Assignment

What does tf-idf mean?

Tf-idf stands for *term frequency-inverse document frequency*, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

One of the simplest ranking functions is computed by summing the tf-idf for each query term; many more sophisticated ranking functions are variants of this simple model.

Tf-idf can be successfully used for stop-words filtering in various subject fields including text summarization and classification.

How to Compute:

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

- **TF:** Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization: $TF(t) = \frac{\text{Number of times term t appears in a document}}{\text{Total number of terms in the document}}$.
- **IDF:** Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following: $IDF(t) = \log_e \frac{\text{Total number of documents}}{\text{Number of documents with term t in it}}$, for numerical stabiltiy we will be changing this formula little bit $IDF(t) = \log_e \frac{\text{Total number of documents}}{\text{Number of documents with term t in it+1}}$.

Example

Consider a document containing 100 words wherein the word cat appears 3 times. The term frequency (i.e., tf) for cat is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word cat appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4. Thus, the Tf-idf weight is the product of these quantities: 0.03 * 4 = 0.12. </p>

Task-1

1. Build a TFIDF Vectorizer & compare its results with Sklearn:

- As a part of this task you will be implementing TFIDF vectorizer on a collection of text documents.

- You should compare the results of your own implementation of TFIDF vectorizer with that of sklearns implemenation TFIDF vectorizer.

- Sklearn does few more tweaks in the implementation of its version of TFIDF vectorizer, so to replicate the exact results you would need to add following things to your custom implementation of tfidf vectorizer:
  1. Sklearn has its vocabulary generated from idf sroted in alphabetical order
  2. Sklearn formula of idf is different from the standard textbook formula. Here the constant **"1"** is added to the numerator and denominator of the idf as if an extra document was seen containing every term in the collection exactly once, which prevents zero divisions. $IDF(t) = 1 + \log_e \frac{1 + \text{Total number of documents in collection}}{1+\text{Number of documents with term t in it}}$.
  3. Sklearn applies L2-normalization on its output matrix.
  4. The final output of sklearn tfidf vectorizer is a sparse matrix.

- Steps to approach this task:
  1. You would have to write both fit and transform methods for your custom implementation of tfidf vectorizer.
  2. Print out the alphabetically sorted voacb after you fit your data and check if its the same as that of the feature names from sklearn tfidf vectorizer.
  3. Print out the idf values from your implementation and check if its the same as that of sklearns tfidf vectorizer idf values.
  4. Once you get your voacb and idf values to be same as that of sklearns implementation of tfidf vectorizer, proceed to the below steps.
  5. Make sure the output of your implementation is a sparse matrix. Before generating the final output, you need to normalize your sparse matrix using L2 normalization. You can refer to this link https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html
  6. After completing the above steps, print the output of your custom implementation and compare it with sklearns implementation of tfidf vectorizer.
  7. To check the output of a single document in your collection of documents, you can convert the sparse matrix related only to that document into dense matrix and print it.

Note-1: All the necessary outputs of sklearns tfidf vectorizer have been provided as reference in this notebook, you can compare your outputs as mentioned in the above steps, with these outputs.
Note-2: The output of your custom implementation and that of sklearns implementation would match only with the collection of document strings provided to you as reference in this notebook. It would not match for strings that contain capital letters or punctuations, etc, because sklearn version of tfidf vectorizer deals with such strings in a different way. To know further details about how sklearn tfidf vectorizer works with such string, you can always refer to its official documentation.
Note-3: During this task, it would be helpful for you to debug the code you write with print statements wherever necessary. But when you are finally submitting the assignment, make sure your code is readable and try not to print things which are not part of this task.

Corpus

In [1]:
```
## SkLearn# Collection of string documents

corpus = ['this is the first document',
     'this document is the second document',
     'and this is the third one',
     'is this the first document']
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore")

from collections import Counter
from tqdm import tqdm
from scipy.sparse import csr_matrix
import math
import operator
from sklearn.preprocessing import normalize
import numpy as np
```

In [3]:
```
corpus = ['this is the first document',
         'this document is t the second document',
         'and this is the third one',
         'is this the first document']
```

In [4]:
```
def fit(doc):
    """given list of document(strings),
       This function splits the string into words,sorts(asc) and enumarates them and adds them into dictionary"""
```

```
    if isinstance(doc,list):
        words=set()         #Set doesn't allow duplicate values.
        for row in doc:
            for i in row.split():
                if len(i)<2:
                    continue
                words.add(i)
        words=sorted(words)      #sorting words in aphabetic order : it helps in enumeration
        words={v:k for k,v in enumerate(words)}
        return words
    else:
        print("Please enter the list of strings")
```

In [5]:
```
#comparison

vocab = fit(corpus)
print("custom:",vocab,'\n')

from sklearn.feature_extraction.text import TfidfVectorizer
# sklearn feature names,sorts by alphabetic_order as default.
print("sklearn:",TfidfVectorizer().fit(corpus).get_feature_names())
```

```
custom: {'and': 0, 'document': 1, 'first': 2, 'is': 3, 'one': 4, 'second': 5, 'the': 6, 'third': 7, 'this': 8}

sklearn: ['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
```

In [6]:
```
def idf(corpus,vocab):
    '''This function fids idf for corpus'''
    idf={}
    n_docs=len(corpus)      #number of documents in corpus
    for ind,row in enumerate(corpus):
         for word in vocab.keys():
            count=0
            for row in corpus:
                if word in set(row.split()):
                    count+=1
            IDF=1+math.log((1+n_docs)/(1+count))
            idf[word]=IDF

    #print(dict(sorted(idf.items(),key=lambda x:x[1],reverse=True)[:50]))
    return idf
z=idf(corpus,vocab)

print(z)
```

```
{'and': 1.916290731874155, 'document': 1.2231435513142097, 'first': 1.5108256237659907, 'is': 1.0, 'one': 1.916290731874155, 'second': 1.916290731874155, 'the': 1.0, 'third': 1.916
290731874155, 'this': 1.0}
```

In [7]:
```
#comparison
print("custom:", list(idf(corpus,vocab).values()),"\n")
print("sklearn:", TfidfVectorizer().fit(corpus).idf_)
```

```
custom: [1.916290731874155, 1.2231435513142097, 1.5108256237659907, 1.0, 1.916290731874155, 1.916290731874155, 1.0, 1.916290731874155, 1.0]

sklearn: [1.91629073 1.22314355 1.51082562 1.         1.91629073 1.91629073
 1.         1.91629073 1.        ]
```

In [8]:
```
def transform(corpus,vocab):
    '''This fuction calculates the Tfidf of the document corpus given vocab:enumaration of words'''
    print('→ Make sure to print instead of raw calling the function\n e.g,x=transform(corpus,vocab)\n\tprint(x[0])')

    if isinstance(corpus,list):
        number_of_docs=len(corpus)
        number_of_unique_words=len(vocab.keys())

        #1
        freq=[row.split()for row in corpus]     #list of lists of words in each row
        frequency_of_words=[]
        for row in freq:                            #getting the word frequencies
            for word in set(row):
                if len(word)<2:
                    continue
                frequency_of_words.append(word)

        #In how many documents each word occured in?
        frequency_of_words=dict(Counter(frequency_of_words))
        print("f",frequency_of_words)
        #2
        csr=csr_matrix((number_of_docs,number_of_unique_words))  #setting the shape
        #help(csr_matrix)
        for ind,row in enumerate(tqdm(corpus)):
                words_counter=dict(Counter(row.split()))   #counting the number of each word in a row
                for word in set(row.split()):
                    if len(word)<2:
                        continue
                    idf=(1+math.log(  ((1+number_of_docs)/(1+frequency_of_words[word]))))
                    csr[ind,vocab[word]]=(words_counter[word]/len(row.split())  *  idf) #sparsematrix[r,c]=value

        tfidf=normalize(csr, norm ='l2')
        return tfidf
    else:
        print("Please enter the list of strings")
```

In [9]:
```
#comparison
x=transform(corpus,vocab)
print(f'custom:\n{x}\nshape:{x.shape}')


from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
vectorizer.fit(corpus)
skl_output = vectorizer.transform(corpus)
print('\n\nsklearn:\n',skl_output,"\nshape:", skl_output.shape)
```

```
→ Make sure to print instead of raw calling the function
 e.g,x=transform(corpus,vocab)
        print(x[0])
f {'is': 4, 'first': 2, 'document': 3, 'the': 4, 'this': 4, 'second': 1, 'third': 1, 'one': 1, 'and': 1}
```

```
100%|██████████████████████████████████████████████████████████████████████████| 4/4 [00:00<00:00, 387.63it/s]
custom:
  (0, 1)        0.4697913855799205
  (0, 2)        0.580285823684436
  (0, 3)        0.3840852409148149
  (0, 6)        0.3840852409148149
  (0, 8)        0.3840852409148149
  (1, 1)        0.6876235979836937
  (1, 3)        0.2810886740337529
  (1, 5)        0.5386476208856762
  (1, 6)        0.2810886740337529
  (1, 8)        0.2810886740337529
  (2, 0)        0.511848512707169
  (2, 3)        0.267103787642168
  (2, 4)        0.511848512707169
  (2, 6)        0.267103787642168
  (2, 7)        0.511848512707169
  (2, 8)        0.267103787642168
  (3, 1)        0.4697913855799205
  (3, 2)        0.580285823684436
  (3, 3)        0.3840852409148149
  (3, 6)        0.3840852409148149
  (3, 8)        0.3840852409148149
shape:(4, 9)


sklearn:
   (0, 8)       0.38408524091481483
  (0, 6)        0.38408524091481483
  (0, 3)        0.38408524091481483
  (0, 2)        0.5802858236844359
  (0, 1)        0.46979138557992045
  (1, 8)        0.281088674033753
  (1, 6)        0.281088674033753
  (1, 5)        0.5386476208856763
  (1, 3)        0.281088674033753
  (1, 1)        0.6876235979836938
  (2, 8)        0.267103787642168
  (2, 7)        0.511848512707169
  (2, 6)        0.267103787642168
  (2, 4)        0.511848512707169
  (2, 3)        0.267103787642168
  (2, 0)        0.511848512707169
  (3, 8)        0.38408524091481483
  (3, 6)        0.38408524091481483
  (3, 3)        0.38408524091481483
  (3, 2)        0.5802858236844359
  (3, 1)        0.46979138557992045
shape: (4, 9)
```

Task-2

2. Implement max features functionality:

- As a part of this task you have to modify your fit and transform functions so that your vocab will contain only 50 terms with top idf scores.

