Job Salary Text Mining and Classification:

In this analysis, 5000 real job descriptions are provided. Using these job descriptions and the normalized salary of these job, a text classifier was built to classify these jobs into “high” salary group and “low” salary group.

The first step is to preprocess the job description.

1. Tokenize each job description: divide a sentence into separated words

2. Remove punctuations and stop words.

3. Lemmatization: usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

The following code shows how to tokenize, remove stop words and lemmatize the text.

```
##Salary prediction#####
import nltk
import pandas as pd
import os
import random
from nltk.corpus import stopwords
stop = stopwords.words("english")
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()


train=pd.read_csv("Train_rev1.csv")
job_des=train.ix[:,2]

#tokenize and preprocess the first 5000 posts
text=list()
for job in job_des[:5000]:
    temp = nltk.word_tokenize(job.strip().lower().translate(None,
"\\!#$%&()*+,-./:;<=>?@[\]^_`{|}~"))
    #remove stopwords in job description
    temp2 = [w for w in temp if not w in stop]
    #lemmatize the word
    temp3=list()
    for w in temp2:
        w=repr(w).replace("\\","")
        temp3.append(wordnet_lemmatizer.lemmatize(w))
    text.append(temp3)
text_series=pd.Series(text)
```

After these three steps, we will have the pure text we need for each job to conduct classification analysis. Before building classification model, each job description will be labelled as high salary job or low salary job. If the salary is higher than the 75% quantile salary, it’s considered as high salary. The following code will label each job post.

```
#salary for each post
salaries = train.ix[:4999,10]
salaries_sort=sorted(salaries)
```

```
percentile = .75*len(salaries_sort)
seven_five_percentile = salaries_sort[int(percentile)]
salary_class=["low"]*5000
for i in range(len(salary_class)):
    if salaries[i]>=seven_five_percentile:
        salary_class[i]="high"
```

Then we can do some preparation work for the classification model. In this analysis, only the 5000 most frequent words will be used.

```
#pick out the most frequent 5000 words
job_tuple=[]
for i in range(len(text_series)):
  job_tuple.append((text_series[i],salary_class[i]))
random.shuffle(job_tuple)

all_text=[]
for text in text_series:
    all_text.extend(text)


all_words = nltk.FreqDist(w.lower() for w in all_text )
word_features = list(all_words)[:5000]

#create a function to decide whether a word is in featured words or not
def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features
```

Then we can split all job description into two group. One training group will be used to train the model and a test group will be used to test the accuracy of the classification model. The result shows that the accuracy rate of the model is 79.4% and the confusion matrix is shown below. From the confusion matrix we can calculate the accuracy for both low salary and high salary. We can see 86% time, low salary are correctly classified and the model is better at predicting low salary than high salary.

Confusion Matrix

		Test		
		Low	High	
reference	Low	64.90%	10.50%	75.40%
	High	10.10%	14.50%	24.60%
		75.00%	25.00%	1

Accuracy Rate of Each Group

	Correct	Wrong
Low	86.07%	13.93%
High	58.94%	41.06%

The following code shows how to build the classification model and generate confusion matrix:

```
featuresets = [(document_features(d), c) for (d,c) in job_tuple]
train_set, test_set = featuresets[1500:], featuresets[:1500]
classifier = nltk.NaiveBayesClassifier.train(train_set)

#accuracy of model
print(nltk.classify.accuracy(classifier, test_set))

#show most imformative words
classifier.show_most_informative_features(100)

#Confusion matrix setup
testoutput=[]
for (jobdesc,label) in test_set:
  a = classifier.classify(jobdesc)
  testoutput.append(a)
actual=[]
for (jobdesc,label) in test_set:
  actual.append(label)

#Confusion matrix
cm = nltk.ConfusionMatrix(actual, testoutput)
print(cm.pp(sort_by_count=True, show_percents=True, truncate=2))
```

From the above code, we can also know the 50 most informative words in the job description. Informative is measure by the ratio of the number of times a word appear in a salary group posts to the number of time it appear in the other salary group posts. From the result, we can see words like “prescribing”, “logic”, “architectural”, “provisioning” are more likely to appear in a high salary job description.

Most Informative Features

contains('prescribing') = True	high : low	=	24.2 : 1.0
contains('ic') = True	high : low	=	20.3 : 1.0
contains('hcpc') = True	high : low	=	18.4 : 1.0
contains('logic') = True	high : low	=	16.4 : 1.0
contains('architectural') = True	high : low	=	16.4 : 1.0
contains('adoption') = True	high : low	=	15.7 : 1.0
contains('sc') = True	high : low	=	15.7 : 1.0
contains('insurer') = True	high : low	=	14.5 : 1.0
contains('langford') = True	high : low	=	14.5 : 1.0
contains('provisioning') = True	high : low	=	14.5 : 1.0
contains('salaried') = True	high : low	=	14.5 : 1.0

contains('round') = True	low : high = 14.1 : 1.0
contains('locum') = True	high : low = 13.0 : 1.0
contains('plsql') = True	high : low = 12.6 : 1.0
contains('julia') = True	high : low = 12.6 : 1.0
contains('customisation') = True	high : low = 12.6 : 1.0
contains('refining') = True	high : low = 12.6 : 1.0
contains('milestones') = True	high : low = 12.6 : 1.0
contains('xml') = True	high : low = 12.3 : 1.0
contains('costing') = True	high : low = 12.2 : 1.0
contains('12mths') = True	high : low = 12.2 : 1.0
contains('caring') = True	low : high = 11.7 : 1.0
contains('calculations') = True	high : low = 11.3 : 1.0
contains('define') = True	high : low = 10.9 : 1.0
contains('formulating') = True	high : low = 10.6 : 1.0
contains('organizations') = True	high : low = 10.6 : 1.0
contains('align') = True	high : low = 10.6 : 1.0
contains('serving') = True	low : high = 10.0 : 1.0
contains('forecasts') = True	high : low = 9.9 : 1.0
contains('http') = True	high : low = 9.9 : 1.0
contains('roadmap') = True	high : low = 9.9 : 1.0
contains('accountant') = True	high : low = 9.1 : 1.0
contains('manipulation') = True	high : low = 8.7 : 1.0
contains('metallic') = True	high : low = 8.7 : 1.0
contains('2003') = True	high : low = 8.7 : 1.0
contains('mechanisms') = True	high : low = 8.7 : 1.0
contains('phases') = True	high : low = 8.7 : 1.0
contains('periodic') = True	high : low = 8.7 : 1.0
contains('permanency') = True	high : low = 8.7 : 1.0
contains('pipework') = True	high : low = 8.7 : 1.0
contains('concurrent') = True	high : low = 8.7 : 1.0
contains('masters') = True	high : low = 8.7 : 1.0
contains('arising') = True	high : low = 8.7 : 1.0
contains('sole') = True	high : low = 8.7 : 1.0
contains('workflows') = True	high : low = 8.7 : 1.0
contains('rickmansworth') = True	high : low = 8.7 : 1.0
contains('httpwwwcmcconsultingcouk') = True	high : low = 8.7 : 1.0
contains('equivalents') = True	high : low = 8.7 : 1.0
contains('breed') = True	high : low = 8.7 : 1.0
contains('impacts') = True	high : low = 8.7 : 1.0