HW 4: Classification

> Each assignment needs to be completed independently. Never ever copy others' work (even with minor modification, e.g. changing variable names). Anti-Plagiarism software will be used to check all submissions.

In this assignment, we use classification to identify deceptive comments. This assignment needs the following two data files:

- hw4_train.csv: dataset for training
- hw4_test.csv: dataset for testing

Both of them have samples in the following format. The `text` column contains documents and the `label` column gives the sentiment of each document.

label	text
1	when i first checked the hotel's website and r...
1	I had really high hopes for this hotel. The lo...
0	My experiences at the Fairmont Chicago were le...
...	...

In [3]:
```
import pandas as pd
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
import numpy as np
np.set_printoptions(precision=3)
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, auc,precision_recall_curve
from matplotlib import pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
```

In [4]:
```
train = pd.read_csv("hw4_train.csv")
train.head(3)
test = pd.read_csv("hw4_test.csv")
test.head(3)
```

Out[4]:

	label	text
0	1	when i first checked the hotel's website and r...
1	1	I had really high hopes for this hotel. The lo...

	label	text
2	1	Hotel Monaco is simply amazing. I travel quite...

Out[4]:

	label	text
0	1	This is the hotel for the discriminating busin...
1	0	What a wonderful experience, super great, help...
2	0	I stayed at the Talbott twice in the last few ...

Q1 Text Vectorization and Classification

For classification, the first step is to compute the word TF-IDF weights for each document. A few options can be configured as given below.

Define a function `classify(train_docs, train_y, test_docs, test_y, classifier = 'naive bayes', binary=False, ngrams = (1,1), stop_words='english', min_df=1, show_plots=False)` , where

- `train_docs` : is a list of documents for training.
- `train_y` : is the ground-truth labels of training documents.
- `test_docs` : is a list of documents for test.
- `test_y` : is the ground-truth labels of test documents.
- `classifier` : the name of classification algorithm. Two possible values: 'svm','naive bayes'. The default value is 'naive bayes'.
- `binary` : if true, within a document, the term frequency of a word is binarized to 1 if present and 0 otherwise. If False, the regular term frequency is considered. The default is False.
- `ngrams` : an option to include unigrams, bigrams, ..., nth grams. The default is (1,1), i.e., only unigrams used.
- `stop_words` : indicate whether stop words should be removed. The default value is 'english', i.e. remove English stopwords.
- `min_df` : only tokens with document frequency above this threshold can be included. The default is 1.
- `show_plots` : controls whether to show classification report AND plots. The default is False.

This function does the following:

- Fit a `TfidfVectorizer` using `train_docs` with options `stop_words, min_df, ngrams, binary` as specified in the function inputs. Extract features from `train_docs` using the fitted `TfidfVectorizer` .
- Train a classifier by the specified `classifier` algorithm using the extracted features from `train_docs` and labels from `train_y` .
- Transform `test_docs` by the fitted `TfidfVectorizer` (hint: use function `transform` not `fit_transform`).
- Predict the labels for `test_docs` with trained model.

- If `show_plots` is True,
  - Print the classification report.
  - Plot the AUC score and PRC score (or Average Precision) for class 1 on the test dataset. On the plot, specify xlabel, ylabel on axis, and the scoring metrics (AUC/PRC/Average Precision) on the title.
  - Note, if the classifier is 'svm', please use `decision_function` instead of predict_prob function. The details can be found at https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC.decisic
- Return the `TfidfVectorizer` and the trained model.

Test your function with following cases:

- `stop_words = 'english', binary = False, classifier='naive bayes', show_plots = True`
- `stop_words = 'english', binary = False, classifier='svm', show_plots = True`

