```
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
```

```
def fit_50(corpus):
  #creating set for storing unique_words
  u_w = set()
  if isinstance(corpus, list):
    #iterating through rows in corpus
    for row in corpus:
      #iterating through each words in row
      for words in row.split():
        #checking for length of words
        if len(words) < 2:
          continue
        #adding each words in set
        u_w.add(words)

    #converting set into sorted list
    u_w = list(u_w)

    #creating dict with words in list as keys and enumerated index as values
    vocab = {j:i for i, j in enumerate(u_w)}

    #creating dict with keys in vocab dict as keys and its idf as values
    vocab_idf = {word:get_idf_50(word,corpus) for word in vocab.keys()}

    #gettingtop 50 values based on idf values
    vocab_idf_50 = dict(sorted(vocab_idf.items(),key=operator.itemgetter(1),reverse=Tru

    #creating dict with top 50 words as keys and its enumerated index as values
    vocab_50 = {j:i for i, j in enumerate(vocab_idf_50.keys())}

    return vocab_50
  else:
    print("Send list of Sentences")
```

```
def transform_50(corpus, vocab_50):
  #creating empty lists for rows, column, values
  rows = []
  columns = []
  values = []
  if isinstance(corpus, list):
    #iterating through rows in corpus
    for idx, row in enumerate(corpus):
      #creating dict with words in row and its count as values
      word_freq = dict(Counter(row.split()))
      #iterating through keys in top 50 vocab dict
```

```
      for word in vocab_50.keys():

        tfidf = (word_freq.get(word, 0) / len(row.split())) * get_idf_50(word, corpus)

        if tfidf != 0:
          #appending row index(idx) into row list
          rows.append(idx)

          #getting column index from top 50 vocab dict
          col_index = vocab_50.get(word, 0)
          columns.append(col_index)

          #appending tfidf to values list
          values.append(tfidf)
    #norm = normalize(values)

    #return csr_matrix(norm)
    return csr_matrix((values, (rows, columns)), shape=(len(corpus),len(vocab_50)))
  else:
    print("Send list of Sentences")
```

```
def get_idf_50(word, corpus):
  count=0
  #iterating through rows in corpus
  for r in corpus:
    #if word in that row increament count by one
    if word in r:
      count += 1
  idf_key= 1 + math.log((1+len(corpus)) / (count+1))
  return idf_key
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pickle
with open('/content/drive/My Drive/Dataset/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
```

```
#printing keys in top 50 vocab dict
vocab_50 = fit_50(corpus)
print(list(vocab_50.keys()))
```

```
['garfield', 'tongue', 'sacrifice', 'superlative', 'cheerless', 'guys', 'letting', 's
```

```
##top 50 idf values after fit method
```

```
#top 50 idf values after fit method
idf_lst = [get_idf_50(i, corpus) for i in vocab_50.keys()]
print(idf_lst)
```

```
[6.922918004572872, 6.922918004572872, 6.922918004572872, 6.922918004572872, 6.92291
```

```
#shape of output matrix
print(numpy.shape(transform_50(corpus, vocab_50).toarray()))
```

```
(746, 50)
```

```
tr = transform_50(corpus, vocab_50)
print(tr[19])
```

```
  (0, 3)        0.015769744885131828
  (0, 25)       0.015769744885131828
  (0, 26)       0.015769744885131828
  (0, 46)       0.015769744885131828
```

```
print(tr[19].toarray())
```

```
[[0.         0.         0.         0.01576974 0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.01576974 0.01576974 0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.01576974 0.
  0.         0.        ]]
```

```
1
```

✓ 0s completed at 15:33