6章ドキュメントフィルタリング（2）

サ　ミンソン

フィッシャー法

- スパムのフィルターで役に立ち、非常に正確な結果を生み出す
- 単純ベイズフィルタ：ドキュメントの確率を算出するために特徴の確率たちを利用
- フィッシャー法：ドキュメント中のそれぞれの特徴のあるカテゴリでの確率たちを算出し、それらをまとめた確率の集合がランダムな集合と比較して高いか低いかをテストする

特徴たちのカテゴリの確率

- 単純ベイズフィルタ:ドキュメント全体の確率を得るために全てのPr(特徴　|　カテゴリ)をつなぎ合わせ、最後にそれを反転

- フィッシャー法:特定の特徴を含むドキュメントが、与えられたカテゴリに属する確率を計算

続き

Pr(カテゴリ　|　特徴)＝

（その特徴を持つドキュメントが
このカテゴリ中に存在する数）/（その特徴
を持つドキュメントの総数）

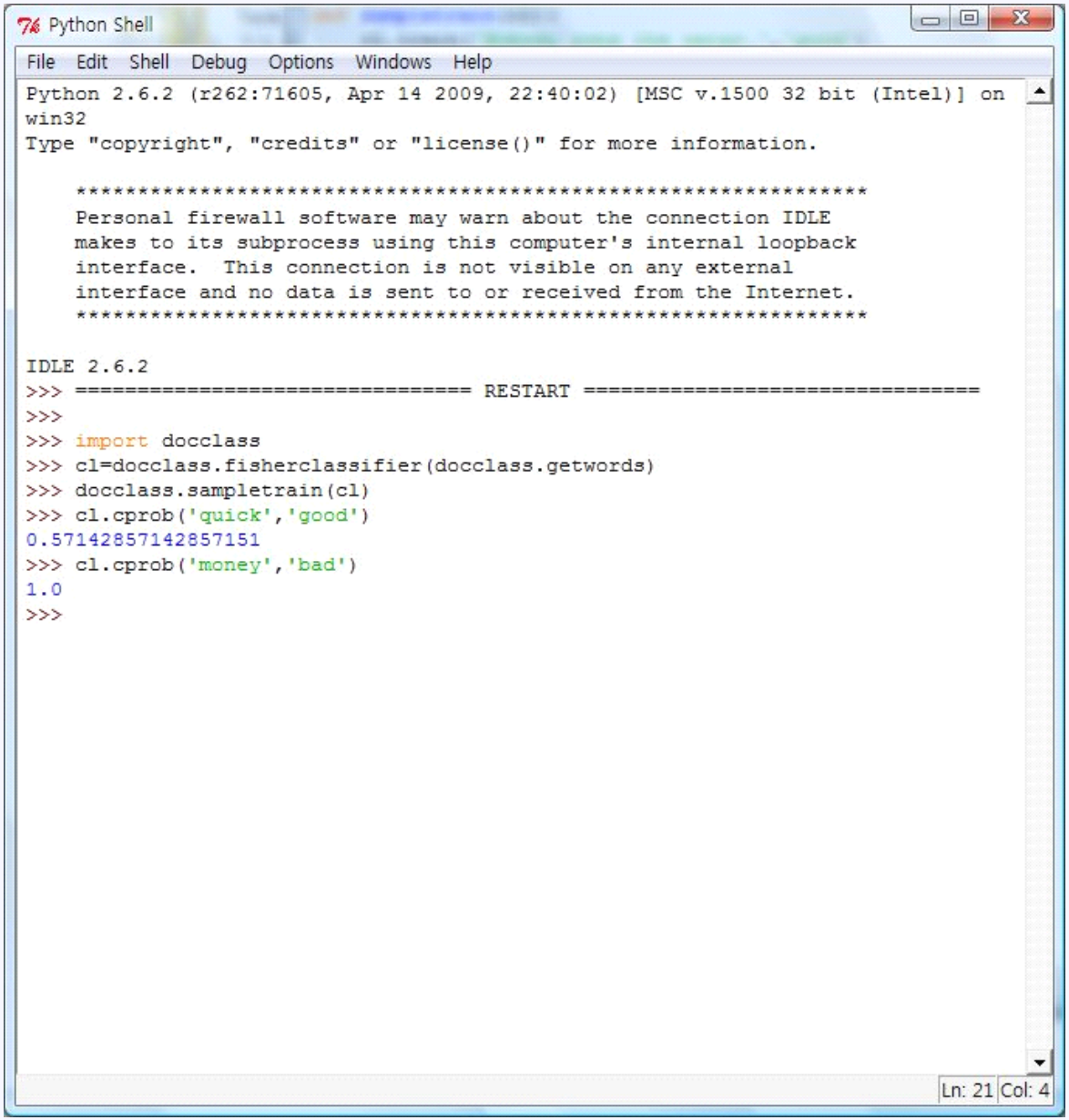
続き

```
def cprob(self,f,cat):
    # The frequency of this feature in this category
    clf=self.fprob(f,cat)
    if clf==0: return 0

    # The frequency of this feature in all the categories
    freqsum=sum([self.fprob(f,c) for c in self.categories()])

    # The probability is the frequency in this category divided by
    # the overall frequency
    p=clf/(freqsum)

    return p
```


Python Shell
File Edit Shell Debug Options Windows Help
Python 2.6.2 (r262:71605, Apr 14 2009, 22:40:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
****
IDLE 2.6.2
>>> ========== RESTART ==========
>>>
>>> import docclass
>>> cl=docclass.fisherclassifier(docclass.getwords)
>>> docclass.sampletrain(cl)
>>> cl.cprob('quick','good')
0.57142857142857151
>>> cl.cprob('money','bad')
1.0
>>>
Ln: 21 Col: 4

確率を統合する

- 全体の確率を見つけ出すため、それぞれの特徴たちの確率をまとめあげる必要がある

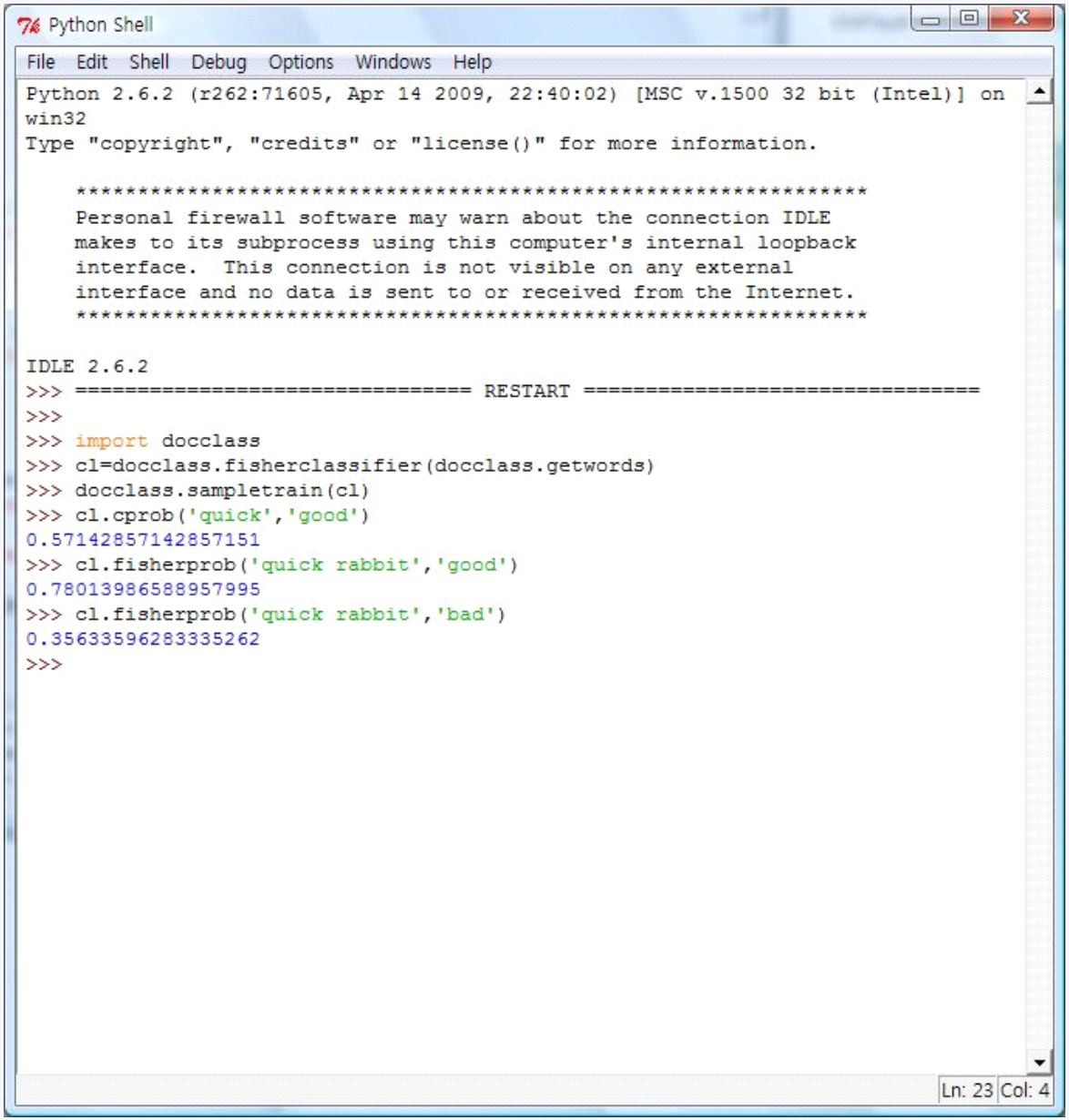
- 単純にすべてを掛け合わせることでカテゴリ同士で比較する際に使えるような確率を算出

確率を統合する

```
def fisherprob(self,item,cat):
  # Multiply all the probabilities together
  p=1
  features=self.getfeatures(item)
  for f in features:
    p*=(self.weightedprob(f,cat,self.cprob))

  # Take the natural log and multiply by -2
  fscore=-2*math.log(p)

  # Use the inverse chi2 function to get a probability
  return self.invchi2(fscore,len(features)*2)
```


Python Shell
File Edit Shell Debug Options Windows Help
Python 2.6.2 (r262:71605, Apr 14 2009, 22:40:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
****
IDLE 2.6.2
>>> ========== RESTART ==========
>>>
>>> import docclass
>>> cl=docclass.fisherclassifier(docclass.getwords)
>>> docclass.sampletrain(cl)
>>> cl.cprob('quick','good')
0.57142857142857151
>>> cl.fisherprob('quick rabbit','good')
0.78013986588957995
>>> cl.fisherprob('quick rabbit','bad')
0.35633596283335262
>>>
Ln: 23 Col: 4

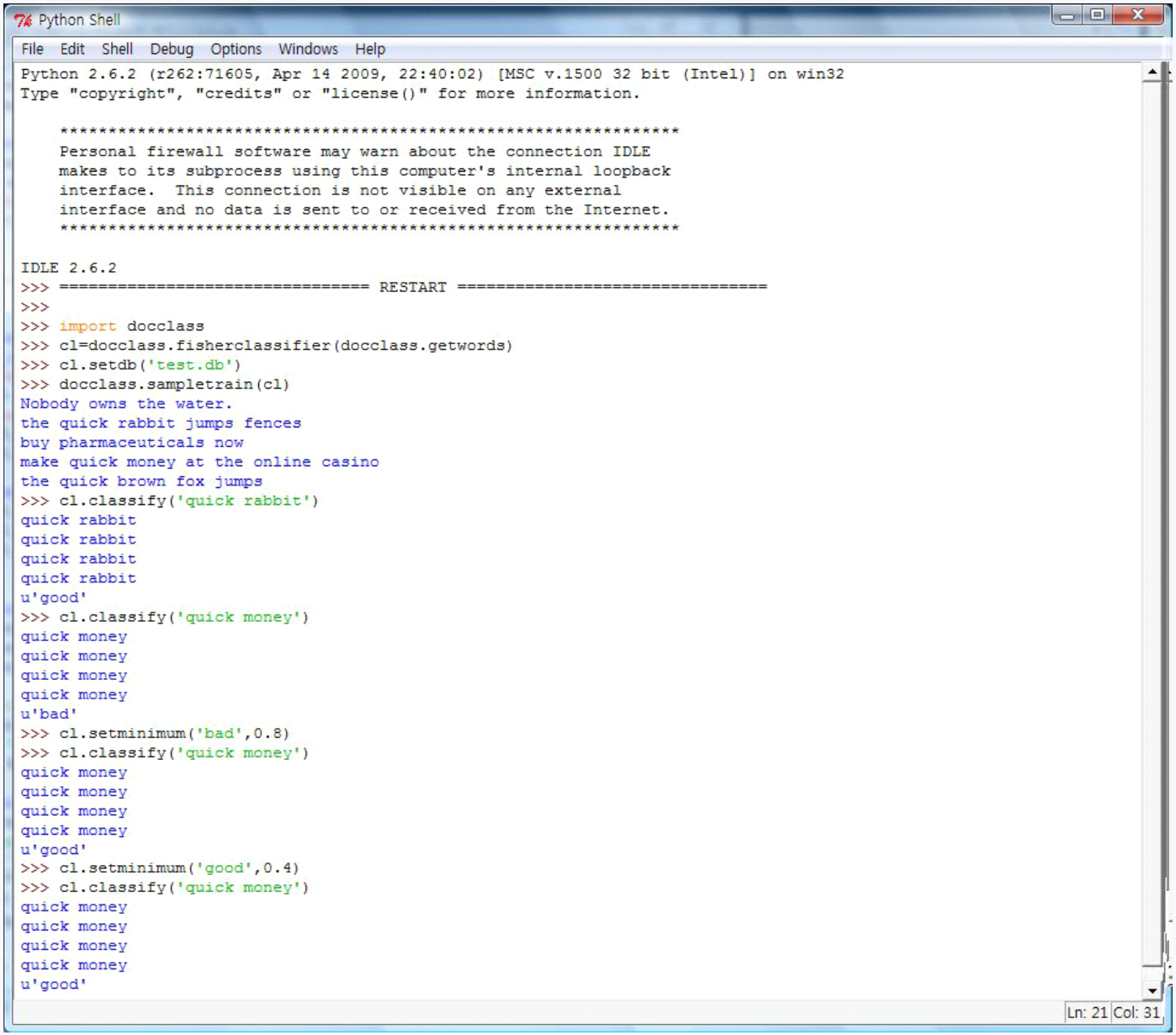
アイテムを分類する

- スパムフィルタでは、それぞれのカテゴリへの下限値を決定する。
- badカテゴリへの下限値はかなり高めに設定
  - 0.6ぐらい
- Goodカテゴリへの下限値
  - 0.2ぐらい

—> ・goodなメールが誤ってbadに分類される危険減らす
・多少のスパムが受信箱に振り分けられる

アイテムを分類

```
def classify(self,item,default=None):
  # もっとも良い結果を探してループする
  best=default
  max=0.0
  for c in self.categories():
    p=self.fisherprob(item,c)
    #下限値を超えていることを確認する
    if p>self.getminimum(c) and p>max:
      max=p
      best=c
  return best
```


Python Shell
File Edit Shell Debug Options Windows Help
Python 2.6.2 (r262:71605, Apr 14 2009, 22:40:02) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
****
IDLE 2.6.2
>>> ========== RESTART ==========
>>>
>>> import docclass
>>> cl=docclass.fisherclassifier(docclass.getwords)
>>> cl.setdb('test.db')
>>> docclass.sampletrain(cl)
Nobody owns the water.
the quick rabbit jumps fences
buy pharmaceuticals now
make quick money at the online casino
the quick brown fox jumps
>>> cl.classify('quick rabbit')
quick rabbit
quick rabbit
quick rabbit
quick rabbit
u'good'
>>> cl.classify('quick money')
quick money
quick money
quick money
quick money
u'bad'
>>> cl.setminimum('bad',0.8)
>>> cl.classify('quick money')
quick money
quick money
quick money
quick money
u'good'
>>> cl.setminimum('good',0.4)
>>> cl.classify('quick money')
quick money
quick money
quick money
quick money
u'good'
Ln: 21 Col: 31

トレーニング済みの分類器を保存する

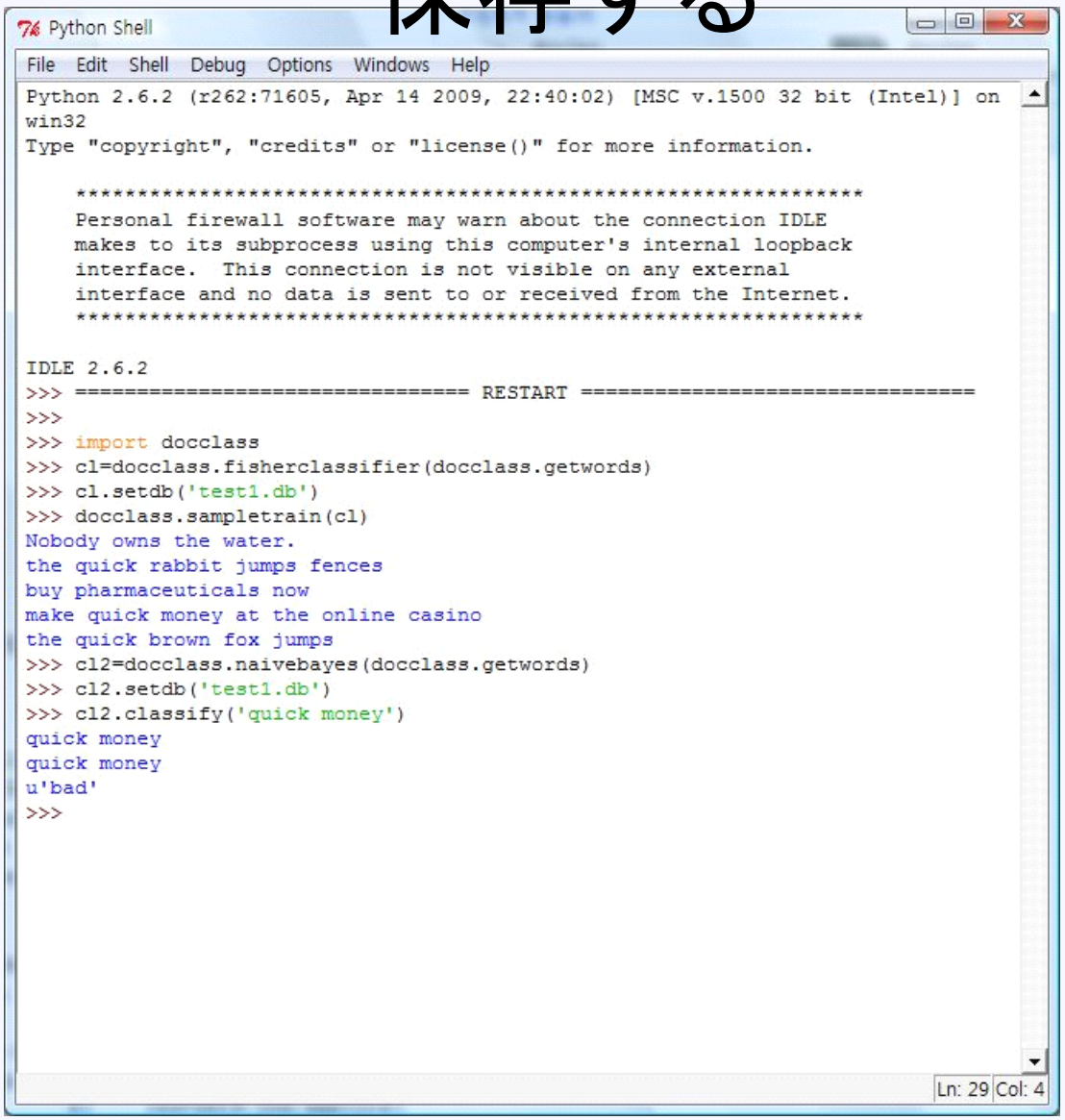
- 分類器がWebベースのアプリケーションの一部として利用される場合

  －ユーザがアプリケーションを利用している間のトレーニングデータは保存されるべき

  －次回ユーザがログインした時にはそれを復元

ー>SQLiteを利用する

トレーニング済みの分類器を保存する



blogフィードをフィルターする