- This task is similar to your previous task, just that here your vocabulary is limited to only top 50 features names based on their idf values. Basically your output will have exactly 50 columns and the number of rows will depend on the number of documents you have in your corpus.

- Here you will be give a pickle file, with file name **cleaned_strings**. You would have to load the corpus from this file and use it as input to your tfidf vectorizer.

- Steps to approach this task:
  1. You would have to write both fit and transform methods for your custom implementation of tfidf vectorizer, just like in the previous task. Additionally, here you have to limit the number of features generated to 50 as described above.
  2. Now sort your vocab based in descending order of idf values and print out the words in the sorted voacb after you fit your data. Here you should be getting only 50 terms in your vocab. And make sure to print idf values for each term in your vocab.
  3. Make sure the output of your implementation is a sparse matrix. Before generating the final output, you need to normalize your sparse matrix using L2 normalization. You can refer to this link https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html
  4. Now check the output of a single document in your collection of documents, you can convert the sparse matrix related only to that document into dense matrix and print it. And this dense matrix should contain 1 row and 50 columns.

In [10]:
```
#Below is the code to load the cleaned_strings pickle file provided
#Here corpus is of list type

import pickle
with open('cleaned_strings', 'rb') as f:
    corpus = pickle.load(f)

#printing the length of the corpus loaded
print("Number of documents in corpus = ",len(corpus))
```

```
Number of documents in corpus =  746
```

In [11]:
```
def fit(doc):
    """given list of document(strings),
       This function splits the string into words,sorts(asc) and enumarates them and adds them into dictionary"""

    if isinstance(doc,list):
        words=set()         #Set doesn't allow duplicate values.
        for row in doc:
            for i in row.split():
                if len(i)>=2:
                    words.add(i)
                else:
                    continue
        words=sorted(words)      #sorting words in aphabetic order : it helps in enumeration
        words={v:k for k,v in enumerate(words)}
        return words
    else:
        print("Please enter the list of strings")
```

