SORBONNE UNIVERSITÉ

Automatisation de la cryptanalyse des cryptosystèmes classiques à l’aide d'algorithmes modernes

De Richemont Jules
Cohen-Solal Léa
Sakkour Zein

LU2IN013
Projet dirigé par Valérie Ménissier Morain

PLAN

I - Introduction

- *Éléments historiques*
- *Objectif du projet*

II - Chiffrement par substitution

- *Algorithme de Hill-climbing*
- *Fitness Function*
- *Automatisation*

III - Conclusion

Histoire du chiffrement

Figure n°1 : Scytale chez les Spartiate

- Les origines de l'utilisation de cryptographie remontent à l'Antiquité avec le plus vieux document chiffré datant du XVIème siècle avant notre ère.

- Plusieurs méthodes de chiffrement :
  - chiffrement par transposition (Scytale chez les Spartiate)
  - chiffrement par substitution mono-alphabétique (Atbash et César)
  - chiffrement par substitution poly-alphabétique (Vigenère)

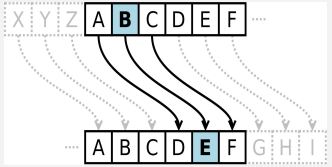


Figure n°2 : Illustration du code de César

Objectifs

- Étudier la méthode de chiffrement par substitution mono-alphabétique par l'intermédiaire de l’algorithme de Hill-climbing.
-
- Souligner les contrastes d'efficacité entres ces différents algorithmes

	Lettre en clair																										
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Lettre de la clé	Lettres chiffrées (au croisement de la colonne *Lettre en clair* et de la ligne *Lettre de la clé*)																										
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure n°3 : chiffre de Vigenere

Algorithme de Hill-climbing

- La méthode de hill-climbing (dite méthode d'escalade) est une méthode itérative locale qui vise à rechercher un optimum local.

- Son but : se rapprocher au maximum de la solution, même si on n’ y accède pas complètement

Evaluation d'un déchiffrement :
La fitness function

- Permet d'évaluer le bon déchiffrement d'un texte

- 2 types de fitness function implémentés :
  - la fitness n-gramme
  - la fitness de Pearson

Fitness n-gramme

- Analyse des fréquences de n-gramme avec n∈ {2, 3, 4, 5}

- Idée de la fonction :

  - Parcourir le texte déchiffré n lettres par n lettres
  - Additionner les fréquences des différentes combinaisons de n lettres
  - Obtenir un score de déchiffrement et le comparer au texte clair

Pseudo-code

```
def fitness_n_gramme ( texte_déchiffré , dictionnaire_n_gramme) :
    score = 0
    liste = liste(texte)
    x = 0
    Pour x allant de 0 à la (longueur du texte-n):
        score = score + fréquence_dictionnaire (liste[x : x + n])

    return score
```

Fitness Pearson

- Analyse des fréquences des mono-grammes

- Idée de la fonction :
  - Calculer la fréquence X_i de chaque lettre i dans le texte déchiffré
  - Calculer la fréquence Y_i de chaque lettre i dans le texte clair
  - Calculer le score r tel que 0 < r < 1

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

Figure 4 : Formule du coefficient de corrélation de Pearson

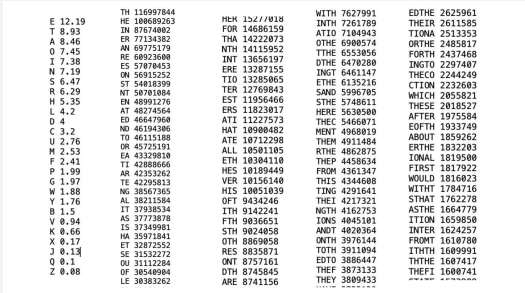
Génération des fréquences

Création des fréquences

- Script python permettant d'obtenir pour chaque n-gramme un fichier de fréquences de n-gramme

Dictionnaires des fréquences : lire-n gramme

- Fonction permettant de passer d'un fichier .txt à un dictionnaire python



	TH 116997844			
E 12.19	HE 100689263	HER 15277018	WITH 7627991	EDTHE 2625961
T 8.93	IN 87674002	FOR 14686159	INTH 7261789	THEIR 2611585
A 8.46	ER 77134382	THA 14222073	ATIO 7104943	TIONA 2513353
O 7.45	AN 69775179	NTH 14115952	OTHE 6900574	ORTHE 2485817
I 7.38	RE 60923600	INT 13656197	TTHE 6553056	FORTH 2437468
S 6.47	ES 57070453	ERE 13287155	DTHE 6470280	INGTO 2297407
N 7.19	ON 56915252	TIO 13285065	INGT 6461147	THECO 2244249
S 6.47	ST 54018399	TER 12769843	ETHE 6135216	CTION 2232603
R 6.29	NT 50701084	EST 11956466	SAND 5996705	WHICH 2055821
H 5.35	EN 48991276	ERS 11823017	STHE 5748611	THESE 2018527
L 4.2	AT 48274564	ATI 11227573	HERE 5630500	AFTER 1975584
D 4	ED 46647960	HAT 10900482	THEC 5466071	EOFTH 1933749
C 3.2	ND 46194306	ATE 10712298	MENT 4960019	ABOUT 1859262
U 2.76	TO 46115188	ALL 10501105	THEM 4911484	ERTHE 1832203
M 2.53	OR 45725191	ETH 10304110	RTHE 4862875	IONAL 1819500
F 2.41	EA 43329810	HES 10189449	THEP 4458634	FIRST 1817922
P 1.99	TI 42888666	VER 10156140	FROM 4361347	WOULD 1816023
G 1.97	AR 42353262	HIS 10051039	THIS 4344608	WITHT 1784716
W 1.88	TE 42295813	OFT 9434246	TING 4291641	STHAT 1762278
Y 1.76	NG 38567365	ITH 9142241	THEI 4217321	ASTHE 1664779
B 1.5	AL 38211584	FTH 9036651	NGTH 4162753	ITION 1659850
V 0.94	IT 37938534	STH 9024058	IONS 4045101	INTER 1624257
K 0.66	AS 37773878	OTH 8869058	ANDT 4028364	FROMT 1610780
X 0.17	IS 37349981	RES 8835871	ONTH 3976144	ITHTH 1609991
J 0.13	HA 35971841	ONT 8757161	TOTH 3911094	THTHE 1607417
Q 0.1	ET 32872552	DTH 8745845	EDTO 3886447	THEFI 1600741
Z 0.08	SE 31532272	ARE 8741156	THEF 3873133	
	OU 31112284		THEY 3809433	
	OF 30540904			
	LE 30383262			

Figure 5 : Extrait des fréquences des bi, tri, tétra, penta -gramme

Hill-Climbing

- Tester en boucle un grand nombre de clés potentielles et évaluer à chaque tour le score de la clé potentielle à l'aide d'une certaine fonction fitness, puis prendre, pour l'itération suivante, la clé au meilleur score

Pseudo- Code

```
def hillClimbing(fitness,texte chiffré,dictionnaire,NBITERGLOB,NBITERSTATIC,clé,fonction de comparaison):
    cle_parent = clé
    texte_déchiffré = decipher ( texte_chiffré, clé_parent)  #Déchiffrement du texte avec la clé initiale
    score_parent = fitness (texte dechiffré , dictionnaire) # Score du déchiffrement avec la fonction fitness de notre choix
    global = 0 et statique = 0
    Tant que global < NBITERGLOB et statique < NBITERSTATIC :
        clé_enfant = permutation de 2 lettres dans la clé_parent  #changement de clé de déchiffrement
        score_enfant = fitness (texte_déchiffré, dictionnaire)    # On déchiffre le texte avec une nouvelle clé
        if (score_enfant est meilleur que score_parent):           # Sera determiné par la fonction compare
            score_parent = score_enfant
            clé_parent = clé_enfant
            statique = 0
        else :
            statique += 1
        global += 1
        affichage du texte déchiffré obtenu, du texte clair et du nombre de bonnes lettres dans la clé_parent
    return cle_parent # clé gagnante
```

Optimisation des paramètres et statistiques

Conclusion empirique sur la fonction de hill-climbing :

- L'efficacité du déchiffrement dépend du n choisi et de la longueur du texte

lancement d'une batterie de tests sur la fonction de hill-climbing

- But : trouver les paramètres permettant de bons résultats rapidement

Fonctionnement du script

- Faire varier le nombre d'itérations globales et statiques pour obtenir un couple (globale, statique) optimal.

- 1000 < global < 10 000 / pas : 500
- 500 < statique < 4000 / pas 250
- 20 tests par couple
- Faire ces tests sur tous les n-gramme

Résultats texte court

Résultats bi-gramme

Iterations Globales	Iterations Locales	Clé	Score	Temps
2000	*750*	*22.25/26*	*39988.600*	*2.037*
2500	*1000*	*22.2/26*	*39992.033*	*2.417*
2500	*1250*	*22.9/26*	*40036.755*	*2.512*
3000	*2500*	*22.2/26*	*39999.609*	*3.082*
3500	*750*	*22.65/26*	*39979.500*	*2.364*
3500	*1000*	*22.6/26*	*39993.850*	*2.608*
3500	*1750*	*22.5/26*	*39996.806*	*3.392*
3500	*3000*	*23.1/26*	*40043.142*	*3.593*
4000	*1750*	*23.05/26*	*40029.701*	*3.460*
4000	*2250*	*22.3/26*	*39972.386*	*3.903*
4000	*2750*	*22.7/26*	*40008.763*	*4.056*
4000	*3000*	*22.6/26*	*39999.047*	*4.083*
4000	*3250*	*24.15/26*	*40112.429*	*4.099*
4500	*2750*	*23.05/26*	*40043.188*	*4.283*
4500	*3000*	*24.55/26*	*40118.422*	*4.504*
4500	*3500*	*24.05/26*	*40048.389*	*4.588*
4500	*4000*	*22.2/26*	*39947.538*	*4.609*
5000	*1750*	*22.0/26*	*39968.217*	*3.704*
5000	*3750*	*22.8/26*	*39975.115*	*5.058*
5000	*4000*	*22.9/26*	*39985.767*	*5.123*
5500	*750*	*24.2/26*	*40118.422*	*2.482*
5500	*1750*	*23.75/26*	*40116.268*	*3.528*
5500	*2000*	*24.45/26*	*40120.036*	*3.921*
5500	*2500*	*22.65/26*	*39993.034*	*4.698*
5500	*2750*	*23.15/26*	*40041.241*	*4.727*
5500	*3000*	*22.7/26*	*39972.830*	*5.027*
5500	*3250*	*23.15/26*	*40044.702*	*5.093*
5500	*3500*	*22.05/26*	*39972.692*	*5.132*
6000	*2250*	*23.15/26*	*40042.456*	*4.156*
6000	*3500*	*22.3/26*	*39968.496*	*5.415*
6000	*3750*	*22.8/26*	*40043.822*	*5.487*
6000	*4000*	*23.8/26*	*40080.056*	*5.786*

Resultats tri-gramme

Iterations Globales	Iterations Locales	Clé	Score	Temps
2500	*1750*	*22.7/26*	*34245.244*	*2.681*
2500	*2000*	*22.2/26*	*34218.207*	*2.678*
3000	*1250*	*23.9/26*	*34498.324*	*3.004*
3000	*2750*	*22.8/26*	*34277.278*	*3.214*
3500	*1000*	*23.55/26*	*34420.275*	*2.709*
3500	*1250*	*23.5/26*	*34486.696*	*3.094*
3500	*2250*	*22.75/26*	*34274.977*	*3.664*
3500	*3250*	*22.8/26*	*34258.847*	*3.749*
4000	*1500*	*22.7/26*	*34255.901*	*3.344*
4000	*3500*	*22.05/26*	*34031.815*	*4.279*
4500	*2250*	*23.2/26*	*34312.671*	*4.113*
4500	*3250*	*22.45/26*	*34247.870*	*4.789*
5000	*1000*	*22.05/26*	*34097.879*	*2.858*
5000	*1500*	*22.4/26*	*34290.460*	*3.365*
5000	*3250*	*23.8/26*	*34499.712*	*5.128*
5500	*750*	*24.45/26*	*34649.691*	*2.395*
5500	*1250*	*22.75/26*	*34278.783*	*3.221*
5500	*1750*	*22.8/26*	*34213.551*	*3.865*
5500	*2000*	*22.55/26*	*34275.536*	*4.183*
5500	*2250*	*22.95/26*	*34261.244*	*4.421*
5500	*4000*	*23.6/26*	*34495.810*	*5.758*
6000	*1250*	*23.5/26*	*34489.919*	*3.252*
6000	*2250*	*22.4/26*	*34272.127*	*4.306*

Résultats tétra-gramme

Iterations Globales	Iterations Locales	Clé	Score	Temps
2000	*750*	*22.5/26*	*28934.638*	*2.083*
2000	*1000*	*23.3/26*	*29284.407*	*2.203*
2000	*1250*	*22.45/26*	*29046.843*	*2.213*
2500	*1250*	*23.0/26*	*29202.284*	*2.698*
2500	*2250*	*21.7/26*	*28705.755*	*2.758*
3000	*1500*	*22.95/26*	*29332.017*	*3.174*
3000	*2500*	*21.55/26*	*28819.780*	*3.308*
3000	*2750*	*22.8/26*	*29234.546*	*3.315*
3500	*750*	*21.55/26*	*28672.206*	*2.589*
3500	*1250*	*23.05/26*	*29231.535*	*2.998*
4000	*3500*	*21.55/26*	*28799.692*	*4.404*
4500	*1000*	*21.8/26*	*28780.991*	*2.801*
4500	*1500*	*21.3/26*	*28764.088*	*3.330*
5000	*2000*	*23.85/26*	*29682.348*	*3.929*
5000	*2250*	*22.5/26*	*29192.210*	*4.323*
5500	*1750*	*23.85/26*	*29684.477*	*3.688*
6000	*3250*	*22.65/26*	*29257.430*	*5.403*
6000	*4000*	*22.1/26*	*28883.534*	*6.242*

Résultats penta-gramme

Iterations Globales	Iterations Locales	Clé	Score	Temps
2500	*2000*	*19.1/26*	*22210.145*	*2.965*
3000	*2000*	*19.35/26*	*22808.969*	*3.527*
3000	*2250*	*19.3/26*	*22838.095*	*3.507*
3500	*1500*	*19.5/26*	*22904.669*	*3.638*
3500	*2250*	*19.1/26*	*22801.731*	*4.035*
4000	*500*	*19.25/26*	*22784.414*	*2.054*
4000	*1500*	*19.45/26*	*22879.637*	*3.589*
4000	*1750*	*20.95/26*	*23605.558*	*4.080*
4000	*3250*	*19.2/26*	*22862.520*	*4.677*
4500	*1250*	*20.2/26*	*23460.553*	*3.448*
4500	*1500*	*19.0/26*	*22891.732*	*3.906*
4500	*2000*	*20.6/26*	*23627.060*	*4.337*
4500	*2250*	*19.35/26*	*22893.935*	*4.401*
4500	*2500*	*19.0/26*	*22927.946*	*4.571*
4500	*3250*	*19.45/26*	*22898.931*	*5.236*
4500	*3500*	*21.45/26*	*24243.010*	*5.262*
5000	*1000*	*20.15/26*	*23584.914*	*3.410*
5000	*1250*	*19.15/26*	*22857.979*	*3.308*
5000	*1500*	*20.05/26*	*23516.637*	*3.881*
5000	*2000*	*21.15/26*	*24287.161*	*4.509*
5500	*750*	*21.4/26*	*24055.284*	*2.857*
6000	*1250*	*20.0/26*	*23545.868*	*3.581*
6000	*1750*	*20.3/26*	*23379.277*	*3.963*
6000	*2250*	*19.45/26*	*22919.881*	*4.482*
6000	*3500*	*19.25/26*	*22858.833*	*5.909*
6000	*3750*	*19.0/26*	*22814.971*	*6.313*

Résultats texte long

Résultats bi-gramme

Iteration GLOBALE	Iteration LOCAL	Cle	SCORE	TIME
3000	*2250*	*24.65/26*	*133266.747*	*10.060*
3500	*1250*	*25.1/26*	*133300.840*	*9.828*
3500	*2000*	*25.9/26*	*133616.306*	*11.165*
3500	*3250*	*24.05/26*	*132928.374*	*11.732*
4000	*1250*	*24.2/26*	*133208.932*	*9.215*
4000	*2500*	*24.7/26*	*133167.713*	*12.979*
4000	*3750*	*24.05/26*	*133059.642*	*13.400*
4500	*1000*	*24.5/26*	*133120.530*	*8.422*
4500	*1500*	*24.6/26*	*133279.965*	*10.705*
4500	*1750*	*24.5/26*	*133048.501*	*11.836*
4500	*2500*	*24.35/26*	*133162.651*	*14.107*
5000	*2250*	*24.5/26*	*133103.568*	*13.293*
5000	*3000*	*26.0/26*	*133639.957*	*15.388*
5000	*3750*	*24.9/26*	*133355.754*	*16.541*
5500	*1750*	*24.6/26*	*133133.653*	*11.857*
5500	*2750*	*24.3/26*	*132975.793*	*15.081*
5500	*3000*	*25.5/26*	*133459.833*	*16.222*
5500	*4000*	*24.75/26*	*133220.516*	*18.126*
6000	*1000*	*24.75/26*	*133227.736*	*9.433*
6000	*1250*	*24.55/26*	*133180.663*	*10.377*
6000	*1500*	*25.4/26*	*133392.808*	*10.188*
6000	*1750*	*24.3/26*	*133159.829*	*11.636*
6000	*2500*	*25.65/26*	*133482.930*	*13.475*
6000	*3750*	*25.2/26*	*133324.888*	*18.083*
6500	*2500*	*24.35/26*	*133144.493*	*14.110*
6500	*3000*	*24.45/26*	*133068.335*	*16.312*
6500	*3500*	*24.15/26*	*132977.012*	*17.193*
7000	*1000*	*24.9/26*	*133283.122*	*8.722*
7000	*2750*	*24.15/26*	*133093.183*	*15.329*
7000	*3000*	*25.15/26*	*133302.686*	*16.248*
7500	*1000*	*24.05/26*	*133081.095*	*9.493*
7500	*1250*	*24.75/26*	*133166.217*	*9.594*
7500	*2000*	*24.2/26*	*133150.237*	*12.862*
7500	*2250*	*24.0/26*	*133051.330*	*12.655*

Resultats tri-gramme

Iterations Globales	Iterations Locales	Clé	Score	Temps
2000	*750*	*24.7/26*	*114374.509*	*6.855*
2000	*1000*	*24.7/26*	*114361.450*	*6.992*
2000	*1250*	*24.85/26*	*114776.080*	*7.016*
2000	*1750*	*24.0/26*	*113874.046*	*7.035*
3000	*2500*	*25.65/26*	*115274.971*	*10.538*
3500	*750*	*24.7/26*	*114824.535*	*7.532*
3500	*3000*	*24.9/26*	*114845.925*	*12.281*
4000	*1250*	*24.25/26*	*114336.447*	*9.384*
4000	*3000*	*24.85/26*	*114835.706*	*14.016*
4000	*3500*	*24.8/26*	*114895.254*	*14.016*
4500	*500*	*24.65/26*	*114261.730*	*8.018*
4500	*1000*	*24.8/26*	*114902.984*	*8.851*
4500	*2750*	*24.9/26*	*114883.472*	*14.558*
4500	*3000*	*24.25/26*	*114329.080*	*15.270*
5000	*750*	*25.25/26*	*114882.292*	*9.039*
5000	*1250*	*24.7/26*	*114726.016*	*10.027*
5000	*2250*	*24.9/26*	*114866.893*	*12.885*
5000	*4000*	*24.0/26*	*114171.043*	*17.534*
5500	*3000*	*25.45/26*	*115125.927*	*15.965*
6000	*750*	*24.3/26*	*114300.160*	*8.224*
6000	*3000*	*24.0/26*	*114099.985*	*16.967*
6500	*500*	*25.7/26*	*115547.727*	*6.747*
6500	*1500*	*24.8/26*	*114923.019*	*10.708*
6500	*2750*	*24.85/26*	*114923.441*	*15.275*
6500	*3750*	*24.75/26*	*114410.211*	*19.444*
7000	*1500*	*24.8/26*	*114904.337*	*10.579*
7000	*2250*	*24.8/26*	*114880.958*	*13.877*
7000	*2500*	*25.9/26*	*115607.459*	*14.349*
7500	*500*	*24.05/26*	*114320.316*	*6.711*
7500	*750*	*24.3/26*	*114262.740*	*8.379*
7500	*2000*	*24.65/26*	*114809.369*	*12.530*
8000	*750*	*24.55/26*	*114421.175*	*7.901*
8000	*1250*	*24.9/26*	*114859.097*	*10.076*
8000	*2750*	*26.0/26*	*115724.936*	*16.058*

Résultats tétra-gramme

Iterations Globales	Iterations Locales	Clé	Score	Temps
8000	*3000*	*24.8/26*	*114929.687*	*16.560*
8000	*3500*	*24.5/26*	*114351.949*	*17.710*
8000	*3750*	*26.0/26*	*115724.936*	*18.356*
8500	*1750*	*25.05/26*	*114909.573*	*11.438*
8500	*2750*	*25.4/26*	*115129.601*	*15.922*
8500	*3000*	*24.75/26*	*114873.000*	*16.385*
9000	*750*	*24.45/26*	*114350.001*	*7.471*
9000	*1250*	*24.6/26*	*114386.287*	*10.019*
9000	*1750*	*24.85/26*	*114924.414*	*11.936*
9500	*1000*	*24.85/26*	*114923.441*	*8.917*
9500	*3500*	*25.5/26*	*115144.665*	*17.931*
9500	*4000*	*24.9/26*	*114926.432*	*19.728*
10000	*500*	*25.9/26*	*115539.106*	*7.052*
10000	*1750*	*24.75/26*	*114638.554*	*12.600*
10000	*2000*	*24.9/26*	*114901.210*	*12.667*
10000	*2750*	*25.05/26*	*114948.405*	*15.896*
10000	*3500*	*24.8/26*	*114886.000*	*18.142*

Résultats penta-gramme

Iteration GLOBALE	Iteration LOCAL	Cle	SCORE	TIME
3000	*1250*	*23.45/26*	*81105.214*	*10.574*
3500	*1250*	*23.6/26*	*81388.341*	*11.025*
4000	*3000*	*24.8/26*	*83660.521*	*15.682*
4500	*2500*	*24.6/26*	*82713.218*	*16.528*
4500	*3250*	*24.75/26*	*83648.834*	*17.131*
5000	*750*	*23.45/26*	*81251.824*	*8.817*
5500	*2750*	*23.8/26*	*81392.992*	*17.462*
5500	*3750*	*24.7/26*	*83623.298*	*20.430*
6500	*3500*	*24.8/26*	*83653.384*	*20.823*
7000	*750*	*23.75/26*	*80862.902*	*8.802*
7500	*2250*	*23.6/26*	*81300.276*	*16.444*
7500	*3750*	*23.7/26*	*81330.953*	*22.243*
8000	*2250*	*24.9/26*	*83645.426*	*15.623*
8000	*2750*	*23.9/26*	*81272.716*	*17.462*
8500	*750*	*23.35/26*	*80424.116*	*9.444*
9500	*1250*	*23.65/26*	*81332.174*	*11.854*
9500	*2000*	*23.6/26*	*81191.380*	*14.141*
10000	*3750*	*23.7/26*	*81283.441*	*22.304*
10000	*4000*	*25.0/26*	*83662.785*	*21.764*

N-gramme optimales

	texte court	texte long
bi-gramme	rapidité +++ efficacité +	efficacité +
tri-gramme	rapidité ++ efficacité +++	rapidité++ efficacité ++
tétra-gramme	rapidité + efficacité +	rapidité + efficacité +++
penta-gramme	rapidité - efficacité -	rapidité - efficacité ++

Résultats Pearson

clé	score obtenu
5/26	*0.2918653*
8/26	*0.395423*
9/26	*0.35289*
11/26	*0.625217*
14/26	*0.7918653*
2/26	*0.995289*

Pourquoi fitness Pearson donne-t-elle des résultats décevants ?

- N’évalue pas l'enchaînement des caractères mais seulement la fréquence des caractères.
- Le score de cette fitness reflète simplement de la “ressemblance” à la langue
- Donne seulement un indice de clarté

Combinaison de n-gramme

Idée de la démarche

- Exécuter la fonction hill-climbing avec un n
- Sauvegarder la clé trouvée (clé n° 1)
- Exécuter une nouvelle fois la fonction de hill-climbing avec comme clé de départ la clé n° 1 (avec un n différent ou pas)

Résultats

- Combiner 2 fonctions fitness avec le même n-gramme
  → pas d'augmentation sur le score final.
- Combinaison (n, n+k) avec k>0
  → parfois une augmentation, mais pas systématiquement

combinaison	itérations globales	itérations statiques	moyenne lettres 1	moyenne lettres 2	moyenne score 1	moyenne score 2	Temps
(2,2)	1500	(750,500)	22.83/26	24.0/26	40115	40115	2.406
(2,3)	1500	(750,500)	20.3/26	25.1/26	40112	34685	2.688
(2,4)	1500	(1000,1000)	22.25/26	25.17/26	40121	30128	3.951
(2,5)	1000	(750,750)	18.8/26	25.2/26	40105	25941	2.603
(3,3)	1500	(1000,750)	23.8/26	25.2/26	34714	34714	3.134
(3,3)	1000	(1000,1000)	9.8/26	12.0/26	31963	31963	2.573
(3,4)	1500	(1250,750)	20.25/26	23.33/26	34355	29380	3.538
(3,5)	1500	(1250,750)	23.2/26	24.4/26	34714	34714	3.771
(4,4)	1500	(1000,1000)	23.8/26	24.8/26	30128	30128	3.507
(4,5)	3500	(1500,1500)	26.0/26	25.6/26	30128	26373	5.434
(4,5)	3500	(1500,1500)	26.0/26	25.6/26	30128	26373	5.434
(5,5)	2000	(1750,1750)	24.0/26	25.2/26	26373	26373	5.591
(5,5)	2000	(1500,1500)	15.8/26	15.2/26	20776	20776	5.520

Conclusion

- L'efficacité du déchiffrement par substitution mono-alphabétique dépend du choix des paramètres de la fonction de hill-climbing (fitness, n-gramme, itérations)

- Pas de résultats réellement concluants sur les combinaisons de n-gramme

MERCI DE NOUS AVOIR ÉCOUTÉS !!

Et merci à notre professeur encadrant Valérie Ménissier-Morain