3. Suppose that after receiving the results of a query Q0 = "dog race", a user has provided relevance feedback by rating

the following 3 document as non-relevant:

DOC1: "greyhound race track betting"
DOC2: "dog race betting"
DOC3: "greyhound dog training"

and the following 4documents as relevant:

DOC4: "iditarod dog sled race"
DOC5: "husky dog sled race malamute dog sled"
DOC6: "betting alaska dog sled race"
DOC7: "dog race alaska iditarod"

Assuming simple term frequency weights, use Rocchio's relevance feedback method to compute a new query Q1 (use a positive feedback factor of 1.0 and negative feedback factor of 0.5). Show Q1 as a vector over the above index terms with the corresponding weights generated by Rocchio. Explain any significant increase or decrease in term weights. Show your work.

In [1]:
```
# from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
docs = ["greyhound race track betting","dog race betting","greyhound dog training","iditarod dog sled race",
"husky dog sled race malamute dog sled","betting alaska dog sled race","dog race alaska iditarod"]

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(docs)

print()
print(vectorizer.get_feature_names())
print(X.toarray())

query_transform = vectorizer.transform(['dog race'])
print()
print(query_transform.toarray())
```

```
['alaska', 'betting', 'dog', 'greyhound', 'husky', 'iditarod', 'malamute', 'race', 'sled', 'track', 'training']
[[0 1 0 1 0 0 0 1 0 1 0]
 [0 1 1 0 0 0 0 1 0 0 0]
 [0 0 1 1 0 0 0 0 0 0 1]
 [0 0 1 0 0 1 0 1 1 0 0]
 [0 0 2 0 1 0 1 1 2 0 0]
 [1 1 1 0 0 0 0 1 1 0 0]
 [1 0 1 0 0 1 0 1 0 0 0]]

[[0 0 1 0 0 0 0 1 0 0 0]]
```

In [3]:
```
import pandas as pd
df = pd.DataFrame(X.toarray(),columns=vectorizer.get_feature_names())
q_df=pd.DataFrame(query_transform.toarray(),columns=vectorizer.get_feature_names(),index=['query'])

rel=pd.DataFrame([0,0,0,1,1,1,1],columns=['relavance']) # Append the relevance indicator column
cc=pd.concat([df, q_df], axis=0, sort=False)

all_results=pd.concat([cc, rel], axis=1)[['dog','race','alaska', 'betting', 'greyhound', 'husky', 'iditarod', 'malamut
e', 'sled', 'track', 'training','relavance']]
all_results
```

Out[3]:

	dog	race	alaska	betting	greyhound	husky	iditarod	malamute	sled	track	training	relavance
0	0	1	0	1	1	0	0	0	0	1	0	0.0
1	1	1	0	1	0	0	0	0	0	0	0	0.0
2	1	0	0	0	1	0	0	0	0	0	1	0.0
3	1	1	0	0	0	0	1	0	1	0	0	1.0
4	2	1	0	0	0	1	0	1	2	0	0	1.0
5	1	1	1	1	0	0	0	0	1	0	0	1.0
6	1	1	1	0	0	0	1	0	0	0	0	1.0
query	1	1	0	0	0	0	0	0	0	0	0	NaN

In [4]:
```
# non-relavant
nr=all_results[all_results['relavance']==0].drop('relavance', axis=1).sum()/3
print('NON-Relavant')
print(nr)
print()
# Relavant
print('Relavant')
r=all_results[all_results['relavance']==1].drop('relavance', axis=1).sum()/4
print(r)
```

```
NON-Relavant
dog          0.666667
race         0.666667
alaska       0.000000
betting      0.666667
greyhound    0.666667
husky        0.000000
iditarod     0.000000
malamute     0.000000
sled         0.000000
track        0.333333
training     0.333333
dtype: float64

Relavant
dog          1.25
race         1.00
alaska       0.50
betting      0.25
greyhound    0.00
husky        0.25
iditarod     0.50
malamute     0.25
sled         1.00
track        0.00
training     0.00
dtype: float64
```

In [5]:
```
# Q0 + beta*rel - gamma*nrel
q1=all_results.loc['query'][:-1]+(1*r)-(0.5*nr)
q1[q1<0] = 0 # negative to zero
q1
```

Out[5]:
```
dog          1.916667
race         1.666667
alaska       0.500000
betting      0.000000
greyhound    0.000000
husky        0.250000
iditarod     0.500000
malamute     0.250000
sled         1.000000
track        0.000000
training     0.000000
dtype: float64
```