In [12]:
```
vocab = fit(corpus)
print(vocab)
```

{'aailiyah': 0, 'abandoned': 1, 'ability': 2, 'abroad': 3, 'absolutely': 4, 'abstruse': 5, 'abysmal': 6, 'academy': 7, 'accents': 8, 'accessible': 9, 'acclaimed': 10, 'accolades': 11, 'accurate': 12, 'accurately': 13, 'accused': 14, 'achievement': 15, 'achille': 16, 'ackerman': 17, 'act': 18, 'acted': 19, 'acting': 20, 'action': 21, 'actions': 22, 'actor': 23, 'actors': 24, 'actress': 25, 'actresses': 26, 'actually': 27, 'adams': 28, 'adaptation': 29, 'add': 30, 'added': 31, 'addition': 32, 'admins': 33, 'admiration': 34, 'admitted': 35, 'adorable': 36, 'adrift': 37, 'adventure': 38, 'advise': 39, 'aerial': 40, 'aesthetically': 41, 'affected': 42, 'affleck': 43, 'afraid': 44, 'africa': 45, 'afternoon': 46, 'age': 47, 'aged': 48, 'ages': 49, 'ago': 50, 'agree': 51, 'agreed': 52, 'aimless': 53, 'air': 54, 'aired': 55, 'akasha': 56, 'akin': 57, 'alert': 58, 'alexander': 59, 'alike': 60, 'allison': 61, 'allow': 62, 'allowing': 63, 'almost': 64, 'along': 65, 'alongside': 66, 'already': 67, 'also': 68, 'although': 69, 'always': 70, 'amateurish': 71, 'amaze': 72, 'amazed': 73, 'amazing': 74, 'amazingly': 75, 'america': 76, 'american': 77, 'americans': 78, 'among': 79, 'amount': 80, 'amusing': 81, 'amust': 82, 'anatomist': 83, 'angel': 84, 'angela': 85, 'angeles': 86, 'angelina': 87, 'angle': 88, 'angles': 89, 'angry': 90, 'anguish': 91, 'angus': 92, 'animals': 93, 'animated': 94, 'animation': 95, 'anita': 96, 'ann': 97, 'anne': 98, 'anniversary': 99, 'annoying': 100, 'another': 101, 'anthony': 102, 'antithesis': 103, 'anyone': 104, 'anything': 105, 'anyway': 106, 'apart': 107, 'appalling': 108, 'appealing': 109, 'appearance': 110, 'appears': 111, 'applauded': 112, 'applause': 113, 'appreciate': 114, 'appropriate': 115, 'apt': 116, 'argued': 117, 'armageddon': 118, 'armand': 119, 'around': 120, 'array': 121, 'art': 122, 'articulated': 123, 'artiness': 124, 'artist': 125, 'artistic': 126, 'artless': 127, 'arts': 128, 'aside': 129, 'ask': 130, 'asleep': 131, 'aspect': 132, 'aspects': 133, 'ass': 134, 'assante': 135, 'assaulted': 136, 'assistant': 137, 'astonishingly': 138, 'astronaut': 139, 'atmosphere': 140, 'atrocious': 141, 'atrocity': 142, 'attempt': 143, 'attempted': 144, 'attempting': 145, 'attempts': 146, 'attention': 147, 'attractive': 148, 'audience': 149, 'audio': 150, 'aurv': 151, 'austen': 152, 'austere': 153, 'author': 154, 'average': 155, 'aversion': 156, 'avoid': 157, 'avoided': 158, 'award': 159, 'awarded': 160, 'awards': 161, 'away': 162, 'awesome': 163, 'awful': 164, 'awkwardly': 165, 'aye': 166, 'baaaaaad': 167, 'babbling': 168, 'babie': 169, 'baby': 170, 'babysitting': 171, 'back': 172, 'backdrop': 173, 'backed': 174, 'bad': 175, 'badly': 176, 'bag': 177, 'bailey': 178, 'bakery': 179, 'balance': 180, 'balanced': 181, 'ball': 182, 'ballet': 183, 'balls': 184, 'band': 185, 'barcelona': 186, 'barely': 187, 'barking': 188, 'barney': 189, 'barren': 190, 'based': 191, 'basic': 192, 'basically': 193, 'bat': 194, 'bates': 195, 'baxendale': 196, 'bear': 197, 'beautiful': 198, 'beautifully': 199, 'bec': 200, 'became': 201, 'bechard': 202, 'become': 203, 'becomes': 204, 'began': 205, 'begin': 206, 'beginning': 207, 'behind': 208, 'behold': 209, 'bela': 210, 'believable': 211, 'believe': 212, 'believed': 213, 'bell': 214, 'bellucci': 215, 'belly': 216, 'belmondo': 217, 'ben': 218, 'bendingly': 219, 'bennett': 220, 'bergen': 221, 'bertolucci': 222, 'best': 223, 'better': 224, 'betty': 225, 'beware': 226, 'beyond': 227, 'bible': 228, 'big': 229, 'biggest': 230, 'billy': 231, 'biographical': 232, 'bipolarity': 233, 'bit': 234, 'bitchy': 235, 'black': 236, 'blah': 237, 'blake': 238, 'bland': 239, 'blandly': 240, 'blare': 241, 'blatant': 242, 'blew': 243, 'blood': 244, 'blown': 245, 'blue': 246, 'blush': 247, 'boasts': 248, 'bob': 249, 'body': 250, 'bohemian': 251, 'boiling': 252, 'bold': 253, 'bombardments': 254, 'bond': 255, 'bonding': 256, 'bonus': 257, 'bonuses': 258, 'boobs': 259, 'boogeyman': 260, 'book': 261, 'boost': 262, 'bop': 263, 'bordered': 264, 'borderlines': 265, 'borders': 266, 'bore': 267, 'bored': 268, 'boring': 269, 'borrowed': 270, 'boss': 271, 'bother': 272, 'bothersome': 273, 'bought': 274, 'box': 275, 'boyfriend': 276, 'boyle': 277, 'brain': 278, 'brainsucking': 279, 'brat': 280, 'breaking': 281, 'breeders': 282, 'brevity': 283, 'brian': 284, 'brief': 285, 'brigand': 286, 'bright': 287, 'brilliance': 288, 'brilliant': 289, 'brilliantly': 290, 'bring': 291, 'brings': 292, 'broad': 293, 'broke': 294, 'brooding': 295, 'brother': 296, 'brutal': 297, 'buddy': 298, 'budget': 299, 'buffalo': 300, 'buffet': 301, 'build': 302, 'builders': 303, 'buildings': 304, 'built': 305, 'bullock': 306, 'bully': 307, 'bunch': 308, 'burton': 309, 'business': 310, 'buy': 311, 'cable': 312, 'cailles': 313, 'california': 314, 'call': 315, 'called': 316, 'calls': 317, 'came': 318, 'cameo': 319, 'camera': 320, 'camerawork': 321, 'camp': 322, 'campy': 323, 'canada': 324, 'cancan': 325, 'candace': 326, 'candle': 327, 'cannot': 328, 'cant': 329, 'captain': 330, 'captured': 331, 'captures': 332, 'car': 333, 'card': 334, 'cardboard': 335, 'cardellini': 336, 'care': 337, 'carol': 338, 'carrell': 339, 'carries': 340, 'carry': 341, 'cars': 342, 'cartoon': 343, 'cartoons': 344, 'case': 345, 'cases': 346, 'cast': 347, 'casted': 348, 'casting': 349, 'cat': 350, 'catchy': 351, 'caught': 352, 'cause': 353, 'ceases': 354, 'celebration': 355, 'celebrity': 356, 'celluloid': 357, 'centers': 358, 'central': 359, 'century': 360, 'certain': 361, 'certainly': 362, 'cg': 363, 'cgi': 364, 'chalkboard': 365, 'challenges': 366, 'chance': 367, 'change': 368, 'changes': 369, 'changing': 370, 'channel': 371, 'character': 372, 'characterisation': 373, 'characters': 374, 'charisma': 375, 'charismatic': 376, 'charles': 377, 'charlie': 378, 'charm': 379, 'charming': 380, 'chase': 381, 'chasing': 382, 'cheap': 383, 'cheaply': 384, 'check': 385, 'checking': 386, 'cheek': 387, 'cheekbones': 388, 'cheerfull': 389, 'cheerless': 390, 'cheesiness': 391, 'cheesy': 392, 'chemistry': 393, 'chick': 394, 'child': 395, 'childhood': 396, 'children': 397, 'childrens': 398, 'chills': 399, 'chilly': 400, 'chimp': 401, 'chodorov': 402, 'choice': 403, 'choices': 404, 'choked': 405, 'chosen': 406, 'chow': 407, 'christmas': 408, 'christopher': 409, 'church': 410, 'cinema': 411, 'cinematic': 412, 'cinematographers': 413, 'cinematography': 414, 'circumstances': 415, 'class': 416, 'classic': 417, 'classical': 418, 'clear': 419, 'clearly': 420, 'clever': 421, 'clich': 422, 'cliche': 423, 'clients': 424, 'cliff': 425, 'climax': 426, 'close': 427, 'closed': 428, 'clothes': 429, 'club': 430, 'co': 431, 'coach': 432, 'coal': 433, 'coastal': 434, 'coaster': 435, 'coherent': 436, 'cold': 437, 'cole': 438, 'collect': 439, 'collective': 440, 'colored': 441, 'colorful': 442, 'colours': 443, 'columbo': 444, 'come': 445, 'comedic': 446, 'comedy': 447, 'comes': 448, 'comfortable': 449, 'comforting': 450, 'comical': 451, 'coming': 452, 'commands': 453, 'comment': 454, 'commentary': 455, 'commented': 456, 'comments': 457, 'commercial': 458, 'community': 459, 'company': 460, 'compelling': 461, 'competent': 462, 'complete': 463, 'completed': 464, 'completely': 465, 'complex': 466, 'complexity': 467, 'composed': 468, 'composition': 469, 'comprehensible': 470, 'compromise': 471, 'computer': 472, 'concentrate': 473, 'conception': 474, 'conceptually': 475, 'concerning': 476, 'concerns': 477, 'concert': 478, 'conclusion': 479, 'condescends': 480, 'confidence': 481, 'configuration': 482, 'confirm': 483, 'conflict': 484, 'confuses': 485, 'confusing': 486, 'connections': 487, 'connery': 488, 'connor': 489, 'conrad': 490, 'consequences': 491, 'consider': 492, 'considerable': 493, 'considered': 494, 'considering': 495, 'considers': 496, 'consistent': 497, 'consolations': 498, 'constant': 499, 'constantine': 500, 'constructed': 501, 'contained': 502, 'containing': 503, 'contains': 504, 'content': 505, 'continually': 506, 'continuation': 507, 'continue': 508, 'continuity': 509, 'continuously': 510, 'contract': 511, 'contrast': 512, 'contributing': 513, 'contributory': 514, 'contrived': 515, 'control': 516, 'controversy': 517, 'convention': 518, 'convey': 519, 'convince': 520, 'convincing': 521, 'convoluted': 522, 'cool': 523, 'coppola': 524, 'cords': 525, 'core': 526, 'corn': 527, 'corny': 528, 'correct': 529, 'cost': 530, 'costs': 531, 'costumes': 532, 'cotton': 533, 'could': 534, 'couple': 535, 'course': 536, 'court': 537, 'courtroom': 538, 'cover': 539, 'cowardice': 540, 'cox': 541, 'crackles': 542, 'crafted': 543, 'crap': 544, 'crash': 545, 'crashed': 546, 'crayon': 547, 'crayons': 548, 'crazy': 549, 'create': 550, 'created': 551, 'creates': 552, 'creative': 553, 'creativity': 554, 'creature': 555, 'credible': 556, 'credit': 557, 'credits': 558, 'crew': 559, 'crime': 560, 'crisp': 561, 'critic': 562, 'critical': 563, 'crocdodile': 564, 'crocs': 565, 'cross': 566, 'crowd': 567, 'crowe': 568, 'cruel': 569, 'cruise': 570, 'cry': 571, 'cult': 572, 'culture': 573, 'curtain': 574, 'custer': 575, 'cute': 576, 'cutest': 577, 'cutie': 578, 'cutouts': 579, 'cuts': 580, 'cutting': 581, 'dads': 582, 'damian': 583, 'damn': 584, 'dance': 585, 'dancing': 586, 'dangerous': 587, 'dark': 588, 'darren': 589, 'daughter': 590, 'daughters': 591, 'day': 592, 'days': 593, 'de': 594, 'dead': 595, 'deadly': 596, 'deadpan': 597, 'deal': 598, 'dealt': 599, 'death': 600, 'debated': 601, 'debbie': 602, 'debits': 603, 'debut': 604, 'decay': 605, 'decent': 606, 'decidely': 607, 'decipher': 608, 'decisions': 609, 'dedication': 610, 'dee': 611, 'deep': 612, 'deeply': 613, 'defensemen': 614, 'defined': 615, 'definitely': 616, 'delete': 617, 'delight': 618, 'delightful': 619, 'delights': 620, 'deliver': 621, 'delivered': 622, 'delivering': 623, 'delivers': 624, 'dependant': 625, 'depending': 626, 'depends': 627, 'depicted': 628, 'depicts': 629, 'depressing': 630, 'depth': 631, 'derivative': 632, 'describe': 633, 'describes': 634, 'desert': 635, 'deserved': 636, 'deserves': 637, 'deserving': 638, 'design': 639, 'designed': 640, 'designer': 641, 'desperately': 642, 'desperation': 643, 'despised': 644, 'despite': 645, 'destroy': 646, 'detailing': 647, 'details': 648, 'develop': 649, 'development': 650, 'developments': 651, 'di': 652, 'diabetic': 653, 'dialog': 654, 'dialogs': 655, 'dialogue': 656, 'diaper': 657, 'dickens': 658, 'difference': 659, 'different': 660, 'dignity': 661, 'dimensional': 662, 'direct': 663, 'directed': 664, 'directing': 665, 'direction': 666, 'director': 667, 'directorial': 668, 'directors': 669, 'disappointed': 670, 'disappointing': 671, 'disappointment': 672, 'disaster': 673, 'disbelief': 674, 'discomfort': 675, 'discovering': 676, 'discovery': 677, 'disgrace': 678, 'disgusting': 679, 'dislike': 680, 'disliked': 681, 'disney': 682, 'disparate': 683, 'distant': 684, 'distinction': 685, 'distorted': 686, 'distract': 687, 'distressed': 688, 'disturbing': 689, 'diving': 690, 'doctor': 691, 'documentaries': 692, 'documentary': 693, 'dodge': 694, 'dogs': 695, 'dollars': 696, 'dominated': 697, 'done': 698, 'donlevy': 699, 'dont': 700, 'doomed': 701, 'dose': 702, 'doubt': 703, 'downs': 704, 'dozen': 705, 'dr': 706, 'dracula': 707, 'draft': 708, 'drag': 709, 'drago': 710, 'drama': 711, 'dramatic': 712, 'drawings': 713, 'drawn': 714, 'dream': 715, 'dreams': 716, 'dreary': 717, 'dribble': 718, 'drift': 719, 'drifting': 720, 'drive': 721, 'drooling': 722, 'dropped': 723, 'dry': 724, 'due': 725, 'duet': 726, 'dull': 727, 'dumb': 728, 'dumbest': 729, 'duper': 730, 'duris': 731, 'dustin': 732, 'dvd': 733, 'dwight': 734, 'dysfunction': 735, 'earlier': 736, 'early': 737, 'earth': 738, 'easily': 739, 'easy': 740, 'eating': 741, 'ebay': 742, 'ebola': 743, 'eccleston': 744, 'ed': 745, 'edge': 746, 'editing': 747, 'edition': 748, 'educational': 749, 'edward': 750, 'effect': 751, 'effective': 752, 'effects': 753, 'effort': 754, 'efforts': 755, 'egotism': 756, 'eighth': 757, 'eiko': 758, 'either': 759, 'elaborately': 760, 'elderly': 761, 'elegant': 762, 'element': 763, 'elias': 764, 'eloquently': 765, 'else': 766, 'elsewhere': 767, 'embarrassed': 768, 'embarrassing': 769, 'embassy': 770, 'emerge': 771, 'emilio': 772, 'emily': 773, 'emoting': 774, 'emotion': 775, 'emotionally': 776, 'emotions': 777, 'emperor': 778, 'empowerment': 779, 'emptiness': 780, 'empty': 781, 'en': 782, 'enchanting': 783, 'end': 784, 'endearing': 785, 'ended': 786, 'ending': 787, 'endlessly': 788, 'ends': 789, 'energetic': 790, 'energy': 791, 'engaging': 792, 'english': 793, 'enhanced': 794, 'enjoy': 795, 'enjoyable': 796, 'enjoyed': 797, 'enjoyment': 798, 'enough': 799, 'enter': 800, 'enterprise': 801, 'entertained': 802, 'entertaining': 803, 'entire': 804, 'entirely': 805, 'entrance': 806, 'episode': 807, 'episodes': 808, 'equivalent': 809, 'era': 810, 'errol': 811, 'errors': 812, 'escalating': 813, 'escapism': 814, 'especially': 815, 'essence': 816, 'establish': 817, 'established': 818, 'estate': 819, 'estevez': 820, 'etc': 821, 'european': 822, 'evaluate': 823, 'even': 824, 'events': 825, 'ever': 826, 'every': 827, 'everybody': 828, 'everyone': 829, 'everything': 830, 'everywhere': 831, 'evidently': 832, 'evil': 833, 'evinced': 834, 'evokes': 835, 'exactly': 836, 'exaggerating': 837, 'example': 838, 'excellent': 839, 'excellently': 840, 'except': 841, 'exceptional': 842, 'exceptionally': 843, 'excerpts': 844, 'excessively': 845, 'exchange': 846, 'exciting': 847, 'excruciatingly': 848, 'excuse': 849, 'excuses': 850, 'executed': 851, 'exemplars': 852, 'existent': 853, 'existential': 854, 'expansive': 855, 'expect': 856, 'expectations': 857, 'expected': 858, 'expecting': 859, 'experience': 860, 'experiences': 861, 'expert': 862, 'explain': 863, 'explains': 864, 'explanation': 865, 'exploit': 866, 'explorations': 867, 'explosion': 868, 'expression': 869, 'exquisite': 870, 'extant': 871, 'exteriors': 872, 'extraneous': 873, 'extraordinary': 874, 'extremely': 875, 'eye': 876, 'eyes': 877, 'fabulous': 878, 'face': 879, 'faces': 880, 'facial': 881, 'facing': 882, 'fact': 883, 'factory': 884, 'failed': 885, 'fails': 886, 'fair': 887, 'fairly': 888, 'faithful': 889, 'fall': 890, 'falling': 891, 'falls': 892, 'falsely': 893, 'falwell': 894, 'fame': 895, 'famed': 896, 'family': 897, 'famous': 898, 'fan': 899, 'fanciful': 900, 'fans': 901, 'fantastic': 902, 'fantasy': 903, 'far': 904, 'farce': 905, 'fare': 906, 'fascinated': 907, 'fascinating': 908, 'fascination': 909, 'fashioned': 910, 'fast': 911, 'faster': 912, 'fat': 913, 'father': 914, 'faultless': 915, 'fausa': 916, 'faux': 917, 'favorite': 918, 'favourite': 919, 'fear': 920, 'feature': 921, 'features': 922, 'feel': 923, 'feeling': 924, 'feelings': 925, 'feet': 926, 'feisty': 927, 'fellowes': 928, 'felt': 929, 'female': 930, 'females': 931, 'ferry': 932, 'fest': 933, 'fi': 934, 'fields': 935, 'fifteen': 936, 'fifties': 937, 'fill': 938, 'film': 939, 'filmed': 940, 'filmiing': 941, 'filmmaker': 942, 'filmography': 943, 'films': 944, 'final': 945, 'finale': 946, 'finally': 947, 'financial': 948, 'find': 949, 'finds': 950, 'fine': 951, 'finest': 952, 'fingernails': 953, 'finished': 954, 'fire': 955, 'first': 956, 'fish': 957, 'fishnet': 958, 'fisted': 959, 'fit': 960, 'five': 961, 'flag': 962, 'flakes': 963, 'flaming': 964, 'flashbacks': 965, 'flat': 966, 'flaw': 967, 'flawed': 968, 'flaws': 969, 'fleshed': 970, 'flick': 971, 'flicks': 972, 'florida': 973, 'flowed': 974, 'flying': 975, 'flynn': 976, 'focus': 977, 'fodder': 978, 'follow': 979, 'following': 980, 'follows': 981, 'foolish': 982, 'footage': 983, 'football': 984, 'force': 985, 'forced': 986, 'forces': 987, 'ford': 988, 'foreign': 989, 'foreigner': 990, 'forever': 991, 'forget': 992, 'forgettable': 993, 'forgetting': 994, 'forgot': 995, 'forgotten': 996, 'form': 997, 'format': 998, 'former': 999, 'fort': 1000, 'forth': 1001, 'forwarded': 1002, 'found': 1003, 'four': 1004, 'fox': 1005, 'foxx': 1006, 'frances': 1007, 'francis': 1008, 'frankly': 1009, 'free': 1010, 'freedom': 1011, 'freeman': 1012, 'french': 1013, 'fresh': 1014, 'freshness': 1015, 'friends': 1016, 'friendship': 1017, 'frightening': 1018, 'front': 1019, 'frontier': 1020, 'frost': 1021, 'frustration': 1022, 'fulci': 1023, 'fulfilling': 1024, 'full': 1025, 'fully': 1026, 'fumbling': 1027, 'fun': 1028, 'function': 1029, 'fundamental': 1030, 'funniest': 1031, 'funny': 1032, 'future': 1033, 'fx': 1034, 'gabriel': 1035, 'gadget': 1036, 'gain': 1037, 'gake': 1038, 'galley': 1039, 'gallon': 1040, 'game': 1041, 'games': 1042, 'garage': 1043, 'garbage': 1044, 'garbo': 1045, 'garfield': 1046, 'gas': 1047, 'gaudi': 1048, 'gave': 1049, 'gay': 1050, 'geek': 1051, 'gem': 1052, 'general': 1053, 'generally': 1054, 'generates': 1055, 'generic': 1056, 'genius': 1057, 'genre': 1058, 'gently': 1059, 'genuine': 1060, 'george': 1061, 'gerardo': 1062, 'gere': 1063, 'get': 1064, 'gets': 1065, 'getting': 1066, 'ghibili': 1067, 'giallo': 1068, 'gibberish': 1069, 'gifted': 1070, 'giovanni': 1071, 'girl': 1072, 'girlfriend': 1073, 'girls': 1074, 'girolamo': 1075, 'give': 1076, 'given': 1077, 'gives': 1078, 'giving': 1079, 'glad': 1080, 'glance': 1081, 'glasses': 1082, 'gloriously': 1083, 'go': 1084, 'goalies': 1085, 'god': 1086, 'goes': 1087, 'going': 1088, 'gone': 1089, 'gonna': 1090, 'good': 1091, 'gore': 1092, 'goremeister': 1093, 'gorman': 1094, 'gosh': 1095, 'got': 1096, 'goth': 1097, 'gotta': 1098, 'gotten': 1099, 'government': 1100, 'grace': 1101, 'grade': 1102, 'gradually': 1103, 'grainy': 1104, 'granted': 1105, 'graphics': 1106, 'grasp': 1107, 'grates': 1108, 'great': 1109, 'greatest': 1110, 'greatness': 1111, 'green': 1112, 'greenstreet': 1113, 'grew': 1114, 'grim': 1115, 'grimes': 1116, 'gripping': 1117, 'groove': 1118, 'gross': 1119, 'ground': 1120, 'guards': 1121, 'guess': 1122, 'guests': 1123, 'guilt': 1124, 'gung': 1125, 'guy': 1126, 'guys': 1127, 'hackneyed': 1128, 'haggis': 1129, 'hair': 1130, 'hairsplitting': 1131, 'half': 1132, 'halfway': 1133, 'ham': 1134, 'hand': 1135, 'handle': 1136, 'handled': 1137, 'handles': 1138, 'hands': 1139, 'hang': 1140, 'hankies': 1141, 'hanks': 1142, 'happen': 1143, 'happened': 1144, 'happiness': 1145, 'happy': 1146, 'hard': 1147, 'harris': 1148, 'hate': 1149, 'hated': 1150, 'hatred': 1151, 'havilland': 1152, 'hay': 1153, 'hayao': 1154, 'hayworth': 1155, 'hbo': 1156, 'head': 1157, 'heads': 1158, 'hear': 1159, 'heard': 1160, 'heart': 1161, 'hearts': 1162, 'heartwarming': 1163, 'heaven': 1164, 'heche': 1165, 'heels': 1166, 'heist': 1167, 'helen': 1168, 'hell': 1169, 'hellish': 1170, 'helms': 1171, 'help': 1172, 'helping': 1173, 'helps': 1174, 'hence': 1175, 'hendrikson': 1176, 'hernandez': 1177, 'hero': 1178, 'heroes': 1179, 'heroine': 1180, 'heroism': 1181, 'hes': 1182, 'hide': 1183, 'high': 1184, 'higher': 1185, 'highest': 1186, 'highlights': 1187, 'highly': 1188, 'hilarious': 1189, 'hill': 1190, 'hilt': 1191, 'hip': 1192, 'history': 1193, 'hitchcock': 1194, 'ho': 1195, 'hockey': 1196, 'hoffman': 1197, 'hold': 1198, 'holding': 1199, 'holds': 1200, 'holes': 1201, 'hollander': 1202, 'hollow': 1203, 'hollywood': 1204, 'home': 1205, 'homework': 1206, 'honest': 1207, 'honestly': 1208, 'hoot': 1209, 'hope': 1210, 'hopefully': 1211, 'hopeless': 1212, 'horrendous': 1213, 'horrendously': 1214, 'horrible': 1215, 'horrid': 1216, 'horrified': 1217, 'horror': 1218, 'horse': 1219, 'hosting': 1220, 'hot': 1221, 'hour': 1222, 'hours': 1223, 'house': 1224, 'houses': 1225, 'howdy': 1226, 'howe': 1227, 'howell': 1228, 'however': 1229, 'huge': 1230, 'hugo': 1231, 'human': 1232, 'humanity': 1233, 'humans': 1234, 'hummh': 1235, 'humor': 1236, 'humorous': 1237, 'humour': 1238, 'hurt': 1239, 'huston': 1240, 'hype': 1241, 'hypocrisy': 1242, 'idea': 1243, 'idealogical': 1244, 'identified': 1245, 'identifies': 1246, 'identify': 1247, 'idiot': 1248, 'idiotic': 1249, 'idyllic': 1250, 'iffy': 1251, 'im': 1252, 'imaginable': 1253, 'imagination': 1254, 'imaginative': 1255, 'imagine': 1256, 'imdb': 1257, 'imitation': 1258, 'impact': 1259, 'imperial': 1260, 'implausible': 1261, 'important': 1262, 'impossible': 1263, 'impressed': 1264, 'impression': 1265, 'impressive': 1266, 'improved': 1267, 'improvement': 1268, 'improvisation': 1269, 'impulse': 1270, 'inappropriate': 1271, 'incendiary': 1272, 'includes': 1273, 'including': 1274, 'incomprehensible': 1275, 'inconsistencies': 1276, 'incorrectness': 1277, 'incredible': 1278, 'incredibly': 1279, 'indeed': 1280, 'indescribably': 1281, 'indication': 1282, 'indictment': 1283, 'indie': 1284, 'individual': 1285, 'indoor': 1286, 'indulgent': 1287, 'industry': 1288, 'ineptly': 1289, 'inexperience': 1290, 'inexplicable': 1291, 'initially': 1292, 'innocence': 1293, 'insane': 1294, 'inside': 1295, 'insincere': 1296, 'insipid': 1297, 'insomniacs': 1298, 'inspiration': 1299, 'inspiring': 1300, 'instant': 1301, 'instead': 1302, 'instruments': 1303, 'insulin': 1304, 'insult': 1305, 'intangibles': 1306, 'integral': 1307, 'integration': 1308, 'intelligence': 1309, 'intelligent': 1310, 'intense': 1311, 'intensity': 1312, 'intentions': 1313, 'interacting': 1314, 'interest': 1315, 'interested': 1316, 'interesting': 1317, 'interim': 1318, 'interplay': 1319, 'interpretations': 1320, 'interview': 1321, 'intoning': 1322, 'intrigued': 1323, 'inventive': 1324, 'involved': 1325, 'involves': 1326, 'involving': 1327, 'iq': 1328, 'ireland': 1329, 'ironically': 1