In [5]:

```
def classify(train_docs, train_y, test_docs, test_y, \
                classifier = 'naive bayes',
                binary=False, ngrams = (1,1), \
                stop_words='english', min_df=1, \
                show_plots=True):

    clf, tfidf_vect = None, None

    #Tfidfvectorizer
    tfidf_vect = TfidfVectorizer(stop_words = stop_words, min_df = min_df, binary = bin
    #Transform
    train_doc_vector = tfidf_vect.fit_transform(train_docs)
    test_doc_vector = tfidf_vect.transform(test_docs)

    if classifier == 'naive bayes':
        clf = MultinomialNB().fit(train_doc_vector, train_y)
        predict_p = clf.predict_proba(test_doc_vector)
        y_pred = predict_p[:,1]

    if classifier == 'svm':
        clf = svm.LinearSVC().fit(train_doc_vector, train_y)
        predict_p = clf.decision_function(test_doc_vector)
        y_pred = predict_p

    binary_y = np.where(test_y == 1,1,0)
    predicted = clf.predict(test_doc_vector)
    print(classification_report(test_y, predicted))

    if show_plots:
        fpr, tpr, thresholds = roc_curve(binary_y, y_pred)
        precision, recall, thresholds = precision_recall_curve(binary_y,y_pred)

        f, ax = plt.subplots(1,2, figsize = (15,5))

        ax[0].set_title('auc')
        ax[0].set(xlabel = 'False Positive Rate',ylabel = 'True Positive Rate')
        ax[0].plot(fpr, tpr, color='darkorange', lw=2)
```

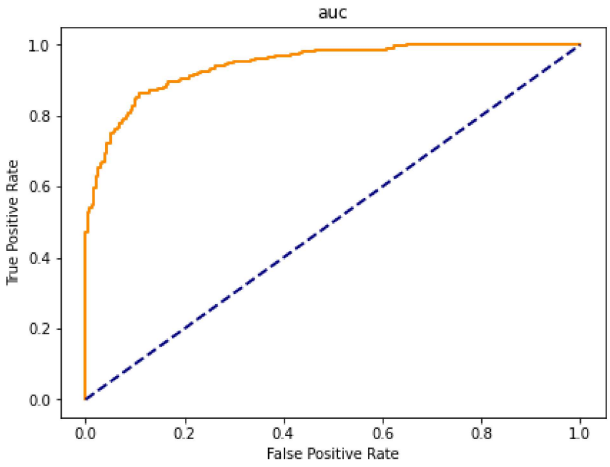
```
        ax[0].plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
        ax[0].set(xlim = [-0.05, 1.05], ylim = [-0.05, 1.05])

        ax[1].set_title('prc');
        ax[1].set(xlabel = 'Recall', ylabel = 'Precision')
        ax[1].plot(recall, precision, color='darkorange', lw=2)
        ax[1].set(xlim = [-0.05, 1.05], ylim = [-0.05, 1.05])

        print("AUC: {:.2%}".format(auc(fpr, tpr)) , ", PRC: {:.2%}".format(auc(recall,

    return  clf, tfidf_vect
```

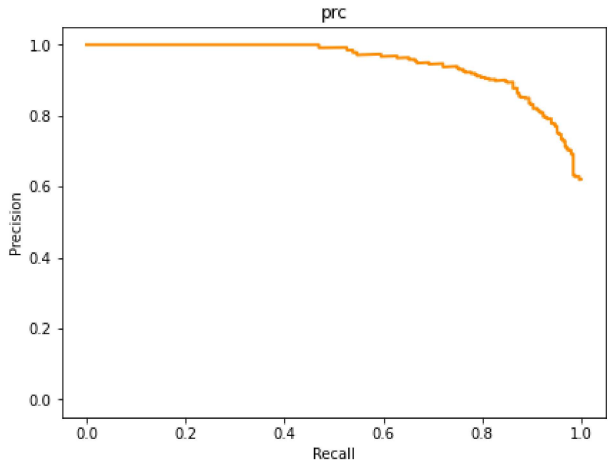
In [6]:

```
clf, vectorizer = classify(train["text"], train["label"],
                           test["text"], test["label"],
                           stop_words = 'english', binary = False,
                           classifier='naive bayes', show_plots = True)
```

	precision	recall	f1-score	support
0	0.88	0.82	0.85	233
1	0.84	0.89	0.86	247
accuracy			0.86	480
macro avg	0.86	0.86	0.86	480
weighted avg	0.86	0.86	0.86	480

