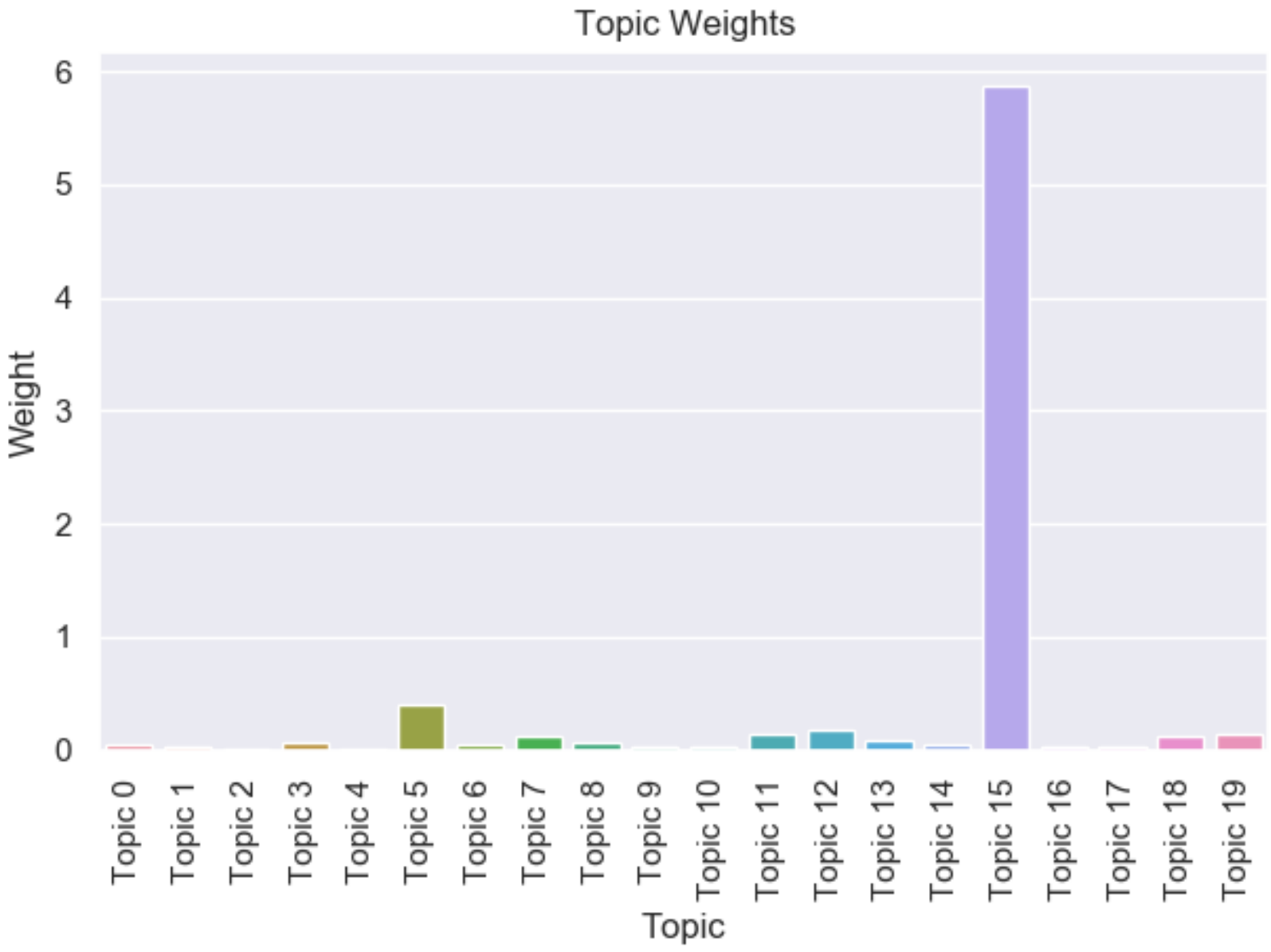
Output 2

(0, '0.018*"paper" + 0.018*"openshaw" + 0.016*"letter" + 0.014*"uncle" + 0.011*"horsham"')
(1, '0.037*"simon" + 0.034*"lord" + 0.018*"frank" + 0.014*"lady" + 0.013*"lestrade"')
(2, '0.032*"mccarthy" + 0.025*"father" + 0.022*"son" + 0.019*"lestrade" + 0.017*"turner"')
(3, '0.021*"photograph" + 0.019*"king" + 0.019*"majesty" + 0.017*"irene" + 0.015*"adler"')
(4, '0.026*"clair" + 0.018*"neville" + 0.016*"inspector" + 0.011*"den" + 0.011*"lascar"')
(5, '0.012*"policeman" + 0.012*"tie" + 0.009*"leaf" + 0.009*"tobacco" + 0.006*"amateur"')
(6, '0.026*"hosmer" + 0.025*"angel" + 0.024*"windibank" + 0.017*"father" + 0.012*"mother"')
(7, '0.020*"french" + 0.013*"sweeping" + 0.013*"larger" + 0.013*"instance" + 0.013*"longer"')
(8, '0.025*"coronet" + 0.018*"arthur" + 0.017*"holder" + 0.014*"son" + 0.014*"mary"')
(9, '0.026*"colonel" + 0.014*"machine" + 0.013*"mile" + 0.013*"hydraulic" + 0.012*"stark"')
(10, '0.021*"wilson" + 0.015*"league" + 0.014*"assistant" + 0.013*"advertisement" + 0.012*"merryweather"')
(11, '0.013*"young" + 0.011*"lonely" + 0.009*"drawer" + 0.009*"situation" + 0.009*"letter"')
(12, '0.022*"low" + 0.012*"tree" + 0.010*"handkerchief" + 0.010*"weary" + 0.010*"madam"')
(13, '0.023*"german" + 0.019*"forefinger" + 0.019*"landau" + 0.014*"temple" + 0.009*"hum"')
(14, '0.021*"bed" + 0.020*"sister" + 0.019*"roylott" + 0.018*"stoner" + 0.014*"ventilator"')
(15, '0.065*"wa" + 0.021*"holmes" + 0.014*"ha" + 0.014*"man" + 0.008*"hand"')
(16, '0.038*"project" + 0.034*"work" + 0.017*"rucastle" + 0.015*"miss" + 0.013*"gutenberg"')
(17, '0.038*"goose" + 0.022*"hat" + 0.020*"bird" + 0.018*"stone" + 0.014*"baker"')
(18, '0.016*"faded" + 0.016*"forever" + 0.010*"prove" + 0.010*"criminal" + 0.010*"nonsense"')
(19, '0.013*"interesting" + 0.010*"traced" + 0.010*"candle" + 0.010*"income" + 0.010*"agitation"')

Code 3

23_gensim2_topicmodel2.py

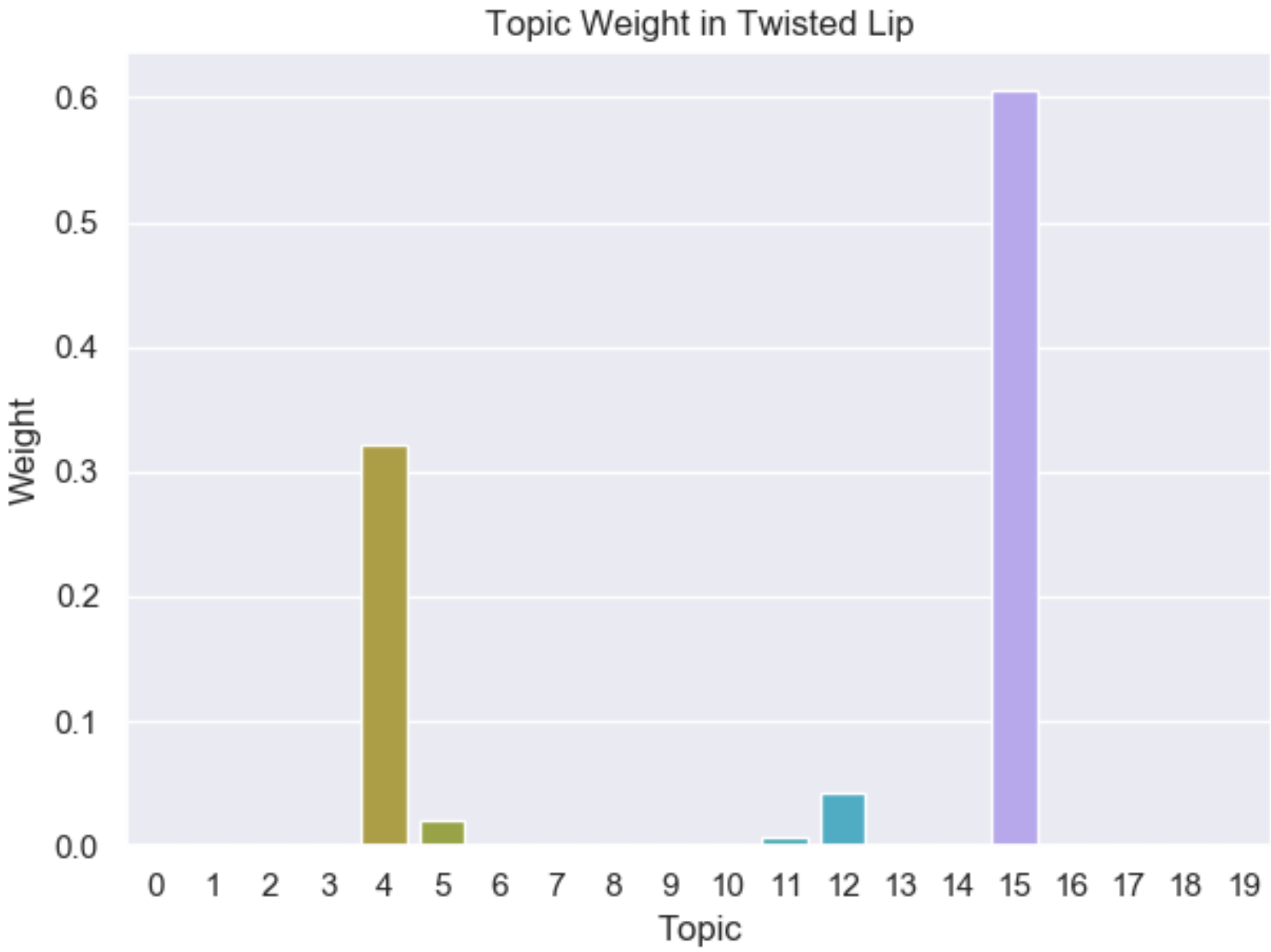
Output 3


Topic Weights
Weight
0
1
2
3
4
5
6
Topic 0
Topic 1
Topic 2
Topic 3
Topic 4
Topic 5
Topic 6
Topic 7
Topic 8
Topic 9
Topic 10
Topic 11
Topic 12
Topic 13
Topic 14
Topic 15
Topic 16
Topic 17
Topic 18
Topic 19
Topic

Code 4

23_gensim2_topicmodel3.py

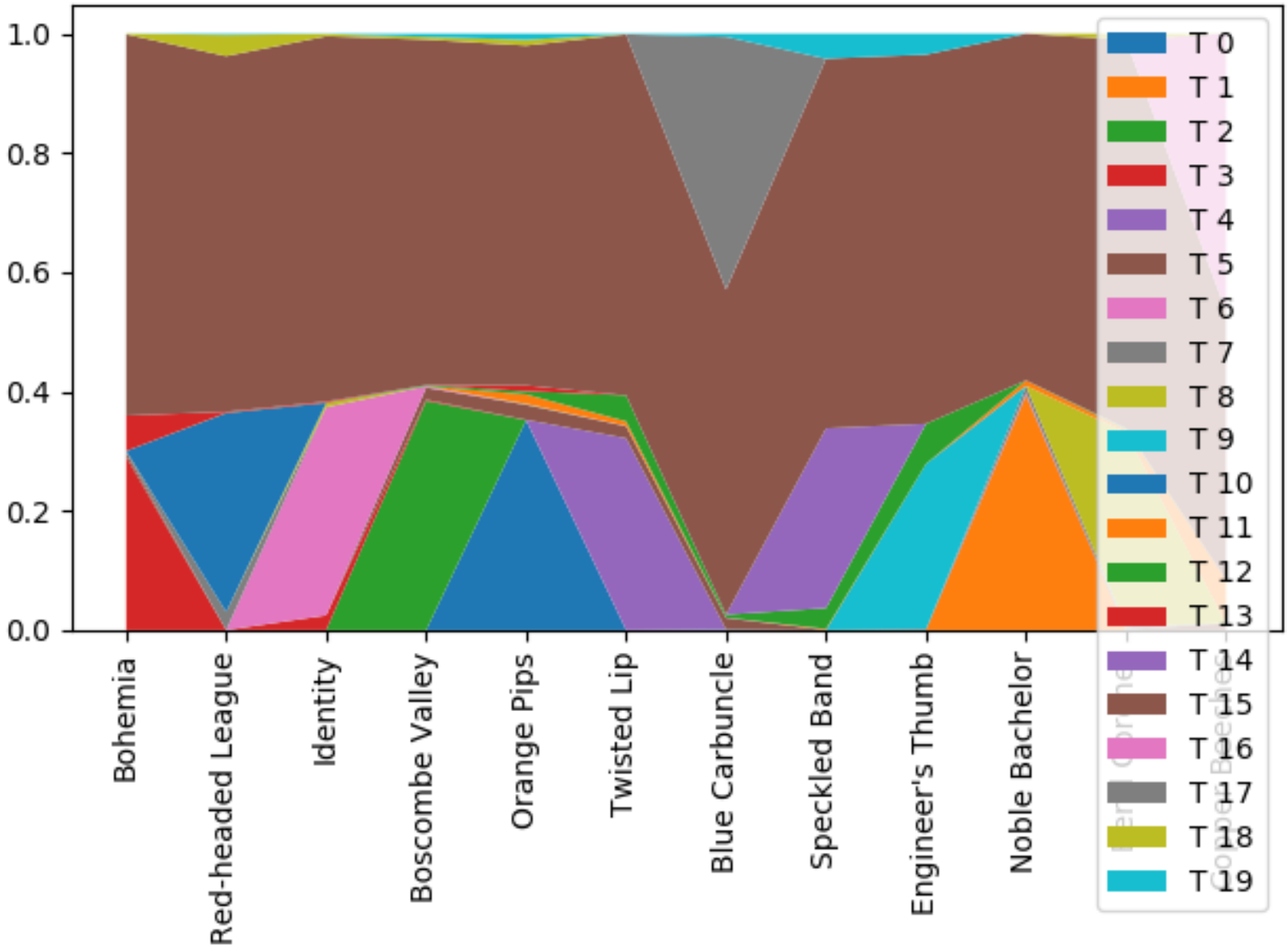
Output 4


Topic Weight in Twisted Lip
Weight
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Topic

Code 5

23_gensim2_topicmodel4.py

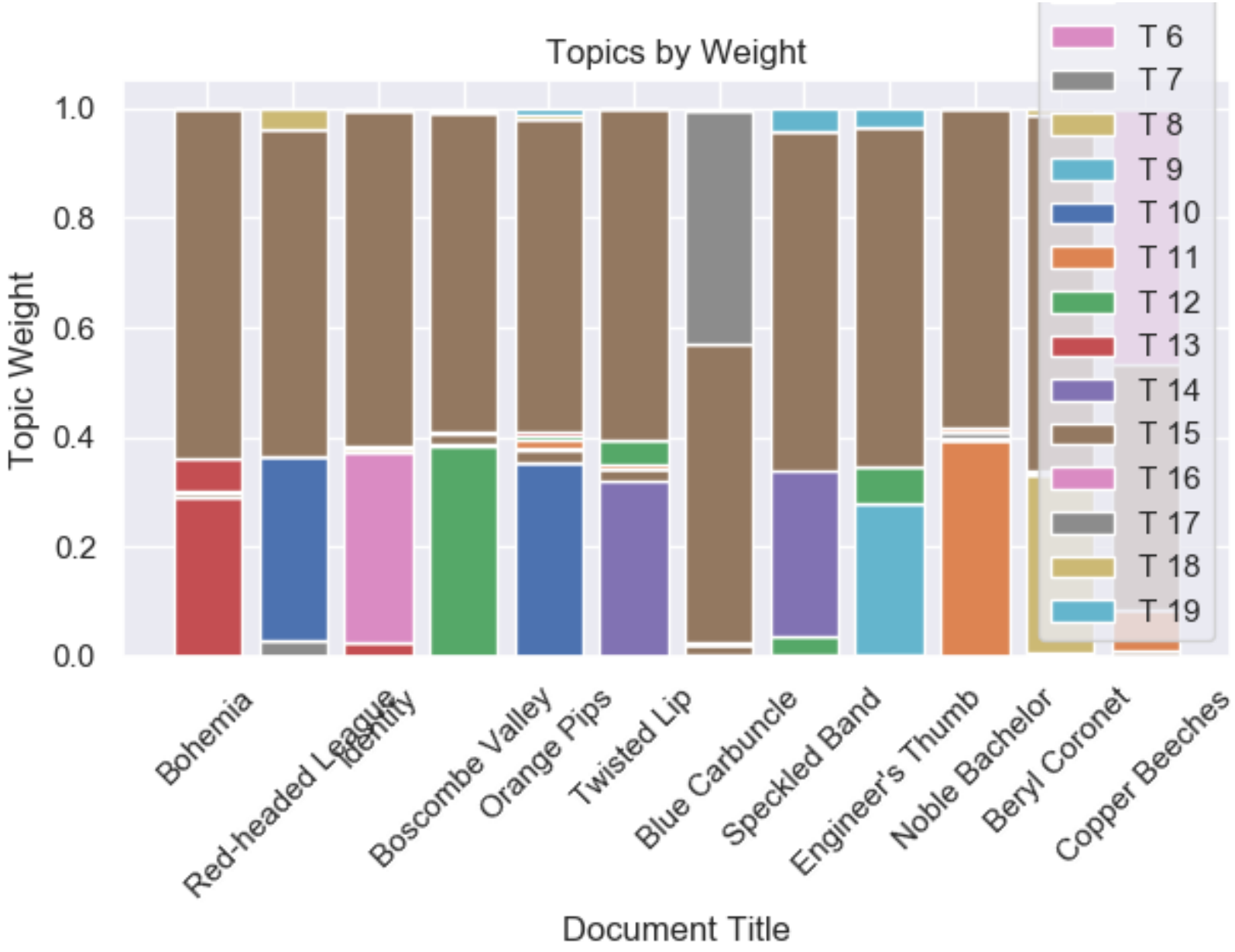
Output 5


1.0
0.8
0.6
0.4
0.2
0.0
Bohemia
Red-headed League
Identity
Boscombe Valley
Orange Pips
Twisted Lip
Blue Carbuncle
Speckled Band
Engineer's Thumb
Noble Bachelor
T 0
T 1
T 2
T 3
T 4
T 5
T 6
T 7
T 8
T 9
T 10
T 11
T 12
T 13
T 14
T 15
T 16
T 17
T 18
T 19

Code 6

23_gensim2_topicmodel5.py

Output 6


Topics by Weight
Topic Weight
1.0
0.8
0.6
0.4
0.2
0.0
Bohemia
Red-headed League
Identity
Boscombe Valley
Orange Pips
Twisted Lip
Blue Carbuncle
Speckled Band
Engineer's Thumb
Noble Bachelor
Beryl Coronet
Copper Beeches
Document Title
T 6
T 7
T 8
T 9
T 10
T 11
T 12
T 13
T 14
T 15
T 16
T 17
T 18
T 19

Word embedding

abstract vector representations of words

based on the assumption that similar words tend to appear in similar contexts

closely related words tend to be embedded close to each other

powerful, because you can perform math on the vectors

Word embedding: maths

$$\vec{king} - \vec{man} = \vec{queen} - \vec{woman}$$

$$\vec{king} - \vec{man} + \vec{woman} = \vec{queen}$$

Code 7

23_gensim3_word2vec.py

Output 7

[('u', 0.999778151512146), ('went', 0.999760508537292 5), ('away', 0.99972641468048 1), ('long', 0.999721527099609 4), ('having', 0.999718725681304 9), ('son', 0.999716818332672 1), ('step', 0.999713778495788 6), ('end', 0.999713480472564 7), ('half', 0.999712705612182 6), ('great', 0.999708473682403 6)] is similar to 'him'
Woman plus gentleman minus man is [('oh', 0.999465823173523), ('must', 0.999442756175994 9), ('nothing', 0.999424934387207), ('anything', 0.999417304992675 8), ('ask', 0.999403297901153 6), ('why', 0.999402225017547 6), ('find', 0.999385893344879 2), ('now', 0.999385476112365 7), ('understand', 0.999354839324951 2), ('done', 0.999316036701202 4)]

Final remark

Corpus size!

millions of words, preferably billions...