In [52]:
```
import sys
print(sys.version)
import pandas as pd
import math
import numpy as np
import matplotlib.pyplot as plt
import json
from datetime import datetime
from scipy.spatial.distance import pdist,squareform,cdist
import subprocess
%matplotlib inline
pd.set_option('display.max_columns', 100)
pd.set_option('display.max_rows', 300)
```

```
3.3.5 |Anaconda 1.9.2 (x86_64)| (default, Mar 10 2014, 11:22:25)
[GCC 4.0.1 (Apple Inc. build 5493)]
```

Первая часть. Распознование сущностей на данных из russian-ner

In [53]:
```
def calc_border_measures_class(truth, pred,class_start, class_in):
    tp = fp = tn = fn = 0
    i = 0
    while i < len(truth):
        state = truth[i]
        if state == class_start:
            #не обнаружили начала
            if pred[i] != class_start:
                fn += 1
                i += 1
                while truth[i] == class_in and i < len(truth):
                    #если обнаружили начала там, где начало быть не долж
но
                    if pred[i] == 'B-ORG' or pred[i] == 'B-PER':
                        fp += 1
                    i += 1
                continue
            else:
                i += 1
                tp += 1
                while truth[i] == class_in and i < len(truth):
                    #не распознали до конца
                    if pred[i] != class_in:
                        tp -= 1
                        fn += 1
                        while truth[i] == class_in and i < len(truth):
                            if pred[i] == 'B-ORG' or pred[i] == 'B-PER'
:
                                fp += 1
                            i += 1
                        continue
                    i += 1
                continue
        else:
            if pred[i] == state:
                tn += 1
            else:
                fp += 1
            i += 1
            continue


    accuracy =  (tp + tn) / (tp + tn + fp + fn)
    recall = tp / (tp + fn)
    fm = 2 * accuracy * recall / (accuracy + recall)
    precision = tp / (tp + fp) if (tp + fp) != 0 else 0
    return pd.Series([accuracy, recall, fm,precision], index = ["accura
cy","recall","Fm","precision"])
```

In [54]:
```
def calc_border_measures(truth, pred):
    tp = fp = tn = fn = 0
    i = 0
    while i < len(truth):
        state = truth[i]
        if state == 'O':
            if pred[i] == 'O':
                tn += 1
            else:
                fp += 1
            i += 1
            continue
        if state == 'B-ORG':
            #не обнаружили начала
            if pred[i] != 'B-ORG':
                fn += 1
                i += 1
                while truth[i] == 'I-ORG' and i < len(truth):
                    #если обнаружили начала там, где начало быть не долж
но
                    if pred[i] == 'B-ORG' or pred[i] == 'B-PER':
                        fp += 1
                    i += 1
                continue
            else:
                i += 1
                tp += 1
                while truth[i] == 'I-ORG' and i < len(truth):
                    #не распознали до конца
                    if pred[i] != 'I-ORG':
                        tp -= 1
                        fn += 1
                        while truth[i] == 'I-ORG' and i < len(truth):
                            if pred[i] == 'B-ORG' or pred[i] == 'B-PER'
:
                                fp += 1
                            i += 1
                        continue
                    i += 1
                continue
        if state == 'B-PER':
            #не обнаружили начала
            if pred[i] != 'B-PER':
                fn += 1
                i += 1
                while truth[i] == 'I-PER' and i < len(truth):
                    #если обнаружили начала там, где начало быть не долж
но
                    if pred[i] == 'B-ORG' or pred[i] == 'B-PER':
                        fp += 1
                    i += 1
                continue
            else:
                i += 1
                tp += 1
                while truth[i] == 'I-PER' and i < len(truth):
                    #не распознали до конца
                    if pred[i] != 'I-PER':
                        tp -= 1
                        fn += 1
                        while truth[i] == 'I-PER' and i < len(truth):
                            if pred[i] == 'B-ORG' or pred[i] == 'B-PER'
:
                                fp += 1
                            i += 1
                        continue
                    i += 1
```

In [55]:
```
def calc_measures(truth, pred):
    truth_true = set(truth[truth].index)
    truth_false = set(truth[truth == False].index)
    pred_true = set(pred[pred].index)
    pred_false = set(pred[pred == False].index)
    tp = len(truth_true.intersection(pred_true))
    fp = len(pred_true - truth_true)
    tn = len(truth_false.intersection(pred_false))
    fn = len(pred_false - truth_false)
    accuracy =  (tp + tn) / (tp + tn + fp + fn)
    recall = tp / (tp + fn)
    fm = 2 * accuracy * recall / (accuracy + recall)
    precision = tp / (tp + fp) if (tp + fp) != 0 else 0
    return pd.Series([accuracy, recall, fm,precision], index = ["accura
cy","recall","Fm","precision"])
```

In [56]:
```
dir_path = "russian-ner/experiment-1/"
```

In [57]:
```
def calc_table(filename = dir_path + "result"):
    df = pd.read_csv(filename,sep="\t",header=None)
    df = df.dropna(axis = 0,subset = [3])
    classes = set(df[df.shape[1]-2])
    results = pd.DataFrame()
    for label in classes:
        truth = df[df.shape[1]-2] == label
        pred = df[df.shape[1]-1] == label
        results[label] = calc_measures(truth,pred)
    results["borders"] = calc_border_measures(df[7].values,df[8].values)
    results["border-org"] = calc_border_measures_class(df[7].values,df[8
].values,"B-ORG","I-ORG")
    results["border-per"] = calc_border_measures_class(df[7].values,df[8
].values,"B-PER","I-PER")
    return results
```

Используем crf++. Используемая модель зависит от параметра → хотим найти параметр, который будет хорошо работать — будем искать с помощью cross-validation

In [58]:
```
cv_folds = 5
```

In [59]:
```
learn_prefix = "ru_corpus_learn"
learn_name = "ru_corpus_learn"
test_name = "ru_corpus_test"
```

In [60]:
```
learn = lambda i: "{}-{}-{}".format(dir_path + learn_prefix,i,"learn")
test = lambda i:  "{}-{}-{}".format(dir_path + learn_prefix,i,"test")
```

In [61]:
```
def run_crf(c = 1.0):
    crf_learn = "crf_learn -c {} template {} {}".format(c,dir_path+learn
_name,dir_path + "model")
    crf_test = "crf_test -m {} {} > {}result".format(dir_path + "model",
dir_path + test_name,dir_path)
    subprocess.call(crf_learn, shell=True)
    subprocess.call(crf_test, shell=True)
    return calc_table("{}result".format(dir_path))
```

In [62]:
```
def run_crf_fold(i,c = 1.0):
    crf_learn = "crf_learn -c {} template {} {}".format(c,learn(i),dir_p
ath + "model")
    crf_test = "crf_test -m {} {} > {}result-{}".format(dir_path + "mode
l",test(i),dir_path,i)
    subprocess.call(crf_learn, shell=True)
    subprocess.call(crf_test, shell=True)
    return calc_table("{}result-{}".format(dir_path,i))
```

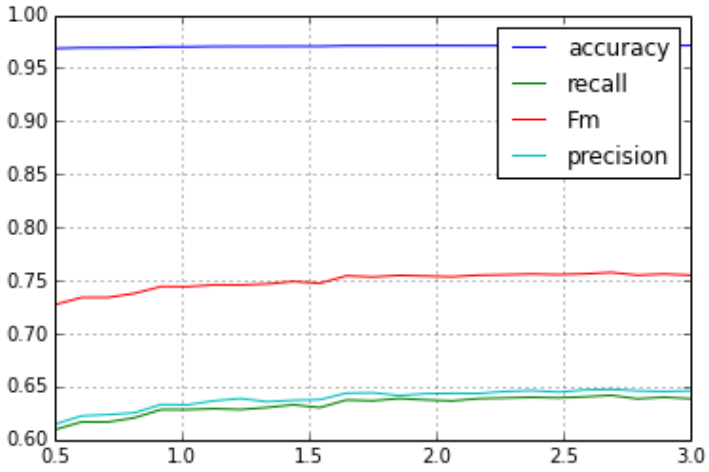
In [63]:
```
def cv_parametr(c):
    result = pd.DataFrame()
    for i in range(0, cv_folds):
        #print(i)
        result[i] = run_crf_fold(i,c).mean(axis=1)
    return result.mean(axis=1)
```

In [64]:
```
params = np.linspace(5e-1,3,25)
params.sort()
```

In [65]:
```
results = pd.DataFrame()
for c in params:
    results[c]  = cv_parametr(c)
```

In [66]:
```
results.T.plot()
```

Out[66]: <matplotlib.axes.AxesSubplot at 0x1090a49d0>



Borders — качество выявления границ сущностей Border-org,border-per — соответсвенно то, насколько качественно выделяются границы сущностей для организаций и людей

In [67]:
```
run_crf(2)
```

Out[67]:

	B-PER	O	I-ORG	B-ORG	I-PER	borders	border-org	border-
accuracy	0.990868	0.962186	0.984732	0.983305	0.994435	0.968027	0.967685	0.95802
recall	0.555556	0.991377	0.471591	0.487685	0.870370	0.464789	0.433498	0.54321
Fm	0.711942	0.976563	0.637758	0.652000	0.928276	0.628033	0.598764	0.69330
precision	0.616438	0.968557	0.855670	0.883929	0.594937	0.670051	0.453608	0.14715

4 rows × 8 columns

Как видно — организации узнаем точнее, чем имена. Имена узнаем не очень точно — вероятно потому, что для имен имеет смысл иметь словарик и тогда качество улучшится, а текущие вычисляемые признаки ориетированны больше на организации (много больших букв, английские буквы, etc)

Распознование именных сущностей с информацией из romip2005-facts

Добавили конвертер с эвристиками — формат файла плохой, эвристики работать будут плохо за счет плохой разметки firstText & secondText Эвристики нормально работать не будут — будет куча ложных срабатываний Пример 1: Порядок содержимого фактов не фиксирован Пример 2: Отличная организация Президентская Администрация — от имени ничем не отличается и либо вообще не можем добывать пользователей, либо будет куча false positive Но раз так просят конвертор, то посмотрим, что с ним получилось в этот раз

In [71]:
```
dir_path = "romip2005-facts/"
learn_prefix = "learn"
learn_name = "learn"
test_name = "test"
```

In [76]:
```
run_crf(2)
```

Out[76]:

	B-PER	O	I-ORG	B-ORG	I-PER	borders	border-org	border-
accuracy	0.988156	0.945491	0.983876	0.978739	0.992437	0.964907	0.957724	0.95125
recall	0.037037	0.997844	0.403409	0.300493	0.055556	0.225352	0.300493	0.03703
Fm	0.071398	0.970962	0.572203	0.459813	0.105221	0.365372	0.457455	0.07129
precision	0.375000	0.946262	0.898734	0.897059	0.600000	0.780488	0.293269	0.01136

4 rows × 8 columns

Ну что — эвристики так себе, только портята датасет. Качество только ухудшилось. Ну и с выявлением личностей все плохо как раз таки из-за того, что их из данных Romip совсем не определить то, что сущность является личностью — если выявлять организации можно, например, по кавычкам, то в обратную сторону это не работает — организация может быть и без кавычек…

In []:
```
```