Natural language processing

IMDB Ratings

by Andreea, Hala, Hritika

SUMMARY

- About NLP
- Applications of NLP

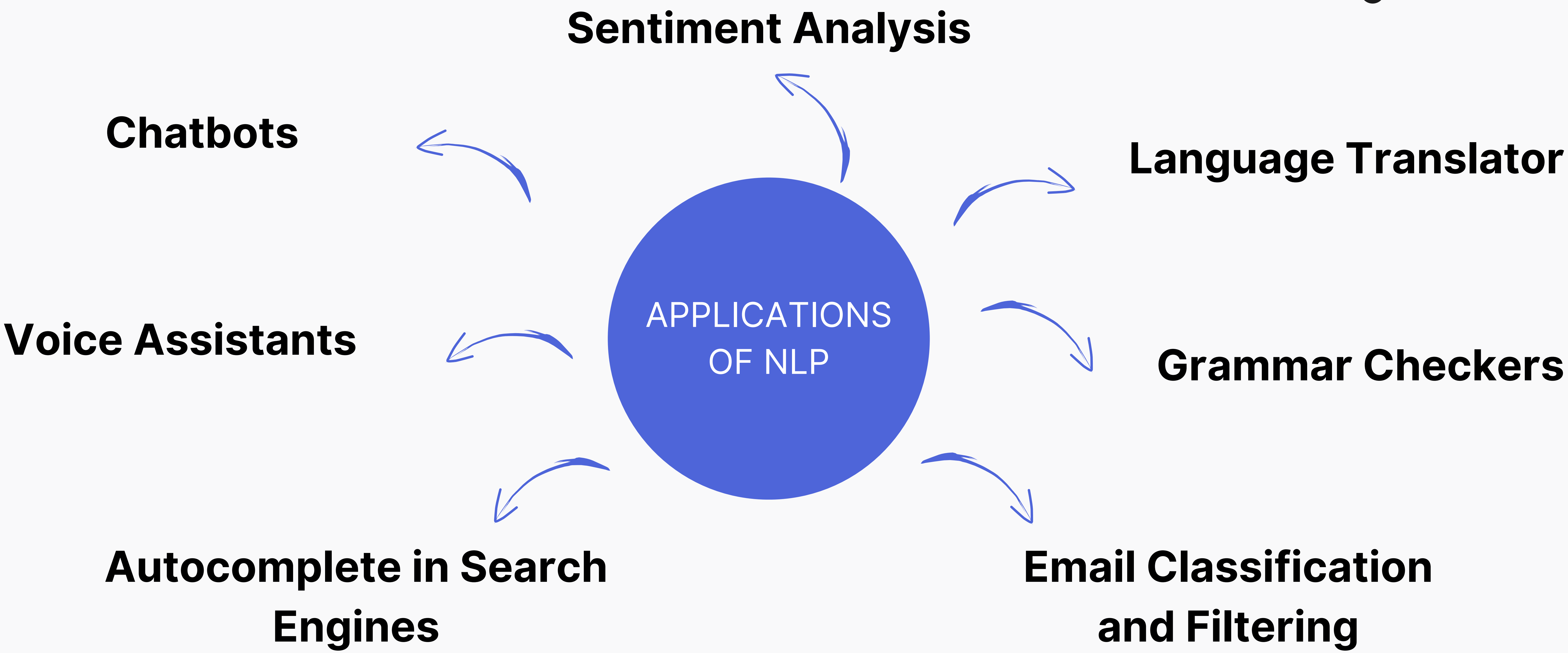
Overview of our data

- Pre processing steps
- Bag of words
- Classification model

- Word2Vector
- Vector Averaging
- Clustering

What is NLP?

giving computers the ability to understand text and spoken words in much the same way human beings can



APPLICATIONS OF NLP

Sentiment Analysis

Chatbots

Language Translator

Voice Assistants

Grammar Checkers

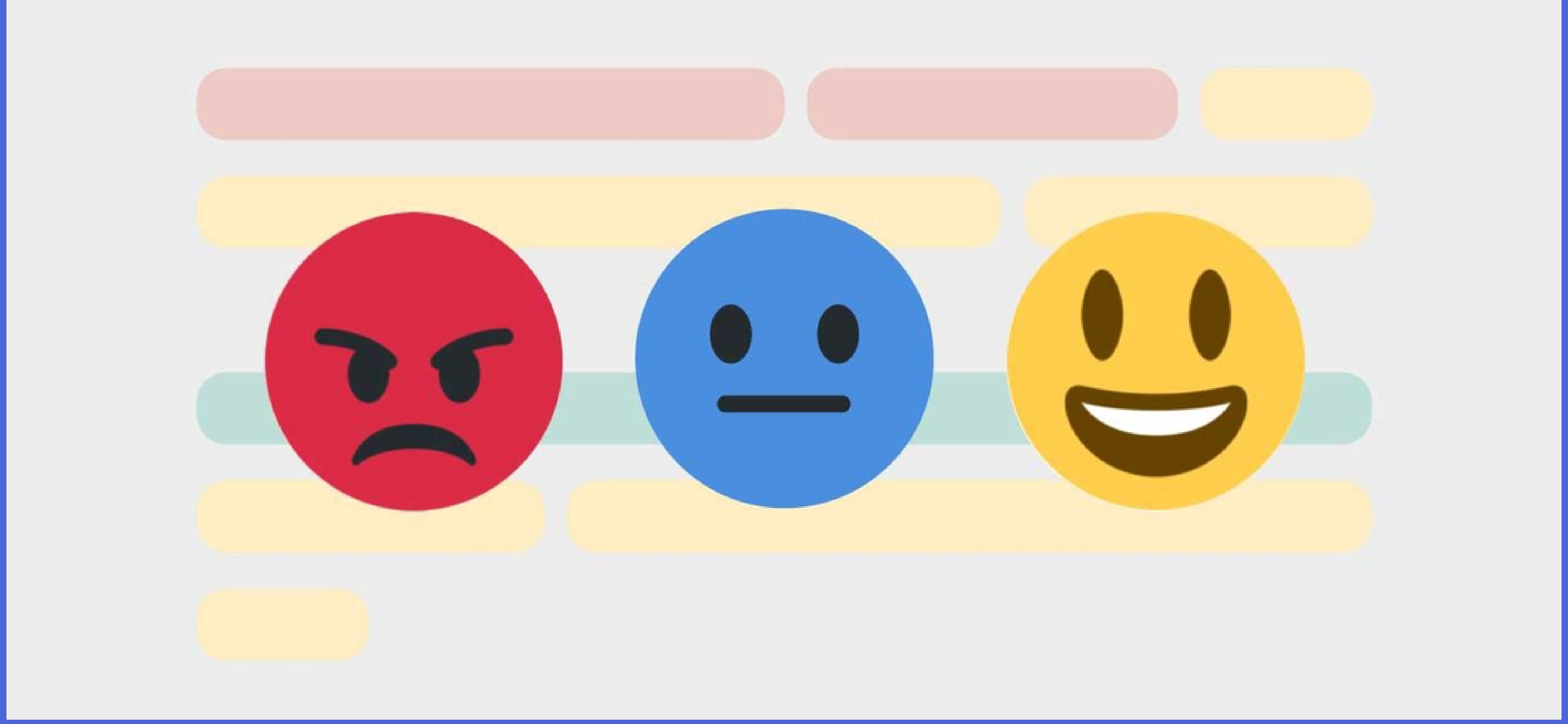
Autocomplete in Search Engines

Email Classification and Filtering

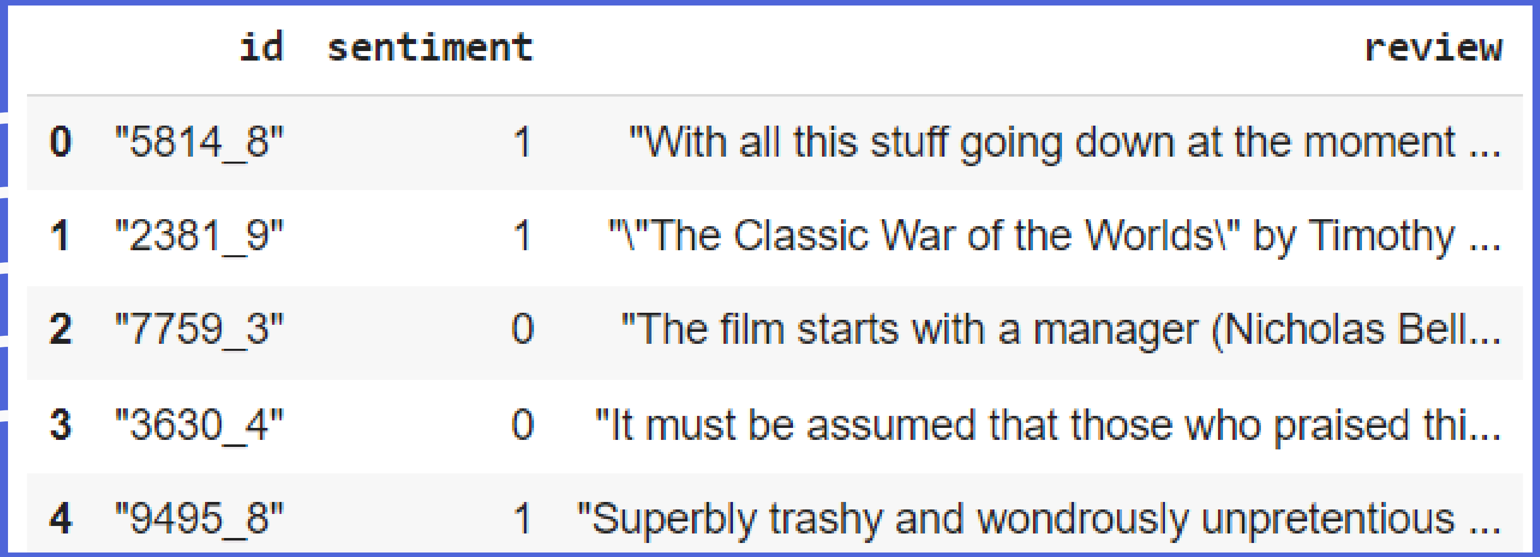
What is our application?

Sentiment Analysis

identifies the emotional tone behind a body of text



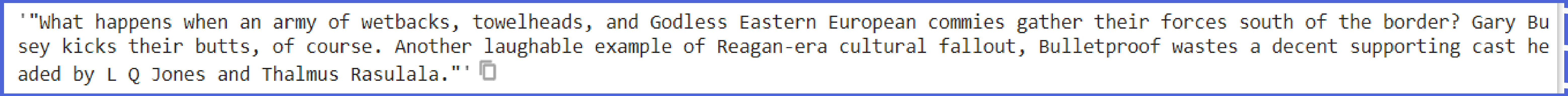
Overview of IMBD Ratings



	id	sentiment	review
0	"5814_8"	1	"With all this stuff going down at the moment ...
1	"2381_9"	1	"\"The Classic War of the Worlds\" by Timothy ...
2	"7759_3"	0	"The film starts with a manager (Nicholas Bell...
3	"3630_4"	0	"It must be assumed that those who praised thi...
4	"9495_8"	1	"Superbly trashy and wondrously unpretentious ...