AUC: 94.20% , PRC: 94.97%



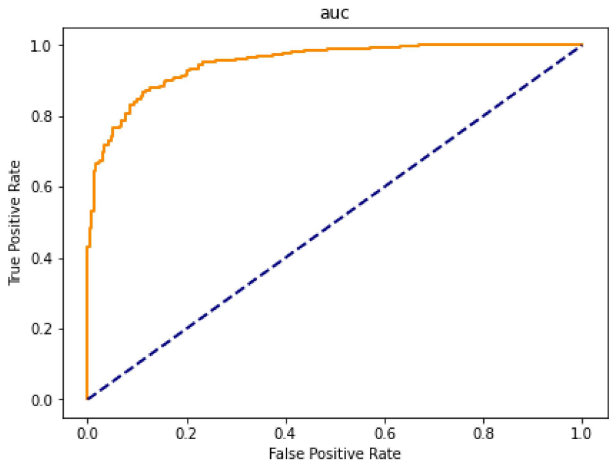


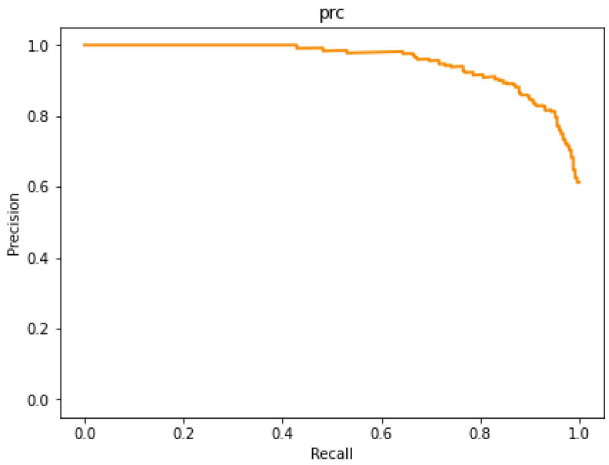
In [7]:

```
clf, vectorizer = classify(train["text"], train["label"],
                           test["text"], test["label"],
                           stop_words = 'english', binary = False,
                           classifier='svm', show_plots = True)
```

	precision	recall	f1-score	support
0	0.87	0.85	0.86	233
1	0.86	0.88	0.87	247
accuracy			0.87	480
macro avg	0.87	0.87	0.87	480
weighted avg	0.87	0.87	0.87	480

AUC: 94.83% , PRC: 95.44%





Q2: Search for best parameters

From Q1, you may find there are many possible ways to configure parameters. Next, let's use grid search to find the optimal parameters.

- Define a function `search_para(docs, y, classifier = 'naive bayes')` where `docs` are training documents, `y` is the ground-truth labels, and `classifier` is the model you use.
- This function does the following:
  - Create a pipleline which integrates `TfidfVectorizer` and the classifier as specified by parameter `classifier` .
  - Define the parameter ranges as follow:
    - `stop_words: [None, 'english']`
    - `min_df:[1, 3]`
    - `ngram_range:[(1,1), (1,2), (1,3)]`
    - `binary: [True, False]`
  - Set the scoring metric to `f1_macro` .
  - Use `GridSearchCV` with `4-fold cross validation` to find the best parameter values based on the training dataset.
  - Print the values of the `best` parameters combination.
- Call this function to find `the best parameters combination` for linear SVM and Naive Bayes models.

- Call the function `classify` again to use `the best parameters combination`

Please briefly answer the following:

- Compared with the model in Q1, how is the performance improved on the test dataset?
- Why do you think the new parameter values help deceptive comment classification?

In [20]:
```
def search_para(docs, y, classifier = 'naive bayes'):

    if classifier == 'svm':

        text_clf = Pipeline([('tfidf', TfidfVectorizer()),('clf', svm.LinearSVC())])
```

```
    if classifier == 'naive bayes':

        text_clf = Pipeline([('tfidf', TfidfVectorizer()),
                             ('clf', MultinomialNB())])

    parameters = {'tfidf__min_df':[1,3],
                  'tfidf__stop_words':[None,"english"],
                  'tfidf__ngram_range':[(1,1), (1,2), (1,3)],
                  'tfidf__binary': [True, False]}

    metric =  "f1_macro"

    gs_clf = GridSearchCV(text_clf, param_grid=parameters, scoring=metric, cv=4)
    gs_clf = gs_clf.fit(docs, y)

    for param_name in gs_clf.best_params_:

        print("{0}:\t{1}".format(param_name,\
                                     gs_clf.best_params_[param_name]))

    print("best f1 score: {:.3f}".format(gs_clf.best_score_))
```

