In this assignment, we will measure the cosine similarity between two given sentences using numpy. Let us assume the two sentences are:

In [2]:

```
A = "I love data mining"

B = "I hate data mining"
```

First thing we need to do is to import numpy

In [3]:

```
import numpy
```

We will get the list of words in each sentence by calling the split() method in strings. lower() will convert the sentence into lowercase.

In [4]:

```
wordsA = A.lower().split()

wordsB = B.lower().split()
```

In [5]:

```
print wordsA

print wordsB
```

```
['i', 'love', 'data', 'mining']
['i', 'hate', 'data', 'mining']
```

Before we represent each sentence using a numpy array, we must know the dimensionality of the feature space (i.e. the total number of unique features in) all instances. We can use the set class for this purpose.

In [6]:

```
vocab = set(wordsA)

vocab = vocab.union(set(wordsB))
```

Lets print all the features in the vocabulary vocab.

In [7]:

```
print vocab
```

```
set(['mining', 'love', 'i', 'hate', 'data'])
```

We see that vocab contains all the features in all sentences. However, we need to have a one-to-one mapping between features and their ids in the feature space. Each feature must be associated with a single dimension in the feature space. This can be done easily by converting the vocab into a list. In a list we have a one-to-one mapping between each element and its position in the list, starting from 0.

In [8]:

```
vocab = list(vocab)
```

You can see the list of unique features as follows:

In [9]:

```
print vocab
```

```
['mining', 'love', 'i', 'hate', 'data']
```

Now lets declare two 1D-arrays (which will act as feature vectors) for representing the two sentences.

In [10]:

```
vA = numpy.zeros(len(vocab), dtype=float)

vB = numpy.zeros(len(vocab), dtype=float)
```

numpy.zeros(shape, data_type) returns arrays of the specified shape and size, filled with zeros. In our case we need 1D arrays filled with float type zeros. dtype is a keyword indicating the data type to store in the array. Unlike python lists where we could store any data type, with numpy.arrays we can store only values for one data type. This makes numpy.arrays efficient in terms of both speed and memory.

We will now go through the list of features for the first sentence and increment the corresponding feature value in the vector.

In [11]:

```
for w in wordsA:

    i = vocab.index(w)

    vA[i] += 1
```

Lets print this vector.

In [12]:

```
print vA
```

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

Take a moment to check this result. The first feature in the vocabulary is "mining". This features occurrs in the first sentence. Therefore, its value is set to 1. Go through all the features in the first sentence and confirm that this is true.

We can do the same procedure to populate the vector for the second sentence as follows:

In [13]:

```
for w in wordsB:

    i = vocab.index(w)

    vB[i] += 1
```

In [14]:

```
print vB
```

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

Again check that the vector is correctly populated for the second sentence.

Next, we will compute the cosine similarity between the two vectors. The cosine similarity between two vectors x and y is defined as follows:
cos(x,y) = numpy.dot(x,y) / (numpy.sqrt(numpy.dot(x,x)) * numpy.sqrt(numpy.dot(y,y)))

In [16]:

```
cos = numpy.dot(vA, vB) / (numpy.sqrt(numpy.dot(vA,vA)) * numpy.sqrt(nu
mpy.dot(vB,vB)))
```

In [17]:

```
print cos
```

```
0.75
```

Therefore, the cosine similarity between the two sentences is 0.75. Here, numpy.dot computes the inner-product between two vectors, and numpy.sqrt computes the square root.
Of course, this is not the only way to compute cosine similarity. Numpy provides the linear algebra routines such as norm of a vector in the numpy.linalg package. If you use this, you do not need to call the numpy.sqrt. For example,

In [18]:

```
print numpy.dot(vA, vB) / (numpy.linalg.norm(vA) * numpy.linalg.norm(vB
))
```

```
0.75
```

As you can see you get the same result but the code looks better.