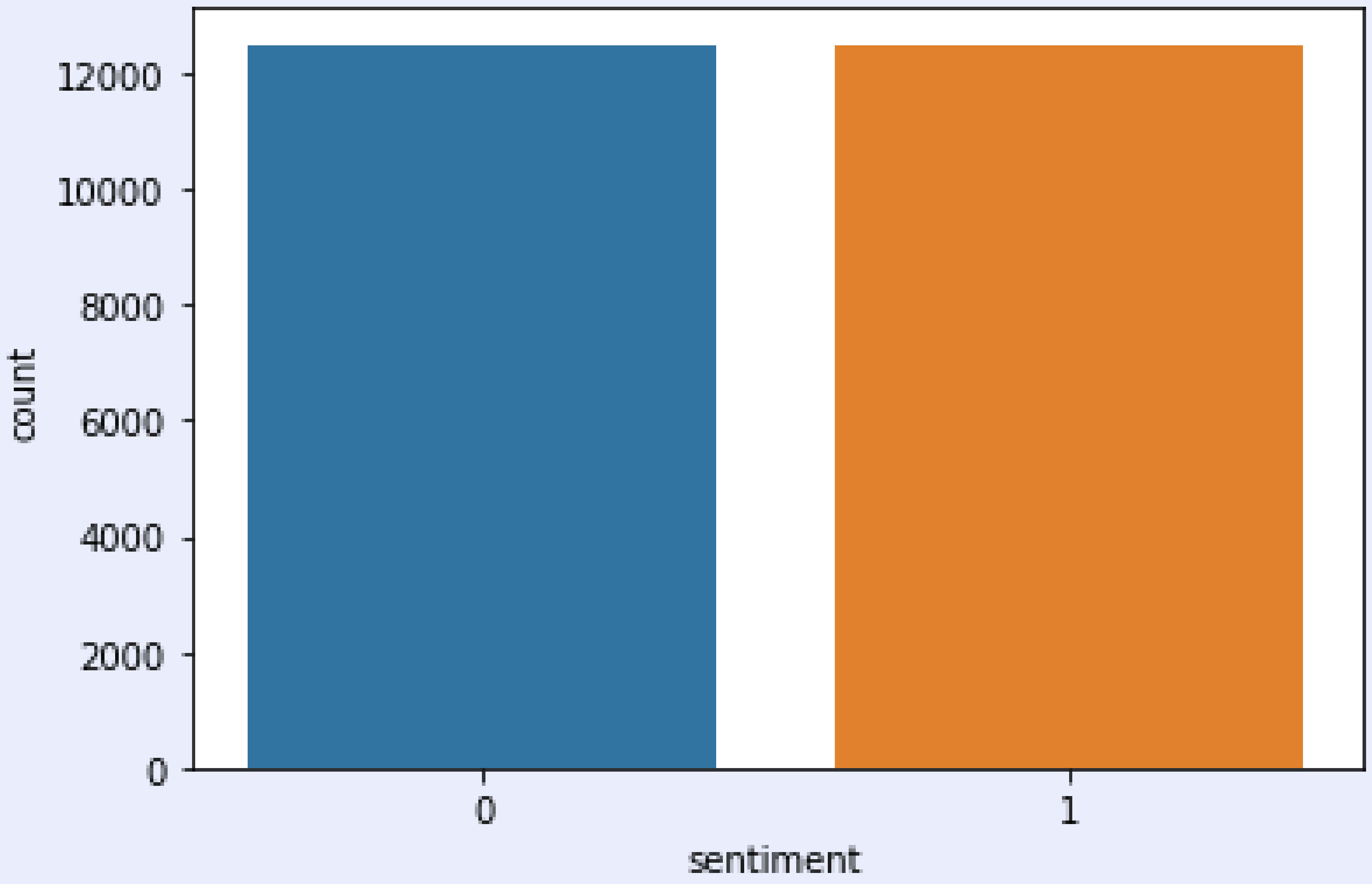
25000 reviews

An example of a review

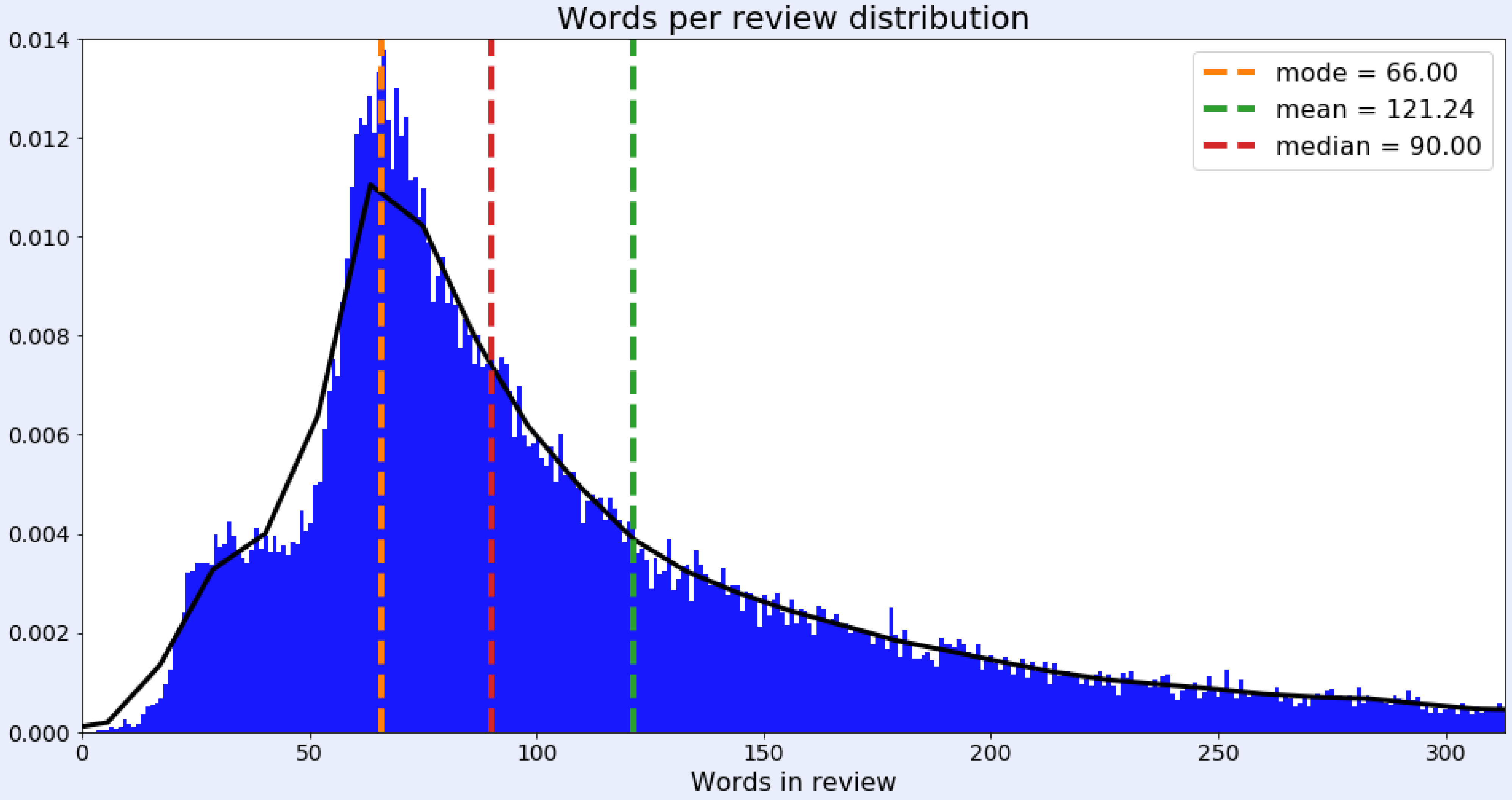


```
'"What happens when an army of wetbacks, towelheads, and Godless Eastern European commies gather their forces south of the border? Gary Bu
sey kicks their butts, of course. Another laughable example of Reagan-era cultural fallout, Bulletproof wastes a decent supporting cast he
aded by L Q Jones and Thalmus Rasulala."'
```

Visualisation

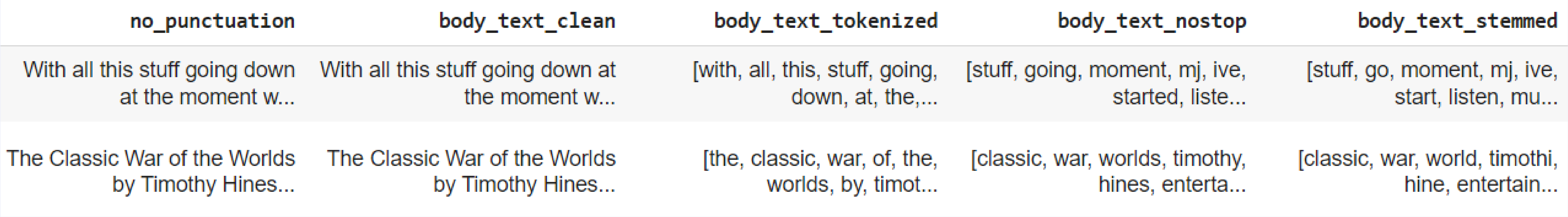


Visualisation



Pre-processing

- Removing punctation
- Lower case only
- Tokenization
- Removing Stop Words
- Stemming



no_punctuation	body_text_clean	body_text_tokenized	body_text_nostop	body_text_stemmed
With all this stuff going down at the moment w...	With all this stuff going down at the moment w...	[with, all, this, stuff, going, down, at, the,...	[stuff, going, moment, mj, ive, started, liste...	[stuff, go, moment, mj, ive, start, listen, mu...
The Classic War of the Worlds by Timothy Hines...	The Classic War of the Worlds by Timothy Hines...	[the, classic, war, of, the, worlds, by, timot...	[classic, war, worlds, timothy, hines, enterta...	[classic, war, world, timothi, hine, entertain...

STEPS

Tokenization

with, all, this, stuff, going, down, at, the

Stemming

stuff, go, moment, mj, ive, start, listen

Lemmatization

reducing a word to its base form

went” is changed to “go”

Stop words

i', 'me', 'my','myself', 'we', 'our', 'ours', 'ourselves', 'you',

Part of Speech (POS)

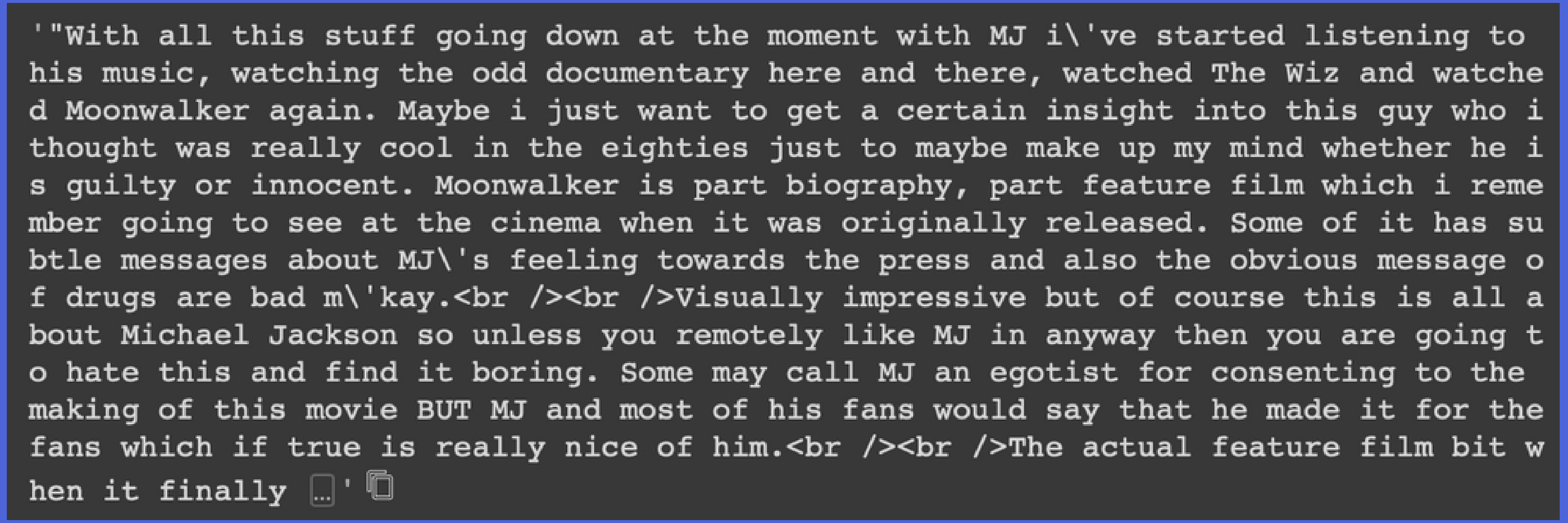
nouns, pronouns, adjectives, verbs, adverbs,

Named-entity-recognition

person names, organizations, locations

We're going to talk about...

