In [2]:

```
import keras
import numpy as np
import os
from pathlib import Path
from keras import layers
import random
import sys
import pandas as pd

current_dir = Path(os.getcwd()).absolute()
results_dir = current_dir.joinpath('results')

results_dir.mkdir(parents=True, exist_ok=True)

output_dir = current_dir.joinpath('output')
output_dir.mkdir(parents = True, exist_ok = True)

corpus = current_dir.joinpath('hounds.txt')

text = open(corpus, encoding="utf8").read().lower()
print('Corpus length:', len(text))
```

```
Using TensorFlow backend.

Corpus length: 373569
```

In [3]:
```
maxlen = 60

step = 3

sentences = []
next_chars = []

for i in range(0, len(text) - maxlen, step):
    sentences.append(text[i: i + maxlen])
    next_chars.append(text[i + maxlen])

print('Number of sequences:', len(sentences))
chars = sorted(list(set(text)))
print('Unique characters:', len(chars))
char_indices = dict((char, chars.index(char)) for char in chars)

print('Vectorization...')

x = np.zeros((len(sentences), maxlen, len(chars)), dtype = np.bool)
y = np.zeros((len(sentences), len(chars)), dtype = np.bool)
for i, sentence in enumerate(sentences):
    for t, char in enumerate(sentence):
        x[i, t, char_indices[char]] = 1
    y[i, char_indices[next_chars[i]]] = 1
```

```
Number of sequences: 124503
Unique characters: 70
Vectorization...
```

In [4]:
```
model = keras.models.Sequential()
model.add(layers.LSTM(128, input_shape = (maxlen, len(chars))))
model.add(layers.Dense(len(chars), activation = 'softmax'))
```

In [5]:
```
optimizer = keras.optimizers.RMSprop(lr = 0.01)
model.compile(loss = 'categorical_crossentropy', optimizer = optimizer)
```

In [6]:

```
def sample(preds, temperature = 1.0):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

In [7]:
```
for epoch in range(1, 61):
        print('epoch', epoch)
        model.fit(x, y, batch_size = 128, epochs = 1)
        start_index = random.randint(0, len(text) - maxlen - 1)
        generated_text = text[start_index: start_index + maxlen]
        start_text = generated_text
        output_text = 'Initial seed: ' + generated_text
        print('--- Generating with seed: "' + generated_text + '"')

        for temperature in [0.2, 0.5, 1.0, 1.2]:
            print('------ temperature:', temperature)
            output_text += '\ntemperature: ' + str(temperature) + '\n' + start_text
            sys.stdout.write(start_text)
            generated_text = start_text

            for i in range(400):
                sampled = np.zeros((1, maxlen, len(chars)))
                for t, char in enumerate(generated_text):
                    sampled[0, t, char_indices[char]] = 1.

                preds = model.predict(sampled, verbose = 0)[0]
                next_index = sample(preds, temperature)
                next_char = chars[next_index]

                generated_text += next_char
                output_text += next_char
                generated_text = generated_text[1:]

                sys.stdout.write(next_char)
        output_path = results_dir.joinpath('training.' + str(epoch) + '.txt')
        output_file = open(output_path, "w")
        output_file.write(output_text)
        output_file.close()
```

```
epoch 1
Epoch 1/1
124503/124503 [==============================] - 92s 741us/step - loss: 1.9240
--- Generating with seed: "      on the moor. he proved to be a blackguard and deserted"
------ temperature: 0.2
      on the moor. he proved to be a blackguard and deserted the a
      that it was the court of the courter that it was a proach of the
      been the courter the coure the coure the coure that it was a coure
      that it was been the came the coure the coure the propect of the
```

stapleter that it was my coure the coure the coured the coure
the courter the came the case a cour the case the coure of the
been the coure of the coure that it------ temperature: 0.5
on the moor. he proved to be a blackguard and deserted the
been liventer had not a meing he up the mass of a rear it which
expace complece to the was his the care his the mare the came
that it for a leans for the gooo” he in the complicale of the
of the race heard it a cloom had here by the dary the case of the
this my that the moor. by a read and he the was a been you have
it was you say peen be a course in my pap------ temperature: 1.0

In [8]:

```
column_names = ["temp", "text"]

dataset = pd.DataFrame(columns = column_names)

for attempt in range(1,1001):
    start_index = random.randint(1560, len(text) - maxlen - 18799)
    generated_text = text[start_index: start_index + 400]
    df2 = {'temp': -1, 'text': generated_text}
    #print(df2)
    dataset = dataset.append(df2, ignore_index = True)

#print("Here's the dataset:\n", dataset)
for attempt in range(1, 1000):
        print('attempt', attempt)
        start_index = random.randint(1560, len(text) - maxlen - 18799)
        generated_text = text[start_index: start_index + maxlen]
        start_text = generated_text
        output_text = 'Initial seed: ' + generated_text
        print('--- Generating with seed: "' + generated_text + '"')

        for temperature in [0.1, .2, .25, .3, .4, .5, .6, .7, .75, .8, .9, 1, 1.1, 1.2, 1.25]:
            print('------ temperature:', temperature)
            output_text += '\ntemperature: ' + str(temperature) + '\n' + start_text
            sys.stdout.write(start_text)
            generated_text = start_text
            data_text = ""
            for i in range(400):
                sampled = np.zeros((1, maxlen, len(chars)))
                for t, char in enumerate(generated_text):
                    sampled[0, t, char_indices[char]] = 1.

                preds = model.predict(sampled, verbose = 0)[0]
                next_index = sample(preds, temperature)
                next_char = chars[next_index]

                generated_text += next_char
                output_text += next_char
                generated_text = generated_text[1:]
                data_text += next_char

                #sys.stdout.write(next_char)
            df2 = {'temp': temperature, 'text': data_text}
```

```
        #   print("\nHere is the output:\n")
            print(data_text)
            #print("\nOutput end.\n")
            dataset = dataset.append(df2, ignore_index = True)
        output_path = output_dir.joinpath('output.' + str(attempt) + '.txt')
        output_file = open(output_path, "w")
        output_file.write(output_text)
        output_file.close()
```

```
------ temperature: 0.8
e a conductor of light. some people without possessing geniustquess
      on the moor.

