

```

1 from collections import Counter
2 from tqdm import tqdm
3 from scipy.sparse import csr_matrix
4 import math
5 import operator
6 from sklearn.preprocessing import normalize
7 import numpy

```

```

1 corpus = [
2     'this is the first document',
3     'this document is the second document',
4     'and this is the third one',
5     'is this the first document',
6 ]

```

```

1 def fit(corpus):
2     #creating set for storing unique_words
3     unique_words = set()
4     if isinstance(corpus, list):
5         #iterating through rows in corpus
6         for row in corpus:
7             #iterating through each words in row
8             for word in row.split():
9                 #checking for length of words
10                if len(word) < 2:
11                    continue
12                #adding each words in set
13                unique_words.add(word)
14            #converting set into sorted list
15            unique_words = sorted(list(unique_words))
16            #creating vocab dict
17            vocab = {j:i for i , j in enumerate(unique_words)}
18
19        return vocab
20    else:
21        print("Send list of Sentences")

```

```

1 def transform(corpus, vocab):
2     #creating empty lists for rows, column, values
3     rows = []
4     columns = []
5     val = []
6     #iterating through rows of corpus
7     for idx, row in enumerate(tqdm(corpus)):
8         values = []
9         #creating word_frequency dict using counter
10        word_freq = dict(Counter(row.split()))
11        #iterating through words in vocab dict
12        for word in vocab.keys():
13            #calculating tfidf using formula
14            tfidf = (word_freq.get(word, 0) / len(row.split())) * get_idf(word, corpus)
15            col_index = vocab.get(word, 0)

```

```

15     col_index = vocab.get(word, 0)
16
17     if tfidf != 0:
18         rows.append(idx)
19         columns.append(col_index)
20         values.append(tfidf)
21     val.append(values)
22 norm = normalize(val)
23 #return csr_matrix((norm), shape=(len(corpus),len(vocab)))
24 #return csr_matrix((norm, (rows, columns)), shape=(len(corpus),len(vocab)))
25 return csr_matrix((norm))

```

```

1 def get_idf(word, corpus):
2     count=0
3     #iterating through rows in corpus
4     for r in corpus:
5         #if word in that row increament count by one
6         if word in r:
7             count += 1
8
9     idf_key= 1 + math.log((1+len(corpus)) / (count+1))
10    return idf_key

```

```

1 #calling fit method
2 vocab = fit(corpus)
3 #creating dictionary with words in vocab as keys and its idf value as values
4 vocab_idf = {word:get_idf(word, corpus) for word in vocab.keys()}

```

```

1 #printing all words in vocab dict
2 print(list(vocab.keys()))

```

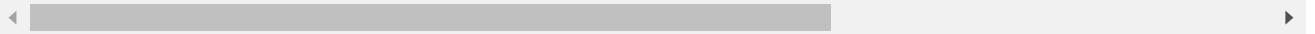
```
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
```

```

1 #printing idf values for words in vocab dict
2 print(list(vocab_idf.values()))

```

```
[1.916290731874155, 1.2231435513142097, 1.5108256237659907, 1.0, 1.916290731874155, 1
```



```

1 #shape of output matrix
2 print(numpy.shape(transform(corpus, vocab)))

```

```
100%|██████████| 4/4 [00:00<00:00, 7588.07it/s](4, 9)
```

```

1 tr = transform(corpus, vocab)
2 print(tr)

```

```

100%|██████████| 4/4 [00:00<00:00, 10094.59it/s] (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
```

```
1 print(tr[0].toarray())
```

```
[[0.          0.46979139 0.58028582 0.38408524 0.          0.
  0.38408524 0.          0.38408524]]
```

```
1
```

✓ 0s completed at 15:32