330, 'irons': 1331, 'ironside': 1332, 'irritating': 1333, 'ishioka': 1334, 'iso': 1335, 'issue': 1336, 'issues': 1337, 'istagey': 1338, 'italian': 1339, 'ive': 1340, 'jack': 1341, 'jaclyn': 1342, 'james': 1343, 'jamie': 1344, 'japanese': 1345, 'jason': 1346, 'jay': 1347, 'jealousy': 1348, 'jean': 1349, 'jennifer': 1350, 'jerky': 1351, 'jerry': 1352, 'jessica': 1353, 'jessice': 1354, 'jet': 1355, 'jim': 1356, 'jimmy': 1357, 'job': 1358, 'jobs': 1359, 'joe': 1360, 'john': 1361, 'joins': 1362, 'joke': 1363, 'jokes': 1364, 'jonah': 1365, 'jones': 1366, 'journey': 1367, 'joy': 1368, 'joyce': 1369, 'juano': 1370, 'judge': 1371, 'judging': 1372, 'judith': 1373, 'judo': 1374, 'julian': 1375, 'june': 1376, 'junk': 1377, 'junkyard': 1378, 'justice': 1379, 'jutland': 1380, 'kanaly': 1381, 'kathy': 1382, 'keep': 1383, 'keeps': 1384, 'keira': 1385, 'keith': 1386, 'kept': 1387, 'kevin': 1388, 'kid': 1389, 'kidnapped': 1390, 'kids': 1391, 'kieslowski': 1392, 'kill': 1393, 'killer': 1394, 'killing': 1395, 'killings': 1396, 'kind': 1397, 'kinda': 1398, 'kirk': 1399, 'kitchy': 1400, 'knew': 1401, 'knightley': 1402, 'knocked': 1403, 'know': 1404, 'known': 1405, 'knows': 1406, 'koteas': 1407, 'kris': 1408, 'kristoffersen': 1409, 'kudos': 1410, 'la': 1411, 'labute': 1412, 'lack': 1413, 'lacked': 1414, 'lacks': 1415, 'ladies': 1416, 'lady': 1417, 'lame': 1418, 'lance': 1419, 'landscapes': 1420, 'lane': 1421, 'lange': 1422, 'largely': 1423, 'laselva': 1424, 'lassie': 1425, 'last': 1426, 'lasting': 1427, 'latched': 1428, 'late': 1429, 'later': 1430, 'latest': 1431, 'latifa': 1432, 'latin': 1433, 'latter': 1434, 'laugh': 1435, 'laughable': 1436, 'laughed': 1437, 'laughs': 1438, 'layers': 1439, 'lazy': 1440, 'lead': 1441, 'leading': 1442, 'leap': 1443, 'learn': 1444, 'least': 1445, 'leave': 1446, 'leaves': 1447, 'leaving': 1448, 'lee': 1449, 'left': 1450, 'legal': 1451, 'legendary': 1452, 'length': 1453, 'leni': 1454, 'less': 1455, 'lesser': 1456, 'lestat': 1457, 'let': 1458, 'lets': 1459, 'letting': 1460, 'level': 1461, 'levels': 1462, 'lewis': 1463, 'lid': 1464, 'lie': 1465, 'lies': 1466, 'lieutenant': 1467, 'life': 1468, 'lifetime': 1469, 'light': 1470, 'lighting': 1471, 'like': 1472, 'liked': 1473, 'likes': 1474, 'lilli': 1475, 'lilt': 1476, 'limitations': 1477, 'limited': 1478, 'linda': 1479, 'line': 1480, 'linear': 1481, 'lines': 1482, 'lino': 1483, 'lion': 1484, 'list': 1485, 'literally': 1486, 'littered': 1487, 'little': 1488, 'lived': 1489, 'lives': 1490, 'living': 1491, 'loads': 1492, 'local': 1493, 'location': 1494, 'locations': 1495, 'loewenhielm': 1496, 'logic': 1497, 'london': 1498, 'loneliness': 1499, 'long': 1500, 'longer': 1501, 'look': 1502, 'looked': 1503, 'looking': 1504, 'looks': 1505, 'loose': 1506, 'loosely': 1507, 'lord': 1508, 'los': 1509, 'losing': 1510, 'lost': 1511, 'lot': 1512, 'lots': 1513, 'lousy': 1514, 'lovable': 1515, 'love': 1516, 'loved': 1517, 'lovely': 1518, 'loves': 1519, 'low': 1520, 'lower': 1521, 'loyalty': 1522, 'lucio': 1523, 'lucy': 1524, 'lugosi': 1525, 'lust': 1526, 'luv': 1527, 'lyrics': 1528, 'macbeth': 1529, 'machine': 1530, 'mad': 1531, 'made': 1532, 'magnificent': 1533, 'main': 1534, 'mainly': 1535, 'major': 1536, 'make': 1537, 'maker': 1538, 'makers': 1539, 'makes': 1540, 'making': 1541, 'male': 1542, 'males': 1543, 'malta': 1544, 'man': 1545, 'managed': 1546, 'manages': 1547, 'manna': 1548, 'mansonites': 1549, 'many': 1550, 'marbles': 1551, 'march': 1552, 'marine': 1553, 'marion': 1554, 'mark': 1555, 'marred': 1556, 'marriage': 1557, 'martin': 1558, 'masculine': 1559, 'masculinity': 1560, 'massive': 1561, 'master': 1562, 'masterful': 1563, 'masterpiece': 1564, 'masterpieces': 1565, 'material': 1566, 'matrix': 1567, 'matter': 1568, 'matthews': 1569, 'mature': 1570, 'may': 1571, 'maybe': 1572, 'mchattie': 1573, 'mclaglen': 1574, 'meagre': 1575, 'mean': 1576, 'meanders': 1577, 'meaning': 1578, 'meanings': 1579, 'meant': 1580, 'medical': 1581, 'mediocre': 1582, 'meld': 1583, 'melodrama': 1584, 'melville': 1585, 'member': 1586, 'members': 1587, 'memorable': 1588, 'memories': 1589, 'memorized': 1590, 'menace': 1591, 'menacing': 1592, 'mention': 1593, 'mercy': 1594, 'meredith': 1595, 'merit': 1596, 'mesmerising': 1597, 'mess': 1598, 'messages': 1599, 'meteorite': 1600, 'mexican': 1601, 'michael': 1602, 'mickey': 1603, 'microsoft': 1604, 'middle': 1605, 'might': 1606, 'mighty': 1607, 'mind': 1608, 'mindblowing': 1609, 'miner': 1610, 'mini': 1611, 'minor': 1612, 'minute': 1613, 'minutes': 1614, 'mirrormask': 1615, 'miserable': 1616, 'miserably': 1617, 'mishima': 1618, 'misplace': 1619, 'miss': 1620, 'missed': 1621, 'mistakes': 1622, 'miyazaki': 1623, 'modern': 1624, 'modest': 1625, 'mollusk': 1626, 'moment': 1627, 'moments': 1628, 'momentum': 1629, 'money': 1630, 'monica': 1631, 'monolog': 1632, 'monotonous': 1633, 'monster': 1634, 'monstrous': 1635, 'monumental': 1636, 'moral': 1637, 'morgan': 1638, 'morons': 1639, 'mostly': 1640, 'mother': 1641, 'motion': 1642, 'motivations': 1643, 'mountain': 1644, 'mouse': 1645, 'mouth': 1646, 'move': 1647, 'moved': 1648, 'movements': 1649, 'moves': 1650, 'movie': 1651, 'movies': 1652, 'moving': 1653, 'ms': 1654, 'much': 1655, 'muddled': 1656, 'muppets': 1657, 'murder': 1658, 'murdered': 1659, 'murdering': 1660, 'murky': 1661, 'music': 1662, 'musician': 1663, 'must': 1664, 'mystifying': 1665, 'naked': 1666, 'narration': 1667, 'narrative': 1668, 'nasty': 1669, 'national': 1670, 'nationalities': 1671, 'native': 1672, 'natural': 1673, 'nature': 1674, 'naughty': 1675, 'nearly': 1676, 'necklace': 1677, 'need': 1678, 'needed': 1679, 'needlessly': 1680, 'negative': 1681, 'negulesco': 1682, 'neighbour': 1683, 'neil': 1684, 'nerves': 1685, 'nervous': 1686, 'net': 1687, 'netflix': 1688, 'network': 1689, 'never': 1690, 'nevertheless': 1691, 'nevsky': 1692, 'new': 1693, 'next': 1694, 'nice': 1695, 'nicola': 1696, 'night': 1697, 'nimoy': 1698, 'nine': 1699, 'no': 1700, 'noble': 1701, 'nobody': 1702, 'noir': 1703, 'non': 1704, 'none': 1705, 'nonetheless': 1706, 'nonsense': 1707, 'nor': 1708, 'normally': 1709, 'northern': 1710, 'nostalgia': 1711, 'not': 1712, 'notable': 1713, 'notch': 1714, 'note': 1715, 'noteworthy': 1716, 'nothing': 1717, 'novella': 1718, 'number': 1719, 'numbers': 1720, 'nun': 1721, 'nuns': 1722, 'nurse': 1723, 'nut': 1724, 'nuts': 1725, 'obliged': 1726, 'obsessed': 1727, 'obvious': 1728, 'obviously': 1729, 'occasionally': 1730, 'occupied': 1731, 'occur': 1732, 'occurs': 1733, 'odd': 1734, 'offend': 1735, 'offensive': 1736, 'offer': 1737, 'offers': 1738, 'often': 1739, 'oh': 1740, 'okay': 1741, 'old': 1742, 'olde': 1743, 'older': 1744, 'ole': 1745, 'olivia': 1746, 'omit': 1747, 'one': 1748, 'ones': 1749, 'open': 1750, 'opened': 1751, 'opening': 1752, 'operas': 1753, 'opinion': 1754, 'ordeal': 1755, 'oriented': 1756, 'original': 1757, 'originality': 1758, 'origins': 1759, 'ortolani': 1760, 'oscar': 1761, 'others': 1762, 'otherwise': 1763, 'ought': 1764, 'outlandish': 1765, 'outlets': 1766, 'outside': 1767, 'outward': 1768, 'overacting': 1769, 'overall': 1770, 'overcome': 1771, 'overdue': 1772, 'overly': 1773, 'overs': 1774, 'overt': 1775, 'overwrought': 1776, 'owed': 1777, 'owls': 1778, 'owned': 1779, 'owns': 1780, 'oy': 1781, 'pace': 1782, 'paced': 1783, 'pacing': 1784, 'pack': 1785, 'paid': 1786, 'painful': 1787, 'painfully': 1788, 'paint': 1789, 'painted': 1790, 'pair': 1791, 'palance': 1792, 'pandering': 1793, 'pans': 1794, 'paolo': 1795, 'pap': 1796, 'paper': 1797, 'par': 1798, 'parents': 1799, 'park': 1800, 'parker': 1801, 'part': 1802, 'partaking': 1803, 'particular': 1804, 'particularly': 1805, 'parts': 1806, 'passed': 1807, 'passion': 1808, 'past': 1809, 'patent': 1810, 'pathetic': 1811, 'patriotism': 1812, 'paul': 1813, 'pay': 1814, 'peaking': 1815, 'pearls': 1816, 'peculiarity': 1817, 'pedestal': 1818, 'pencil': 1819, 'people': 1820, 'perabo': 1821, 'perfect': 1822, 'perfected': 1823, 'perfectly': 1824, 'performance': 1825, 'performances': 1826, 'perhaps': 1827, 'period': 1828, 'perplexing': 1829, 'person': 1830, 'personalities': 1831, 'personally': 1832, 'peter': 1833, 'pg': 1834, 'phantasm': 1835, 'phenomenal': 1836, 'philippa': 1837, 'phony': 1838, 'photograph': 1839, 'photography': 1840, 'phrase': 1841, 'physical': 1842, 'pi': 1843, 'picked': 1844, 'picture': 1845, 'pictures': 1846, 'piece': 1847, 'pieces': 1848, 'pile': 1849, 'pillow': 1850, 'pitch': 1851, 'pitiful': 1852, 'pixar': 1853, 'place': 1854, 'places': 1855, 'plain': 1856, 'plane': 1857, 'planned': 1858, 'plants': 1859, 'play': 1860, 'played': 1861, 'player': 1862, 'players': 1863, 'playing': 1864, 'plays': 1865, 'pleasant': 1866, 'pleased': 1867, 'pleaser': 1868, 'pleasing': 1869, 'pledge': 1870, 'plenty': 1871, 'plmer': 1872, 'plot': 1873, 'plug': 1874, 'plus': 1875, 'pm': 1876, 'poet': 1877, 'poetry': 1878, 'poignant': 1879, 'point': 1880, 'pointillistic': 1881, 'pointless': 1882, 'poised': 1883, 'poler': 1884, 'political': 1885, 'politically': 1886, 'politics': 1887, 'ponyo': 1888, 'poor': 1889, 'poorly': 1890, 'popcorn': 1891, 'popular': 1892, 'portrayal': 1893, 'portrayals': 1894, 'portrayed': 1895, 'portraying': 1896, 'positive': 1897, 'possible': 1898, 'possibly': 1899, 'post': 1900, 'potted': 1901, 'power': 1902, 'powerful': 1903, 'powerhouse': 1904, 'practical': 1905, 'practically': 1906, 'pray': 1907, 'precisely': 1908, 'predict': 1909, 'predictable': 1910, 'predictably': 1911, 'prejudice': 1912, 'prelude': 1913, 'premise': 1914, 'prepared': 1915, 'presence': 1916, 'presents': 1917, 'preservation': 1918, 'president': 1919, 'pretentious': 1920, 'pretext': 1921, 'pretty': 1922, 'previous': 1923, 'primal': 1924, 'primary': 1925, 'probably': 1926, 'problem': 1927, 'problems': 1928, 'proceedings': 1929, 'process': 1930, 'produce': 1931, 'produced': 1932, 'producer': 1933, 'producers': 1934, 'product': 1935, 'production': 1936, 'professionals': 1937, 'professor': 1938, 'progresses': 1939, 'promote': 1940, 'prompted': 1941, 'prone': 1942, 'propaganda': 1943, 'properly': 1944, 'proud': 1945, 'proudly': 1946, 'provided': 1947, 'provokes': 1948, 'provoking': 1949, 'ps': 1950, 'pseudo': 1951, 'psychological': 1952, 'psychotic': 1953, 'public': 1954, 'pull': 1955, 'pulling': 1956, 'pulls': 1957, 'punched': 1958, 'punches': 1959, 'punish': 1960, 'punishment': 1961, 'puppet': 1962, 'puppets': 1963, 'pure': 1964, 'purity': 1965, 'put': 1966, 'putting': 1967, 'puzzle': 1968, 'pyromaniac': 1969, 'qu': 1970, 'quaid': 1971, 'qualities': 1972, 'quality': 1973, 'question': 1974, 'questioning': 1975, 'quick': 1976, 'quicker': 1977, 'quiet': 1978, 'quinn': 1979, 'quite': 1980, 'race': 1981, 'racial': 1982, 'racism': 1983, 'radiant': 1984, 'raging': 1985, 'random': 1986, 'range': 1987, 'ranks': 1988, 'rare': 1989, 'rate': 1990, 'rated': 1991, 'rather': 1992, 'rating': 1993, 'ratings': 1994, 'raver': 1995, 'raw': 1996, 'ray': 1997, 'reactions': 1998, 'readers': 1999, 'reading': 2000, 'ready': 2001, 'real': 2002, 'realised': 2003, 'realistic': 2004, 'reality': 2005, 'realize': 2006, 'realized': 2007, 'really': 2008, 'reason': 2009, 'reasonable': 2010, 'reasons': 2011, 'receive': 2012, 'received': 2013, 'recent': 2014, 'recently': 2015, 'recommend': 2016, 'recommended': 2017, 'reconciliation': 2018, 'recover': 2019, 'recurring': 2020, 'redeemed': 2021, 'redeeming': 2022, 'reenactments': 2023, 'references': 2024, 'reflected': 2025, 'refreshing': 2026, 'regardless': 2027, 'regret': 2028, 'regrettable': 2029, 'regrettably': 2030, 'rejection': 2031, 'relate': 2032, 'related': 2033, 'relation': 2034, 'relations': 2035, 'relationship': 2036, 'relationships': 2037, 'relatively': 2038, 'relaxing': 2039, 'release': 2040, 'released': 2041, 'relief': 2042, 'relying': 2043, 'remaining': 2044, 'remake': 2045, 'remarkable': 2046, 'remember': 2047, 'reminded': 2048, 'remotely': 2049, 'removing': 2050, 'rendering': 2051, 'rendition': 2052, 'renowned': 2053, 'rent': 2054, 'repair': 2055, 'repeated': 2056, 'repeating': 2057, 'repeats': 2058, 'repertory': 2059, 'reporter': 2060, 'represents': 2061, 'require': 2062, 'rescue': 2063, 'researched': 2064, 'resounding': 2065, 'respecting': 2066, 'rest': 2067, 'restrained': 2068, 'result': 2069, 'results': 2070, 'resume': 2071, 'retarded': 2072, 'retreat': 2073, 'return': 2074, 'revealing': 2075, 'revenge': 2076, 'revere': 2077, 'reverse': 2078, 'review': 2079, 'reviewer': 2080, 'reviewers': 2081, 'reviews': 2082, 'rice': 2083, 'rickman': 2084, 'ridiculous': 2085, 'ridiculousness': 2086, 'right': 2087, 'riot': 2088, 'rips': 2089, 'rise': 2090, 'rita': 2091, 'rivalry': 2092, 'riveted': 2093, 'riz': 2094, 'road': 2095, 'robert': 2096, 'robotic': 2097, 'rochon': 2098, 'rocked': 2099, 'rocks': 2100, 'roeg': 2101, 'role': 2102, 'roles': 2103, 'roller': 2104, 'rolls': 2105, 'romantic': 2106, 'room': 2107, 'roosevelt': 2108, 'roth': 2109, 'rough': 2110, 'round': 2111, 'routine': 2112, 