      “followed the other swoppes. i could have not be a placed and
      hearted in the project gutenberg-tm to tugg-pack linist goin and
            actopy of the prisate ofe, and there came that the ruiftly too
      elded there, and to was knowledger for a small upon no house
      between with a composs of paid on the darkn to me. he may
      as them, drance the m

------ temperature: 0.9
e a conductor of light. some people without possessing genius cleasal
      evidence of the hutuated inflitue. a gever to have the wife has
      opit perfumbers and down fathilute list of his escaped blight
      and of sure. it was a silent, but continguring, in little donate the
            detection, i could have found him bitterly my who doing such a syone
      that on distertainision as asay there was the stane.. we have
      cussed to go a twire o
```

In [9]:
```
dataset
```

Out[9]:

	temp	text
0	-1	ide porter will send for the hall porter,\n ...
1	-1	ge that i am making up\n for lost time, a...
2	-1	what about the convict on the moor?"\n\n...
3	-1	ad given to me. "a\n warder, no doubt," s...
4	-1	together in the face of what must have been a...
5	-1	heavy, solid\n person, very limited, int...
6	-1	"hardly that."\n\n "well, it cannot be...
7	-1	g you some information now, in return for all ...
8	-1	been at your club all day, i\n perceive."...
9	-1	d sufficient roof to act as a\n screen ag...
10	-1	\n only this one letter, so i took the mo...

In [13]:
```
generated_text
```

Out[13]:
```
'iend dr. mortimer has\n      see that there was a states and '
```

In [10]:
```
dataset.to_excel("output.xlsx")
```

In [11]:
```
dataset
```

Out[11]:

	temp	text
0	-1	ide porter will send for the hall porter,\n ...
1	-1	ge that i am making up\n for lost time, a...
2	-1	what about the convict on the moor?”\n\n...
3	-1	ad given to me. “a\n warder, no doubt,” s...
4	-1	together in the face of what must have been a...
5	-1	heavy, solid\n person, very limited, int...
6	-1	“hardly that.”\n\n “well, it cannot be...
7	-1	g you some information now, in return for all ...
8	-1	been at your club all day, i\n perceive.”...
9	-1	d sufficient roof to act as a\n screen ag...
10	-1	\n only this one letter, so i took the mo...
11	-1	our\n own prisoner. such are the adventur...
12	-1	on the table was blurred by it. as i entered,\...
13	-1	a dream.”\n\n “i heard it distinctly, and...
14	-1	s head in strong dissent. i stood among the ro...
15	-1	’s well worth troubling about.”\n\n “why,...
16	-1	has brought my narrative up to the eighteenth...
17	-1	o seize it, and had we not been there to drag ...
18	-1	et broad on either\n side.”\n\n “i u...
19	-1	les away, over yonder, i think.”\n\n “it ...
20	-1	made your fresh arrangements, but you will\n ...
21	-1	-hall, his bedroom candle in his hand, and he ...
22	-1	y fallen over here and broken his neck.”\n\n ...
23	-1	the direct accessory to murder. she was ready...
24	-1	given my reasons for not\n wishing to do...

	temp	text
25	-1	aid nothing?”\n\n “what was the use?”\n\n...
26	-1	life. if\n she had to leave him he had ra...
27	-1	nearer to our supreme adventure.\n\n ...
28	-1	and he wrung his hands\n together like on...
29	-1	living unannounced under another name so clos...
...	...	...
15955	0.1	i have not alon\n stapleton was all the ...
15956	0.2	he was not along.\n his face of the word...
15957	0.25	he was not the\n disappoor to me, ...
15958	0.3	i have stapleton\n the stapletons.”\n\n ...
15959	0.4	the farther of\n the figual on the moor ...
15960	0.5	where the probable of the\n ...
15961	0.6	it is very intees\n of the lond purpose ...
15962	0.7	”\n\n “i rememberled him, which suppress ...
15963	0.75	”\n\n “it was the let of short was laugh ...
15964	0.8	you see him camewable\n foot th...
15965	0.9	we shall the folked out him that the baronet\...
15966	1	”\n\n “i am long and sicl of the little. ...
15967	1.1	to mus the moor\n of him for i hou...
15968	1.2	”\n\n holmes was a tally verledge frankla...
15969	1.25	another vidded but\n falling a recessed....
15970	0.1	y own straight of the hound was a death, and t...
15971	0.2	y own straight of the project gutenberg litera...
15972	0.25	y own street the stood and have been the old\n...
15973	0.3	y other since i had not her alient to me, and\...
15974	0.4	y own fellow is the door of the point of the\n...
15975	0.5	y own struck, and the point of the other state...

	temp	text
15976	0.6	y own death of the wide," said he. “i asked.\n...
15977	0.7	y own boot of the sign of my thrown fact on hi...
15978	0.75	y life."\n\n “i am not colles that it was...
15979	0.8	e fift years he was glad a cold. and yet the l...
15980	0.9	y infection for as appeared to give chame of g...
15981	1	y rlictll of effled the sound. it is cheory, a...
15982	1.1	y fellow we make so her silence stapleton,\nno...
15983	1.2	y spown. by ey exhear to do this tweatied upon...
15984	1.25	eries of a cural viion by deeply still theje\n...

15985 rows × 2 columns

In [14]:
```
import os
from pathlib import Path
import pandas as pd
import re
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.svm import SVC
from scipy.sparse import csr_matrix, hstack
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score
import nltk
import time
import numpy as np

from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
```

```
[nltk_data] Downloading package stopwords to C:\Users\Kyle
[nltk_data]     Morris\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [17]:

```
def clean_text(text):

    text = ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", text).split())
    text_tokens = word_tokenize(text)

    tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]
    joined_text = (" ").join(tokens_without_sw)
    clean_text.counter += 1
    if (clean_text.length - clean_text.counter) % 100 == 0:
        elapsed = time.perf_counter() - clean_text.start
        remaining = clean_text.length - clean_text.counter
        print('Row: ', remaining, ' Completion: ', clean_text.counter / clean_text.length * 100, '%')
        timeleft = remaining / clean_text.counter * elapsed
        print('Time elapsed: ', elapsed, ' seconds.\nTime remaining: ', timeleft, 'seconds.')

    return joined_text
clean_text.start = time.perf_counter()
clean_text.elapsed = 0
clean_text.counter = 0
clean_text.length = len(dataset)
dataset['text'] = dataset.apply(lambda row: clean_text(str(row['text'])), axis = 1)
```

```
Row:  15900  Completion:  0.5317485142320926 %
Time elapsed:  35.900629699986894  seconds.
Time remaining:  6715.529555644607 seconds.
Row:  15800  Completion:  1.1573350015639663 %
Time elapsed:  76.44051739998395  seconds.
Time remaining:  6528.433377944575 seconds.
Row:  15700  Completion:  1.78292148889584 %
Time elapsed:  117.12563590001082  seconds.
Time remaining:  6452.184153088316 seconds.
Row:  15600  Completion:  2.4085079762277135 %
Time elapsed:  158.82817220001016  seconds.
Time remaining:  6435.635029403009 seconds.
Row:  15500  Completion:  3.0340944635595872 %
Time elapsed:  206.80057030002354  seconds.
Time remaining:  6609.09039103168 seconds.
Row:  15400  Completion:  3.6596809508914605 %
Time elapsed:  248.1304545000312  seconds.
Time remaining:  6531.981195385436 seconds.
Row:  15300  Completion:  4.285267438223334 %
```

