breakciphers.py

Modulo python per il recupero della chiave di cifratura nel caso dell'utilizzo di cifrari a trasposizione o a sostituzione (Cesare)

breakciphers.py contiene funzioni che consentono il recupero di una probabile chiave di cifratura. È in grado di recuperare la chiave nel caso di testi cifrati derivanti dall'applicazione di cifrari a trasposizione (colonnare) e a sostituzione (con una sola chiave, come nel caso del cifrario di Cesare).

L'attacco si divide in 2 fasi:

- Rilevazione cifrario usato
- Recupero della chiave

In entrambe le fasi vengono applicati metodi statistici quali analisi delle frequenze degli n-grammi e il test di verifica dell'ipotesi (test χ^2). Le funzioni python che vengono usate per l'applicazione di questi metodi si trovano nel modulo frequtil.py .

Fase 1: rilevazione del cifrario usato

La rilevazione del cifrario utilizzato è eseguita con la funzione `transpose_or_caesar` che controlla per prima cosa se il testo in input contiene il carattere speciale utilizzato per il padding dal cifrario a trasposizione colonnare, rilevando come cifrario utilizzato il cifrario a trasposizione. Se il carattere per il padding non viene trovato, provvede a rilevare il cifrario eseguendo il test χ^2 sulla frequenza delle lettere e confrontando il risultato del test con il valore *alpha* dato in input. Questo valore deriva da una serie di test su un set di dati (frasi di un libro) su cui viene applicata la cifratura a trasposizione.

Il senso di questi test è che il cifrario a trasposizione non alterando la distribuzione delle frequenze delle singole lettere presenterà un valore per il test chi-quadrato molto piccolo rispetto allo stesso test eseguito su testi cifrati con un cifrario per sostituzione (che invece alterano la distribuzione).

Fase 2: recupero della chiave

Recupero della chiave per i cifrari a sostituzione (Cesare)

La modalità di recupero della chiave per i cifrari a sostituzione si basa sul test chi-quadrato. Questo test applicato alla distribuzione delle occorrenze delle lettere del messaggio dà una misura di quanto questa distribuzione si distacchi dalla distribuzione comune della lingua in cui è scritto il messaggio.
Si procede dunque a decifrare il testo con tutte le possibili chiavi (spazio di cardinalità l*en(alfabeto)*), e successivamente si ricerca quale dei possibili testi decifrati ha il valore del test chi-quadrato minore. Il testo con queste caratteristiche presenterà quindi il minor numero di differenze rispetto alla distribuzione delle frequenze della lingua in cui è scritto il testo originale.

Esempio di utilizzo

Di seguito viene mostrato un esempio di utilizzo del modulo per il recupero della chiave di un messaggio criptato con il cifrario di Cesare (i progressi dell'attacco vengono visualizzati grazie al parametro di input printprog)

```
>>> import breakciphers as bc
>>> ct = bc.oc.caesar_enc('attaccaallalba', 7)
>>> key = bc.break_caesar(ct, printprog = True)
key : 1 plaintext : gzzgiiggrrgrhg  chi_squared_val : 187.50079091
key : 2 plaintext : fyyfhhffqqfqgf  chi_squared_val : 547.03799326
key : 3 plaintext : exxeggeeppepfe  chi_squared_val : 6522.5435914
key : 4 plaintext : dwwdffddoodoed  chi_squared_val : 1107.7339826
key : 5 plaintext : cvvceeccnncndc  chi_squared_val : 64.183473398
key : 6 plaintext : buubddbbmmbmcb  chi_squared_val : 273.84508770
key : 7 plaintext : attaccaallalba  chi_squared_val : 35.572682781
key : 8 plaintext : zsszbbzzkkzkaz  chi_squared_val : 3886.9942504
key : 9 plaintext : yrryaayyjjyjzy  chi_squared_val : 8198.6143874
key : 10 plaintext : xqqxzzxxiixiyx chi_squared_val : 58330.784401
key : 11 plaintext : wppwyywwhhwhxw chi_squared_val : 11022.488733
key : 12 plaintext : voovxxvvggvgwv chi_squared_val : 6906.8878390
key : 13 plaintext : unnuwwuuffufvu chi_squared_val : 1154.3054931
key : 14 plaintext : tmmtvvtteeteut chi_squared_val : 64.81027643
key : 15 plaintext : sllsuussddsdts chi_squared_val : 64.184541081
key : 16 plaintext : rkkrttrrccrcsr chi_squared_val : 1613.8398857
key : 17 plaintext : qjjqssqqbbqbrq chi_squared_val : 3483.4377253
key : 18 plaintext : piiprrppaapaqp chi_squared_val : 107.46126699
key : 19 plaintext : ohhoqqoozzozpo chi_squared_val : 176.36082386
key : 20 plaintext : nggnppnnyynyon chi_squared_val : 443.98829899
key : 21 plaintext : mffmoommxxmxnm chi_squared_val : 14655.020818
key : 22 plaintext : leelnnllwwlwml chi_squared_val : 2332.8050691
key : 23 plaintext : kddkmmkkvvkvlk chi_squared_val : 14125.326501
key : 24 plaintext : jccjlljjuujukj chi_squared_val : 26862.016767
key : 25 plaintext : ibbikkiittitji chi_squared_val : 2347.8772569
key : 26 plaintext : haahjjhhsshsih chi_squared_val : 3115.4871898
plaintext: attaccaallalba
key is probably: 7
>>> print key
7
```

Recupero della chiave per i cifrari a trasposizione

La modalità di recupero della chiave per i cifrari a trasposizione si basa sull'analisi delle occorrenze dei quadrigrammi più comuni nella lingua in cui è scritto il testo originale.
L'approccio utilizzato è sempre quello brute-force. In questo modo avremo varie permutazioni delle lettere che formano il testo cifrato. Ad ogni iterazione (ogni presunta chiave) si calcola attraverso una *score function* un punteggio (*fit score*) che dipende dai quadrigrammi presenti nella permutazione calcolata (e dalle occorrenze degli stessi). Successivamente si seleziona il testo per il quale il punteggio è stato più elevato.
Il testo selezionato (e la chiave che lo ha generato) sarà con più probabilità il testo ricercato.

Esempio di utilizzo

Di seguito viene mostrato un esempio di utilizzo del modulo per il recupero della chiave di un messaggio criptato con il cifrario a trasposizione colonnare (i progressi dell'attacco vengono visualizzati grazie al parametro di input printprog)

```
>>> import breakciphers as bc
>>> ct = bc.oc.transpose_enc('attaccaallalba', 4)
>>> ct
'aclbtclataa*aal*'
>>> key = bc.break_transpose(ct, printprog = True)
key : 1 plaintext: aclbtclataa*aal*  score: 0.0036542804352
key : 2 plaintext: atcalab*tacalla*  score: 0.0029562718127
key : 3 plaintext: aca*cl**laa*bta*tal*  score: 0.0030383904742
key : 4 plaintext: attaccaallalba**  score: 0.042044754670
key : 5 plaintext: ablaa*ctaaa*lct*l*    score: 0.00049271196879
key : 6 plaintext: altltaalcbcaa*a*  score: 0.0027920344898
key : 7 plaintext: altltaalcbcaa*a*  score: 0.0027920344898
key : 8 plaintext: altltaalcbcaa*a*  score: 0.0027920344898
key is probably 4 and the message is: attaccaallalba**
>>> key
4
```

Percentuale di successo di attacchi effettuati con l'ausilio del modulo

Con lo script *statistical_test.py* si è provato a dare una stima della percentuale di successo dei metodi utilizzati.
Il modulo divide i file di testo nella cartella del modulo in frasi (utilizzando la funzione *split([sep [,maxsplit]])* classe *str*) e cifrandole scegliendo casualmente tra i due cifrari. Successivamente vengono applicate le funzioni del modulo a questo set di dati e vengono calcolate le volte in cui il recupero della chiave va a buon fine. Di seguito è listata una esecuzione dello script:

```
[antonio] $ python statistical_tests.py
Initializing the tests...
Testing phrases in Soffocare-20--20Chuck-20Palahniuk.txt
Testing phrases in promessi_sposi_cap1.txt
Testing phrases in mattiapascal.txt
Tested 9067 and found 8849 valid keys
Success percentage: 97.6%
```