'row': 2113, 'rpg': 2114, 'rpger': 2115, 'rubbish': 2116, 'rubin': 2117, 'rumbles': 2118, 'run': 2119, 'running': 2120, 'ruthless': 2121, 'ryan': 2122, 'ryans': 2123, 'sabotages': 2124, 'sack': 2125, 'sacrifice': 2126, 'sad': 2127, 'said': 2128, 'sake': 2129, 'salesman': 2130, 'sam': 2131, 'sample': 2132, 'sand': 2133, 'sandra': 2134, 'sappiest': 2135, 'sarcophage': 2136, 'sat': 2137, 'satanic': 2138, 'savalas': 2139, 'savant': 2140, 'save': 2141, 'savor': 2142, 'saw': 2143, 'say': 2144, 'says': 2145, 'scale': 2146, 'scamp': 2147, 'scare': 2148, 'scared': 2149, 'scares': 2150, 'scary': 2151, 'scene': 2152, 'scenery': 2153, 'scenes': 2154, 'schilling': 2155, 'schizophrenic': 2156, 'school': 2157, 'schoolers': 2158, 'schrader': 2159, 'schultz': 2160, 'sci': 2161, 'science': 2162, 'scientist': 2163, 'score': 2164, 'scot': 2165, 'scream': 2166, 'screamy': 2167, 'screen': 2168, 'screened': 2169, 'screenplay': 2170, 'screenwriter': 2171, 'scrimm': 2172, 'script': 2173, 'scripted': 2174, 'scripting': 2175, 'scripts': 2176, 'sculpture': 2177, 'sea': 2178, 'seamless': 2179, 'seamlessly': 2180, 'sean': 2181, 'season': 2182, 'seat': 2183, 'second': 2184, 'secondary': 2185, 'secondly': 2186, 'see': 2187, 'seeing': 2188, 'seem': 2189, 'seemed': 2190, 'seems': 2191, 'seen': 2192, 'selections': 2193, 'self': 2194, 'sells': 2195, 'semi': 2196, 'senior': 2197, 'sense': 2198, 'senses': 2199, 'sensibility': 2200, 'sensitivities': 2201, 'sentiment': 2202, 'seperate': 2203, 'sequel': 2204, 'sequels': 2205, 'sequence': 2206, 'sequences': 2207, 'series': 2208, 'serious': 2209, 'seriously': 2210, 'served': 2211, 'set': 2212, 'sets': 2213, 'setting': 2214, 'settings': 2215, 'seuss': 2216, 'several': 2217, 'sex': 2218, 'shakespear': 2219, 'shakespears': 2220, 'shallow': 2221, 'shame': 2222, 'shameful': 2223, 'share': 2224, 'sharing': 2225, 'sharply': 2226, 'shatner': 2227, 'shattered': 2228, 'shed': 2229, 'sheer': 2230, 'shelf': 2231, 'shell': 2232, 'shelves': 2233, 'shenanigans': 2234, 'shepard': 2235, 'shined': 2236, 'shirley': 2237, 'shocking': 2238, 'shooting': 2239, 'short': 2240, 'shortlist': 2241, 'shot': 2242, 'shots': 2243, 'show': 2244, 'showcasing': 2245, 'showed': 2246, 'shows': 2247, 'shut': 2248, 'sibling': 2249, 'sick': 2250, 'side': 2251, 'sidelined': 2252, 'sign': 2253, 'significant': 2254, 'silent': 2255, 'silly': 2256, 'simmering': 2257, 'simplifying': 2258, 'simply': 2259, 'since': 2260, 'sincere': 2261, 'sing': 2262, 'singing': 2263, 'single': 2264, 'sinister': 2265, 'sink': 2266, 'sinking': 2267, 'sister': 2268, 'sisters': 2269, 'sit': 2270, 'sitcoms': 2271, 'site': 2272, 'sites': 2273, 'sits': 2274, 'situation': 2275, 'situations': 2276, 'skilled': 2277, 'skip': 2278, 'slackers': 2279, 'slavic': 2280, 'sleep': 2281, 'slideshow': 2282, 'slightest': 2283, 'slightly': 2284, 'slimy': 2285, 'sloppy': 2286, 'slow': 2287, 'slurs': 2288, 'smack': 2289, 'small': 2290, 'smart': 2291, 'smells': 2292, 'smile': 2293, 'smiling': 2294, 'smith': 2295, 'smoothly': 2296, 'snider': 2297, 'snow': 2298, 'soap': 2299, 'sobering': 2300, 'social': 2301, 'soldiers': 2302, 'sole': 2303, 'solid': 2304, 'solidifying': 2305, 'solving': 2306, 'someone': 2307, 'something': 2308, 'sometimes': 2309, 'somewhat': 2310, 'son': 2311, 'song': 2312, 'songs': 2313, 'soon': 2314, 'sophisticated': 2315, 'sorrentino': 2316, 'sorry': 2317, 'sort': 2318, 'soul': 2319, 'sound': 2320, 'sounded': 2321, 'sounds': 2322, 'soundtrack': 2323, 'sour': 2324, 'south': 2325, 'southern': 2326, 'space': 2327, 'spacek': 2328, 'spacey': 2329, 'span': 2330, 'speak': 2331, 'speaking': 2332, 'special': 2333, 'speed': 2334, 'spend': 2335, 'spent': 2336, 'spew': 2337, 'sphere': 2338, 'spiffy': 2339, 'splendid': 2340, 'spock': 2341, 'spoil': 2342, 'spoiled': 2343, 'spoiler': 2344, 'spoilers': 2345, 'spot': 2346, 'spy': 2347, 'squibs': 2348, 'stable': 2349, 'stage': 2350, 'stagy': 2351, 'stand': 2352, 'standout': 2353, 'stanwyck': 2354, 'star': 2355, 'starlet': 2356, 'starring': 2357, 'stars': 2358, 'start': 2359, 'started': 2360, 'starts': 2361, 'state': 2362, 'stay': 2363, 'stayed': 2364, 'stealing': 2365, 'steamboat': 2366, 'steele': 2367, 'step': 2368, 'stephen': 2369, 'stereotypes': 2370, 'stereotypically': 2371, 'steve': 2372, 'stewart': 2373, 'stick': 2374, 'still': 2375, 'stinker': 2376, 'stinks': 2377, 'stocking': 2378, 'stockings': 2379, 'stoic': 2380, 'store': 2381, 'stories': 2382, 'storm': 2383, 'story': 2384, 'storyline': 2385, 'storytelling': 2386, 'stowe': 2387, 'strange': 2388, 'stranger': 2389, 'stratus': 2390, 'straw': 2391, 'street': 2392, 'strident': 2393, 'string': 2394, 'strives': 2395, 'strokes': 2396, 'strong': 2397, 'struck': 2398, 'structure': 2399, 'struggle': 2400, 'stuart': 2401, 'student': 2402, 'students': 2403, 'studio': 2404, 'study': 2405, 'stuff': 2406, 'stunning': 2407, 'stupid': 2408, 'stupidity': 2409, 'style': 2410, 'stylized': 2411, 'sub': 2412, 'subject': 2413, 'subjects': 2414, 'sublime': 2415, 'sublimely': 2416, 'subplots': 2417, 'subtitles': 2418, 'subtle': 2419, 'subversive': 2420, 'subverting': 2421, 'succeeded': 2422, 'succeeds': 2423, 'success': 2424, 'suck': 2425, 'sucked': 2426, 'sucks': 2427, 'suffered': 2428, 'suffering': 2429, 'suggest': 2430, 'suggests': 2431, 'suited': 2432, 'sum': 2433, 'summary': 2434, 'sundays': 2435, 'super': 2436, 'superb': 2437, 'superbad': 2438, 'superbly': 2439, 'superficial': 2440, 'superlative': 2441, 'supernatural': 2442, 'supporting': 2443, 'supposed': 2444, 'supposedly': 2445, 'sure': 2446, 'surely': 2447, 'surf': 2448, 'surface': 2449, 'surprised': 2450, 'surprises': 2451, 'surprising': 2452, 'surprisingly': 2453, 'surrounding': 2454, 'surroundings': 2455, 'survivors': 2456, 'suspense': 2457, 'suspension': 2458, 'sven': 2459, 'swamp': 2460, 'sweep': 2461, 'sweet': 2462, 'switched': 2463, 'swords': 2464, 'sydney': 2465, 'sympathetic': 2466, 'syrupy': 2467, 'system': 2468, 'tacky': 2469, 'taelons': 2470, 'take': 2471, 'taken': 2472, 'takes': 2473, 'taking': 2474, 'tale': 2475, 'talent': 2476, 'talented': 2477, 'talents': 2478, 'talk': 2479, 'tanks': 2480, 'taped': 2481, 'tardis': 2482, 'task': 2483, 'taste': 2484, 'taxidermists': 2485, 'taylor': 2486, 'teacher': 2487, 'teaches': 2488, 'team': 2489, 'tear': 2490, 'tears': 2491, 'technically': 2492, 'teddy': 2493, 'tedium': 2494, 'teen': 2495, 'teenagers': 2496, 'teeth': 2497, 'telephone': 2498, 'television': 2499, 'tell': 2500, 'telly': 2501, 'temperaments': 2502, 'ten': 2503, 'tender': 2504, 'tension': 2505, 'tensions': 2506, 'terminology': 2507, 'terms': 2508, 'terrible': 2509, 'terribly': 2510, 'terrific': 2511, 'terror': 2512, 'th': 2513, 'thanks': 2514, 'theater': 2515, 'theatre': 2516, 'theatres': 2517, 'theatrical': 2518, 'theme': 2519, 'themes': 2520, 'therapy': 2521, 'thick': 2522, 'thing': 2523, 'things': 2524, 'think': 2525, 'thinking': 2526, 'thomerson': 2527, 'thoroughly': 2528, 'thorsen': 2529, 'though': 2530, 'thought': 2531, 'thoughts': 2532, 'thousand': 2533, 'thread': 2534, 'three': 2535, 'threshold': 2536, 'thrilled': 2537, 'thriller': 2538, 'thrillers': 2539, 'throughout': 2540, 'throwback': 2541, 'thrown': 2542, 'thug': 2543, 'thumper': 2544, 'thunderbirds': 2545, 'thus': 2546, 'ticker': 2547, 'tickets': 2548, 'tightly': 2549, 'time': 2550, 'timeless': 2551, 'timely': 2552, 'timers': 2553, 'times': 2554, 'timing': 2555, 'tiny': 2556, 'tired': 2557, 'title': 2558, 'titta': 2559, 'today': 2560, 'together': 2561, 'told': 2562, 'tolerable': 2563, 'tolerate': 2564, 'tom': 2565, 'tomorrow': 2566, 'tone': 2567, 'tongue': 2568, 'tonight': 2569, 'tons': 2570, 'tony': 2571, 'took': 2572, 'toons': 2573, 'top': 2574, 'tops': 2575, 'torture': 2576, 'tortured': 2577, 'total': 2578, 'totally': 2579, 'touch': 2580, 'touches': 2581, 'touching': 2582, 'tough': 2583, 'towards': 2584, 'towers': 2585, 'townsend': 2586, 'track': 2587, 'tract': 2588, 'traditional': 2589, 'traffic': 2590, 'trailer': 2591, 'train': 2592, 'tranquillity': 2593, 'transcend': 2594, 'transfer

```
s': 2595, 'translate': 2596, 'translating': 2597, 'trap': 2598, 'trash': 2599, 'trashy': 2600, 'treachery': 2601, 'treasure': 2602, 'treat': 2603, 'treatments': 2604, 'trek': 2605,
'tremendous': 2606, 'tremendously': 2607, 'tries': 2608, 'trilogy': 2609, 'trinity': 2610, 'trip': 2611, 'triumphed': 2612, 'trond': 2613, 'trooper': 2614, 'trouble': 2615, 'truc
k': 2616, 'true': 2617, 'truly': 2618, 'trumbull': 2619, 'trumpeter': 2620, 'truth': 2621, 'try': 2622, 'trying': 2623, 'trysts': 2624, 'tsunami': 2625, 'tuneful': 2626, 'turkey':
2627, 'turn': 2628, 'turned': 2629, 'turns': 2630, 'tv': 2631, 'twice': 2632, 'twirling': 2633, 'twist': 2634, 'twists': 2635, 'two': 2636, 'tying': 2637, 'type': 2638, 'typical':
2639, 'ue': 2640, 'ugliest': 2641, 'ugly': 2642, 'uhura': 2643, 'ultra': 2644, 'um': 2645, 'unaccompanied': 2646, 'unbearable': 2647, 'unbearably': 2648, 'unbelievable': 2649, 'unc
alled': 2650, 'unconditional': 2651, 'unconvincing': 2652, 'underacting': 2653, 'underappreciated': 2654, 'underbite': 2655, 'underlines': 2656, 'underlying': 2657, 'underneath': 2
658, 'underrated': 2659, 'understand': 2660, 'understanding': 2661, 'understated': 2662, 'understatement': 2663, 'understood': 2664, 'undertone': 2665, 'underwater': 2666, 'undoubt
edly': 2667, 'uneasy': 2668, 'unemployed': 2669, 'unethical': 2670, 'unfaithful': 2671, 'unfolds': 2672, 'unforgettable': 2673, 'unfortunate': 2674, 'unfortunately': 2675, 'unfunn
y': 2676, 'unintentionally': 2677, 'uninteresting': 2678, 'union': 2679, 'unique': 2680, 'uniqueness': 2681, 'universal': 2682, 'universe': 2683, 'unless': 2684, 'unlockable': 268
5, 'unmatched': 2686, 'unmitigated': 2687, 'unmoving': 2688, 'unnecessary': 2689, 'unneeded': 2690, 'unoriginal': 2691, 'unpleasant': 2692, 'unpredictability': 2693, 'unpredictabl
e': 2694, 'unrealistic': 2695, 'unrecognizable': 2696, 'unrecommended': 2697, 'unremarkable': 2698, 'unrestrained': 2699, 'unsatisfactory': 2700, 'unwatchable': 2701, 'upa': 2702,
'uplifting': 2703, 'upper': 2704, 'ups': 2705, 'uptight': 2706, 'ursula': 2707, 'us': 2708, 'use': 2709, 'used': 2710, 'user': 2711, 'uses': 2712, 'using': 2713, 'ussr': 2714, 'usu
al': 2715, 'utter': 2716, 'utterly': 2717, 'valentine': 2718, 'value': 2719, 'values': 2720, 'vampire': 2721, 'vandiver': 2722, 'variation': 2723, 'vehicles': 2724, 'ventura': 272
5, 'verbal': 2726, 'verbatim': 2727, 'versatile': 2728, 'version': 2729, 'versus': 2730, 'vessel': 2731, 'veteran': 2732, 'vey': 2733, 'vibe': 2734, 'victor': 2735, 'video': 2736,
'view': 2737, 'viewer': 2738, 'viewing': 2739, 'views': 2740, 'villain': 2741, 'villains': 2742, 'violence': 2743, 'violin': 2744, 'virtue': 2745, 'virus': 2746, 'vision': 2747, 'v
isual': 2748, 'visually': 2749, 'vitally': 2750, 'vivian': 2751, 'vivid': 2752, 'vocal': 2753, 'voice': 2754, 'volatile': 2755, 'volcano': 2756, 'vomit': 2757, 'vomited': 2758, 'vo
yage': 2759, 'vulcan': 2760, 'waiting': 2761, 'waitress': 2762, 'walk': 2763, 'walked': 2764, 'wall': 2765, 'want': 2766, 'wanted': 2767, 'wanting': 2768, 'wants': 2769, 'war': 277
0, 'warmth': 2771, 'warn': 2772, 'warning': 2773, 'wartime': 2774, 'warts': 2775, 'washed': 2776, 'washing': 2777, 'waste': 2778, 'wasted': 2779, 'waster': 2780, 'wasting': 2781,
'watch': 2782, 'watchable': 2783, 'watched': 2784, 'watching': 2785, 'water': 2786, 'watkins': 2787, 'watson': 2788, 'wave': 2789, 'way': 2790, 'waylaid': 2791, 'wayne': 2792, 'way
s': 2793, 'wb': 2794, 'weak': 2795, 'weaker': 2796, 'weariness': 2797, 'weaving': 2798, 'website': 2799, 'wedding': 2800, 'weight': 2801, 'weird': 2802, 'well': 2803, 'welsh': 280
4, 'went': 2805, 'whatever': 2806, 'whatsoever': 2807, 'whenever': 2808, 'whether': 2809, 'whine': 2810, 'whiny': 2811, 'white': 2812, 'whites': 2813, 'whoever': 2814, 'whole': 281
5, 'wholesome': 2816, 'wide': 2817, 'widmark': 2818, 'wife': 2819, 'wih': 2820, 'wild': 2821, 'wilkinson': 2822, 'william': 2823, 'willie': 2824, 'wily': 2825, 'win': 2826, 'wind':
2827, 'wise': 2828, 'wish': 2829, 'within': 2830, 'without': 2831, 'witticisms': 2832, 'witty': 2833, 'woa': 2834, 'women': 2835, 'wonder': 2836, 'wondered': 2837, 'wonderful': 283
8, 'wonderfully': 2839, 'wong': 2840, 'wont': 2841, 'woo': 2842, 'wooden': 2843, 'word': 2844, 'words': 2845, 'work': 2846, 'worked': 2847, 'working': 2848, 'works': 2849, 'world':
2850, 'worry': 2851, 'worse': 2852, 'worst': 2853, 'worth': 2854, 'worthless': 2855, 'worthwhile': 2856, 'worthy': 2857, 'would': 2858, 'wouldnt': 2859, 'woven': 2860, 'wow': 2861,
'wrap': 2862, 'write': 2863, 'writer': 2864, 'writers': 2865, 'writing': 2866, 'written': 2867, 'wrong': 2868, 'wrote': 2869, 'yardley': 2870, 'yawn': 2871, 'yeah': 2872, 'year': 2
873, 'years': 2874, 'yelps': 2875, 'yes': 2876, 'yet': 2877, 'young': 2878, 'younger': 2879, 'youthful': 2880, 'youtube': 2881, 'yun': 2882, 'zillion': 2883, 'zombie': 2884, 'zombi
```