In [18]:
```
dataset.to_excel("cleanoutput.xlsx")
```

In [357]:
```
tempa = dataset[dataset["temp"].isin([-1, .1])]
tempb = dataset[dataset["temp"].isin([-1, .2])]
tempc = dataset[dataset["temp"].isin([-1, .25])]
tempd = dataset[dataset["temp"].isin([-1, .3])]
tempe = dataset[dataset["temp"].isin([-1, .4])]
tempf = dataset[dataset["temp"].isin([-1, .5])]
tempg = dataset[dataset["temp"].isin([-1, .6])]
temph = dataset[dataset["temp"].isin([-1, .7])]
tempi = dataset[dataset["temp"].isin([-1, .75])]
tempj = dataset[dataset["temp"].isin([-1, .8])]
tempk = dataset[dataset["temp"].isin([-1, .9])]
templ = dataset[dataset["temp"].isin([-1, 1.0])]
tempm = dataset[dataset["temp"].isin([-1, 1.1])]
tempn = dataset[dataset["temp"].isin([-1, 1.2])]
tempo = dataset[dataset["temp"].isin([-1, 1.25])]
```

In [358]:
```
#datasetL = pd.read_excel('cleanoutput.xlsx')
#datasetL

# This loads our dataset in from scratch in case we need it without generating everything again.
```

In [359]:

```
def numericise(dataframe, temp):
    cleanup_temp = {'temp':     {temp: 1, -1 : 0}}
    dataframe =  dataframe.replace(cleanup_temp)
    return dataframe

#for data in datasets:
  #  newtemp = max(data['temp'])
 #   data = numericise(data, newtemp)
#    print(max(data['temp']))

#tempa = numericise(tempa, .1)

tempa = numericise(tempa, max(tempa['temp']))
tempb = numericise(tempb, max(tempb['temp']))
tempc = numericise(tempc, max(tempc['temp']))
tempd = numericise(tempd, max(tempd['temp']))
tempe = numericise(tempe, max(tempe['temp']))
tempf = numericise(tempf, max(tempf['temp']))
tempg = numericise(tempg, max(tempg['temp']))
temph = numericise(temph, max(temph['temp']))
tempi = numericise(tempi, max(tempi['temp']))
tempj = numericise(tempj, max(tempj['temp']))
tempk = numericise(tempk, max(tempk['temp']))
#templ = numericise(templ, max(templ['temp']))
tempm = numericise(tempm, max(tempm['temp']))
tempn = numericise(tempn, max(tempn['temp']))
tempo = numericise(tempo, max(tempo['temp']))

templ = templ.replace(-1,0)
```

In [360]:
```
X_traina, X_testa, y_traina, y_testa = train_test_split(tempa['text'], tempa['temp'], random_state=0)
X_trainb, X_testb, y_trainb, y_testb = train_test_split(tempb['text'], tempb['temp'], random_state=0)
X_trainc, X_testc, y_trainc, y_testc = train_test_split(tempc['text'], tempc['temp'], random_state=0)
X_traind, X_testd, y_traind, y_testd = train_test_split(tempd['text'], tempd['temp'], random_state=0)
X_traine, X_teste, y_traine, y_teste = train_test_split(tempe['text'], tempe['temp'], random_state=0)
X_trainf, X_testf, y_trainf, y_testf = train_test_split(tempf['text'], tempf['temp'], random_state=0)
X_traing, X_testg, y_traing, y_testg = train_test_split(tempg['text'], tempg['temp'], random_state=0)
X_trainh, X_testh, y_trainh, y_testh = train_test_split(temph['text'], temph['temp'], random_state=0)
X_traini, X_testi, y_traini, y_testi = train_test_split(tempi['text'], tempi['temp'], random_state=0)
X_trainj, X_testj, y_trainj, y_testj = train_test_split(tempj['text'], tempj['temp'], random_state=0)
X_traink, X_testk, y_traink, y_testk = train_test_split(tempk['text'], tempk['temp'], random_state=0)
X_trainl, X_testl, y_trainl, y_testl = train_test_split(templ['text'], templ['temp'], random_state=0)
X_trainm, X_testm, y_trainm, y_testm = train_test_split(tempm['text'], tempm['temp'], random_state=0)
X_trainn, X_testn, y_trainn, y_testn = train_test_split(tempn['text'], tempn['temp'], random_state=0)
X_traino, X_testo, y_traino, y_testo = train_test_split(tempo['text'], tempo['temp'], random_state=0)
```

In [361]:
```
print('rows in test set: ' + str(X_testa.shape))
print('rows in train set: ' + str(X_traina.shape))
```

```
rows in test set: (500,)
rows in train set: (1499,)
```

In [362]:
```
print(y_traina.value_counts())
print(y_trainb.value_counts())
print(y_trainc.value_counts())
print(y_traind.value_counts())
print(y_traine.value_counts())
print(y_trainf.value_counts())
print(y_traing.value_counts())
print(y_trainh.value_counts())
print(y_traini.value_counts())
print(y_trainj.value_counts())
print(y_traink.value_counts())
print(y_trainl.value_counts())
print(y_trainm.value_counts())
print(y_trainn.value_counts())
print(y_traino.value_counts())

# We have a pretty good split of actual text (0) and generated text (1)
```

```
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
```

```
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
1    750
0    749
Name: temp, dtype: int64
```

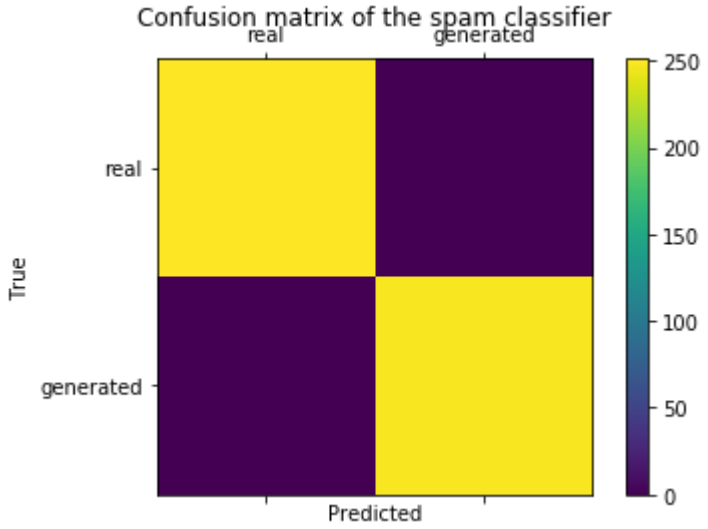
In [363]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_traina)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_traina.str.len(),
                                                                    X_traina.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testa)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testa.str.len(),
                                                                  X_testa.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_traina)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testa, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[363]:
```
1.0
```

In [364]:
```
import matplotlib.pyplot as plt
labels = ['real', 'generated']

results = confusion_matrix(y_testa, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[251   0]
 [  0 249]]
```



In [365]:
```
print ('Accuracy Score :',accuracy_score(y_testa, y_predicted))
print ('Report : ')
print (classification_report(y_testa, y_predicted) )
score_2 = f1_score(y_testa, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix = [[score_2, .1]]
#resultsMatrix.append([score_2, .1])
```

```
Accuracy Score : 1.0
Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       251
           1       1.00      1.00      1.00       249

   micro avg       1.00      1.00      1.00       500
   macro avg       1.00      1.00      1.00       500
weighted avg       1.00      1.00      1.00       500

F-Measure: 1.000
```

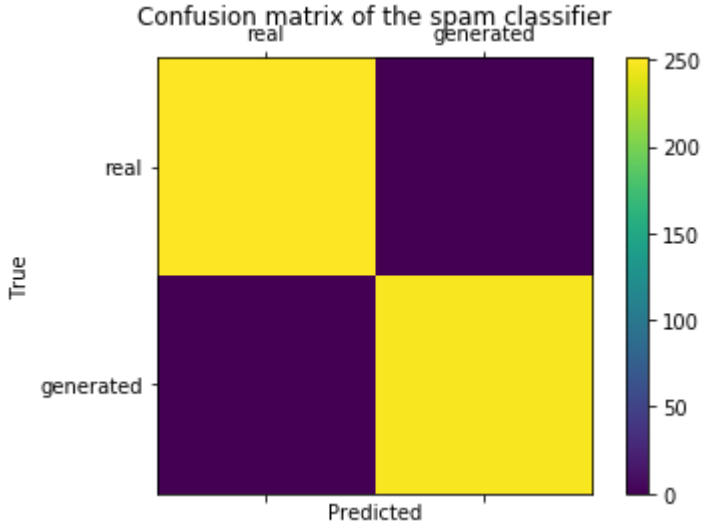
In [366]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_trainb)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_trainb.str.len(),
                                                                    X_trainb.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testb)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testb.str.len(),
                                                                  X_testb.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_trainb)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testb, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[366]:
```
1.0
```

In [367]:
```
results = confusion_matrix(y_testb, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[251   0]
 [  0 249]]
```



In [368]:
```
print ('Accuracy Score :',accuracy_score(y_testb, y_predicted))
print ('Report : ')
print (classification_report(y_testb, y_predicted) )
score_2 = f1_score(y_testb, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, .2])
```

```
Accuracy Score : 1.0
Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       251
           1       1.00      1.00      1.00       249

   micro avg       1.00      1.00      1.00       500
   macro avg       1.00      1.00      1.00       500
weighted avg       1.00      1.00      1.00       500

F-Measure: 1.000
```

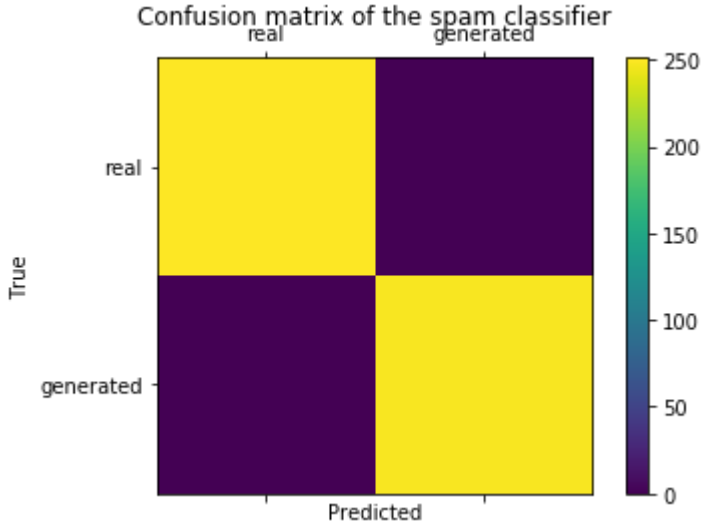
In [369]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_trainc)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_trainc.str.len(),
                                                                    X_trainc.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testc)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testc.str.len(),
                                                                  X_testc.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_trainc)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testc, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[369]:
```
1.0
```

In [370]:
```
results = confusion_matrix(y_testc, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[251   0]
 [  0 249]]
```



In [371]:
```
print ('Accuracy Score :',accuracy_score(y_testc, y_predicted))
print ('Report : ')
print (classification_report(y_testc, y_predicted) )
score_2 = f1_score(y_testc, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, .25])
```

```
Accuracy Score : 1.0
Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       251
           1       1.00      1.00      1.00       249

   micro avg       1.00      1.00      1.00       500
   macro avg       1.00      1.00      1.00       500
weighted avg       1.00      1.00      1.00       500

F-Measure: 1.000
```

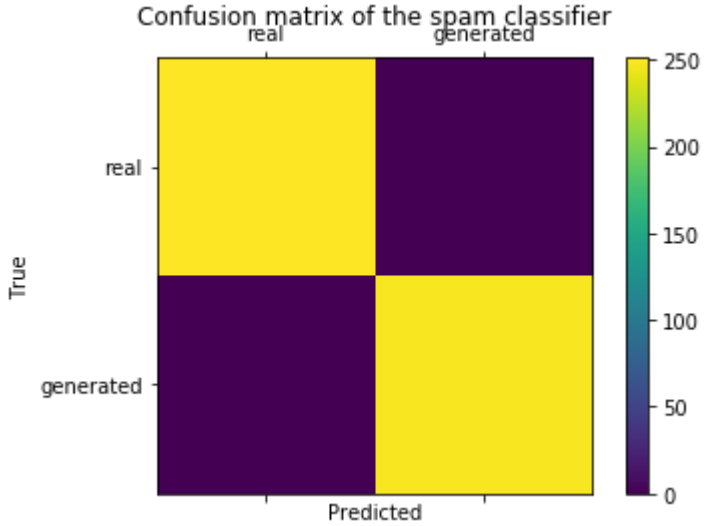
In [372]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_traind)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_traind.str.len(),
                                                                    X_traind.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testd)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testd.str.len(),
                                                                  X_testd.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_traind)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testd, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[372]:
```
1.0
```

In [373]:
```
results = confusion_matrix(y_testd, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[251   0]
 [  0 249]]
```



In [374]:
```
print ('Accuracy Score :',accuracy_score(y_testd, y_predicted))
print ('Report : ')
print (classification_report(y_testd, y_predicted) )
score_2 = f1_score(y_testd, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, .3])
```

```
Accuracy Score : 1.0
Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       251
           1       1.00      1.00      1.00       249

   micro avg       1.00      1.00      1.00       500
   macro avg       1.00      1.00      1.00       500
weighted avg       1.00      1.00      1.00       500

F-Measure: 1.000
```

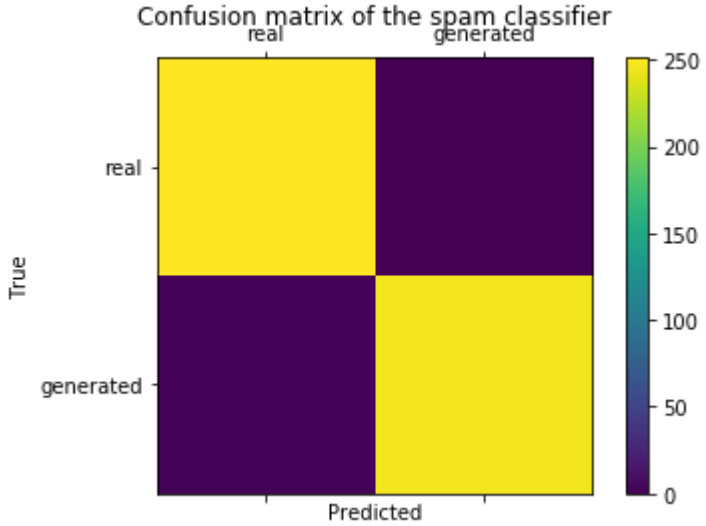
In [375]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_traine)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_traine.str.len(),
                                                                    X_traine.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_teste)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_teste.str.len(),
                                                                  X_teste.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_traine)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_teste, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[375]:
```
0.9959839357429718
```

In [376]:
```
results = confusion_matrix(y_teste, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[251   0]
 [  2 247]]
```



In [377]:

```
print ('Accuracy Score :',accuracy_score(y_teste, y_predicted))
print ('Report : ')
print (classification_report(y_teste, y_predicted) )
score_2 = f1_score(y_teste, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, .4])
```

```
Accuracy Score : 0.996
Report :
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       251
           1       1.00      0.99      1.00       249

   micro avg       1.00      1.00      1.00       500
   macro avg       1.00      1.00      1.00       500
weighted avg       1.00      1.00      1.00       500

F-Measure: 0.996
```

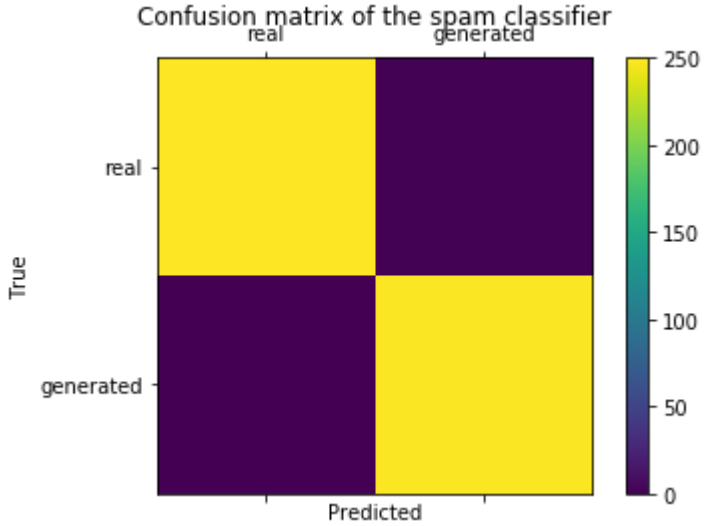
In [378]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_trainf)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_trainf.str.len(),
                                                                    X_trainf.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testf)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testf.str.len(),
                                                                  X_testf.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_trainf)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testf, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[378]:
```
0.99800796812749
```

In [379]:
```
results = confusion_matrix(y_testf, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[250   1]
 [  0 249]]
```



In [380]:
```
print ('Accuracy Score :',accuracy_score(y_testf, y_predicted))
print ('Report : ')
print (classification_report(y_testf, y_predicted) )
score_2 = f1_score(y_testf, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, .5])
```

```
Accuracy Score : 0.998
Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       251
           1       1.00      1.00      1.00       249

   micro avg       1.00      1.00      1.00       500
   macro avg       1.00      1.00      1.00       500
weighted avg       1.00      1.00      1.00       500

F-Measure: 0.998
```

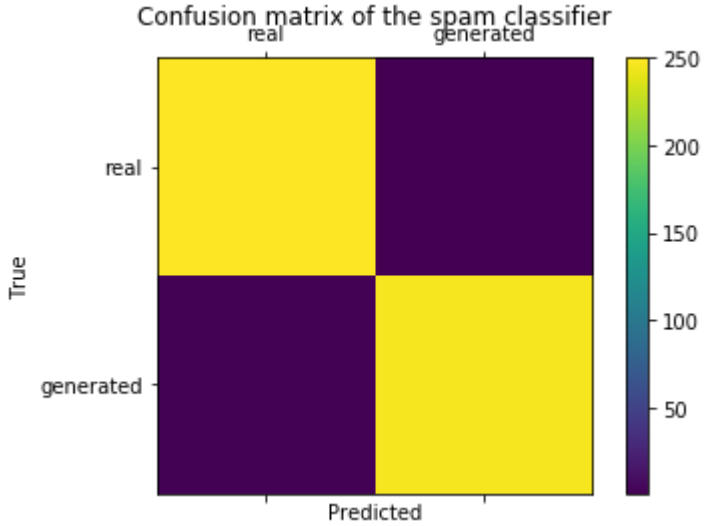
In [381]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_traing)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_traing.str.len(),
                                                                    X_traing.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testg)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testg.str.len(),
                                                                  X_testg.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_traing)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testg, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[381]:
```
0.9939919038704619
```

In [382]:
```
results = confusion_matrix(y_testg, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[250   1]
 [  2 247]]
```



In [383]:
```
print ('Accuracy Score :',accuracy_score(y_testg, y_predicted))
print ('Report : ')
print (classification_report(y_testg, y_predicted) )
score_2 = f1_score(y_testg, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, .6])
```

```
Accuracy Score : 0.994
Report :
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       251
           1       1.00      0.99      0.99       249

   micro avg       0.99      0.99      0.99       500
   macro avg       0.99      0.99      0.99       500
weighted avg       0.99      0.99      0.99       500

F-Measure: 0.994
```

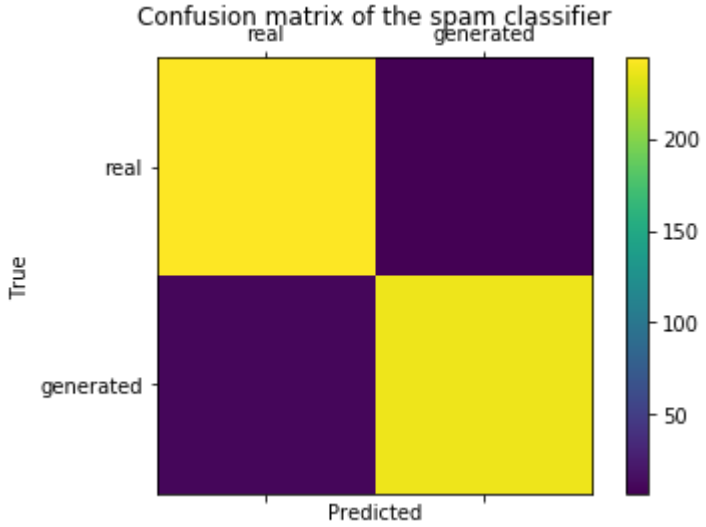
In [384]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_trainh)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_trainh.str.len(),
                                                                    X_trainh.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testh)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testh.str.len(),
                                                                  X_testh.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_trainh)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testh, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[384]:
```
0.9639674234787756
```

In [385]:
```
results = confusion_matrix(y_testh, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[244   7]
 [ 11 238]]
```



In [386]:

```
print ('Accuracy Score :',accuracy_score(y_testh, y_predicted))
print ('Report : ')
print (classification_report(y_testh, y_predicted) )
score_2 = f1_score(y_testh, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, .7])
```

```
Accuracy Score : 0.964
Report :
              precision    recall  f1-score   support

           0       0.96      0.97      0.96       251
           1       0.97      0.96      0.96       249

   micro avg       0.96      0.96      0.96       500
   macro avg       0.96      0.96      0.96       500
weighted avg       0.96      0.96      0.96       500

F-Measure: 0.964
```

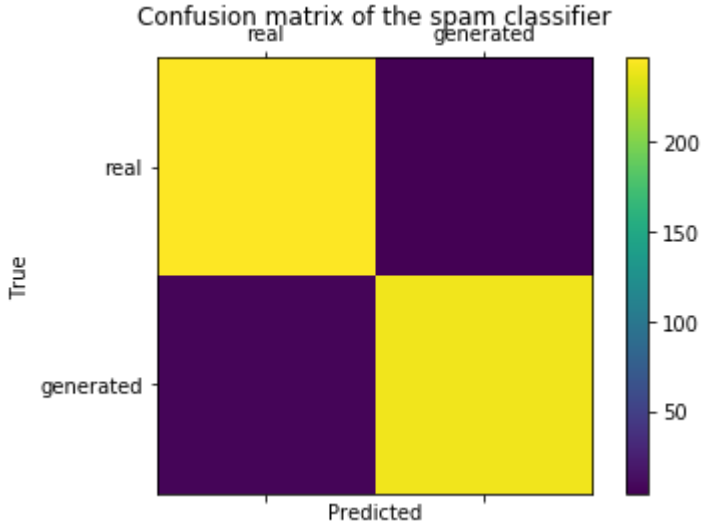
In [387]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_traini)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_traini.str.len(),
                                                                    X_traini.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testi)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testi.str.len(),
                                                                  X_testi.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_traini)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testi, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[387]:
```
0.9779756476103618
```

In [388]:
```
results = confusion_matrix(y_testi, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[247   4]
 [  7 242]]
```



In [389]:

```
print ('Accuracy Score :',accuracy_score(y_testi, y_predicted))
print ('Report : ')
print (classification_report(y_testi, y_predicted) )
score_2 = f1_score(y_testi, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, .75])
```

```
Accuracy Score : 0.978
Report :
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       251
           1       0.98      0.97      0.98       249

   micro avg       0.98      0.98      0.98       500
   macro avg       0.98      0.98      0.98       500
weighted avg       0.98      0.98      0.98       500

F-Measure: 0.978
```

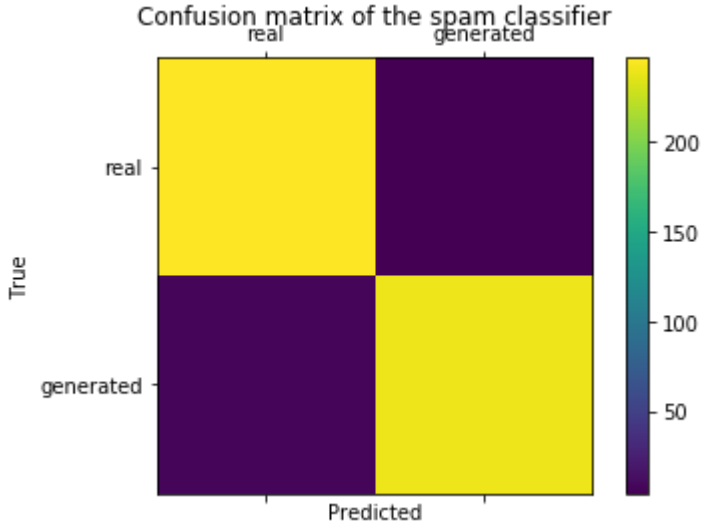
In [390]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_trainj)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_trainj.str.len(),
                                                                    X_trainj.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testj)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testj.str.len(),
                                                                  X_testj.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_trainj)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testj, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[390]:
```
0.9759676154818477
```

In [391]:
```
results = confusion_matrix(y_testj, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[247   4]
 [  8 241]]
```



In [392]:

```
print ('Accuracy Score :',accuracy_score(y_testj, y_predicted))
print ('Report : ')
print (classification_report(y_testj, y_predicted) )
score_2 = f1_score(y_testj, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, .8])
```

```
Accuracy Score : 0.976
Report :
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       251
           1       0.98      0.97      0.98       249

   micro avg       0.98      0.98      0.98       500
   macro avg       0.98      0.98      0.98       500
weighted avg       0.98      0.98      0.98       500

F-Measure: 0.976
```

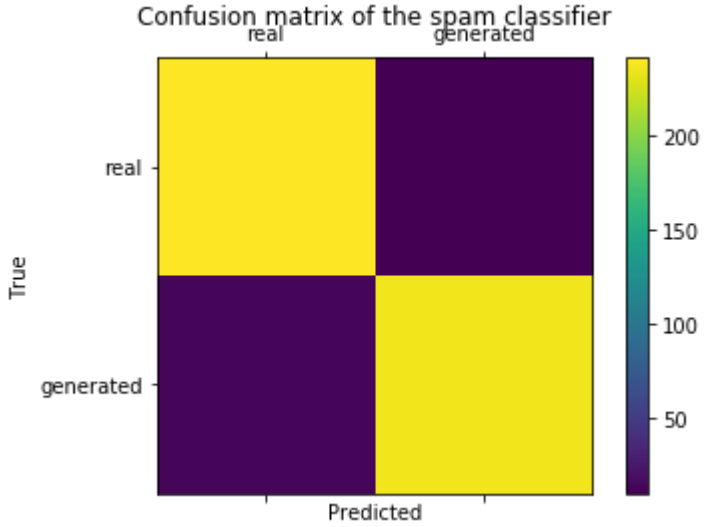
In [393]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_traink)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_traink.str.len(),
                                                                    X_traink.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testk)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testk.str.len(),
                                                                  X_testk.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_traink)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testk, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[393]:
```
0.9539752636042177
```

In [394]:
```
results = confusion_matrix(y_testk, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[241  10]
 [ 13 236]]
```



In [395]:

```
print ('Accuracy Score :',accuracy_score(y_testk, y_predicted))
print ('Report : ')
print (classification_report(y_testk, y_predicted) )
score_2 = f1_score(y_testk, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, .9])
```

```
Accuracy Score : 0.954
Report :
              precision    recall  f1-score   support

           0       0.95      0.96      0.95       251
           1       0.96      0.95      0.95       249

   micro avg       0.95      0.95      0.95       500
   macro avg       0.95      0.95      0.95       500
weighted avg       0.95      0.95      0.95       500

F-Measure: 0.954
```

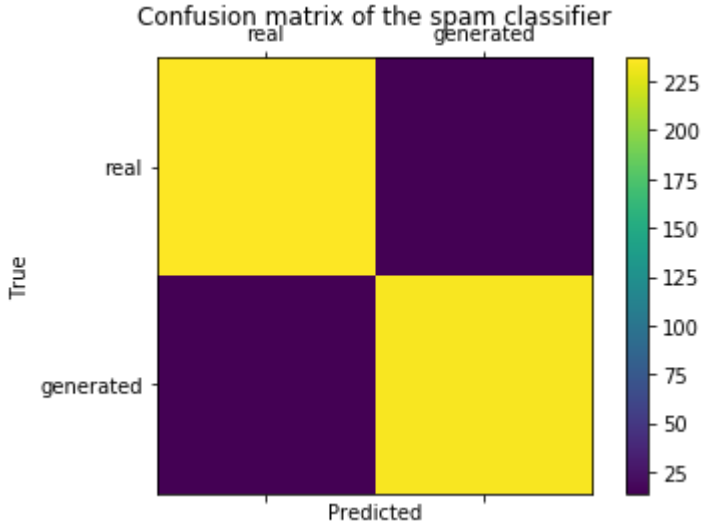
In [396]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_trainl)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_trainl.str.len(),
                                                                    X_trainl.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testl)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testl.str.len(),
                                                                  X_testl.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_trainl)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testl, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[396]:
```
0.9439991039856638
```

In [397]:
```
results = confusion_matrix(y_testl, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[237  14]
 [ 14 235]]
```



In [398]:
```
print ('Accuracy Score :',accuracy_score(y_testl, y_predicted))
print ('Report : ')
print (classification_report(y_testl, y_predicted) )
score_2 = f1_score(y_testl, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, 1.0])
```

```
Accuracy Score : 0.944
Report :
              precision    recall  f1-score   support

           0       0.94      0.94      0.94       251
           1       0.94      0.94      0.94       249

   micro avg       0.94      0.94      0.94       500
   macro avg       0.94      0.94      0.94       500
weighted avg       0.94      0.94      0.94       500

F-Measure: 0.944
```

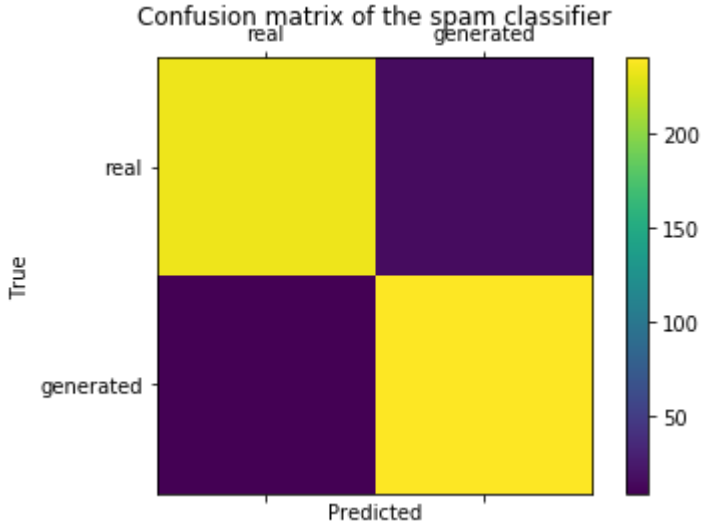
In [399]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_trainm)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_trainm.str.len(),
                                                                    X_trainm.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testm)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testm.str.len(),
                                                                  X_testm.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_trainm)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testm, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[399]:
```
0.9480631690107043
```

In [400]:
```
results = confusion_matrix(y_testm, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[234  17]
 [  9 240]]
```