In [21]:
```
search_para(train["text"], train["label"])
```

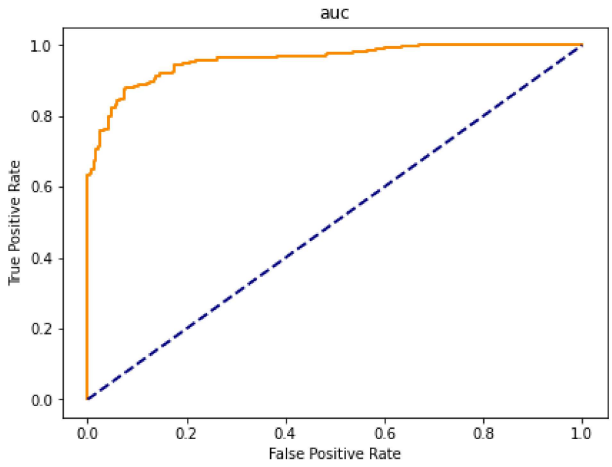
```
tfidf__binary:  True
tfidf__min_df:  3
tfidf__ngram_range:     (1, 2)
tfidf__stop_words:      None
best f1 score: 0.888
```

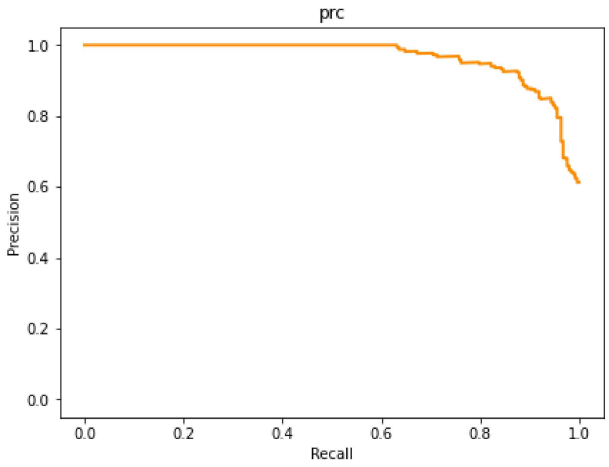
In [22]:
```
search_para(train["text"], train["label"], classifier ='svm')
```

```
tfidf__binary:  True
tfidf__min_df:  1
tfidf__ngram_range:     (1, 2)
tfidf__stop_words:      None
best f1 score: 0.895
```

In [133…
```
clf, vectorizer = classify(train["text"], train["label"],
                           test["text"], test["label"],
                           stop_words= None, min_df = 3, binary=True,
                           ngrams = (1,2), classifier = 'naive bayes', show_plots=Tr
```

```
              precision    recall  f1-score   support

           0       0.91      0.85      0.88       233
           1       0.87      0.92      0.89       247

    accuracy                           0.89       480
   macro avg       0.89      0.89      0.89       480
weighted avg       0.89      0.89      0.89       480

AUC: 95.70% , PRC: 96.52%
```

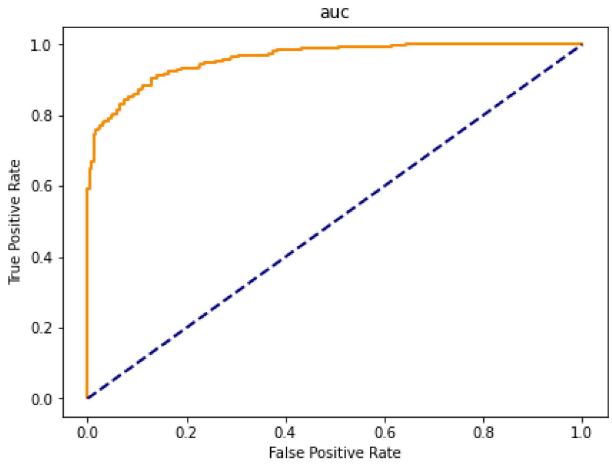


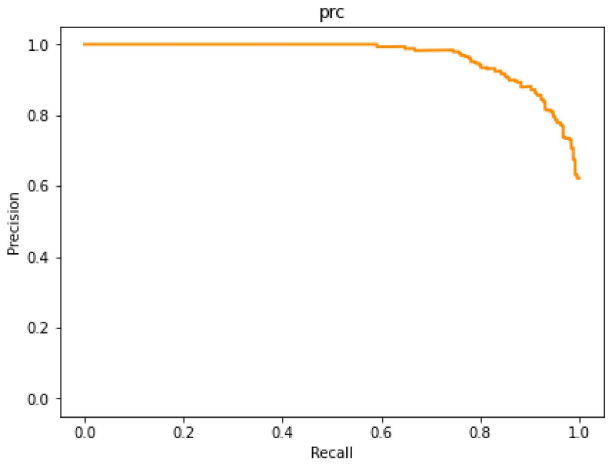


In [134…

```
clf, vectorizer = classify(train["text"], train["label"],
                           test["text"], test["label"],
                           stop_words= 'english', min_df = 2, binary=True,
                           ngrams = (1,3), classifier = 'svm', show_plots=True)
```

```
              precision    recall  f1-score   support

           0       0.90      0.86      0.88       233
           1       0.87      0.91      0.89       247

    accuracy                           0.88       480
   macro avg       0.88      0.88      0.88       480
weighted avg       0.88      0.88      0.88       480

AUC: 95.85% , PRC: 96.55%
```





In Q1, we get 86% and 87% in Naive Bayes and SVM classification. After we tune the parameters, we increased the value to 89% and 88%. I think the reason new parameters help deceptive comment it because some deceptive are bigrams term and we haven't count them in Q1. Also, we kept the stop words because there are some negation words we should kept them in our model. If we remove the stop words it will lose some information.

Q3. Impact of Sample Size

This task is to help you understand the impact of sample size on classifier performance.

Define a function `show_sample_size_impact(train_docs, train_y)` where:

- `train_docs` : is a list of documents for training.
- `train_y` : is the ground-truth labels of training documents.

Conduct the experiment as follows:

- Starting with 100 samples, in each round you build a classifier with 100 more samples. i.e. in round 1, you use samples from 0 to 100, and in round 2, you use samples from 0 to 200, ..., until you use up all samples.
- In each round, you'll conduct `4-fold cross validation` for both Naive Bayes and Linear SVM algorithms with all `default paramater values` in the TFIDF vectorizer and classifiers. Record the average testing F1-macro score.
- Plot a line chart to show the relationship between sample size and the F1-macro score for SVM and Naive Bayes models.
- This function has no return.

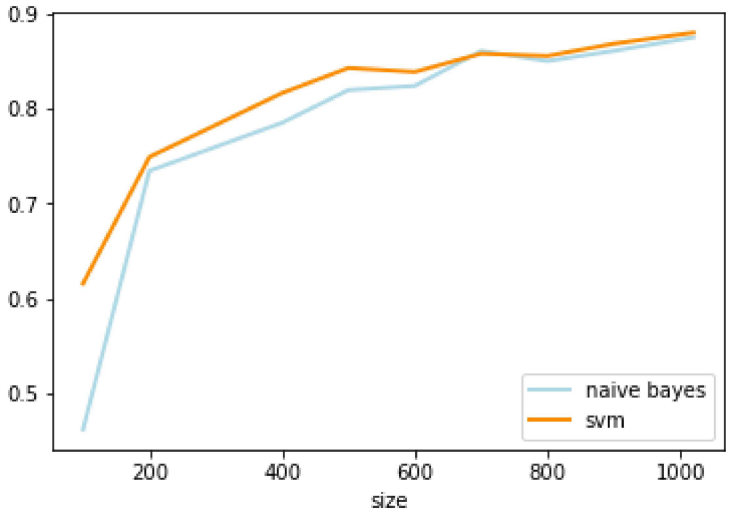
- Write your analysis on the following:
  - How does sample size affect each classifier's performance?
  - If it is expensive to collect and label samples, can you decide an optimal sample size with model performance and the cost of samples both considered?

In [69]:

```
def show_sample_size_impact(train_docs, train_labels):


    clf_NB = Pipeline([('tfidf', TfidfVectorizer()),
                               ('clf', MultinomialNB())])
    clf_svm = Pipeline([('tfidf', TfidfVectorizer()),
                               ('clf', svm.LinearSVC())])
    metric = "f1_macro"

    parameters = {} #default paramater values

    gs_clf_NB= GridSearchCV(clf_NB,param_grid = parameters,scoring=metric, cv = 4)
    gs_clf_svm= GridSearchCV(clf_svm,param_grid = parameters,scoring=metric, cv = 4)

    seq = list([100,200,400,500,600,700,800,900,1020])

    list_NB,list_svm = [],[]

    for i in seq:

        temp_docs = train_docs[0:i]
        temp_labels = train_labels[0:i]

        gs_clf_NB = gs_clf_NB.fit(temp_docs, temp_labels)
        gs_clf_svm = gs_clf_svm.fit(temp_docs, temp_labels)

        list_NB.append(gs_clf_NB.best_score_)
        list_svm.append(gs_clf_svm.best_score_)

    plt.plot(seq, list_NB, color='lightblue', lw=2, label = "naive bayes")
    plt.plot(seq, list_svm, color='darkorange', lw=2, label ='svm')
```

```
    plt.xlabel("size")
    plt.legend(loc = 4)
```

In [70]:
```
show_sample_size_impact(train["text"], train["label"])
```



The model with larger sample size has better performance.