In [86]:

```
def transform(corpus):

    '''This fuction calculates the Tfidf of the document corpus given vocab:enumaration of words'''
    print("→ Make sure to print instead of raw calling the function\ne.g,x=transform(corpus,vocab)\n   print(x[0])")

    if isinstance(corpus,list):
        global vocab1
        number_of_docs=len(corpus)

        #1:Calculating frequency
        freq=[row.split()for row in corpus]     #list of lists of words in each row
        frequency_of_words=[]
        for row in freq:                    #getting the word frequencies
            for word in set(row):
                if len(word)<2:             #avoiding sigle letters
                    continue
                frequency_of_words.append(word)
        #In how many documents each word occured in?
        frequency_of_words=dict(Counter(frequency_of_words))

        #2:Calculating idf of words
        idf={}
        vocab={}
        for row in corpus:
            for word in set(row.split()):     #.set() for avoiding repeated words
                if len(word)<2:
                    continue
                idf[word]=1+math.log((1+number_of_docs)/(1+frequency_of_words[word]))
        idf=dict(sorted(idf.items(),key=lambda x:x[1],reverse=True)[:50])  #top idf 50 values

        #3:getting the vocab for top 50 idf values
        vocab={}
        vocab1={}
        ind=0
        for word in idf.keys():
                vocab[word]=ind       # setting vocab for the words from top 50 idf values
                vocab1[word]=idf[word],ind
                ind+=1

        #4:calculating tfidf
        number_of_unique_words=len(vocab)
        csr=csr_matrix((number_of_docs,number_of_unique_words))  #setting the shape
        #help(csr_matrix)
        for ind,row in enumerate(tqdm(corpus)):
                words_counter=dict(Counter(row.split()))   #counting the number of each word in a row
                for word in row.split():
                    if word in idf.keys():
                        csr[ind,vocab[word]]=(words_counter[word]/len(row.split())  * idf[word]) #sparsematrix[r,c]=value
        tfidf=normalize(csr, norm ='l2')
        return tfidf
    else:
        print("Please enter the list of strings")
```