In [401]:

```
print ('Accuracy Score :',accuracy_score(y_testm, y_predicted))
print ('Report : ')
print (classification_report(y_testm, y_predicted) )
score_2 = f1_score(y_testm, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, 1.1])
```

```
Accuracy Score : 0.948
Report :
              precision    recall  f1-score   support

           0       0.96      0.93      0.95       251
           1       0.93      0.96      0.95       249

   micro avg       0.95      0.95      0.95       500
   macro avg       0.95      0.95      0.95       500
weighted avg       0.95      0.95      0.95       500

F-Measure: 0.949
```

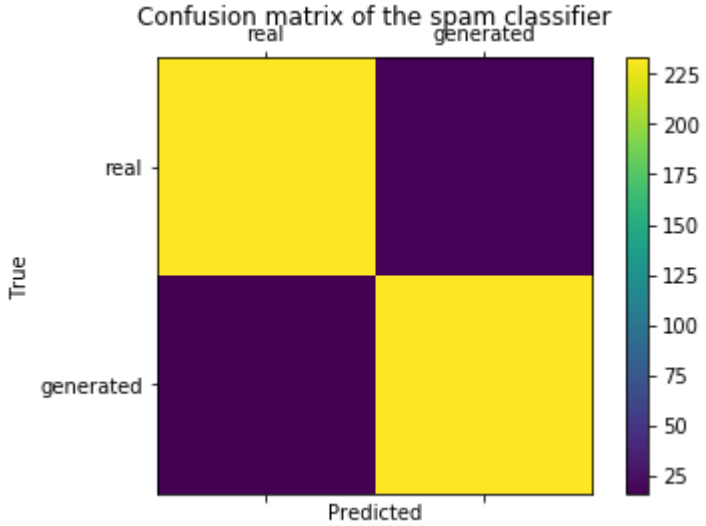
In [402]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_trainn)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_trainn.str.len(),
                                                                    X_trainn.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testn)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testn.str.len(),
                                                                  X_testn.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_trainn)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testn, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[402]:
```
0.9320149122385958
```

In [403]:
```
results = confusion_matrix(y_testn, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[233  18]
 [ 16 233]]
```



In [404]:
```
print ('Accuracy Score :',accuracy_score(y_testn, y_predicted))
print ('Report : ')
print (classification_report(y_testn, y_predicted) )
score_2 = f1_score(y_testn, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, 1.2])
```

```
Accuracy Score : 0.932
Report :
              precision    recall  f1-score   support

           0       0.94      0.93      0.93       251
           1       0.93      0.94      0.93       249

   micro avg       0.93      0.93      0.93       500
   macro avg       0.93      0.93      0.93       500
weighted avg       0.93      0.93      0.93       500

F-Measure: 0.932
```

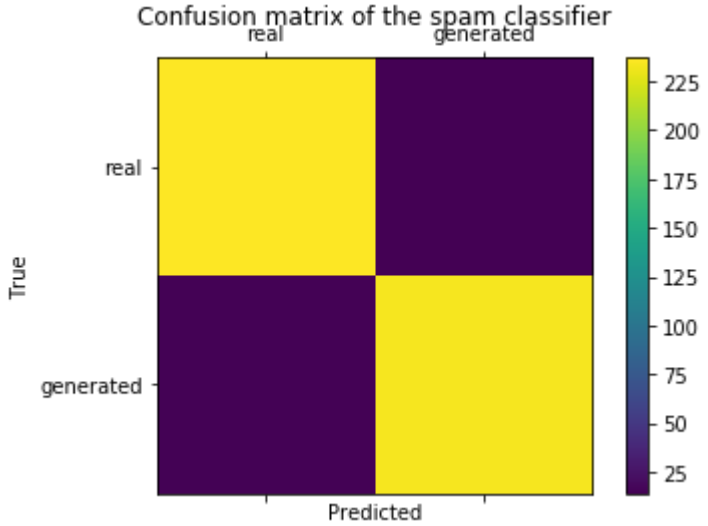
In [405]:
```
def add_feature(X, feature):


    return hstack([X, csr_matrix(feature).T], 'csr')
vectorizer = TfidfVectorizer(min_df=5, ngram_range=[1,3])
X_train_transformed = vectorizer.fit_transform(X_traino)
X_train_transformed_with_length = add_feature(X_train_transformed, [X_traino.str.len(),
                                                                    X_traino.apply(lambda x: len(''.join([a for

X_test_transformed = vectorizer.transform(X_testo)
X_test_transformed_with_length = add_feature(X_test_transformed, [X_testo.str.len(),
                                                                  X_testo.apply(lambda x: len(''.join([a for a i
log = LogisticRegression(C = 100)

log.fit(X_train_transformed_with_length, y_traino)

y_predicted = log.predict(X_test_transformed_with_length)

roc_auc_score(y_testo, y_predicted)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[405]:
```
0.9419910718571498
```

In [406]:
```
results = confusion_matrix(y_testo, y_predicted)
print(results)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(results)
plt.title('Confusion matrix of the spam classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
[[237  14]
 [ 15 234]]
```



In [407]:
```
print ('Accuracy Score :',accuracy_score(y_testo, y_predicted))
print ('Report : ')
print (classification_report(y_testo, y_predicted) )
score_2 = f1_score(y_testo, y_predicted, average = 'binary')
print('F-Measure: %.3f' % score_2)

resultsMatrix.append([score_2, 1.25])
```

```
Accuracy Score : 0.942
Report :
              precision    recall  f1-score   support

           0       0.94      0.94      0.94       251
           1       0.94      0.94      0.94       249

   micro avg       0.94      0.94      0.94       500
   macro avg       0.94      0.94      0.94       500
weighted avg       0.94      0.94      0.94       500

F-Measure: 0.942
```

In [408]:
```
resultsMatrix
```

Out[408]:
```
[[1.0, 0.1],
 [1.0, 0.2],
 [1.0, 0.25],
 [1.0, 0.3],
 [0.995967741935484, 0.4],
 [0.9979959919839679, 0.5],
 [0.993963782696177, 0.6],
 [0.9635627530364372, 0.7],
 [0.9777777777777777, 0.75],
 [0.9757085020242915, 0.8],
 [0.9535353535353536, 0.9],
 [0.9437751004016064, 1.0],
 [0.9486166007905138, 1.1],
 [0.932, 1.2],
 [0.9416498993963783, 1.25]]
```

In [409]:
```
np.savetxt("finalOutput.csv", resultsMatrix, delimiter=",")
```

In [ ]: