In [172]:
```
corpus=['this is the first document',
     'this is the second document',
     'and this is the third one',
     'is this the first document',]
```

In [173]:
```
from tqdm import tqdm # tqdm is a library that helps us to visualize the runtime
#https://tqdm.github.io/
# it accepts only list of sentances
def fit(dataset):
    unique_words = set() # at first we will initialize an empty set
    # check if its list type or not
    if isinstance(dataset, (list,)):
        for row in dataset: # for each review in the dataset
            for word in row.split(" "): # for each word in the review. #split met
                if len(word) < 2:
                    continue
                unique_words.add(word)
        unique_words = sorted(list(unique_words))
        vocab = {j:i for i,j in enumerate(unique_words)}
        #vocab = dict.fromkeys(unique_words, 0)
        return vocab
    else:
        print("you need to pass list of sentance")
```

In [174]:
```
unique_words=[]
vocab = fit(corpus)
print(vocab)
for i in vocab.keys():
    unique_words.append(i)
print(unique_words)
```

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

In [179]:
```
#Returs a sparse matrix of the all non-zero values along with their row and col

import math
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 transform(dataset,vocab):
        row = []
        col = []
        values = []
        for ibx,document in enumerate(dataset):
            #print(dataset)
            word_freq = dict(Counter(document.split()))
            print(word_freq)
            for word, freq in word_freq.items():
                col_index = vocab.get(word,-1)
                if col_index != -1:
                    if len(word)<2:
                        continue
                col.append(col_index)
                row.append(ibx)

                tf=freq/float(len(word_freq))
                #print(word,tf)
                #td = freq/float(len(document)) # the number of times a word occu
                idf=IDF1(corpus,unique_words,word)
                #print(word,freq,td,idf)
                values.append((tf)*(idf))
        print(len(values),len(row),len(col),len(vocab),len(dataset))
        return normalize(csr_matrix( ((values),(row,col)), shape=(len(dataset),le

print(transform(corpus,vocab))

#   (0, 8)  0.38408524091481483
#   (0, 6)  0.38408524091481483
#   (0, 3)  0.38408524091481483
#   (0, 2)  0.5802858236844359
#   (0, 1)  0.46979138557992045
```

```
{'this': 1, 'is': 1, 'the': 1, 'first': 1, 'document': 1}
{'this': 1, 'is': 1, 'the': 1, 'second': 1, 'document': 1}
{'and': 1, 'this': 1, 'is': 1, 'the': 1, 'third': 1, 'one': 1}
{'is': 1, 'this': 1, 'the': 1, 'first': 1, 'document': 1}
21 21 21 9 4
  (0, 1)        0.4697913855799205
  (0, 2)        0.580285823684436
  (0, 3)        0.3840852409148149
  (0, 6)        0.3840852409148149
  (0, 8)        0.3840852409148149
  (1, 1)        0.4279695901493821
  (1, 3)        0.34989318276628206
  (1, 5)        0.6704970632809761
```

```
(1, 6)        0.34989318276628206
(1, 8)        0.34989318276628206
(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
```

In [176]:
```
import numpy as np
def IDF1(corpus,unique_words,word):
    idf_dict={}
    N=len(corpus)
    for i in unique_words:
        count=0
        for row in corpus:
            if i in row.split():
                count+=1
    # increase the count for every occurrence as +1
                idf_dict[i]=1+math.log((N+1)/(count+1))

    if word in idf_dict.keys():
        pass
    return idf_dict[word]
```

In [ ]:
```

```