In [87]:

```
x=transform(corpus)
print(x)
```

```
→ Make sure to print instead of raw calling the function
e.g,x=transform(corpus,vocab)
   print(x[0])
100%|██████████████████████████████████████████████████████████| 746/746 [00:00<00:00, 28734.44it/s]
  (0, 0)        0.5773502691896257
  (0, 1)        0.5773502691896257
  (0, 2)        0.5773502691896257
  (1, 3)        1.0
  (2, 4)        0.5773502691896258
  (2, 5)        0.5773502691896258
  (2, 6)        0.5773502691896258
  (4, 7)        1.0
  (5, 8)        1.0
  (7, 9)        0.7071067811865475
  (7, 10)       0.7071067811865475
  (9, 11)       0.5773502691896257
  (9, 12)       0.5773502691896257
  (9, 13)       0.5773502691896257
  (10, 14)      0.7071067811865476
  (10, 15)      0.7071067811865476
  (11, 16)      1.0
  (12, 17)      1.0
  (15, 18)      1.0
  (16, 19)      1.0
```

```
  (17, 20)      0.7071067811865475
  (17, 21)      0.7071067811865475
  (18, 22)      1.0
  (19, 23)      0.1690308509457033
  (19, 24)      0.1690308509457033
  (19, 25)      0.1690308509457033
  (19, 26)      0.1690308509457033
  (19, 27)      0.1690308509457033
  (19, 28)      0.1690308509457033
  (19, 29)      0.1690308509457033
  (19, 30)      0.1690308509457033
  (19, 31)      0.1690308509457033
  (19, 32)      0.1690308509457033
  (19, 33)      0.1690308509457033
  (19, 34)      0.1690308509457033
  (19, 35)      0.1690308509457033
  (19, 36)      0.1690308509457033
  (19, 37)      0.1690308509457033
  (19, 38)      0.5070925528371099
  (19, 39)      0.1690308509457033
  (19, 40)      0.1690308509457033
  (19, 41)      0.1690308509457033
  (19, 42)      0.1690308509457033
  (19, 43)      0.1690308509457033
  (19, 44)      0.1690308509457033
  (19, 45)      0.1690308509457033
  (19, 46)      0.1690308509457033
  (19, 47)      0.1690308509457033
  (19, 48)      0.1690308509457033
  (19, 49)      0.1690308509457033
```

In [88]:

```
x=transform(corpus)
y=x[0].todense()
print(y,"\nshape:",y.shape)
```

```
→ Make sure to print instead of raw calling the function
e.g,x=transform(corpus,vocab)
   print(x[0])
100%|[redacted]| 746/746 [00:00<00:00, 125464.16it/s]
[[0.57735027 0.57735027 0.57735027 0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.        ]]
shape: (1, 50)
```

In [89]:

```
print("vocab[index]\t word\t  \tidf_value")
for key,value in vocab1.items():
    print(f"\t{value[1]} \t{key}    \t {value[0]}")
print("TotalWords:",len(vocab1))
```

```
vocab[index]     word           idf_value
        0       drifting         6.922918004572872
        1       aimless          6.922918004572872
        2       distressed       6.922918004572872
        3       nearly           6.922918004572872
        4       artiness         6.922918004572872
        5       attempting       6.922918004572872
        6       existent         6.922918004572872
        7       gerardo          6.922918004572872
        8       emptiness        6.922918004572872
        9       effort           6.922918004572872
        10      messages         6.922918004572872
        11      science          6.922918004572872
        12      teacher          6.922918004572872
        13      buffet           6.922918004572872
        14      baby             6.922918004572872
        15      owls             6.922918004572872
        16      florida          6.922918004572872
        17      muppets          6.922918004572872
        18      person           6.922918004572872
        19      overdue          6.922918004572872
        20      post             6.922918004572872
        21      screenplay       6.922918004572872
        22      practically      6.922918004572872
        23      cross            6.922918004572872
        24      helms            6.922918004572872
        25      nerves           6.922918004572872
        26      ladies           6.922918004572872
        27      succeeds         6.922918004572872
        28      integration      6.922918004572872
        29      lived            6.922918004572872
        30      choices          6.922918004572872
        31      dodge            6.922918004572872
        32      pulls            6.922918004572872
        33      blue             6.922918004572872
        34      freeman          6.922918004572872
        35      efforts          6.922918004572872
        36      translate        6.922918004572872
        37      rocks            6.922918004572872
        38      massive          6.922918004572872
        39      cutest           6.922918004572872
        40      solving          6.922918004572872
        41      heche            6.922918004572872
        42      meagre           6.922918004572872
        43      require          6.922918004572872
        44      help             6.922918004572872
        45      managed          6.922918004572872
        46      concentrate      6.922918004572872
        47      content          6.922918004572872
        48      unlockable       6.922918004572872
        49      crackles         6.922918004572872
TotalWords: 50
```