In [14]:
```
import numpy
from collections import defaultdict
```

In [15]:
```
train_file = open("pa4train.txt", "r")
train = []
for line in train_file:
    spl = line.split(' ')
    x = spl[0]
    y = 0
    if spl[1][0] == '+':
        y = 1
    else :
        y= -1
    train.append([x,y])
test_file = open("pa4test.txt", "r")
test = []
for line in test_file:
    spl = line.split(' ')
    x = spl[0]
    y = 0
    if spl[1][0] == '+':
        y = 1
    else :
        y= -1
    test.append([x,y])
```

In [104]:
```
def string_kernel(s, t, p):
    count = 0
    s_dict = defaultdict(int)
    t_dict = defaultdict(int)
    for ind_s in range(len(s)-p+1):
        s_dict[s[ind_s:ind_s+p]] += 1
    for ind_t in range(len(t)-p+1):
        t_dict[t[ind_t:ind_t+p]] += 1

    for k, v in s_dict.items():
        if k in t_dict:
            count += v*t_dict[k]

    return count
```

In [82]:
```
def m_p(s, t, p):
    count = 0
    s_dict = defaultdict(int)
    t_dict = defaultdict(int)
    for ind_s in range(len(s)-p+1):
        s_dict[s[ind_s:ind_s+p]] += 1
    for ind_t in range(len(t)-p+1):
        t_dict[t[ind_t:ind_t+p]] += 1

    for k, v in s_dict.items():
        if k in t_dict:
            count += 1

    return count
```

In [71]:
```
def percepie(p):
    s = []
    for ind, point in enumerate(train):
        num = 0
        for prev in s:
            num += point[1]*train[prev][1] * string_kernel(point[0], train[prev][
        if num <= 0:
            s.append(ind)
        if ind % 100 == 0:
            print(ind)
    return s
```

In [84]:
```
def percepie_m(p):
    s = []
    for ind, point in enumerate(train):
        num = 0
        for prev in s:
            num += point[1]*train[prev][1] * m_p(point[0], train[prev][0], p)
        if num <= 0:
            s.append(ind)
        if ind % 100 == 0:
            print(ind)
    return s
```

In [89]:
```
def test_perceptron(t, s, p):
    preds = []
    correct = 0
    for point in t:
        pred = 0
        num = 0
        for prev in s:
            num += train[prev][1] * string_kernel(point[0], train[prev][0], p)
        if num <= 0:
            pred = -1
        else:
            pred = 1
        preds.append(pred)
        if pred == point[1]:
            correct += 1
        if len(preds) % 100 == 0:
            print(len(preds))
    return correct, preds
```

In [90]:
```
def test_perceptron_m(t, s, p):
    preds = []
    correct = 0
    for point in t:
        pred = 0
        num = 0
        for prev in s:
            num += train[prev][1] * m_p(point[0], train[prev][0], p)
        if num <= 0:
            pred = -1
        else:
            pred = 1
        preds.append(pred)
        if pred == point[1]:
            correct += 1
        if len(preds) % 100 == 0:
            print(len(preds))
    return correct, preds
```

In [91]:
```
res = []
for p in range(3,6):
    print("p =", p)
    w = percepie(p)
    train_ret = test_perceptron(train, w, p)
    test_ret = test_perceptron(test, w, p)
    x = (len(train)-train_ret[0])/len(train)
    y = (len(test)-test_ret[0])/len(test)
    print("training error for p =", p, "-", x)
    print("test error for p =", p, "-", y)
    res.append([x,y])
```

...

In [92]:
```
res
```

Out[92]:
```
[[0.012396694214876033, 0.040897097625329816],
 [0.006887052341597796, 0.026385224274406333],
 [0.006887052341597796, 0.03430079155672823]]
```

In [93]:
```
res_m = []
for p in range(3,6):
    print("p =", p)
    w = percepie_m(p)
    train_ret = test_perceptron_m(train, w, p)
    test_ret = test_perceptron_m(test, w, p)
    x = (len(train)-train_ret[0])/len(train)
    y = (len(test)-test_ret[0])/len(test)
    print("training error for p =", p, "-", x)
    print("test error for p =", p, "-", y)
    res_m.append([x,y])
```

...

In [95]:
```
res_m
```

Out[95]:
```
[[0.012672176308539946, 0.05408970976253298],
 [0.00743801652892562, 0.029023746701846966],
 [0.006887052341597796, 0.03430079155672823]]
```

In [108]:
```
from collections import Counter
def most_c(s, p):
    substrings = Counter()

    for prev in s:
        strng = train[prev][0]
        for ind_s in range(len(strng)-p+1):
            substrings[strng[ind_s:ind_s+p]] += train[prev][1]


    return substrings.most_common(2)
```

In [106]:
```
w = percepie(5)
```

...

In [109]:
```
x = most_c(w, 5)
```

In [110]:
```
x
```

Out[110]:
```
[('WDTAG', 3), ('DTAGQ', 3)]
```

```
1.
    training error for p = 3 - 0.012396694214876033
    test error for p = 3 - 0.040897097625329816
    training error for p = 4 - 0.006887052341597796
    test error for p = 4 - 0.026385224274406333
    training error for p = 5 - 0.006887052341597796
    test error for p = 5 - 0.03430079155672823

2.
    training error for p = 3 - 0.012672176308539946
    test error for p = 3 - 0.05408970976253298
    training error for p = 4 - 0.00743801652892562
    test error for p = 4 - 0.029023746701846966
    training error for p = 5 - 0.006887052341597796
    test error for p = 5 - 0.03430079155672823

3. The 2 most common substrings are 'WDTAG' and 'DTAGQ'
```