CODE:

```
import textmining
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
import numpy as np

def listToString(s):

   str1 = " "
   return (str1.join(s))

def lemmatize(doc):
        lemmatizer = WordNetLemmatizer()
        ps = PorterStemmer()

        stop_words = set(stopwords.words('english'))

        word_tokens = word_tokenize(doc)

        filtered_sentence = [w for w in word_tokens if not w in stop_words]

        filtered_sentence = []

        for w in word_tokens:
                if w not in stop_words:
                        filtered_sentence.append(lemmatizer.lemmatize(w))

        return listToString(filtered_sentence)

def termdocumentmatrix_example():
   doc1 = 'Natural language processing is becoming important since soon we will begin talking
to our computers'
   doc2 = 'If computers understand natural language they will become much simpler to use'
   doc3 = 'Speech recognition is the first step to build computers like us'
   q1   = 'natural language processing for computer'
   q2   = 'speech recognition'

   tdm = textmining.TermDocumentMatrix()
   # print lemmatize(doc1)
   tdm.add_doc(lemmatize(doc1))
```

```
    tdm.add_doc(lemmatize(doc2))
    tdm.add_doc(lemmatize(doc3))
    tdm.add_doc(lemmatize(q1))
    tdm.add_doc(lemmatize(q2))

    tdm.write_csv('matrix.csv', cutoff=1)
    lines = tdm.rows(cutoff=1)
    # lines = lines[1:]
    # print lines
    temp = []
    for row in lines:
            temp.append(row)
    temp = temp[1:]

    return temp[:3], temp[3:]

doctdm, querytdm = termdocumentmatrix_example()

doctdm = np.matrix(doctdm)
querytdm = np.matrix(querytdm)
print "document tdm"
print doctdm
print "query tdm"
print querytdm
score = doctdm.dot(np.transpose(querytdm))
print "score"
print score
```

OUTPUT:

```
document tdm
[[1 1 1 0 0 0 1 0 0 0 1 1 1 0 1 0 0 1 0 1 0 0 0]
 [0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 1 1 0 1 0 1 0]
 [0 0 1 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 0 1]]
query tdm
[[0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0]
 [0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
score
[[4 0]
 [3 0]
 [1 2]]
ranks for query1 doc1,doc2,doc3
ranks for query2 doc3,doc1,doc2
```

Note: due to poor net connectivity in my home town, I am unable to download the test data from google drive. So I consider my own small dataset. I heartly request you to please consider it.