Train a TF-IDF model using a tiny dataset

In [1]:
```
from pprint import pprint # pretty-printer
from gensim import corpora, models, similarities


#corpus
documents = ["new york times",
             "new york post",
             "los angeles times"]

# Preprocessing (here, we only do tokenization)
toknized_documents = [[token for token in document.lower().split()] for document in documents]
pprint(toknized_documents)
```

```
[['new', 'york', 'times'], ['new', 'york', 'post'], ['los', 'angeles', 'times']]
```

In [2]:
```
# Create dictionary (aka : id => word (id2word) mapping)
dictionary = corpora.Dictionary(toknized_documents)
print("\n")
print(dictionary)
print("\n")
print(dictionary.num_docs)
# save dictionary as text for corpus inspection
dictionary.save_as_text("doc.txt")
```

```
Dictionary(6 unique tokens: ['new', 'times', 'york', 'post', 'angeles']...)


3
```

In [3]:
```
# To see the mapping between words and their ids:
print(dictionary.token2id)
print("\n",dictionary[0],dictionary[1])
```

```
{'new': 0, 'times': 1, 'york': 2, 'post': 3, 'angeles': 4, 'los': 5}

 new times
```

In [4]:
```
# vectorization: bag-of-word vector for each doc
corpus_doc2bow_vec = [dictionary.doc2bow(tok_doc) for tok_doc in toknized_documents]
print("doc2bow_vectors of the three documents : [(id_word, tf) , (id_word, tf) ,...,(id_word, tf)]:")
for c in corpus_doc2bow_vec:
    print(c)
```

```
doc2bow_vectors of the three documents : [(id_word, tf) , (id_word, tf) ,...,(id_word, tf)]:
[(0, 1), (1, 1), (2, 1)]
[(0, 1), (2, 1), (3, 1)]
[(1, 1), (4, 1), (5, 1)]
```

In [5]:
```
# train (compute) TF-IDF
tfidf_model = models.TfidfModel(corpus_doc2bow_vec,id2word = dictionary,normalize = False) #fit model
# Apply model
corpus_tfidf_vectors = tfidf_model[corpus_doc2bow_vec]
print("tfidf_vectors of the three documents : [(id_word, tf-idf) , (id_word, tf-idf) ,...,(id_word, tf-idf)]
for doc_vector in corpus_tfidf_vectors:
    print(doc_vector)
```

```
tfidf_vectors of the three documents : [(id_word, tf-idf) , (id_word, tf-idf) ,...,(id_word, tf-idf)]:
[(0, 0.5849625007211562), (1, 0.5849625007211562), (2, 0.5849625007211562)]
[(0, 0.5849625007211562), (2, 0.5849625007211562), (3, 1.5849625007211563)]
[(1, 0.5849625007211562), (4, 1.5849625007211563), (5, 1.5849625007211563)]
```

In [6]:
```
# query
query="new new times"
query_bow_vector = dictionary.doc2bow(query.lower().split())
print(query_bow_vector)
```

```
[(0, 2), (1, 1)]
```

In [7]:
```
# Calculate (compute) TF-IDF vector of the query
query_tfidf_vector = tfidf_model[query_bow_vector] # Apply model
print(query_tfidf_vector)
```

```
[(0, 1.1699250014423124), (1, 0.5849625007211562)]
```

In [8]:
```
# Compute the cosine similarity between the query and the 3 documents
index_matrix = similarities.SparseMatrixSimilarity(corpus_tfidf_vectors,num_features=6)
sims = index_matrix[query_tfidf_vector]
print(list(enumerate(sims)))
```

```
[(0, 0.7745967), (1, 0.2926428), (2, 0.112928025)]
```