# Exploring the Enron dataset:

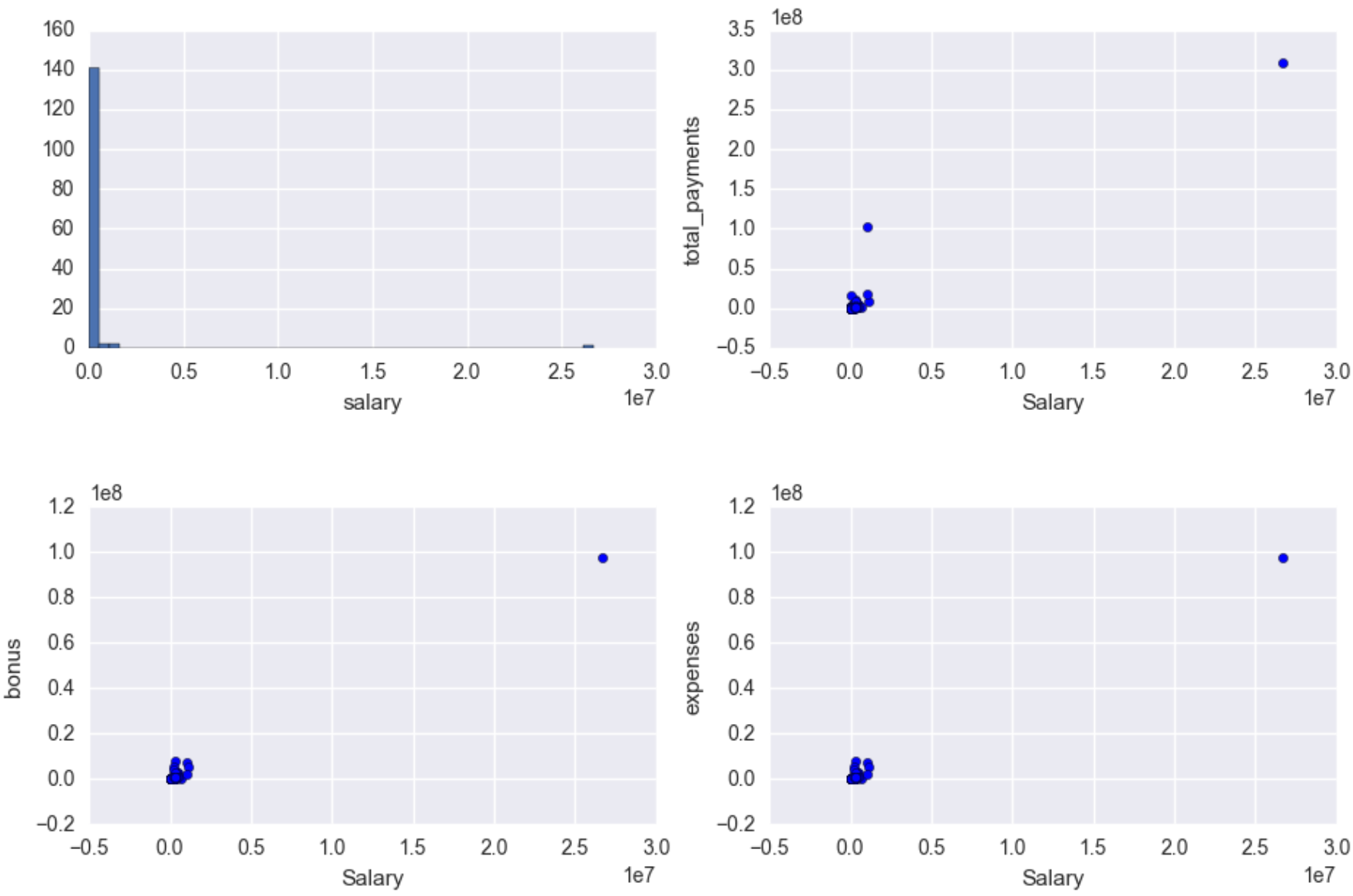
In this project I will explore the ENron email dataset and examine a number of clasifiers to predictthe POI (i.e. Point of Interest) person, based on a number of features extracted from the indivisuals' emails. This process will include identifying and removing the outliers, creating new features based on the previous ones, and engineering classifiers and tuning them to improve the overall prediction accuracy.

In [3]:
```
'''First we will read and examine teh dataset'''


import sys
import pickle
import pandas as pd
import os
import seaborn as sb
import matplotlib.pyplot as plt
from feature_format import featureFormat, targetFeatureSplit


os.chdir("C:/Users/sur216/Box Sync/school stuff/Udacity (sur216@psu.edu)/Data Analyst/p5_enron/ud120-projects-maste
r/final_project")
enron_data = pickle.load(open("final_project_dataset.pkl", "r"))
print "the number of items in the dictionary is: {0}".format(len(enron_data.keys()))
print "the email sensers/recipients are: {0}".format(enron_data.keys())
print "and the current features for each row in the dataste are: {0}".format(enron_data.values()[1].keys())
```

```
the number of items in the dictionary is: 146
the email sensers/recipients are: ['METTS MARK', 'BAXTER JOHN C', 'ELLIOTT STEVEN', 'CORDES WILLIAM R', 'HANNON KEVI
N P', 'MORDAUNT KRISTINA M', 'MEYER ROCKFORD G', 'MCMAHON JEFFREY', 'HORTON STANLEY C', 'PIPER GREGORY F', 'HUMPHREY
 GENE E', 'UMANOFF ADAM S', 'BLACHMAN JEREMY M', 'SUNDE MARTIN', 'GIBBS DANA R', 'LOWRY CHARLES P', 'COLWELL WESLE
Y', 'MULLER MARK S', 'JACKSON CHARLENE R', 'WESTFAHL RICHARD K', 'WALTERS GARETH W', 'WALLS JR ROBERT H', 'KITCHEN L
OUISE', 'CHAN RONNIE', 'BELFER ROBERT', 'SHANKMAN JEFFREY A', 'WODRASKA JOHN', 'BERGSIEKER RICHARD P', 'URQUHART JOH
N A', 'BIBI PHILIPPE A', 'RIEKER PAULA H', 'WHALEY DAVID A', 'BECK SALLY W', 'HAUG DAVID L', 'ECHOLS JOHN B', 'MENDE
LSOHN JOHN', 'HICKERSON GARY J', 'CLINE KENNETH W', 'LEWIS RICHARD', 'HAYES ROBERT E', 'MCCARTY DANNY J', 'KOPPER MI
CHAEL J', 'LEFF DANIEL P', 'LAVORATO JOHN J', 'BERBERIAN DAVID', 'DETMERING TIMOTHY J', 'WAKEHAM JOHN', 'POWERS WILL
IAM', 'GOLD JOSEPH', 'BANNANTINE JAMES M', 'DUNCAN JOHN H', 'SHAPIRO RICHARD S', 'SHERRIFF JOHN R', 'SHELBY REX', 'L
EMAISTRE CHARLES', 'DEFFNER JOSEPH M', 'KISHKILL JOSEPH G', 'WHALLEY LAWRENCE G', 'MCCONNELL MICHAEL S', 'PIRO JIM',
 'DELAINEY DAVID W', 'SULLIVAN-SHAKLOVITZ COLLEEN', 'WROBEL BRUCE', 'LINDHOLM TOD A', 'MEYER JEROME J', 'LAY KENNETH
 L', 'BUTTS ROBERT H', 'OLSON CINDY K', 'MCDONALD REBECCA', 'CUMBERLAND MICHAEL S', 'GAHN ROBERT S', 'MCCLELLAN GEOR
GE', 'HERMANN ROBERT J', 'SCRIMSHAW MATTHEW', 'GATHMANN WILLIAM D', 'HAEDICKE MARK E', 'BOWEN JR RAYMOND M', 'GILLIS
 JOHN', 'FITZGERALD JAY L', 'MORAN MICHAEL P', 'REDMOND BRIAN L', 'BAZELIDES PHILIP J', 'BELDEN TIMOTHY N', 'DURAN W
ILLIAM D', 'THORN TERENCE H', 'FASTOW ANDREW S', 'FOY JOE', 'CALGER CHRISTOPHER F', 'RICE KENNETH D', 'KAMINSKI WINC
ENTY J', 'LOCKHART EUGENE E', 'COX DAVID', 'OVERDYKE JR JERE C', 'PEREIRA PAULO V. FERRAZ', 'STABLER FRANK', 'SKILLI
NG JEFFREY K', 'BLAKE JR. NORMAN P', 'SHERRICK JEFFREY B', 'PRENTICE JAMES', 'GRAY RODNEY', 'PICKERING MARK R', 'THE
 TRAVEL AGENCY IN THE PARK', 'NOLES JAMES L', 'KEAN STEVEN J', 'TOTAL', 'FOWLER PEGGY', 'WASAFF GEORGE', 'WHITE JR T
HOMAS E', 'CHRISTODOULOU DIOMEDES', 'ALLEN PHILLIP K', 'SHARP VICTORIA T', 'JAEDICKE ROBERT', 'WINOKUR JR. HERBERT
 S', 'BROWN MICHAEL', 'BADUM JAMES P', 'HUGHES JAMES A', 'REYNOLDS LAWRENCE', 'DIMICHELE RICHARD G', 'BHATNAGAR SANJ
AY', 'CARTER REBECCA C', 'BUCHANAN HAROLD G', 'YEAP SOON', 'MURRAY JULIA H', 'GARLAND C KEVIN', 'DODSON KEITH', 'YEA
GER F SCOTT', 'HIRKO JOSEPH', 'DIETRICH JANET R', 'DERRICK JR. JAMES V', 'FREVERT MARK A', 'PAI LOU L', 'BAY FRANKLI
N R', 'HAYSLETT RODERICK J', 'FUGH JOHN L', 'FALLON JAMES B', 'KOENIG MARK E', 'SAVAGE FRANK', 'IZZO LAWRENCE L', 'T
ILNEY ELIZABETH A', 'MARTIN AMANDA K', 'BUY RICHARD B', 'GRAMM WENDY L', 'CAUSEY RICHARD A', 'TAYLOR MITCHELL S', 'D
ONAHUE JR JEFFREY M', 'GLISAN JR BEN F']
and the current features for each row in the dataste are: ['salary', 'to_messages', 'deferral_payments', 'total_paym
ents', 'exercised_stock_options', 'bonus', 'restricted_stock', 'shared_receipt_with_poi', 'restricted_stock_deferre
d', 'total_stock_value', 'expenses', 'loan_advances', 'from_messages', 'other', 'from_this_person_to_poi', 'poi', 'd
irector_fees', 'deferred_income', 'long_term_incentive', 'email_address', 'from_poi_to_this_person']
```

In [98]:
```
# extract features from the dictionary
feature_1 = "salary"
feature_2 = "exercised_stock_options"
feature_3 = "total_payments"
feature_4 = "bonus"
feature_5 = "expenses"
poi  = "poi"
features_list = [poi, feature_1, feature_2, feature_3,feature_4,feature_5]

# make lists from the dataset for our scatter plots
def finance_to_list (input_data):
    data = featureFormat(input_data, features_list, remove_all_zeroes=False, remove_any_zeroes=False)
    poi, finance_feat = targetFeatureSplit( data )
    salary = []
    ex_stock = []
    tot_pay = []
    bonus = []
    expens = []
    for point in finance_feat:
        for i in [0,1,2,3,4]:
            if point[i] == "NaN": point[i] = 0
        salary.append(point[0])
        ex_stock.append(point[1])
        tot_pay.append(point[2])
        bonus.append(point[3])
        expens.append(point[4]/100000)
    return ([salary,ex_stock,tot_pay,bonus,expens])


print(len(finance_to_list(enron_data)[2]))

# plot multiple subplots
%matplotlib inline
plt.rcParams['figure.figsize'] = (11, 7)
f, axarr = plt.subplots(2, 2)
axarr[0, 0].hist(finance_to_list(enron_data)[0], bins = 50)
axarr[0, 0].set_xlabel('salary')
axarr[0, 1].scatter(finance_to_list(enron_data)[0], finance_to_list(enron_data)[2])
axarr[0, 1].set_xlabel('Salary')
axarr[0, 1].set_ylabel('total_payments')
axarr[1, 0].scatter(finance_to_list(enron_data)[0], finance_to_list(enron_data)[3])
axarr[1, 0].set_xlabel('Salary')
axarr[1, 0].set_ylabel('bonus')
axarr[1, 1].scatter(finance_to_list(enron_data)[0], finance_to_list(enron_data)[3])
axarr[1, 1].set_xlabel('Salary')
axarr[1, 1].set_ylabel('expenses')
f.subplots_adjust(hspace=0.5)
```

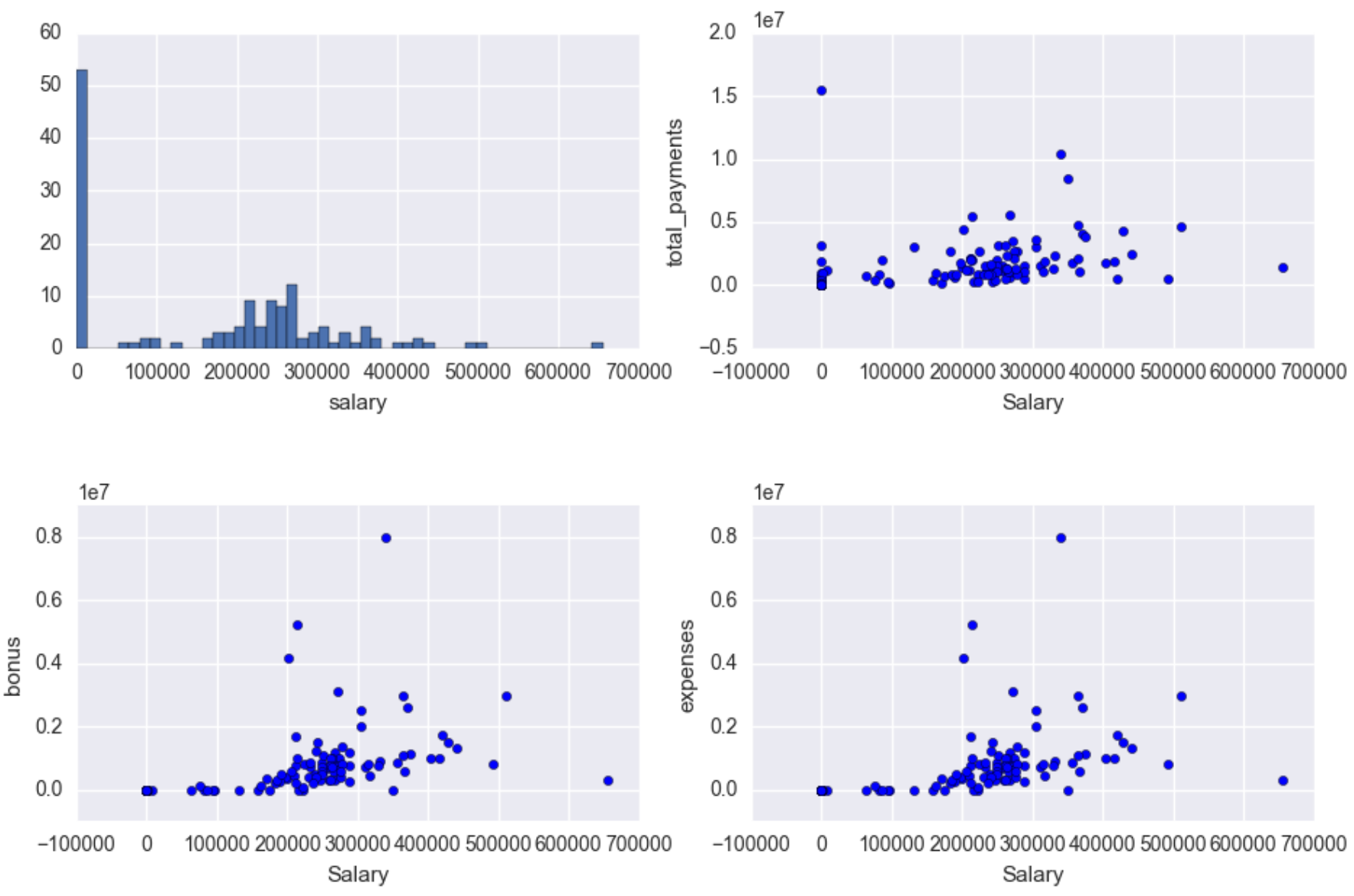
146



In [99]:
```
''' Clearly, there is an outlier that prevents us
from investigating the scatterplots.Since there is only one obvious outlier
, I will simply remove it manually '''

### remove NAN's from dataset and save the top total payment values
outliers = []
for key in enron_data:
    val = enron_data[key]['salary']
    if not val == 'NaN': outliers.append((key, int(val)))

outliers = (sorted(outliers,key=lambda x:x[1],reverse=True)[:5])
print outliers


''' As we learned from the previous histogram of salary, we simply remove those with salaries of higher than 1m. '''

enron_no_outliers ={}
for k,v in enron_data.iteritems():
    if v['salary'] == "NaN": v['salary'] = 0
    if v['salary']< 1000000: enron_no_outliers.update({k:v})

plt.rcParams['figure.figsize'] = (11, 7)
f, axarr = plt.subplots(2, 2)
axarr[0, 0].hist(finance_to_list(enron_no_outliers)[0], bins = 50)
axarr[0, 0].set_xlabel('salary')
axarr[0, 1].scatter(finance_to_list(enron_no_outliers)[0], finance_to_list(enron_no_outliers)[2])
axarr[0, 1].set_xlabel('Salary')
axarr[0, 1].set_ylabel('total_payments')
axarr[1, 0].scatter(finance_to_list(enron_no_outliers)[0], finance_to_list(enron_no_outliers)[3])
axarr[1, 0].set_xlabel('Salary')
axarr[1, 0].set_ylabel('bonus')
axarr[1, 1].scatter(finance_to_list(enron_no_outliers)[0], finance_to_list(enron_no_outliers)[3])
axarr[1, 1].set_xlabel('Salary')
axarr[1, 1].set_ylabel('expenses')
f.subplots_adjust(hspace=0.5)

''' As you can see there are individuals that have significanly higher total payments than others.
We will let them be for now unless they cause problems. '''
```

```
[('TOTAL', 26704229), ('SKILLING JEFFREY K', 1111258), ('LAY KENNETH L', 1072321), ('FREVERT MARK A', 1060932), ('PI
CKERING MARK R', 655037)]
```

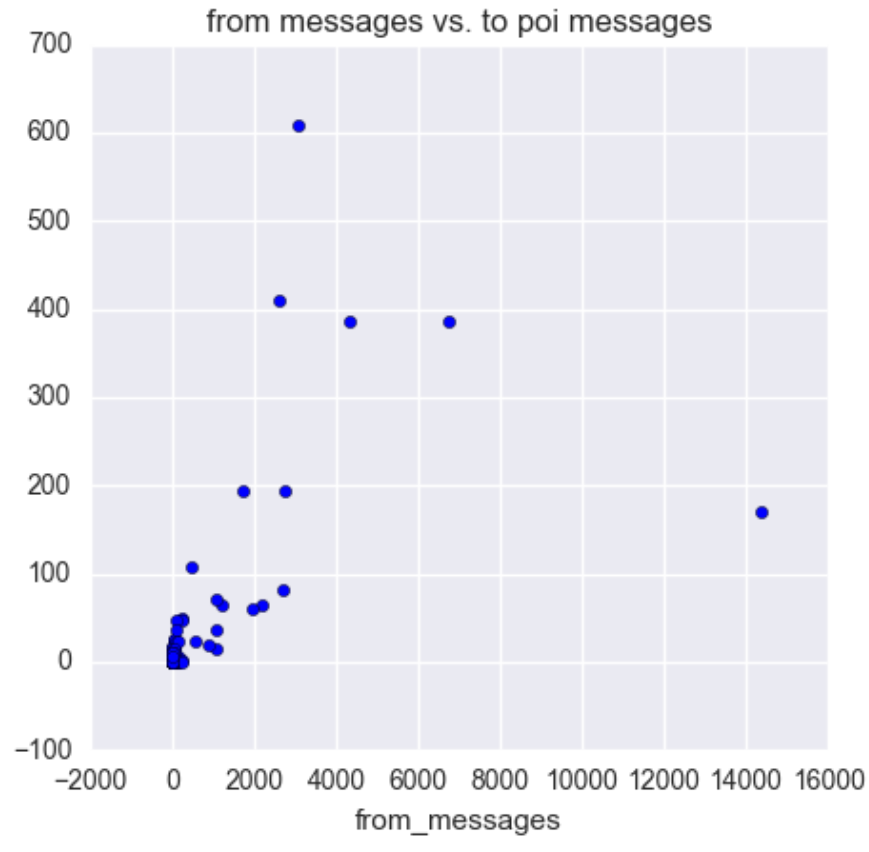
Out[99]:
```
' As you can see there are individuals that have significanly higher total payments than others. \nWe will let them
 be for now unless they cause problems. '
```

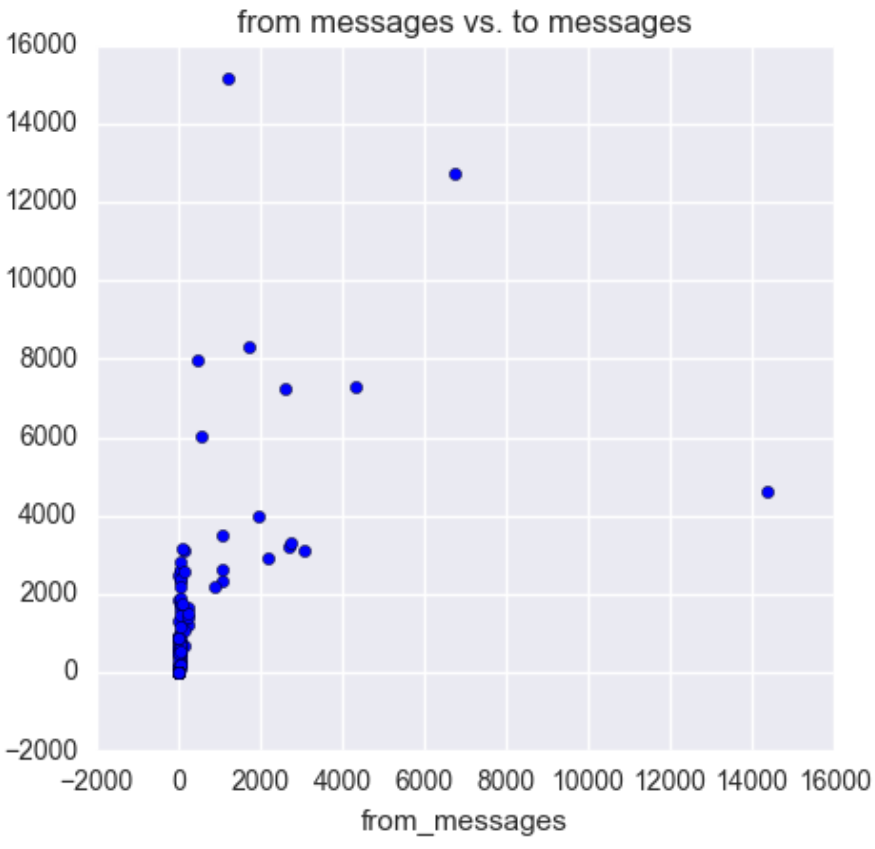


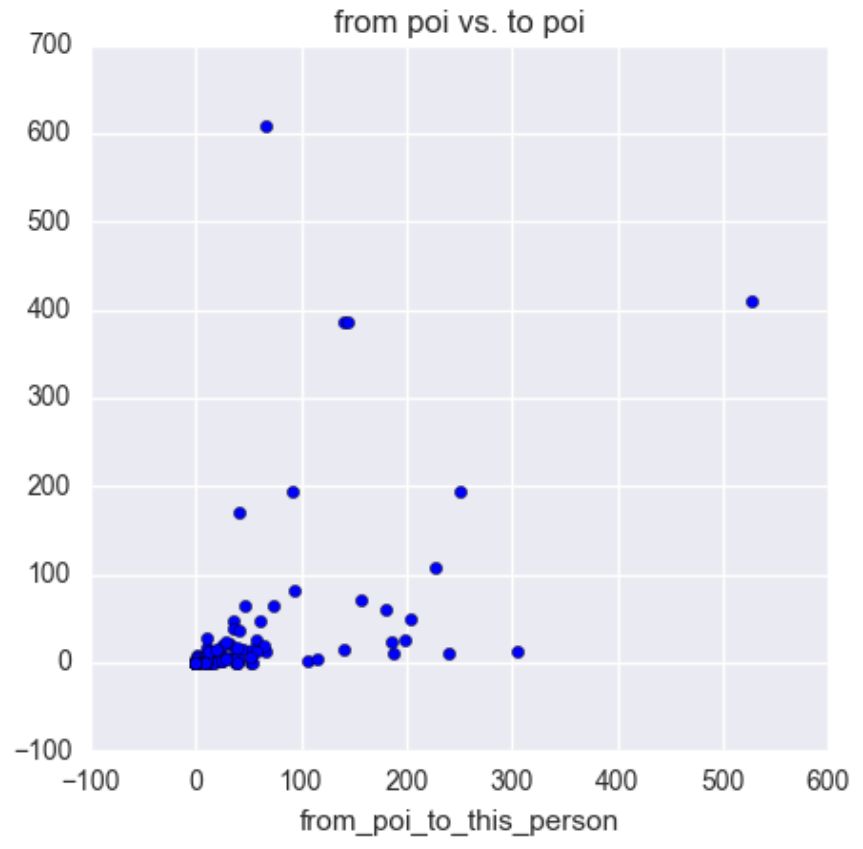

## Task 3: Creat New Feature(s)

It would make sense to take a closer look at the communication patterns of these names via email. So, I will first take a look at the relevant features.

In [101]:
```
['salary', 'to_messages', 'deferral_payments',
 'total_payments', 'exercised_stock_options', 'bonus', 'restricted_stock',
 'shared_receipt_with_poi', 'restricted_stock_deferred', 'total_stock_value',
 'expenses', 'loan_advances', 'from_messages', 'other', 'from_this_person_to_poi',
 'poi', 'director_fees', 'deferred_income', 'long_term_incentive', 'email_address', 'from_poi_to_this_person']

feature_0 = "salary"
feature_1 = "to_messages"
feature_2 = "from_messages"
feature_3 = "from_this_person_to_poi"
feature_4 = "from_poi_to_this_person"
comm_list = [poi,feature_0, feature_1, feature_2, feature_3,feature_4]

# make lists for our scatterplots
def comm_to_list (input_data):
    data = featureFormat(input_data, comm_list,remove_NaN=True, remove_all_zeroes=False, remove_any_zeroes=False)
    poi, comm_feat = targetFeatureSplit( data )
    salary = []
    to_msg = []
    from_msg = []
    to_poi = []
    from_poi = []
    for point in comm_feat:
        for i in [0,1,2,3,4]:
            if point[i] == "NaN": point[i] = 0
        salary.append(point[0])
        to_msg.append(point[1])
        from_msg.append(point[2])
        to_poi.append(point[3])
        from_poi.append(point[4])
    return ([salary,to_msg,from_msg,to_poi,from_poi])



salary = comm_to_list(enron_no_outliers)[0]
to_msg = comm_to_list(enron_no_outliers)[1]
from_msg = comm_to_list(enron_no_outliers)[2]
to_poi = comm_to_list(enron_no_outliers)[3]
from_poi = comm_to_list(enron_no_outliers)[4]


plt.rcParams['figure.figsize'] = (11, 11)
f, axarr = plt.subplots(2, 2)
axarr[0, 0].scatter(to_msg,from_poi)
axarr[0, 0].set_xlabel('to_messages')
axarr[0, 0].set_title('to messages vs. from poi messages')
axarr[0, 1].scatter(from_msg,to_poi)
axarr[0, 1].set_xlabel('from_messages')
axarr[0, 1].set_title('from messages vs. to poi messages')
axarr[1, 0].scatter(from_msg,to_msg)
axarr[1, 0].set_xlabel('from_messages')
axarr[1, 0].set_title('from messages vs. to messages')
axarr[1, 1].scatter(from_poi,to_poi)
axarr[1, 1].set_xlabel('from_poi_to_this_person')
axarr[1, 1].set_title('from poi vs. to poi')
f.subplots_adjust(hspace=0.3)
```

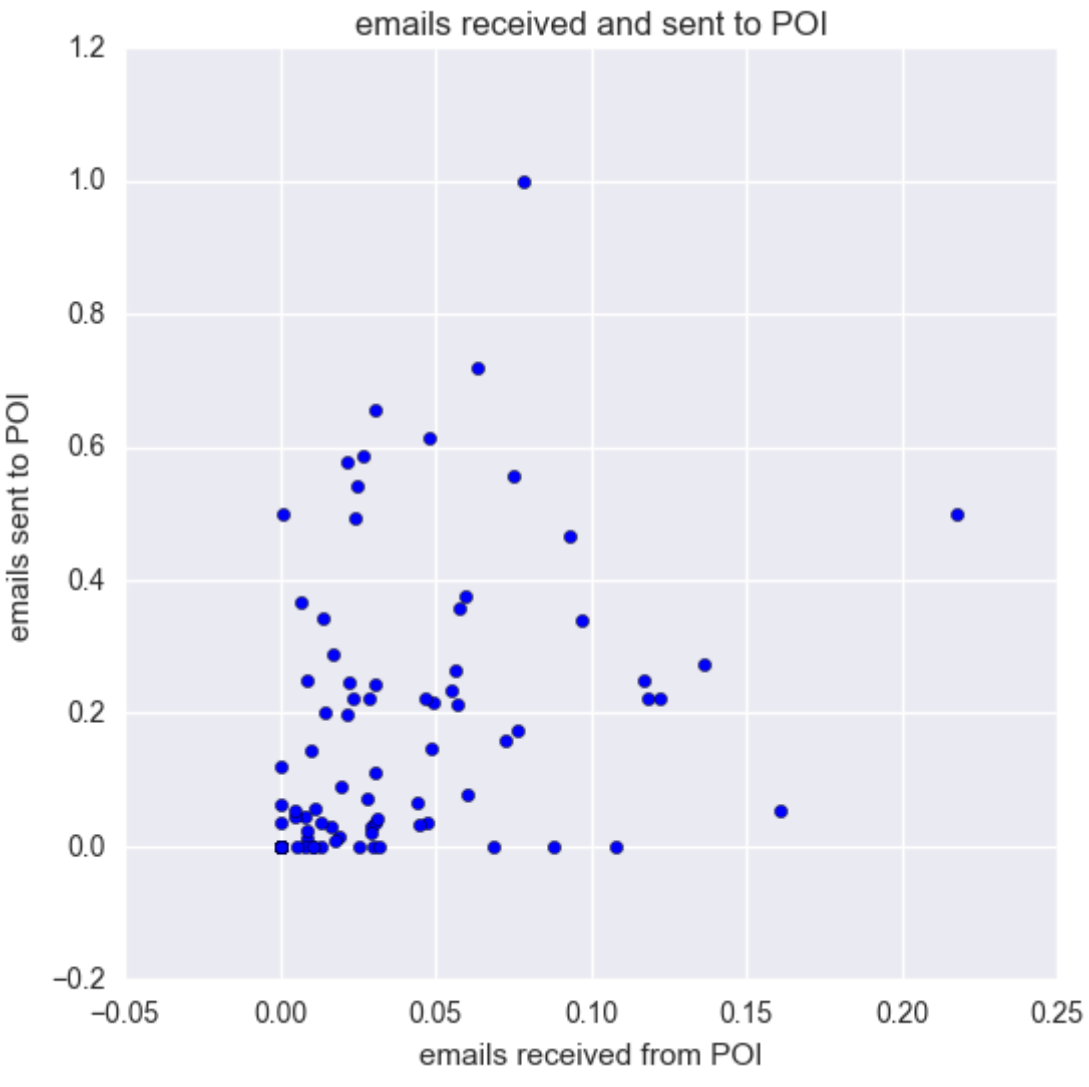




The number of these emails alone will not inform us anything specific. A better idea would be to make a few features from these to see the extent of a person's communication with the poi as a proportion of the total number of his/her emails. Therefor, I will make two additional features: first, from_poi_to_this_person/to_messages and seconf, from_this_person_to_poi/from_messages. we will then add these two features to the list of features.

In [102]:
```
to_msg = comm_to_list(enron_no_outliers)[1]
from_msg = comm_to_list(enron_no_outliers)[2]
to_poi = comm_to_list(enron_no_outliers)[3]
from_poi = comm_to_list(enron_no_outliers)[4]


from_poi_ratio = []
to_poi_ratio = []
for f,t in zip(from_poi, to_msg):
    if not t == 0: q= f / t
    else: q = 0
    from_poi_ratio.append(q)

for t,f in zip(to_poi, from_msg):
    if not f == 0: w= t / f
    else: w = 0
    to_poi_ratio.append(w)
```

In [108]:
```
#make a copy of the data and name it "my_dataset"
my_dataset = enron_no_outliers
#add the features to the my_dataset
c = -1
for i in my_dataset:
    c += 1
    my_dataset[i].update({"from_poi_ratio":from_poi_ratio[c]})
    my_dataset[i].update({"to_poi_ratio": to_poi_ratio[c]})

features_list = ["poi", "from_poi_ratio", "to_poi_ratio"]
data = featureFormat(my_dataset, features_list,remove_NaN=True, remove_all_zeroes=False, remove_any_zeroes=False)
plt.rcParams['figure.figsize'] = (6, 6)
# substtude "NaN" values with 0
for d in data:
    if d[1] == "NaN": d[1] = 0
    if d[2] == "NaN": d[2] = 0
    from_poi = d[1]
    to_poi = d[2]
    plt.scatter(from_poi,to_poi)
plt.xlabel("emails received from POI")
plt.ylabel("emails sent to POI")
plt.title("emails received and sent to POI")
```

Out[108]: <matplotlib.text.Text at 0x1104fe80>




## Extract features and labels from dataset

In [132]:
```
### split into labels and features (this line assumes that the first
### feature in the array is the label, which is why "poi" must always
### be first in features_list

feature_list = list(set([i for i in my_dataset.values()[1].keys()]) - set(['from_poi_to_this_person','from_this_pers
on_to_poi',]))
feature_list[0], feature_list[16] = feature_list[16], feature_list[0]
print feature_list
labels, features = targetFeatureSplit(data)

# defining the training and testing sets
from sklearn import cross_validation
features_train, features_test, labels_train, labels_test = cross_validation.train_test_split(features,
labels, test_size=0.4, random_state=0)

# select features based on PCA
def doPCA():
    from sklearn.decomposition import PCA
    pca = PCA(n_components = 2)
    pca.fit(data)
    return pca
pca = doPCA()
# variance explained by first and second components respectively
print pca.explained_variance_ratio_

first_pc = pca.components_[0]
second_pc = pca.components_[1]
pca.fit(features_train)
# transform the train and test data into a new subspace via PCA
new_train = pca.transform(features_train)
new_test = pca.transform(features_test)
```

```
['poi', 'to_messages', 'from_poi_ratio', 'deferral_payments', 'total_payments', 'exercised_stock_options', 'bonus',
 'director_fees', 'restricted_stock_deferred', 'restricted_stock', 'total_stock_value', 'expenses', 'loan_advances',
 'from_messages', 'other', 'to_poi_ratio', 'salary', 'long_term_incentive', 'shared_receipt_with_poi', 'email_addres
s', 'deferred_income']
[ 0.77238344  0.2202557 ]
```

In [133]:
```
# try different classifiers on new_train and new_test
#selecting best parameters for Naive Bayse using cross_validation
from time import time
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

t0= time()
clf = GaussianNB()
clf.fit(new_train, labels_train)
y_pred = clf.predict(new_test)

print classification_report(labels_test, y_pred)
print "Naive Bayse algorithm time:{0}s".format(round(time()-t0, 3))
```

```
             precision    recall  f1-score   support

        0.0       0.91      0.92      0.91        52
        1.0       0.00      0.00      0.00         5

avg / total       0.83      0.84      0.83        57

Naive Bayse algorithm time:0.004s
```

In [134]:
```
#SVM
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV


t0= time()
parameters = {'kernel':('linear','rbf','poly'), 'C': range(1,200,20), 'gamma': [0.0001, 0.001, 0.005, 0.01, 0.1,0.5,
 1,2,10]}
svr = SVC()
gs = GridSearchCV(svr,parameters)
gs.fit(new_train, labels_train)
clf = gs.best_estimator_

y_pred = clf.predict(new_test)

print classification_report(labels_test, y_pred)
print "SVM algorithm time:{0}s".format(round(time()-t0, 3))
```

```
             precision    recall  f1-score   support

        0.0       0.91      1.00      0.95        52
        1.0       0.00      0.00      0.00         5

avg / total       0.83      0.91      0.87        57

SVM algorithm time:4.229s
```

In [135]:
```
#Adaboost
from sklearn.ensemble import AdaBoostClassifier
import numpy as np
t0= time()
parameters = {'n_estimators':range(10,100,10), 'learning_rate': [0.1,0.3,0.5,0.7,0.8,0.9]}
bdt = AdaBoostClassifier()
gs = GridSearchCV(bdt,parameters)
gs.fit(new_train, labels_train)
clf = gs.best_estimator_
pred = clf.predict(new_test)
print classification_report(labels_test, pred)
print "adaboost algorithm time:{0}s".format(round(time()-t0, 3))
```

```
             precision    recall  f1-score   support

        0.0       0.91      0.92      0.91        52
        1.0       0.00      0.00      0.00         5

avg / total       0.83      0.84      0.83        57

adaboost algorithm time:34.342s
```

In [175]:
```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
t0= time()
parameters = {'n_estimators':range(10,50,10), 'min_samples_split': [2,3,5],
              'min_impurity_split' : [1e-10,1e-15,1e-18],'warm_start' : ['TRUE']}
rnf = RandomForestClassifier()
gs = GridSearchCV(rnf,parameters)
gs.fit(new_train, labels_train)
clf = gs.best_estimator_
pred = clf.predict(new_test)
print classification_report(labels_test, pred)
print "randomforest algorithm time:{0}s".format(round(time()-t0, 3))
```

```
             precision    recall  f1-score   support

        0.0       0.92      0.88      0.90        52
        1.0       0.14      0.20      0.17         5

avg / total       0.85      0.82      0.84        57

randomforest algorithm time:29.837s
```

In [176]:
```
# get the parameters of the best model for Random Forest
clf.get_params()
```

Out[176]:
```
{'bootstrap': True,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'min_impurity_split': 1e-18,
 'min_samples_leaf': 1,
 'min_samples_split': 3,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 20,
 'n_jobs': 1,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': 'TRUE'}
```

In [177]:
```
pickle.dump(clf, open("my_classifier.pkl", "w") )
pickle.dump(my_dataset, open("my_dataset.pkl", "w") )
pickle.dump(features_list, open("my_feature_list.pkl", "w") )
```

# Discussion

In this project I intended to predict the Points of Interests based on a number of features extracted from the Enron email dataset. I took th efollowing steps to accomplish this project: first, I identified a number of outliers in the dataset and removed them. Next, I created two features and added to the dataset as the new features , which represented the ratio of emails sent and received from the point of interest to an individual and I found thiese features to be more informative. After adding these features, I ended up with a high-dimensional dataset which is of course prone to overfitting. In order to remedy this, I conducted a PCA and selected only the two first components and these two components alone explained approximately 99.2 % of the variance in the dataset. Using the first two components, I tried a number of different classifiers on this dataset. First, the Gaussian Naive Base model which resulted in an overall accuracy of 0.83 (recall:0.84,percision:0.83). This model did not do a good job identifying the points of interest and did a very good job in identifying non-POI individuals.

Next I tried a number of other classifiers and simultanioulsy tuned them using the GridSearchCV() command which enabled me to easily try a number of parameters for each classifier and choose the best parameters for each. Although the overall accuracy of SVM is higher than others (i.e. 87 %) this algorithm, same as Naive Basye does a bad job in predicting labels with 1.0 value. The random forest, however, seems to perfom better in this respect, although the overall accuracay is 84%.