Pre processing



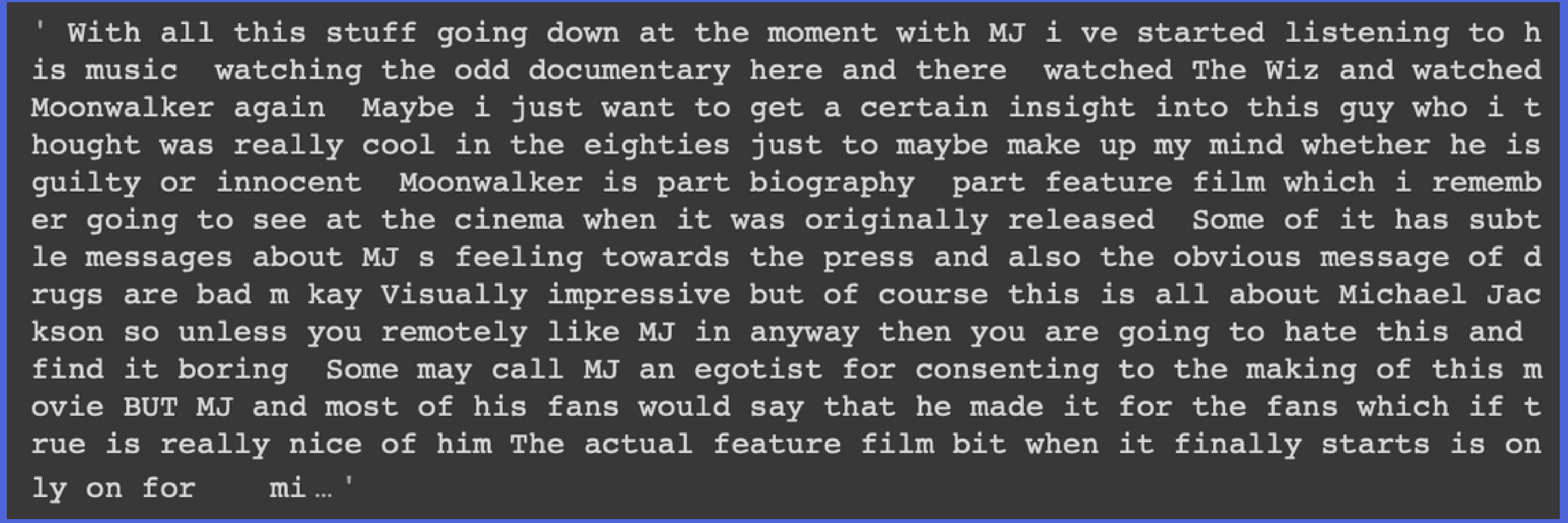
```
'"With all this stuff going down at the moment with MJ i\'ve started listening to
his music, watching the odd documentary here and there, watched The Wiz and watche
d Moonwalker again. Maybe i just want to get a certain insight into this guy who i
thought was really cool in the eighties just to maybe make up my mind whether he i
s guilty or innocent. Moonwalker is part biography, part feature film which i reme
mber going to see at the cinema when it was originally released. Some of it has su
btle messages about MJ\'s feeling towards the press and also the obvious message o
f drugs are bad m\'kay.<br /><br />Visually impressive but of course this is all a
bout Michael Jackson so unless you remotely like MJ in anyway then you are going t
o hate this and find it boring. Some may call MJ an egotist for consenting to the
making of this movie BUT MJ and most of his fans would say that he made it for the
fans which if true is really nice of him.<br /><br />The actual feature film bit w
hen it finally …'
```

Using Beautiful Soup

Splitting, and lower case

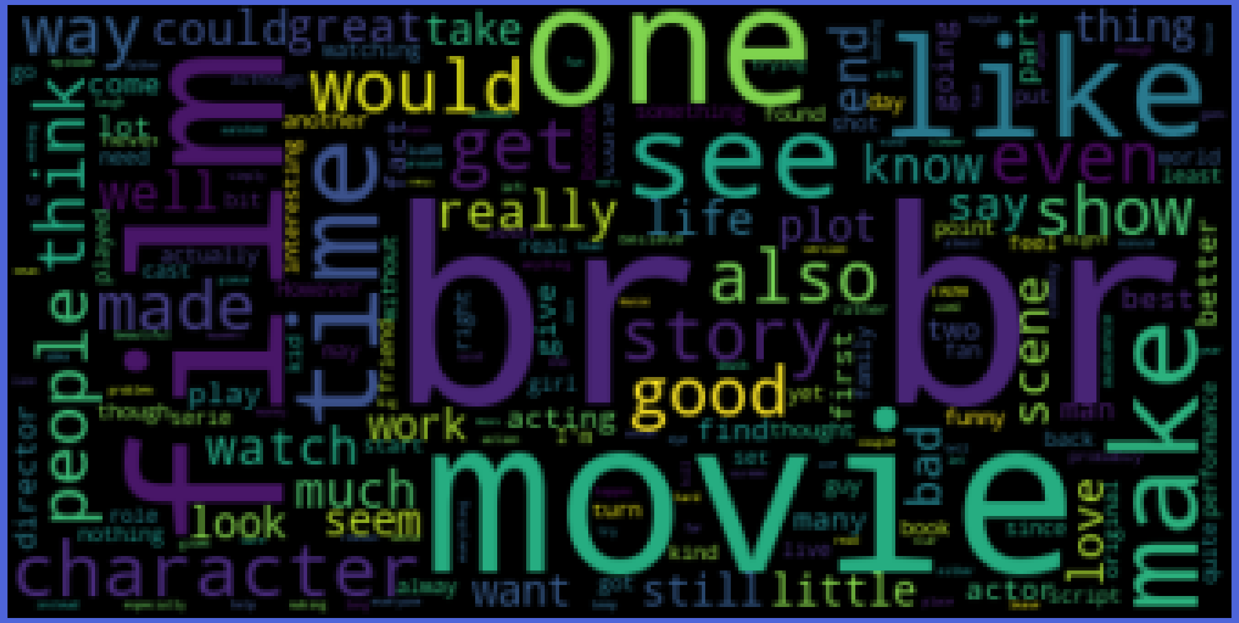
Only letters

Stopwords -yes and no

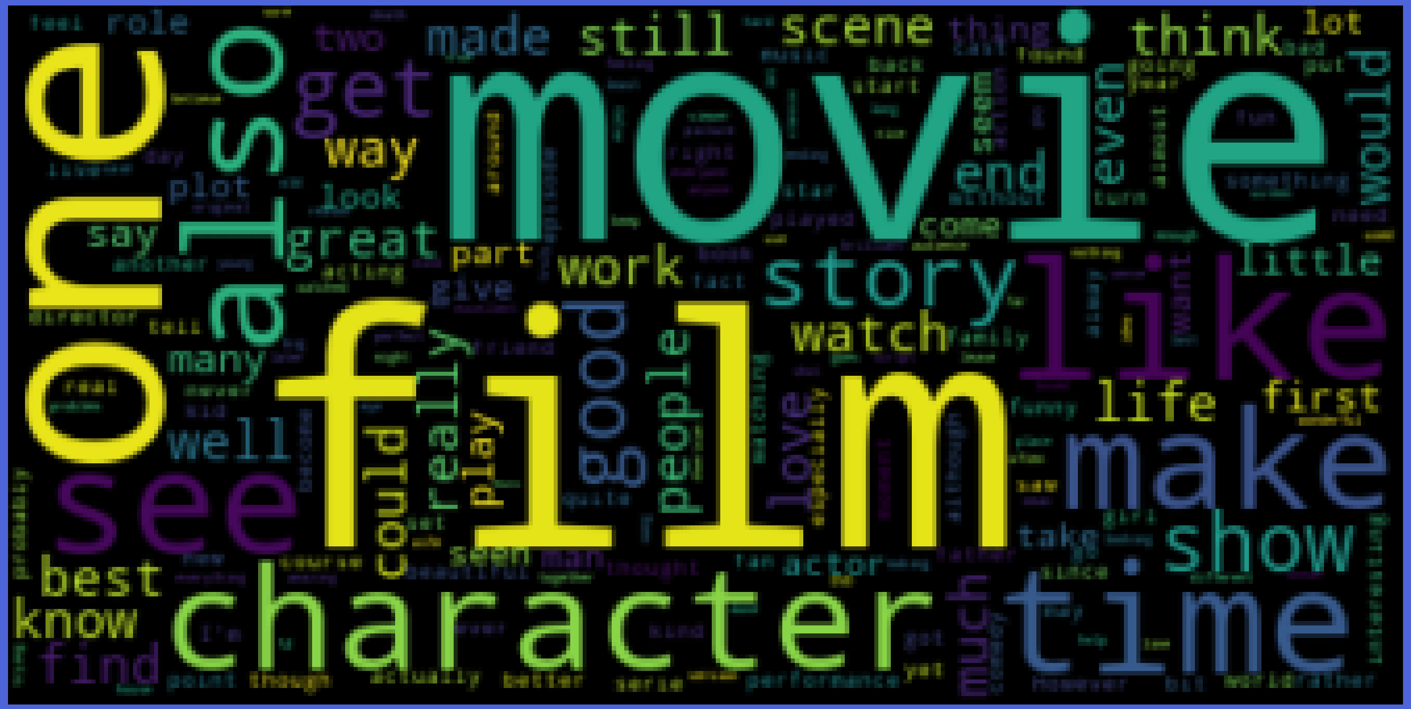


```
' With all this stuff going down at the moment with MJ i ve started listening to h
is music  watching the odd documentary here and there  watched The Wiz and watched
Moonwalker again  Maybe i just want to get a certain insight into this guy who i t
hought was really cool in the eighties just to maybe make up my mind whether he is
guilty or innocent  Moonwalker is part biography  part feature film which i rememb
er going to see at the cinema when it was originally released  Some of it has subt
le messages about MJ s feeling towards the press and also the obvious message of d
rugs are bad m kay Visually impressive but of course this is all about Michael Jac
kson so unless you remotely like MJ in anyway then you are going to hate this and
find it boring  Some may call MJ an egotist for consenting to the making of this m
ovie BUT MJ and most of his fans would say that he made it for the fans which if t
rue is really nice of him The actual feature film bit when it finally starts is on
ly on for    mi…'
```

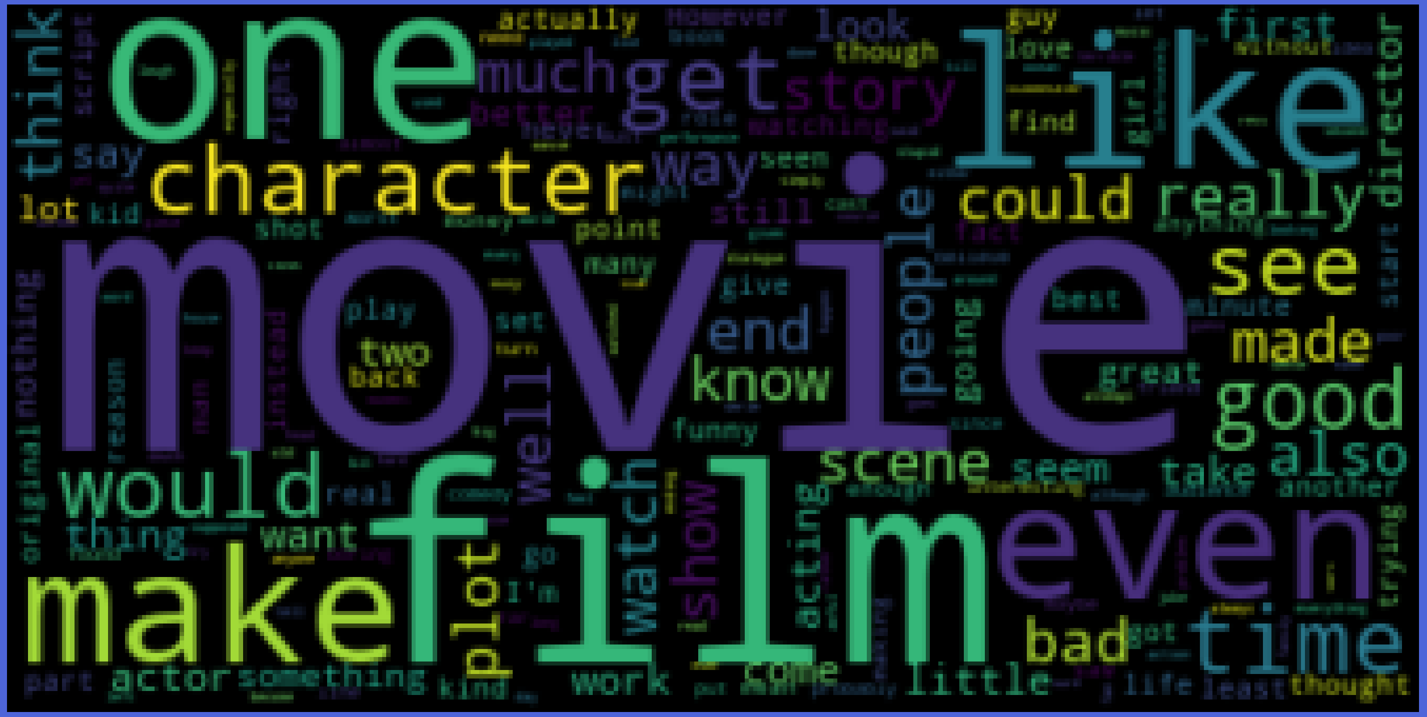
What is the reviewer point of view?



How did people manifest positive sentiment?

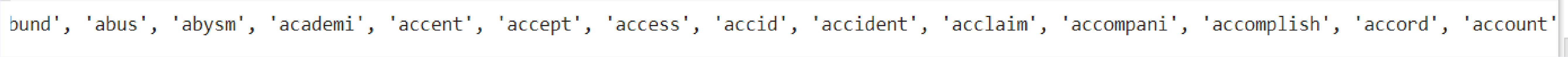


How did people manifest negative sentiment?

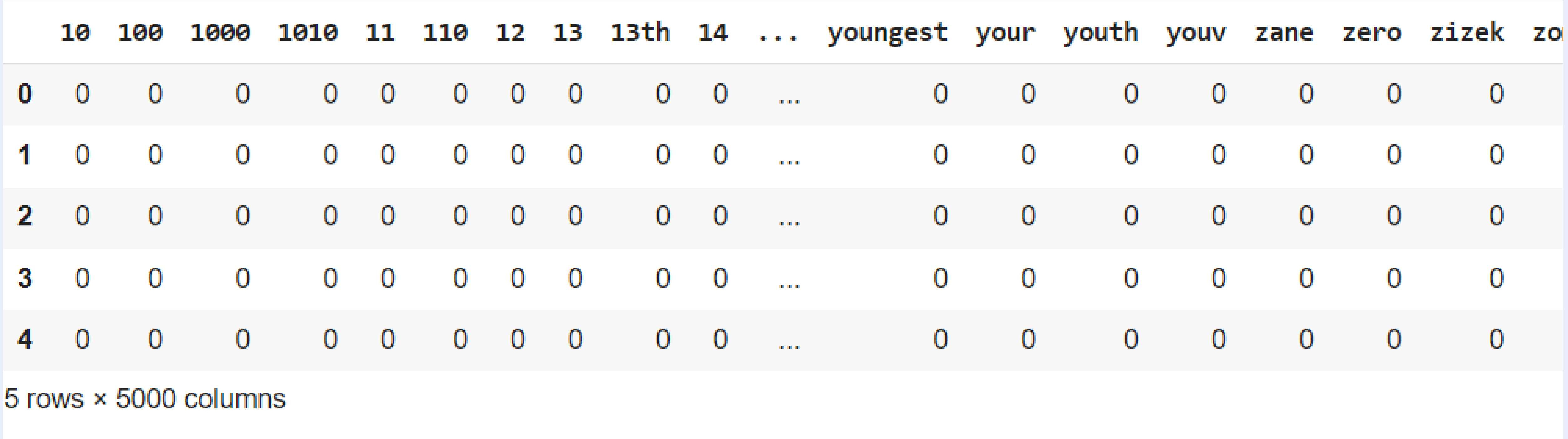


Bag of words... Count Vectorizer

- ***Converting reviews into a numerical representation***
- ***Creating a vocabulary***



```
bund', 'abus', 'abysm', 'academi', 'accent', 'accept', 'access', 'accid', 'accident', 'acclaim', 'accompani', 'accomplish', 'accord', 'account'
```



	10	100	1000	1010	11	110	12	13	13th	14	...	youngest	your	youth	youv	zane	zero	zizek	zo
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	

5 rows × 5000 columns

TF-IDF Vectorizer:

It calculates two things :

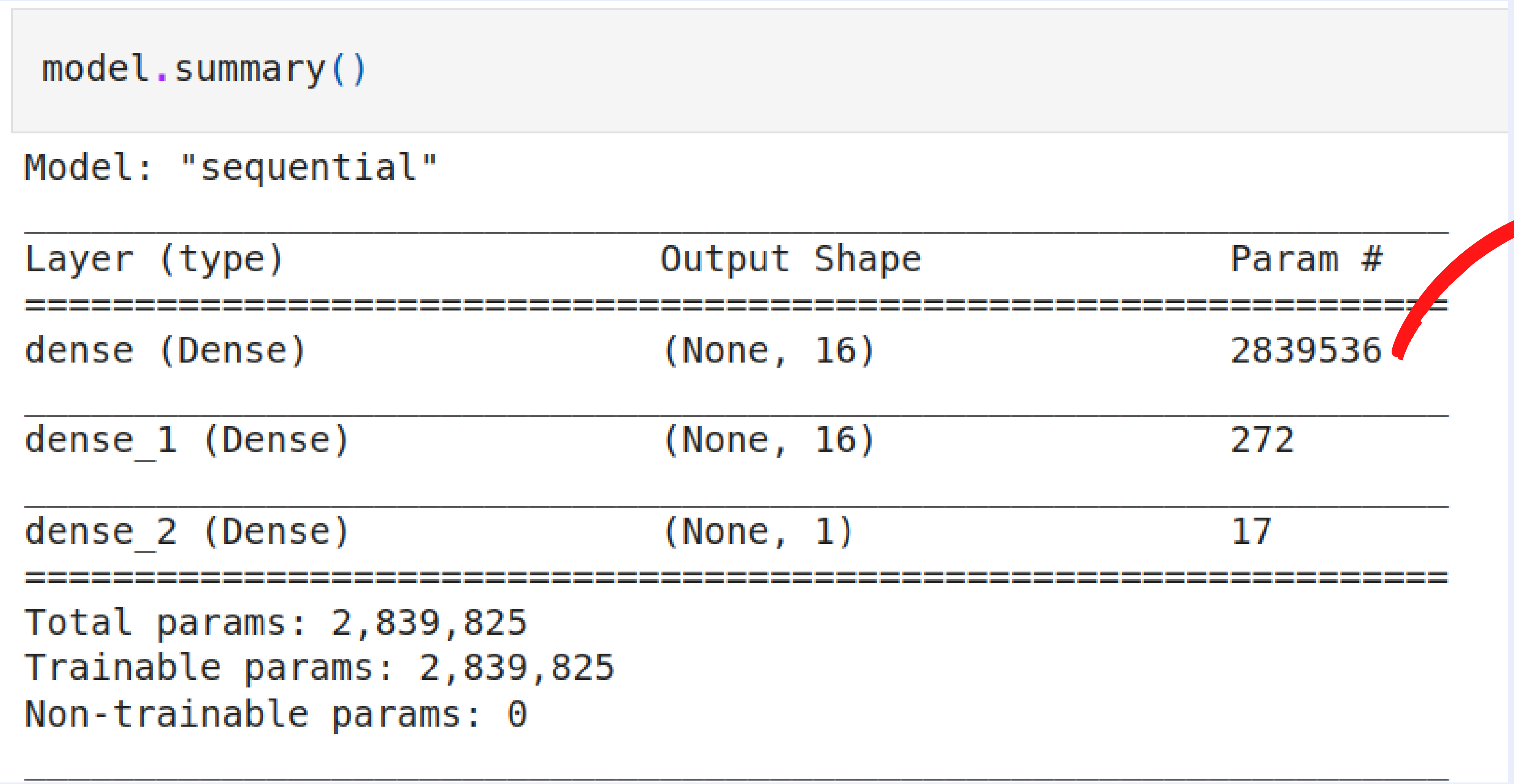
TF= No. of times the word appears in the sample.

IDF = log (No. of times the word appears in the sample/number of times the word appears in the whole document).

So, these TF and IDF values of each word for a specific sample are multiplied to obtain the feature vectors for that sample.

Classification:
Count Vectorization+Sequential Model

accuracy: 0.8731



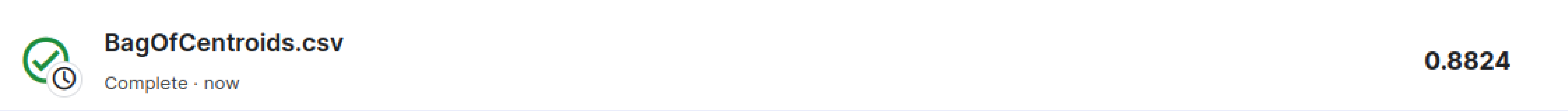
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 16)                2839536
_________________________________________________________________
dense_1 (Dense)              (None, 16)                272
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 17
=================================================================
Total params: 2,839,825
Trainable params: 2,839,825
Non-trainable params: 0
_________________________________________________________________
```

overfitting

Classification:
TF-IDF Vectorization+Logistic Regression

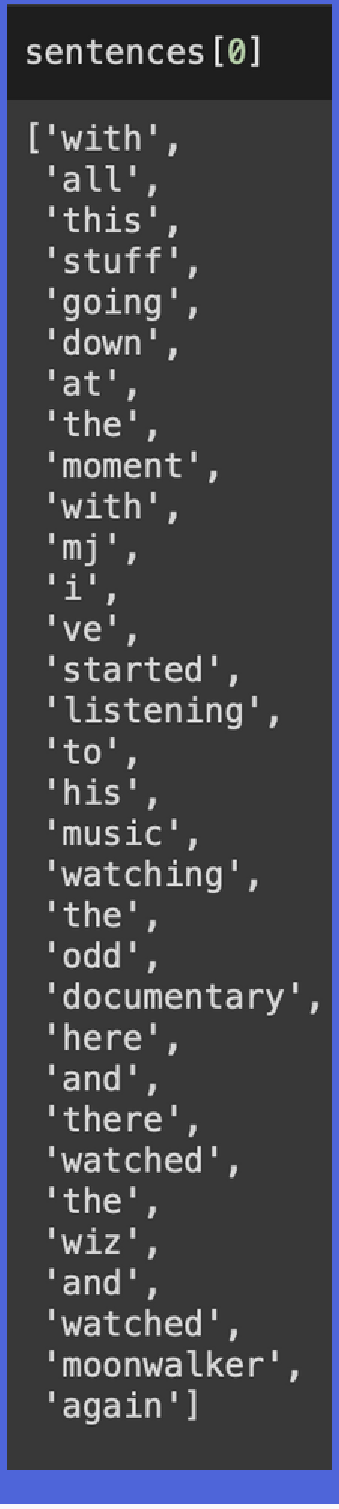
accuracy: 0.89413



BagOfCentroids.csv
Complete · now
0.8824

Post processing

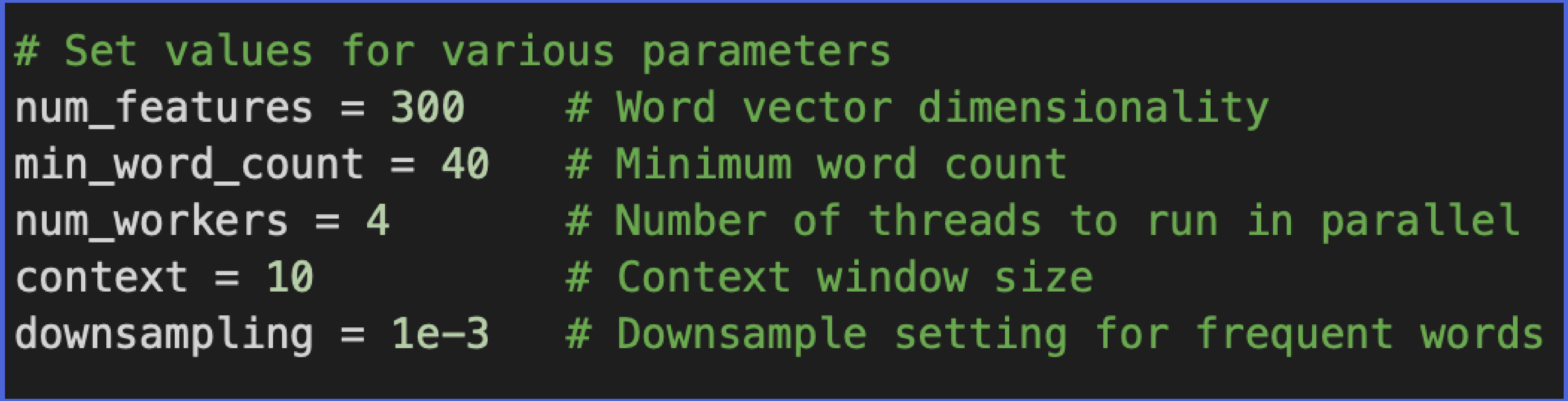
An example of a sentence



```
sentences[0]
```

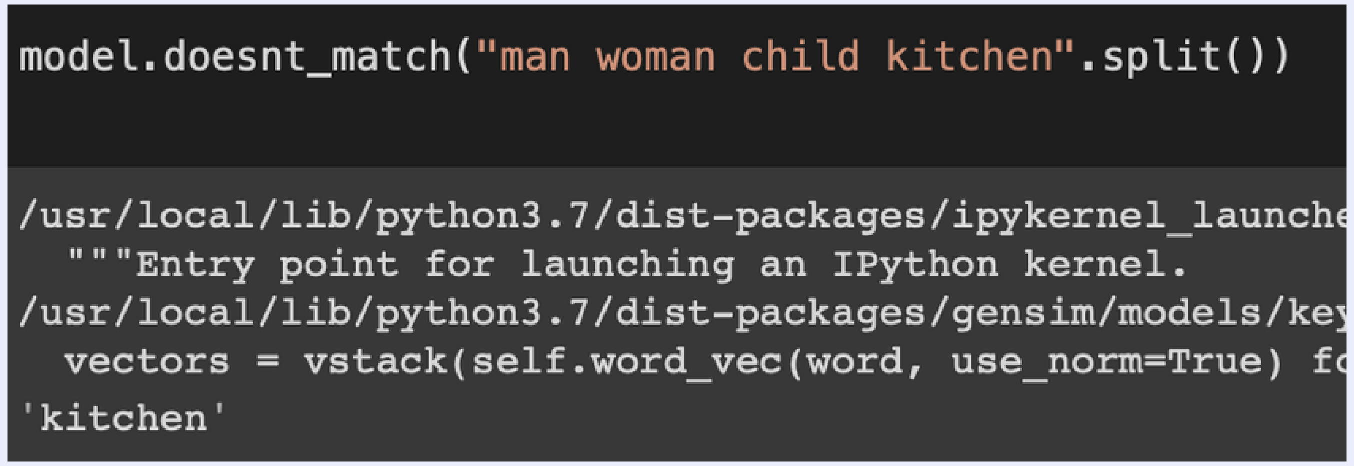
```
['with',
 'all',
 'this',
 'stuff',
 'going',
 'down',
 'at',
 'the',
 'moment',
 'with',
 'mj',
 'i',
 've',
 'started',
 'listening',
 'to',
 'his',
 'music',
 'watching',
 'the',
 'odd',
 'documentary',
 'here',
 'and',
 'there',
 'watched',
 'the',
 'wiz',
 'and',
 'watched',
 'moonwalker',
 'again']
```

training the model



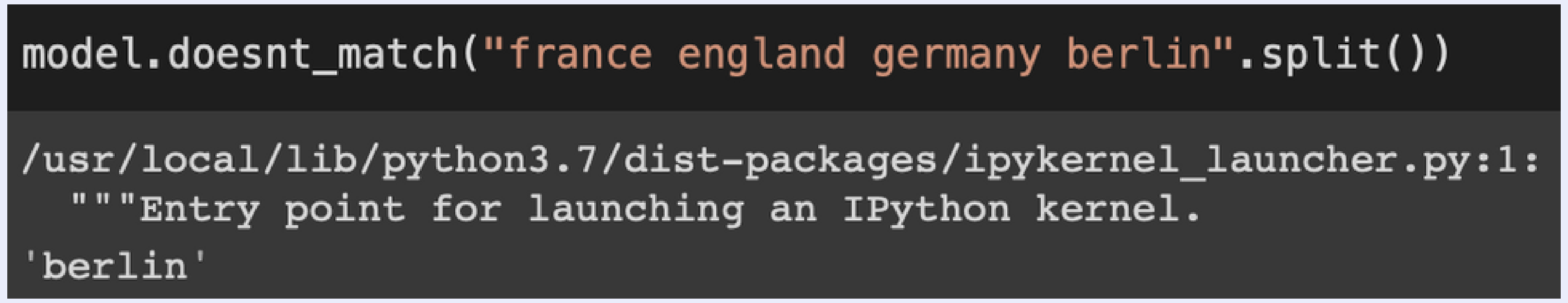
```
# Set values for various parameters
num_features = 300    # Word vector dimensionality
min_word_count = 40   # Minimum word count
num_workers = 4       # Number of threads to run in parallel
context = 10          # Context window size
downsampling = 1e-3   # Downsample setting for frequent words
```

Testing our trained model



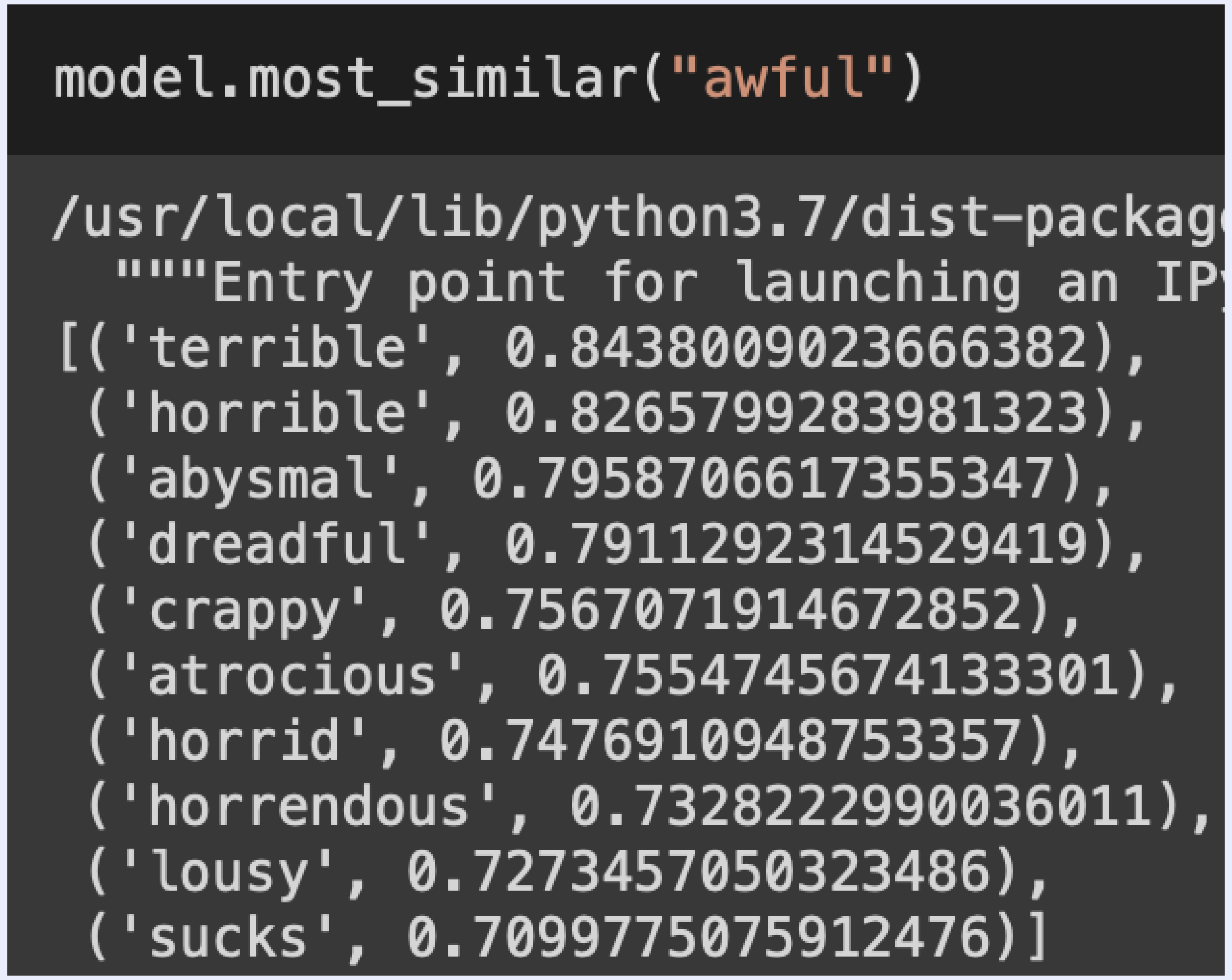
```
model.doesnt_match("man woman child kitchen".split())
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launche
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/gensim/models/key
  vectors = vstack(self.word_vec(word, use_norm=True) fo
'kitchen'
```



```
model.doesnt_match("france england germany berlin".split())
```

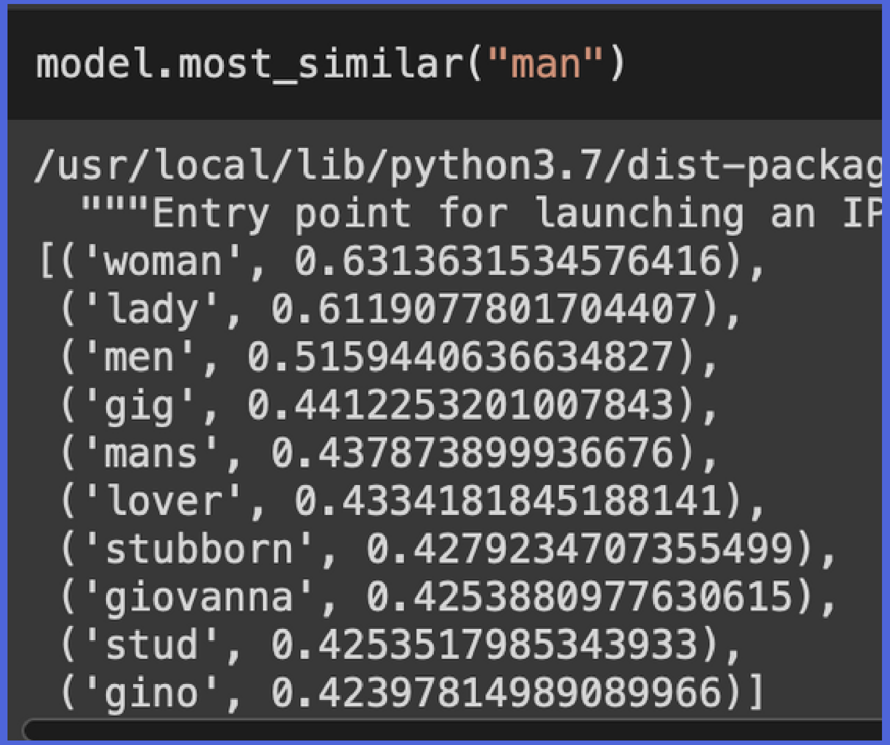
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
  """Entry point for launching an IPython kernel.
'berlin'
```



```
model.most_similar("awful")
```

```
/usr/local/lib/python3.7/dist-packag
  """Entry point for launching an IP
[('terrible', 0.8438009023666382),
 ('horrible', 0.8265799283981323),
 ('abysmal', 0.7958706617355347),
 ('dreadful', 0.7911292314529419),
 ('crappy', 0.756707191467285 2),
 ('atrocious', 0.7554745674133301),
 ('horrid', 0.7476910948753357),
 ('horrendous', 0.7328222990036011),
 ('lousy', 0.7273457050323486),
 ('sucks', 0.7099775075912476)]
```

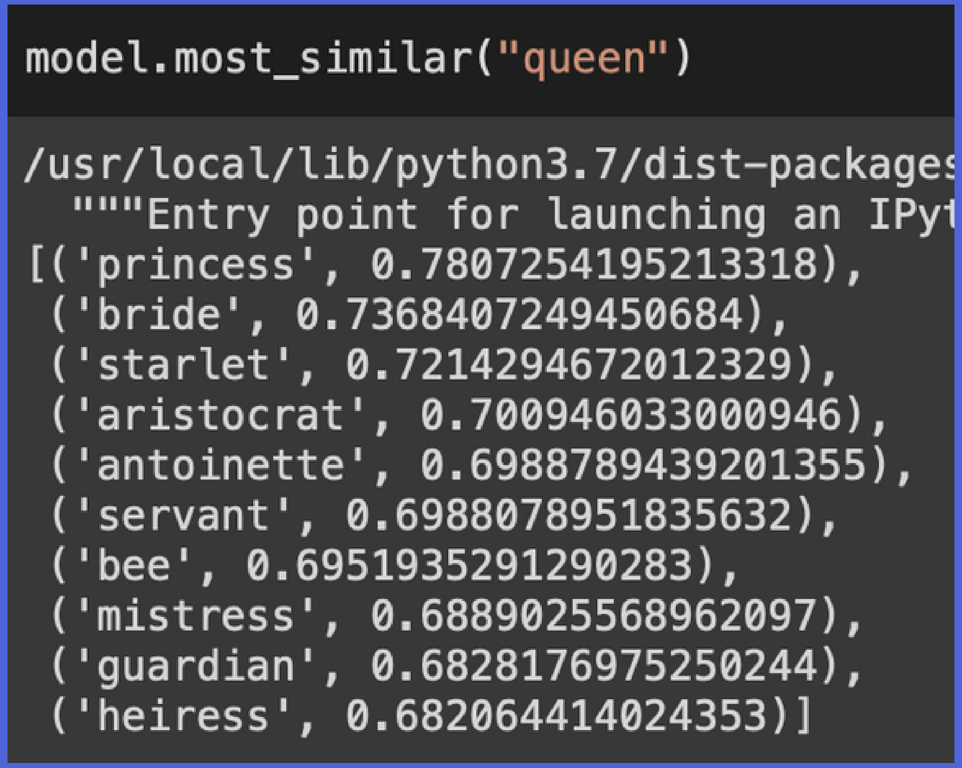
Removing stopwords vs not



```
model.most_similar("man")
```

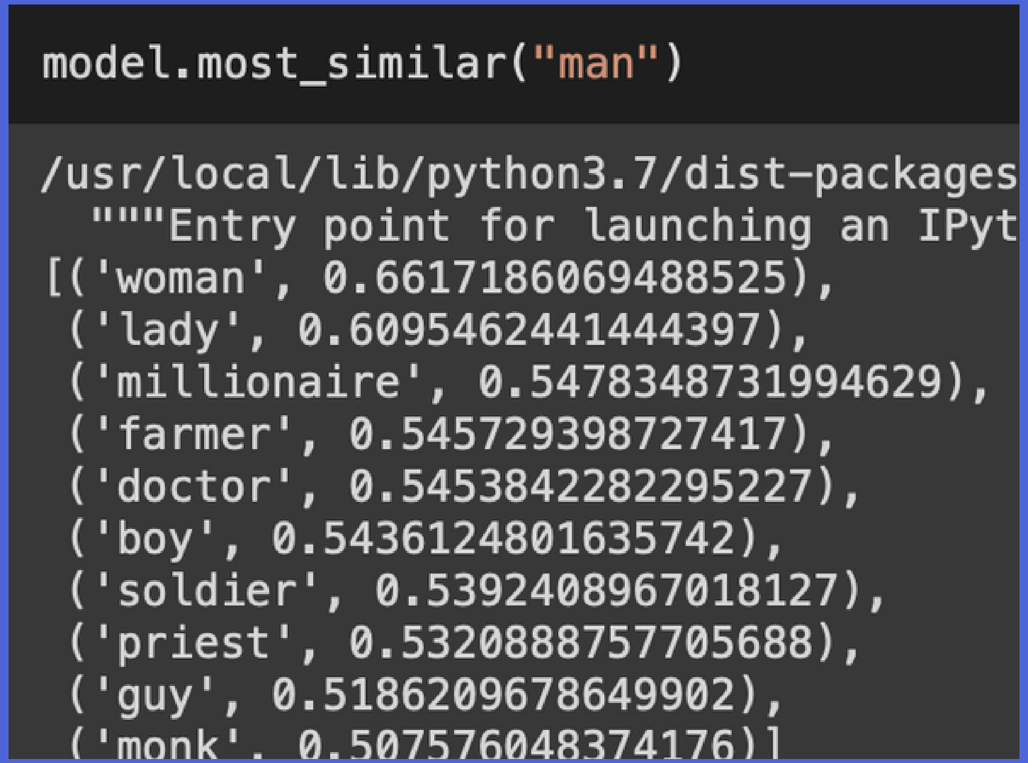
```
/usr/local/lib/python3.7/dist-packag
  """Entry point for launching an IP
[('woman', 0.6313631534576416),
 ('lady', 0.6119077801704407),
 ('men', 0.5159440636634827),
 ('gig', 0.4412253201007843),
 ('mans', 0.437873899936676),
 ('lover', 0.4334181845188141),
 ('stubborn', 0.4279234707355499),
 ('giovanna', 0.4253880977630615),
 ('stud', 0.4253517985343933),
 ('gino', 0.42397814989089966)]
```

Removed



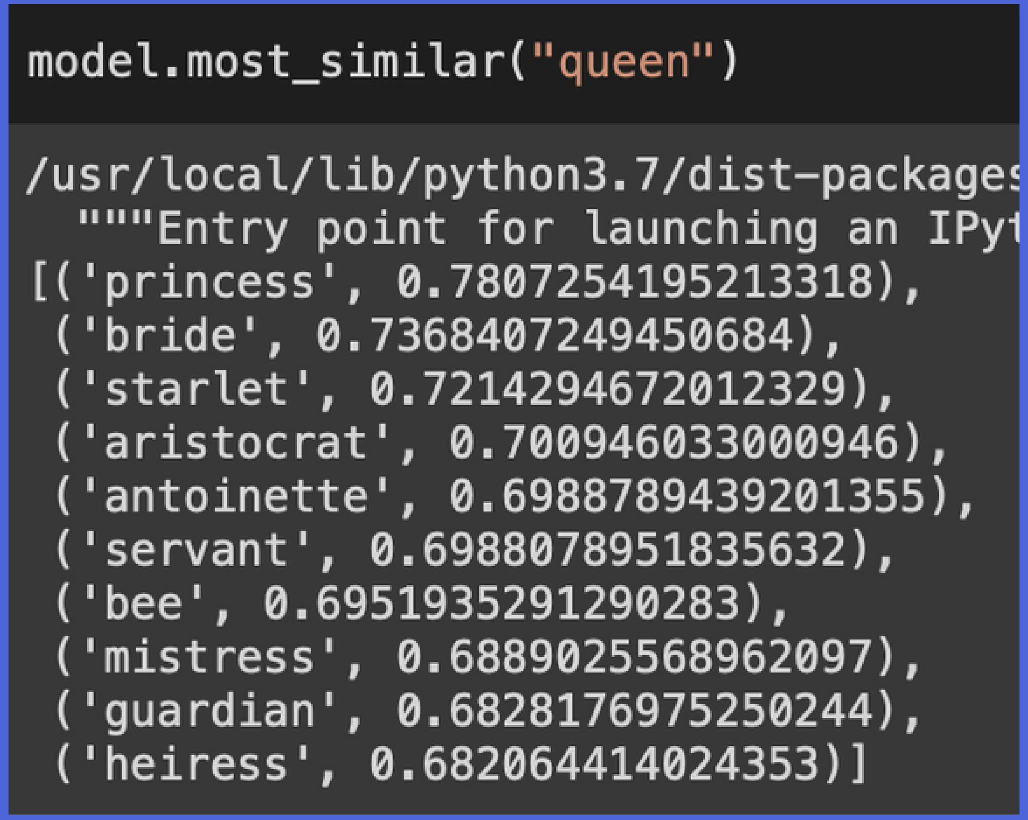
```
model.most_similar("queen")
```

```
/usr/local/lib/python3.7/dist-packages
  """Entry point for launching an IPyt
[('princess', 0.7807254195213318),
 ('bride', 0.7368407249450684),
 ('starlet', 0.7214294672012329),
 ('aristocrat', 0.700946033000946),
 ('antoinette', 0.6988789439201355),
 ('servant', 0.6988078951835632),
 ('bee', 0.6951935291290283),
 ('mistress', 0.6889025568962097),
 ('guardian', 0.6828176975250244),
 ('heiress', 0.682064414024353)]
```



```
model.most_similar("man")
```

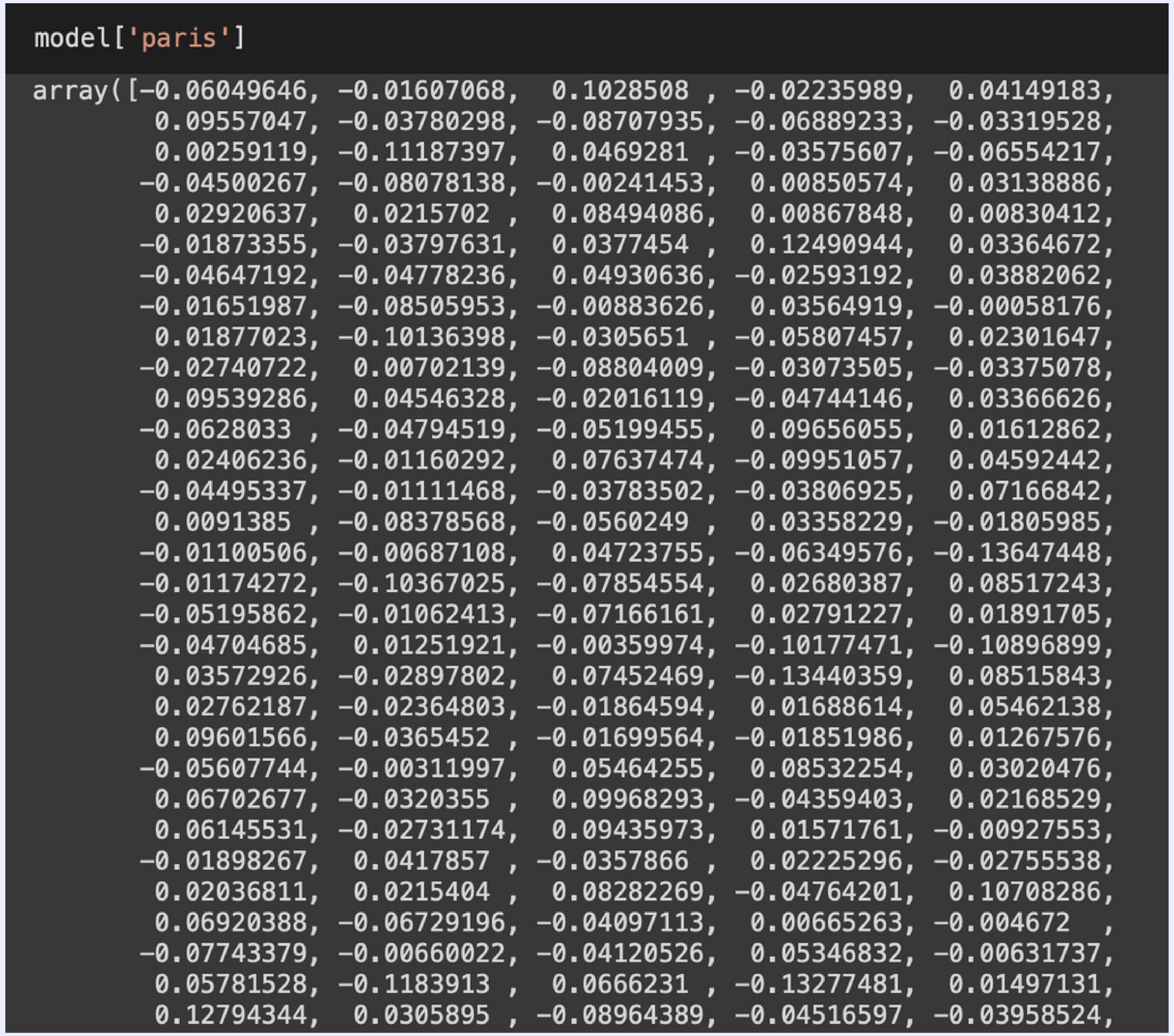
```
/usr/local/lib/python3.7/dist-packages
  """Entry point for launching an IPyt
[('woman', 0.6617186069488525),
 ('lady', 0.6095462441444397),
 ('millionaire', 0.5478348731994629),
 ('farmer', 0.545729398727417),
 ('doctor', 0.5453842282295227),
 ('boy', 0.5436124801635742),
 ('soldier', 0.5392408967018127),
 ('priest', 0.5320888757705688),
 ('guy', 0.5186209678649902),
 ('monk', 0.507576048374176)]
```



```
model.most_similar("queen")
```

```
/usr/local/lib/python3.7/dist-packages
  """Entry point for launching an IPyt
[('princess', 0.7807254195213318),
 ('bride', 0.7368407249450684),
 ('starlet', 0.7214294672012329),
 ('aristocrat', 0.700946033000946),
 ('antoinette', 0.6988789439201355),
 ('servant', 0.6988078951835632),
 ('bee', 0.6951935291290283),
 ('mistress', 0.6889025568962097),
 ('guardian', 0.6828176975250244),
 ('heiress', 0.682064414024353)]
```

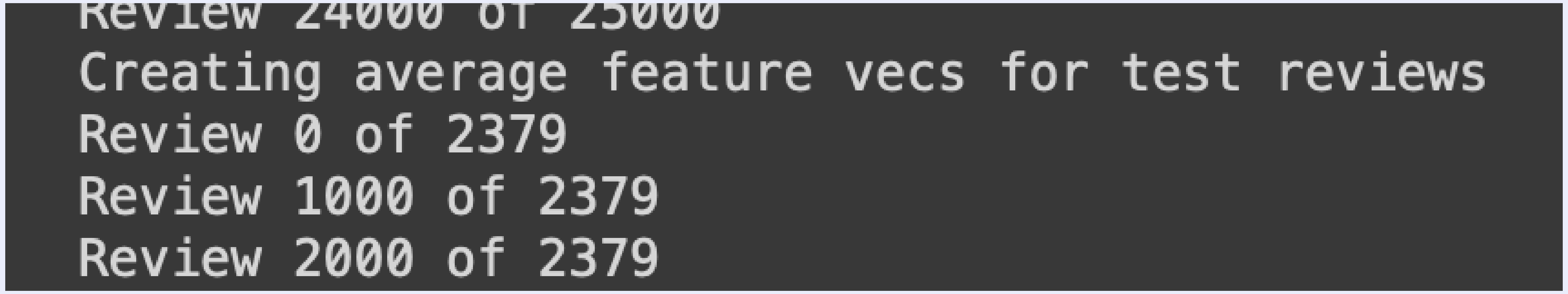
An example of encoded words



```
model['paris']
```

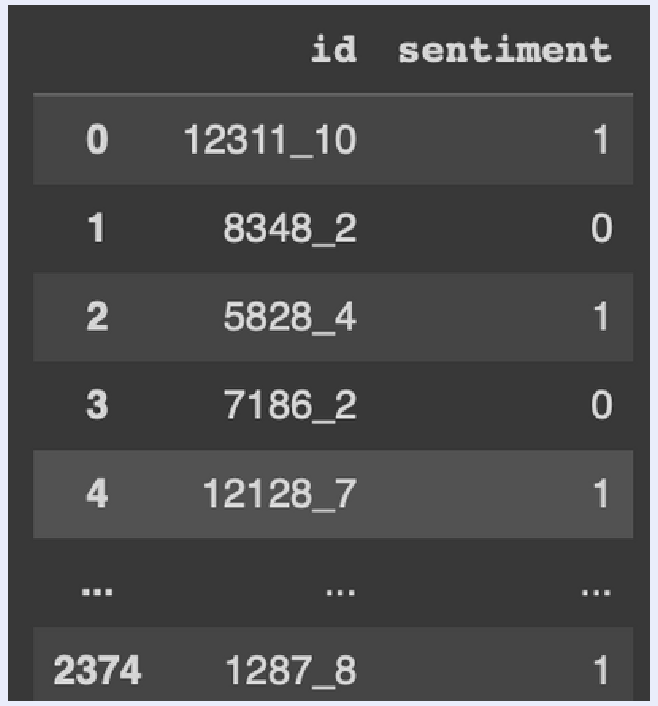
```
array([-0.06049646, -0.01607068,  0.1028508 , -0.02235989,  0.04149183,
        0.09557047, -0.03780298, -0.08707935, -0.06889233, -0.03319528,
        0.00259119, -0.11187397,  0.0469281 , -0.03575607, -0.06554217,
       -0.04500267, -0.08078138, -0.00241453,  0.00850574,  0.03138886,
        0.02920637,  0.0215702 ,  0.08494086,  0.00867848,  0.00830412,
       -0.01873355, -0.03797631,  0.0377454 ,  0.12490944,  0.03364672,
       -0.04647192, -0.04778236,  0.04930636, -0.02593192,  0.03882062,
       -0.01651987, -0.08505953, -0.00883626,  0.03564919, -0.00058176,
        0.01877023, -0.10136398, -0.0305651 , -0.05807457,  0.02301647,
       -0.02740722,  0.00702139, -0.08804009, -0.03073505, -0.03375078,
        0.09539286,  0.04546328, -0.02016119, -0.04744146,  0.03366626,
       -0.0628033 , -0.04794519, -0.05199455,  0.09656055,  0.01612862,
        0.02406236, -0.01160292,  0.07637474, -0.09951057,  0.04592442,
       -0.04495337, -0.01111468, -0.03783502, -0.03806925,  0.07166842,
        0.0091385 , -0.08378568, -0.0560249 ,  0.03358229, -0.01805985,
       -0.01100506, -0.00687108,  0.04723755, -0.06349576, -0.13647448,
       -0.01174272, -0.10367025, -0.07854554,  0.02680387,  0.08517243,
       -0.05195862, -0.01062413, -0.07166161,  0.02791227,  0.01891705,
       -0.04704685,  0.01251921, -0.00359974, -0.10177471, -0.10896899,
        0.03572926, -0.02897802,  0.07452469, -0.13440359,  0.08515843,
        0.02762187, -0.02364803, -0.01864594,  0.01688614,  0.05462138,
        0.09601566, -0.0365452 , -0.01699564, -0.01851986,  0.01267576,
       -0.05607744, -0.00311997,  0.05464255,  0.08532254,  0.03020476,
        0.06702677, -0.0320355 ,  0.09968293, -0.04359403,  0.02168529,
        0.06145531, -0.02731174,  0.09435973,  0.01571761, -0.00927553,
       -0.01898267,  0.0417857 , -0.0357866 ,  0.02225296, -0.02755538,
        0.02036811,  0.0215404 ,  0.08282269, -0.04764201,  0.10708286,
        0.06920388, -0.06729196, -0.04097113,  0.00665263, -0.004672  ,
       -0.07743379, -0.00660022, -0.04120526,  0.05346832, -0.00631737,
        0.05781528, -0.1183913 ,  0.0666231 , -0.13277481,  0.01497131,
        0.12794344,  0.0305895 , -0.08964389, -0.04516597, -0.03958524,
```

Averaging Vectors

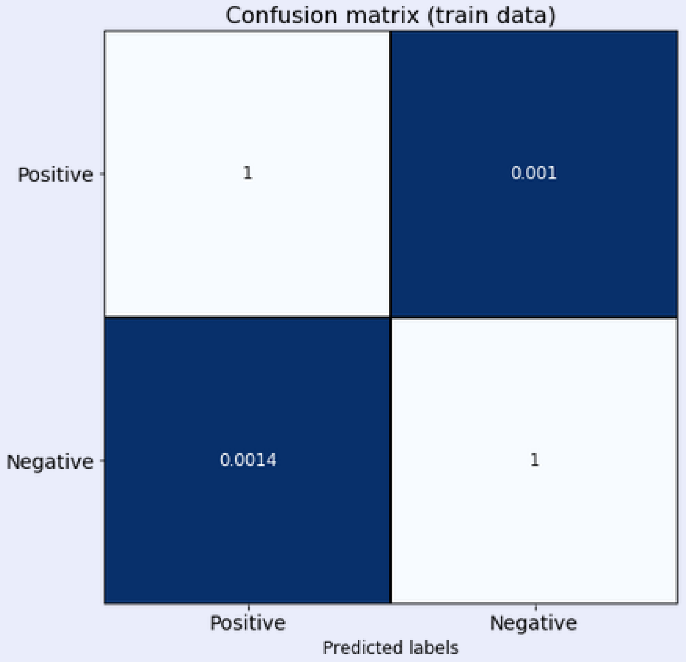


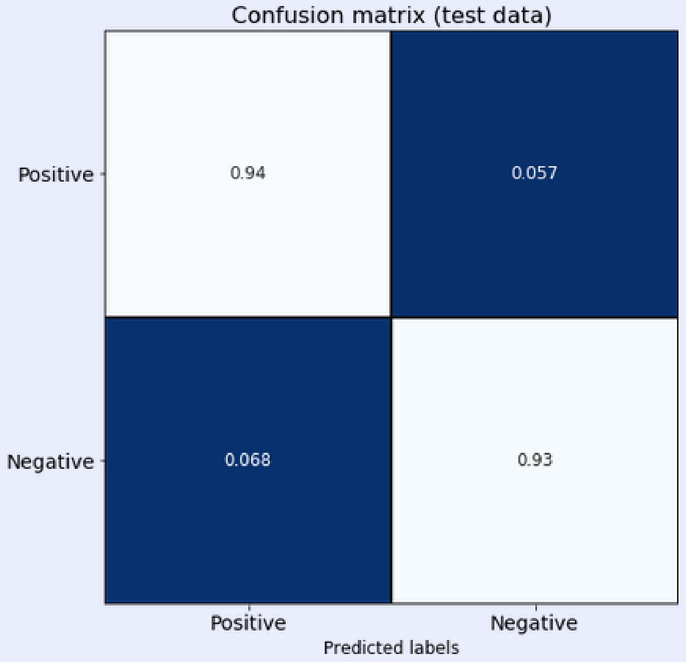
```
Review 24000 of 25000
Creating average feature vecs for test reviews
Review 0 of 2379
Review 1000 of 2379
Review 2000 of 2379
```

- Given a set of reviews (each one a list of words)
- calculate the average feature vector for each one



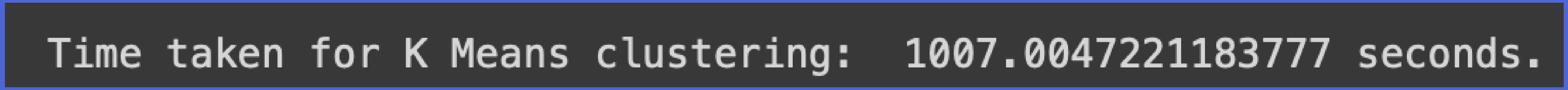
	id	sentiment
0	12311_10	1
1	8348_2	0
2	5828_4	1
3	7186_2	0
4	12128_7	1
...	...	...
2374	1287_8	1



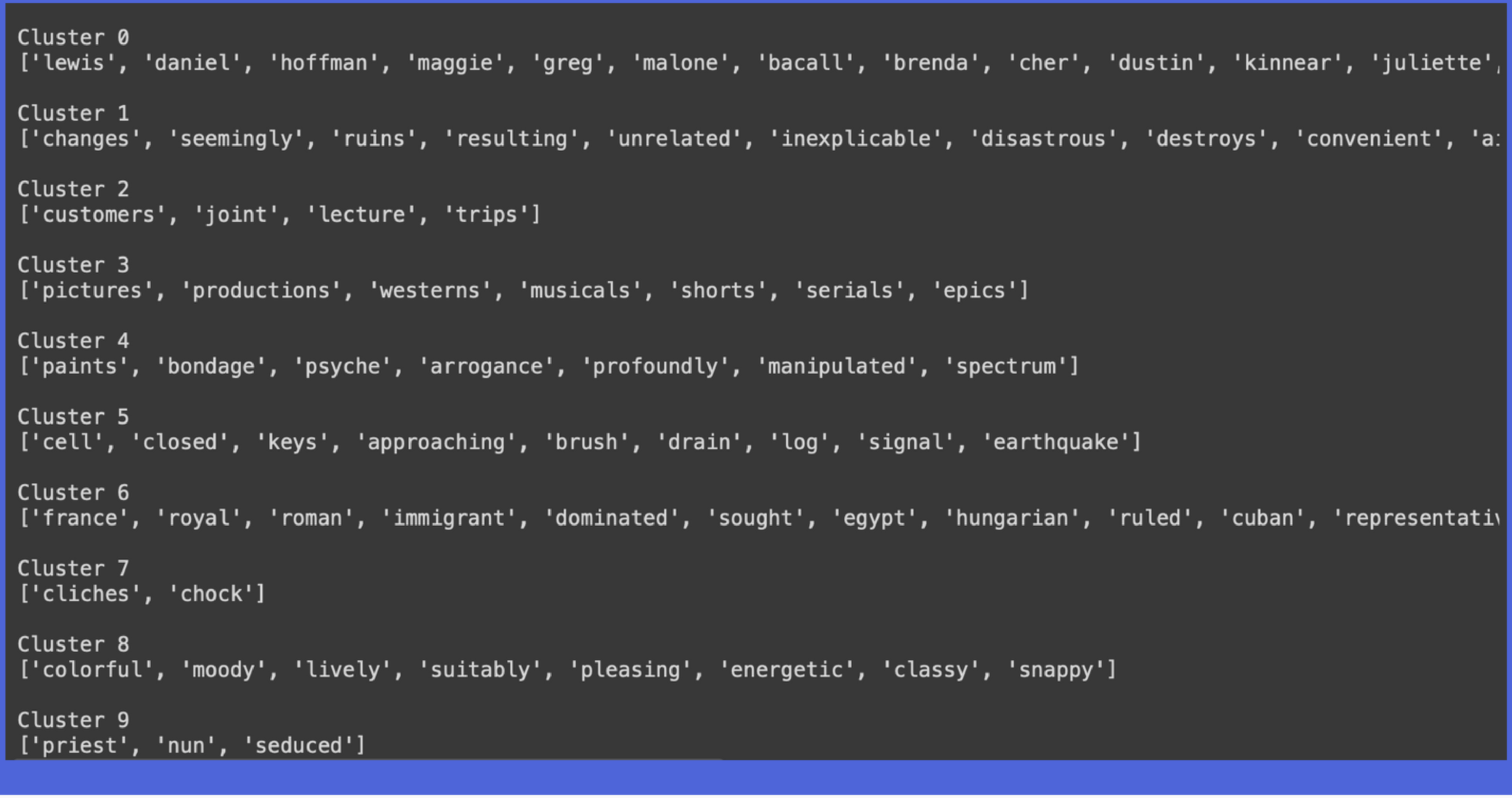


Clustering Using Word2Vec *Using K-means*

exploit the similarity of words within a cluster

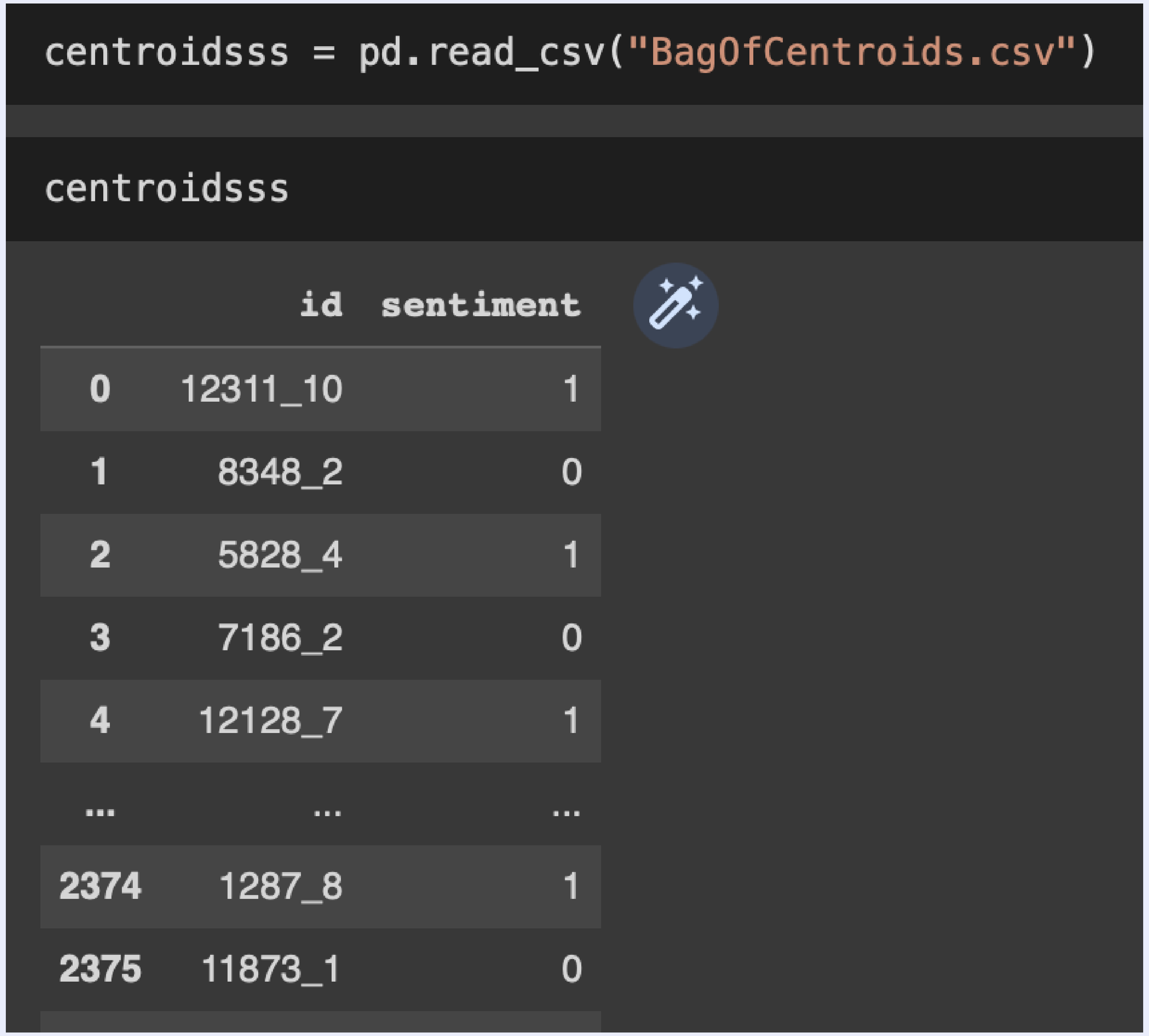


```
Time taken for K Means clustering:  1007.0047221183777 seconds.
```



```
Cluster 0
['lewis', 'daniel', 'hoffman', 'maggie', 'greg', 'malone', 'bacall', 'brenda', 'cher', 'dustin', 'kinnear', 'juliette',

Cluster 1
['changes', 'seemingly', 'ruins', 'resulting', 'unrelated', 'inexplicable', 'disastrous', 'destroys', 'convenient', 'a

Cluster 2
['customers', 'joint', 'lecture', 'trips']

Cluster 3
['pictures', 'productions', 'westerns', 'musicals', 'shorts', 'serials', 'epics']

Cluster 4
['paints', 'bondage', 'psyche', 'arrogance', 'profoundly', 'manipulated', 'spectrum']

Cluster 5
['cell', 'closed', 'keys', 'approaching', 'brush', 'drain', 'log', 'signal', 'earthquake']

Cluster 6
['france', 'royal', 'roman', 'immigrant', 'dominated', 'sought', 'egypt', 'hungarian', 'ruled', 'cuban', 'representati

Cluster 7
['cliches', 'chock']

Cluster 8
['colorful', 'moody', 'lively', 'suitably', 'pleasing', 'energetic', 'classy', 'snappy']

Cluster 9
['priest', 'nun', 'seduced']
```

This works just like Bag of Words but uses semantically related clusters instead of individual words



```python
centroidsss = pd.read_csv("BagOfCentroids.csv")
```

```python
centroidsss
```

	id	sentiment
0	12311_10	1
1	8348_2	0
2	5828_4	1
3	7186_2	0
4	12128_7	1
...	...	...
2374	1287_8	1
2375	11873_1	0

Kaggle submission score - 91 (cluster)
88 (averaging vectors)

Thoughts -

The biggest reason is averaging the vectors and using the centroids lose the order of words, making it very similar to the concept of Bag of Words

Thank you for attention!