In [44]:

```
corpus = ['it was the best of times',
          'it was the worst of times',
          'it was the age of wisdom',
          'it was the age of foolishness']
```

Caculating TFidf Vectorizer using sklearn library

In [45]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
vectorizer.fit(corpus)
skoutput = vectorizer.transform(corpus)
print(skoutput)
```

```
  (0, 7) 0.3169454420370736
  (0, 6) 0.4788492951654494
  (0, 5) 0.3169454420370736
  (0, 4) 0.3169454420370736
  (0, 3) 0.3169454420370736
  (0, 1) 0.6073596130854014
  (1, 9) 0.6073596130854014
  (1, 7) 0.3169454420370736
  (1, 6) 0.4788492951654494
  (1, 5) 0.3169454420370736
  (1, 4) 0.3169454420370736
  (1, 3) 0.3169454420370736
  (2, 8) 0.6073596130854014
  (2, 7) 0.3169454420370736
  (2, 5) 0.3169454420370736
  (2, 4) 0.3169454420370736
  (2, 3) 0.3169454420370736
  (2, 0) 0.4788492951654494
  (3, 7) 0.3169454420370736
  (3, 5) 0.3169454420370736
  (3, 4) 0.3169454420370736
  (3, 3) 0.3169454420370736
  (3, 2) 0.6073596130854014
  (3, 0) 0.4788492951654494
```

We found as sparse matrix ((row,column),value)

In [46]:

```
print(vectorizer.get_feature_names())
```

```
['age', 'best', 'foolishness', 'it', 'of', 'the', 'times', 'was', 'wisdom', 'worst']
```

Sklearn get_feature_names ,They are sorted in alphabetic order by default

In [47]:

```
print(vectorizer.idf_)
```

```
[1.51082562 1.91629073 1.91629073 1.         1.         1.
 1.51082562 1.         1.91629073 1.91629073]
```

IDF values are calculated for each word in corpus = 1+log (base(e))(total number of document/Number of document where the words occur)

In [48]:

```
print(skoutput.shape)
```

```
(4, 10)
```

In [49]:

```
print(skoutput[0])
```

```
  (0, 7) 0.3169454420370736
  (0, 6) 0.4788492951654494
  (0, 5) 0.3169454420370736
  (0, 4) 0.3169454420370736
  (0, 3) 0.3169454420370736
  (0, 1) 0.6073596130854014
```

Here output is the sparse matrix

In [50]:

```
print(skoutput[0].toarray())
```

```
[[0.         0.60735961 0.         0.31694544 0.31694544 0.31694544
  0.4788493  0.31694544 0.         0.         ]]
```

We have converted the sparse matrix to array with length of 10

Using Scratch

Let's import Library

In [51]:

```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
from tqdm import tqdm
```

In [55]:

```
from sklearn.preprocessing import normalize

def fit(dataset):
    unique_word = set()#at first we will intialize the empty set
    #check if dataset is list or not
    if isinstance(dataset,(list)):
        for row in dataset:
            for word in row.split(" "):
                if len(word)<2:
                    continue
                unique_word.add(word)
        unique_word = sorted(list(unique_word))
        vocab = {j:i for i,j in enumerate(unique_word)}
        return vocab
    else:
        print("you need to pass list of sentence")
```

In [56]:

```
# caculating the IDF values and Sparse matrix
import math
def transform(dataset,vocab):
    total_num_doc = len(dataset)
    #return total_num_doc
    dataset1 = ' '.join(dataset)
    #print(dataset1)
    vocab2 = ' '.join(vocab)
```

```
    #print(vocab2)
    print("IDF values")
    for word in vocab2.split(" "):
        number_of_word_occur = 0
        for word1 in dataset1.split(" "):
            if word == word1:
                number_of_word_occur += 1
        print(math.log(float(total_num_doc+1)/(number_of_word_occur+1))+1)
```

In [57]:

```
vocab = fit(["it was the best of times","it was the worst of times", "it was the age of wisdom","it was the age of foolishness"])
print(list(vocab.keys()))
answer = transform(corpus,vocab)
#print(answer)
```

```
['age', 'best', 'foolishness', 'it', 'of', 'the', 'times', 'was', 'wisdom', 'worst']
IDF values
1.5108256237659907
1.916290731874155
1.916290731874155
1.0
1.0
1.0
1.5108256237659907
1.0
1.916290731874155
1.916290731874155
```

Observation: We got the Unique features values and IDF values

In [ ]:

```
```