I would choose sample size 500 as the optimal sample size. Since it only cost half of total expense and it reached to 85% f1-macro score.

Q4 (Bonus): Model Interpretation

For this dataset, both Naive Bayes and SVM model can provide good classification accuracy. The question is, how the models conclude that a document is deceptive. What features have the discriminative power? Do your research to find the most descriminative features in a document. To illustrate, you can randomly select a few samples from the test subset to highlight these most discriminative featues.

In [83]:
```
test['text'][479] # label 1
test['label'][479]

test['text'][227] # label 1
test['label'][227]

test['text'][239] # label 0
test['label'][239]
```

Out[83]:
```
"My stay at the Fairmont Chicago Millennium Hotel was a short one; although, it was inte
nded to be for five days, it only last two. When me and my husband arrived we were met b
y a valet who knicked our Mercedez. The clerk/concierge was on his phone for nearly 5 FU
LL minutes before he noticed that we were standing in front of him. When we checked in,
we lugged all of our luggage upstairs to our room to find that someone was already in i
t. We then returned to Lobby, and to our dismay, the hotel had Double Booked the room! A
fter another 30 minutes of haggling and whatnot we were able to be moved to a 'better' r
oom. The bed was unbearable, the shower looked like it hadnt been cleaned properly and t
he room smelled of cigarrettes (although it was supposedly a non-smoking room). Too tire
d and jet lagged to complain we went to sleep. However, it was hard for us to do so beca
```

```
use voices could be heard all thoughout the night due to the paper thin walls of the roo
m. I needed an Ambien to go to sleep. I woke up with backaches, the continental breakfas
t had me in the bathroom most of the day, and the maids conveniently forgot to clean our
room. Me and my husband had had enough and decided to go to the Hyatt.\n"
```

Out[83]: 1

Out[83]:
```
'This is one of the best hotels I have ever been in. I went there to see about schedulin
g my reception party, and was amazed at the architecture of the building, the interior d
esign of the rooms, and the massive spacing provided for each room. Now I recommend this
hotel to my associates to hold their important business meetings in the James Club, so t
hat after the meetings, we can partake in a nice meal and drinks. Which brings me to my
next point. The James has some of the BEST food I have ever eaten!!! The James Private C
lub, I feel was made for the elite and upper class. Very classy, very elegant. The custo
mer service is outstanding. I recommend this hotel to anyone needing to hold a receptio
n, business meeting, or just need to have a day at the spa. 5 stars all the way!!! \n'
```

Out[83]: 1

Out[83]:
```
"We stayed at Amalfi for 2 nights and i have to say after reading all the reviews here i
expected a little bit more. It's a fine place with clean bathrooms and comfy beds but i
t's just not worth the money. It does have comfy beds and free continental breakfast. Th
e sheets though, made us itch for some reason and we are normally not sensitive to these
things. The breakfast could really benefit from some bagels or yougurt. Otherwise, it's
just an early morning sugar attach of muffins, danishes, sweet breads, and croissants. T
here are plenty of restaurants around but they are all of the big chain variety, if you
are into that sort of thing. Also, the windows do not open and there is a climate contro
l device in the room. The problem with it that it keeps blowing air all the time, altern
ating between cold and warm, right over the bed. There is a sign not to turn it off but
we did once just to see what happens and the air in the room got uncomfortably stale. Th
ey do have a business center where you can send fax and use internet for free. The probl
em is it is only one tiny room with one computer in there. One person at a time... All a
nd all, we will not stay there again. If we feel like luxury, we have to find some other
place. But if we feel like a regular hotel in the heart of the city, maybe we will try R
ed Roof Inn or something like that. At least we'll know what we are paying for.\n"
```

Out[83]: 0

In [ ]: