4 Recognizing Plaintext: SINKOV's Log-Weight Test

The MFL-test is simple and efficient. SINKOV in [8] proposed a more refined test that uses the information given by all single letter frequencies, not just by separating the letters into two classes. We won't explore the power of this method but treat it only as a motivation for Section 5.

As in Section 1 we assign a probability p_s to each letter s of the alphabet Σ. We enumerate the alphabet as $(s_1, \ldots, s_n)$ and write $p_i := p_{s_i}$. For a string $a = (a_1, \ldots, a_r) \in \Sigma^r$ we denote by $N_i(a) = \#\{j \mid a_j = s_i\}$ the multiplicity of the letter s_i in a. Then for an n-tuple $k = (k_1, \ldots, k_n) \in \mathbb{N}^n$ of natural numbers the probability for a string a to have multiplicities exactly given by k follows the multinomial distribution:

$$P(a \in \Sigma^r \mid N_i(a) = k_i \text{ for all } i = 1, \ldots, n) = \frac{r!}{k_1! \cdots k_n!} \cdot p_1^{k_1} \cdots p_n^{k_n}.$$

The Log-Weight (LW) Score

A heuristic derivation of the LW-score of a string $a \in \Sigma^r$ considers the "null hypothesis" (H_0): *a belongs to a given language with letter probabilities p_i*, and the "alternative hypothesis" (H_1): *a is a random string*. The probabilities for a having k as its set of multiplicities if (H_1) or (H_0) is true, are (in a somewhat sloppy notation)

$$P(k \mid \mathrm{H}_1) = \frac{r!}{k_1! \cdots k_n!} \cdot \frac{1}{n^r}, \quad P(k \mid \mathrm{H}_0) = \frac{r!}{k_1! \cdots k_n!} \cdot p_1^{k_1} \cdots p_n^{k_n}.$$

The quotient of these two values, the "likelihood ratio"

$$\lambda(k) = \frac{P(k \mid \mathrm{H}_0)}{P(k \mid \mathrm{H}_1)} = n^r \cdot p_1^{k_1} \cdots p_n^{k_n},$$

makes a good score for deciding between (H_0) and (H_1).

> Usually one takes the reciprocal value, that is H_1 in the numerator, and H_0 in the denominator. We deviate from this convention because we want to have the score large for true texts and small for random texts.

For convenience one considers the logarithm (to any base) of this number:

$$\log \lambda(k) = r \log n + \sum_{i=1}^{n} k_i \cdot \log p_i.$$

Table 9: *Log weights of the letters for English (base-10 logarithms)*

s	A	B	C	D	E	F	G
$1000p_s$	82	15	28	43	127	22	20
Log weight	1.9	1.2	1.4	1.6	2.1	1.3	1.3
s	H	I	J	K	L	M	N
$1000p_s$	61	70	2	8	40	24	67
Log weight	1.8	1.8	0.3	0.9	1.6	1.4	1.8
s	O	P	Q	R	S	T	U
$1000p_s$	75	19	1	60	63	91	28
Log weight	1.9	1.3	0.0	1.8	1.8	1.9	1.4
s	V	W	X	Y	Z		
$1000p_s$	10	23	1	20	1		
Log weight	1.0	1.4	0.0	1.3	0.0		

(We assume all $p_i > 0$, otherwise we would omit s_i from our alphabet.) Noting that the summand $r \log n$ is the same for all $a \in \Sigma^r$ one considers

$$\log \lambda(k) - r \log n = \sum_{i=1}^{n} k_i \cdot \log p_i = \sum_{j=1}^{r} \log p_{a_j}.$$

Because $0 < p_i < 1$ the summands are negative. Adding a constant doesn't affect the use of this score, so finally we define SINKOV's **Log-Weight (LW) score** as

$$S_1(a) := \sum_{i=1}^{n} k_i \cdot \log(1000 \cdot p_i) = \sum_{j=1}^{r} \log(1000 \cdot p_{a_j}) = r \cdot \log 1000 + \sum_{j=1}^{r} \log p_{a_j}.$$

The numbers $\log(1000 \cdot p_i)$ are the "log weights". More frequent letters have higher weights. Table 9 gives the weights for the English alphabet with base-10 logarithms (so $\log 1000 = 3$). The MFL-method in contrast uses the weights 1 for `ETOANIRSHD`, and 0 else.

Note that the definition of the LW score doesn't depend on its heuristic motivation. Just take the weights given in Table 9 and use them for the definition of S_1.

Examples

We won't analyze the LW-method in detail, but rework the examples from Section 1. The LW scores for the CAESAR example are in Table 10.

The correct solution stands out clearly, the order of the non-solutions is somewhat permuted compared with the MFL score.

Table 10: *LW scores for the exhausion of a shift cipher*

FDHVDU	8.7	OMQEMD	8.4		XVZNVM	5.2	
GEIWEV	9.7	PNRFNE	10.1	<---	YWAOWN	9.7	
HFJXFW	6.1	QOSGOF	8.2		ZXBPXO	4.4	
IGKYGX	6.6	RPTHPG	9.4		AYCQYP	7.2	
JHLZHY	6.8	SQUIQH	6.8		BZDRZQ	4.6	
KIMAIZ	7.8	TRVJRI	8.6		CAESAR	10.9	<===
LJNBJA	7.1	USWKSJ	7.6		DBFTBS	9.0	
MKOCKB	7.7	VTXLTK	7.3		ECGUCT	9.5	
NLPDLC	9.3	WUYMUL	8.5				

For the period-4 example the LW scores are in Tables 11 to 14. The method unambiguously picks the correct solution except for column 3 where the top score occurs twice.

In summary the examples show no clear advantage of the LW-method over the MFL-method, notwithstanding the higher granularity of the information used to compute the scores.

As for MFL scores we might define the LW rate as the quotient of the LW score be the length of the string. This makes the values for strings of different lengths comparable.

Table 11: *LW scores for column 1 of a period 4 cipher*

UDHWHUPLSLWD	18.7	DMQFQDYUBUFM	13.9	MVZOZMHDKDOV	14.5	
VEIXIVQMTMXE	14.5	ENRGREZVCVGN	17.4	NWAPANIELEPW	20.4	<--
WFJYJWRNUNYF	15.4	FOSHSFAWDWHO	19.9	OXBQBOJFMFQX	10.5	
XGKZKXSOVOZG	11.0	GPTITGBXEXIP	15.9	PYCRCPKGNGRY	16.9	
YHLALYTPWPAH	19.1	HQUJUHCYFYJQ	12.3	QZDSDQLHOHSZ	13.9	
ZIMBMZUQXQBI	10.2	IRVKVIDZGZKR	13.9	RAETERMIPITA	21.7	<==
AJNCNAVRYRCJ	16.7	JSWLWJEAHALS	17.9	SBFUFSNJQJUB	13.8	
BKODOBWSZSDK	16.2	KTXMXKFBIBMT	13.9	TCGVGTOKRKVC	16.7	
CLPEPCXTATEL	18.5	LUYNYLGCJCNU	16.6			

Exercise. Give a more detailed analysis of the distribution of the LW scores for English and for random texts (with "English" weights). You may use the Perl script `LWscore.pl` in the directory `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/Perl/`.

Table 15 gives log weights for German and French.

Table 12: *LW scores for column 2 of a period 4 cipher*

MBTWZWIBWJWL	15.0	VKCFIFRKFSFU	16.2	ETLOROATOBOD	21.6	<==
NCUXAXJCXKXM	10.5	WLDGJGSLGTGV	16.4	FUMPSPBUPCPE	17.2	
ODVYBYKDYLYN	16.8	XMEHKHTMHUHW	17.7	GVNQTQCVQDQF	11.3	
PEWZCZLEZMZO	13.2	YNFILIUNIVIX	17.4	HWORURDWRERG	20.1	<--
QFXADAMFANAP	16.3	ZOGJMJVOJWJY	11.4	IXPSVSEXSFSH	16.5	
RGYBEBNGBOBQ	16.3	APHKNKWPKXKZ	13.1	JYQTWTFYTGTI	16.3	
SHZCFCOHCPCR	17.3	BQILOLXQLYLA	14.5	KZRUXUGZUHUJ	11.7	
TIADGDPIDQDS	18.2	CRJMPMYRMZMB	14.7	LASVYVHAVIVK	17.0	
UJBEHEQJERET	17.1	DSKNQNZSNANC	16.6			

Table 13: *LW scores for column 3 of a period 4 cipher*

HLSJWJCAKDJ	13.3		QUBSFSLJTMS	14.5	ZDKBOBUSCVB	13.6
IMTKXKDBLEK	14.3		RVCTGTMKUNT	16.7	AELCPCVTDWC	17.0
JNULYLECMFL	15.8		SWDUHUNLVOU	17.1	BFMDQDWUEXD	13.6
KOVMZMFDNGM	14.0		TXEVIVOMWPV	14.8	CGNEREXVFYE	16.2
LPWNANGEOHN	18.7	<-	UYFWJWPNXQW	11.6	DHOFSFYWGZF	15.0
MQXOBOHFPIO	14.5		VZGXKXQOYRX	8.2	EIPGTGZXHAG	14.7
NRYPCPIGQJP	13.6		WAHYLYRPZSY	15.5	FJQHUHAYIBH	14.6
OSZQDQJHRKQ	10.1		XBIZMZSQATZ	10.0	GKRIVIBZJCI	13.3
PTARERKISLR	18.7	<-	YCJANATRBUA	16.8		

Table 14: *LW scores for column 4 of a period 4 cipher*

ORCNBCOWCOO	18.0		XALWKLXFLXX	10.3		GJUFTUGOUGG	14.8
PSDOCDPXDPP	15.1		YBMXLMYGMYY	13.5		HKVGUVHPVHH	15.1
QTEPDEQYEQQ	12.4		ZCNYMNZHNZZ	11.3		ILWHVWIQWII	15.8
RUFQEFRZFRR	14.6		ADOZNOAIOAA	18.5		JMXIWXJRXJJ	7.6
SVGRFGSAGSS	17.1		BEPAOPBJPBB	14.9		KNYJXYKSYKK	11.4
TWHSGHTBHTT	18.7	<-	CFQBPQCKQCC	10.3		LOZKYZLTZLL	12.4
UXITHIUCIUU	16.1		DGRCQRDLRDD	16.1		MPALZAMUAMM	15.6
VYJUIJVDJVV	11.0		EHSDRSEMSEE	20.4	<=	NQBMABNVBNN	15.1
WZKVJKWEKWW	11.7		FITESTFNTFF	18.4			

Table 15: *Log weights of the letters for German and French (base-10 logarithms)*

s	A	B	C	D	E	F	G
German	1.8	1.3	1.4	1.7	2.2	1.2	1.5
French	1.9	1.0	1.5	1.6	2.2	1.1	1.0
s	H	I	J	K	L	M	N
German	1.6	1.9	0.5	1.2	1.5	1.4	2.0
French	0.8	1.8	0.5	0.0	1.8	1.4	1.9
s	O	P	Q	R	S	T	U
German	1.5	1.0	0.0	1.9	1.8	1.8	1.6
French	1.7	1.4	1.0	1.8	1.9	1.9	1.8
s	V	W	X	Y	Z		
German	1.0	1.2	0.0	0.0	1.0		
French	1.2	0.0	0.6	0.3	0.0		