cyber & data

"From bits to information"

Similarity and Matching

Outline

- Similarity Metrics.
- Similarity Hashes.
- Regular Expressions.

cyber & data

"From bits to information"

Similarity

Similarity

5324-9990-1234-5555

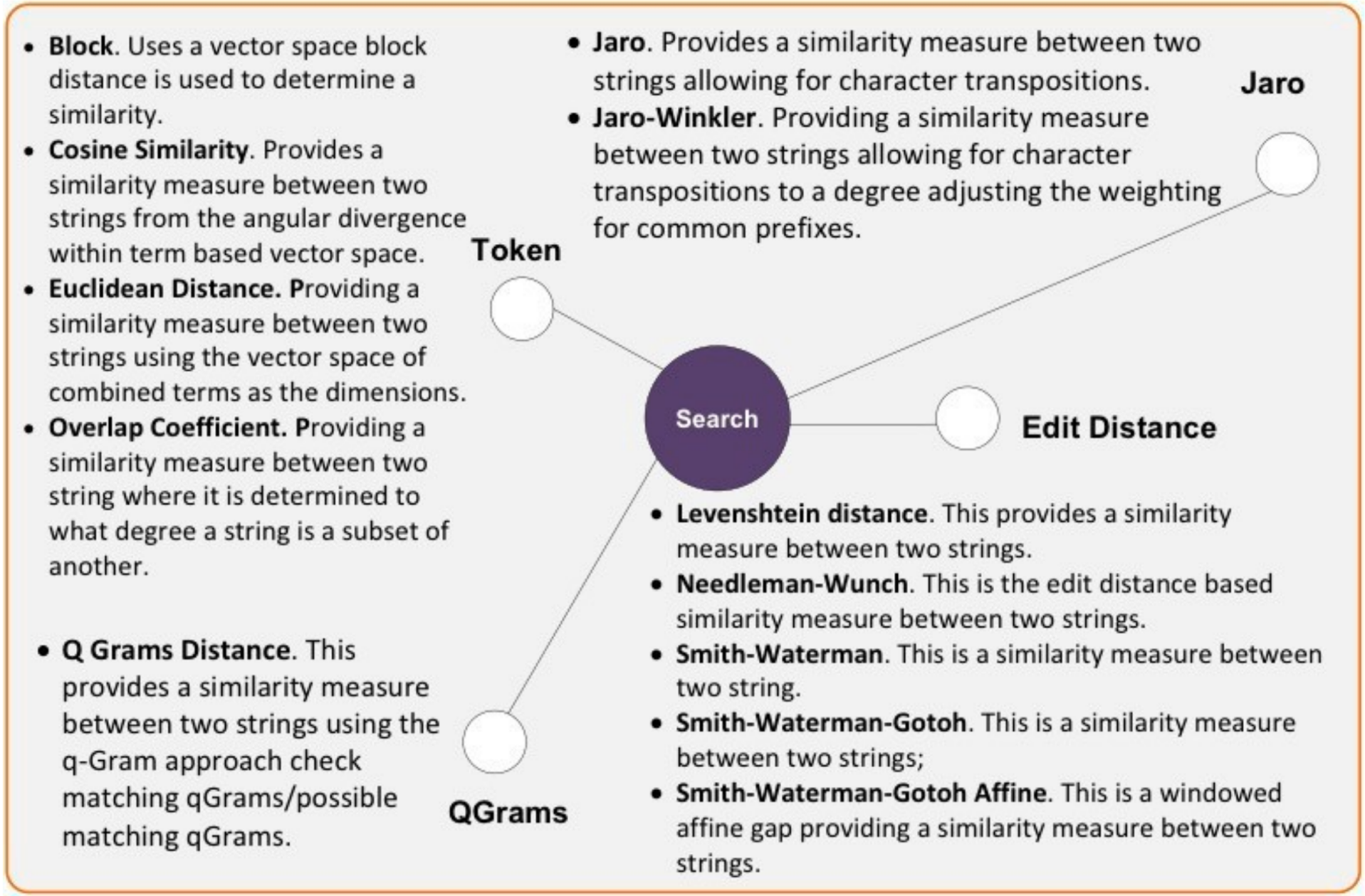
5824 9999 4234 7666

A	Q	F	M	U	G	X	E	H	U	W	V	F	G	I
S	F	F	V	B	M	A	H	G	N	I	M	R	I	B
U	X	I	U	R	B	B	O	O	N	K	X	F	U	V
I	D	D	Z	I	V	A	C	R	E	G	F	D	E	P
O	I	R	Z	G	H	J	H	I	L	E	S	J	B	H
H	I	A	F	H	U	K	V	A	M	V	D	K	R	K
E	L	C	P	T	M	B	S	R	R	C	Y	N	Z	A
D	E	W	H	O	T	G	N	S	L	E	S	E	U	A
R	E	K	X	N	O	L	O	N	D	O	N	W	F	D
O	D	V	Y	W	F	W	S	V	J	Q	B	C	D	K
F	S	E	D	I	N	B	U	R	G	H	N	A	Z	N
X	C	A	M	B	R	I	D	G	E	N	K	S	J	E
O	W	W	H	S	N	O	A	T	L	K	W	T	F	Z
Z	M	X	D	B	M	Z	V	S	Q	S	G	L	E	K
O	B	B	V	M	A	N	C	H	E	S	T	E	R	K

EDINBURGH
GLASGOW
DUNDEE
LONDON
MANCHESTER
LEEDS
BRIGHTON
CARDIFF
BIRMINGHAM
NEWCASTLE
OXFORD
CAMBRIDGE

ber
&data

Similarity


Search
Token
Jaro
Edit Distance
QGrams
• Block. Uses a vector space block distance is used to determine a similarity.
• Cosine Similarity. Provides a similarity measure between two strings from the angular divergence within term based vector space.
• Euclidean Distance. Providing a similarity measure between two strings using the vector space of combined terms as the dimensions.
• Overlap Coefficient. Providing a similarity measure between two string where it is determined to what degree a string is a subset of another.
• Jaro. Provides a similarity measure between two strings allowing for character transpositions.
• Jaro-Winkler. Providing a similarity measure between two strings allowing for character transpositions to a degree adjusting the weighting for common prefixes.
• Levenshtein distance. This provides a similarity measure between two strings.
• Needleman-Wunch. This is the edit distance based similarity measure between two strings.
• Smith-Waterman. This is a similarity measure between two string.
• Smith-Waterman-Gotoh. This is a similarity measure between two strings;
• Smith-Waterman-Gotoh Affine. This is a windowed affine gap providing a similarity measure between two strings.
• Q Grams Distance. This provides a similarity measure between two strings using the q-Gram approach check matching qGrams/possible matching qGrams.

Similarity

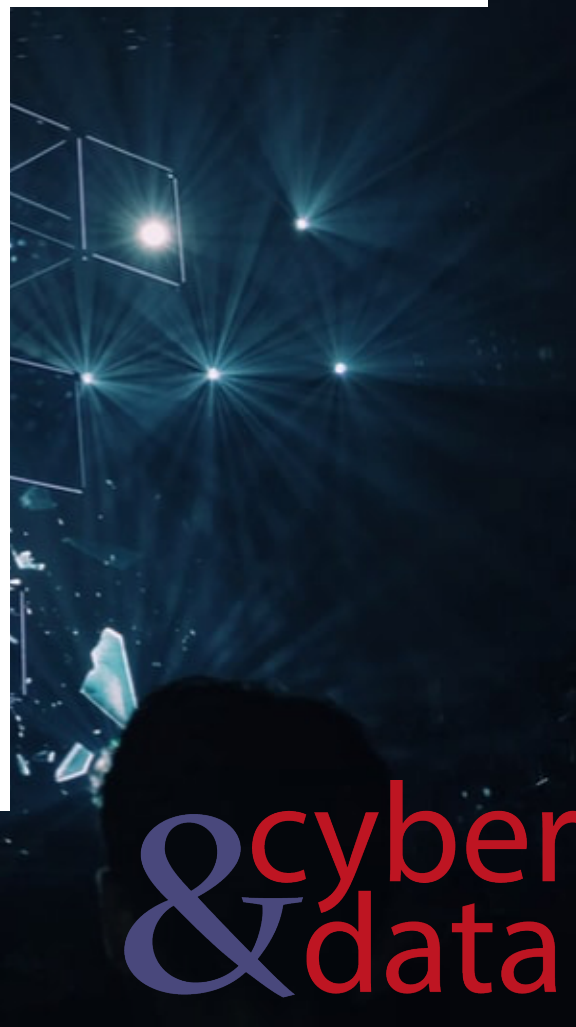
Method	Loss of insig word	Small changes	Rearrangement of words	Punctuation	Case	Spacing
Levenshtein	78	89	44	84	17	77
NeedlemanWunch	81	89	61	84	52	80
Smith-Waterman	86	97	44	90	9	78
Smith-Waterman Gotoh	89	94	47	84	44	78
Smith-Waterman Gotoh Windowed Affine	89	94	47	84	44	78
Jaro	88	96	0	95	41	87
Jaro Winkler	93	98	0	97	47	91
QGrams Distance	89	74	70	69	4	68
Block Distance	80	33	100	25	0	0
Cosine Similarity	82	33	100	25	0	0
Euclidean Distance	55	18	100	13	0	0
Chapman Length Deviation	78	89	100	84	92	82
Overlap Coefficient	100	33	100	25	0	0
	Loans and Accounts	loans and accounts	loans and accounts	fishing, "camping"; and 'forest	Loan Account and Dealing	LoanAccountDealing
	Loans Accounts	loan and account	accounts and loans	fishing camping and forest	LOAN ACCOUNTS DEALINGS	Load, Account, Dealing

Levenshtein

$$d_{ij} = \min \begin{cases} d_{i-1,j} + c_{\text{del}}(b_i) \\ d_{i,j-1} + c_{\text{ins}}(a_j) \\ d_{i-1,j-1} + [a_j \neq b_i] \cdot c_{\text{sub}}(a_j, b_i) \end{cases}$$

		A	p	p	l	e	c	a	i	n
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
p	2	1	0	1	2	3	4	5	6	7
l	3	2	1	2	1	2	3	4	5	6
e	4	3	2	3	2	1	2	3	4	5
c	5	4	3	4	3	2	1	2	3	4
o	6	5	4	5	4	3	2	3	4	5
r	7	6	5	6	5	4	3	4	5	6
e	8	7	6	7	6	5	4	5	6	7

```
A     p l e c o r e
A p p l e c a i n
```



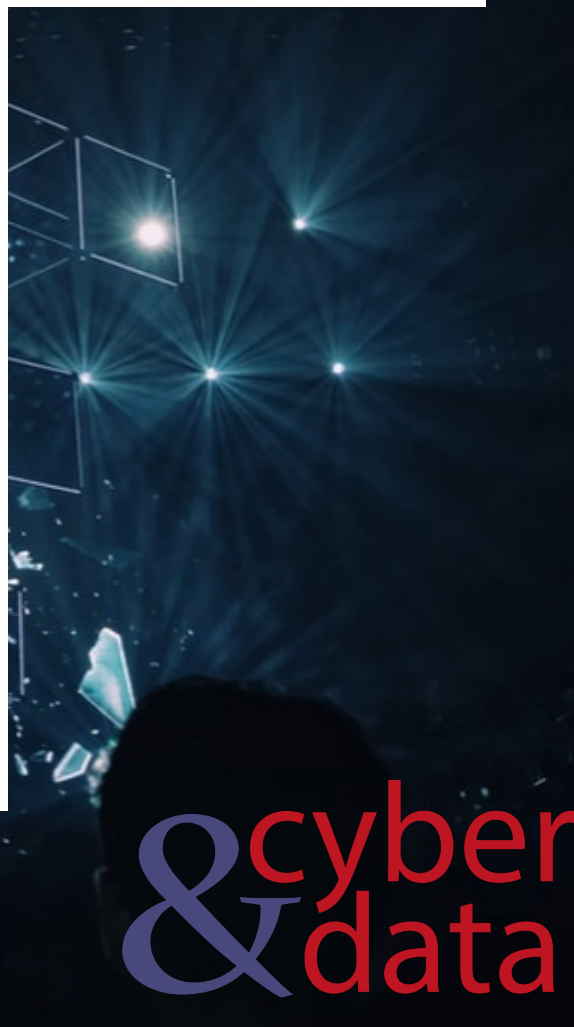
&cyber data

Levenshtein

$$d_{ij} = \min \begin{cases} d_{i-1,j} + c_{\text{del}}(b_i) \\ d_{i,j-1} + c_{\text{ins}}(a_j) \\ d_{i-1,j-1} + [a_j \neq b_i] \cdot c_{\text{sub}}(a_j, b_i) \end{cases}$$

```
var levenshtein = require('fast-levenshtein');

str1='Aplecore'
str2='Applecain';

var distance = levenshtein.get(str1, str2);
console.log('Distance:\t',distance);

length = Math.max(str1.length,str2.length);

ratio = 100-100*(distance /length);
console.log('Similarity:\t',parseFloat(Math.round(ratio).toFixed(2)))
```

```
A     p l e c o r e
A p p l e c a i n
```



Needleman-Wunsch

```
12345678
ap-lecore
Applecain
++++--+- -> 1x4 + (-1)*3 = 1
```

```
erocel-pa
niacelppA
```

- Match. This is where two letters match at the same index value. The two letters at the current index are the same. For this we could assign a score of +1.
- Mismatch: This is where the letters do not match the same index. For this we could assign a score of -1.
- Indel (INsertion or DELetion). This is a deletion or insertion of a character within the alignment. For this we could assign a score of -1.

		A	p	p	l	e	c	a	i	n
	0	◂-1	◂-2	◂-3	◂-4	◂-5	◂-6	◂-7	◂-8	◂-9
a	▲-1	▸◂-1	▸◂-2	▸◂-3	▸◂-4	▸◂-5	▸◂-6	▸-5	▸◂-6	▸◂-7
p	▲-2	▸▲-2	▸ 0	▸◂-1	▸◂-2	▸◂-3	▸◂-4	▸◂-5	▸◂-6	▸◂-7
-	▲-3	▸▲-3	▲-1	▸-1	▸◂-2	▸◂-3	▸◂-4	▸◂-5	▸◂-6	▸◂-7
l	▲-4	▸▲-4	▲-2	▸▲-2	▸ 0	◂-1	◂-2	◂-3	◂-4	◂-5
e	▲-5	▸▲-5	▲-3	▸▲-3	▲-1	▸ 1	◂0	◂-1	◂-2	◂-3
c	▲-6	▸▲-6	▲-4	▸▲-4	▲-2	▲ 0	▸ 2	◂1	◂0	◂-1
o	▲-7	▸▲-7	▲-5	▸▲-5	▲-3	▲-1	▲ 1	▸ 1	▸◂0	▸◂-1
r	▲-8	▸▲-8	▲-6	▸▲-6	▲-4	▲-2	▲ 0	▸▲ 0	▸ 0	▸◂-1
e	▲-9	▸▲-9	▲-7	▸▲-7	▲-5	▸▲-3	▲-1	▸▲-1	▸▲-1	▸-1

Smith-Waterman

- Similar to Needleman-Wunsh, but negative scoring cells are set to zero. The traceback for the sequence then begins within the highest scoring matrix cell and continues until we reach a zero scoring cell.
- Figure outlines an example with a scoring of +1 for a match, 0 for a mismatch, and -1 for both an insertion and a deletion, and for the string of "Aplecore" and "Applecain".

The scoring for each cell is then the highest of the three candidate scores. We then make a path from the bottom right cell to the top left by tracing the arrows. In the example:

```
erocel-pa
niacelppA
```

```
A p l - e c o r e
A p p l e c a i n
```

		A	p	l	e	c	o	r	e	s
	0	0	0	0	0	0	0	0	0	0
A	0	1	0	0	0	0	0	0	0	0
p	0	0	2	1	0	0	0	0	0	0
p	0	0	1	2	1	0	0	0	0	0
l	0	0	0	2	2	1	0	0	0	0
e	0	0	0	1	3	2	1	0	1	0
c	0	0	0	0	2	4	3	2	1	1
a	0	0	0	0	1	3	4	3	2	1
i	0	0	0	0	0	2	3	4	3	2
n	0	0	0	0	0	1	2	3	4	3

cyber & data

"From bits to information"

Phonetic matching

Phonetic matching

Phonetically:

“Castle”

Then becomes

“k-a-s-e-l”

or more formally as “kɑːs(ə)l”

Table 1.1-1: Phonemes

ID Phoneme	IPA Symbol	Graphemes	Example
1	b	b, bb	big
2	d	d, dd, ed	dare
3	f	f, ff, ph, gh, lf, ft	four
4	g	g, gg, gh,gu,gue	great
5	h	h, wh	hope
6	d	j, ge, g, dge, di, gg	jam
7	k	k, c, ch, cc, lk, qu ,q(u), ck, x	cat
8	l	l, ll	love
9	m	m, mm, mb, mn, lm	men
10	n	n, nn,kn, gn, pn	need
11	p	p, pp	pipe
12	r	r, rr, wr, rh	rat
13	s	s, ss, c, sc, ps, st, ce, se	sign
14	t	t, tt, th, ed	top
15	v	v, f, ph, ve	venue
16	w	w, wh, u, o	whip
17	z	z, zz, s, ss, x, ze, se	zone
18		s, si, z	azure
19	t	ch, tch, tu, ti, te	chop
20		sh, ce, s, ci, si, ch, sci, ti	ship
21		th	throw
22		th	leather
23		ng, n, ngue	wrong
24	j	y, i, j	your
25	æ	a, ai, au	cat
26	e	a, ai, eigh, aigh, ay, er, et, ei, au, a_e, ea, ey	pay
27	e	e, ea, u, ie, ai, a, eo, ei, ae	end
28	i:	e, ee, ea, y, ey, oe, ie, i, ei, eo, ay	bee
29		i, e, o, u, ui, y, ie	it
30	a	i, y, igh, ie, uy, ye, ai, is, eigh, i_e	kite
31		a, ho, au, aw, ough	bought
32	o	o, oa, o_e, oe, ow, ough, eau, oo, ew	sew
33		o, oo, u,ou	look
34		u, o, oo, ou	blood
35	u:	o, oo, ew, ue, u_e, *oe*, *ough*, *ui*, *oew*, *ou*	shoe
36		oi, oy, uoy	boy
37	a	ow, ou, ough	cow
38		a, er, i, ar, our, ur	dollar
39	e	air, are, ear, ere, eir, ayer	dare
40	:	a	arm
41	:	ir, er, ur, ear, or, our, yr	burn

Soundex

Soundex uses a phonetic algorithm to classify a sound as it is pronounced. It focuses on matching phrases which have minor spelling errors. A Soundex code has a letter followed by three numbers, such as C253. The first letter is the first letter of the surname.

```
Soundex code for tailer:        T460
Soundex code for taylor:        T460

NYSIIS for tailer:      TALAR
NYSIIS for taylor:      TAYLAR

Phonex for tailer:      T460
Phonex for taylor:      T460

==Metrics==
String          String          Jaro W  Distance        Damerau Jaro    SmithW
tailer          taylor          82.22   66.67           66.67   87.04   66.67
```

Coding

```
Number  Letters
    1   B, F, P, V
    2   C, G, J, K, Q, S, X, Z
    3   D, T
    4   L
    5   M, N
    6   R
```

We disregard the letters of A, E, I, O, U, H, W, and Y. For example, “Buchanan” becomes [here]:

```
B255 – "B" ... "C" ... "N" ... "N"
```

The name “Lee” becomes:

```
L000 = "L"
```

cyber & data

"From bits to information"

Similarity Hashes

Charikar similarity

[Back] The Charikar similarity method is often used for documents and metadata in order to located duplicates

Parameters

Word 1:

this is the first string

Word 2:

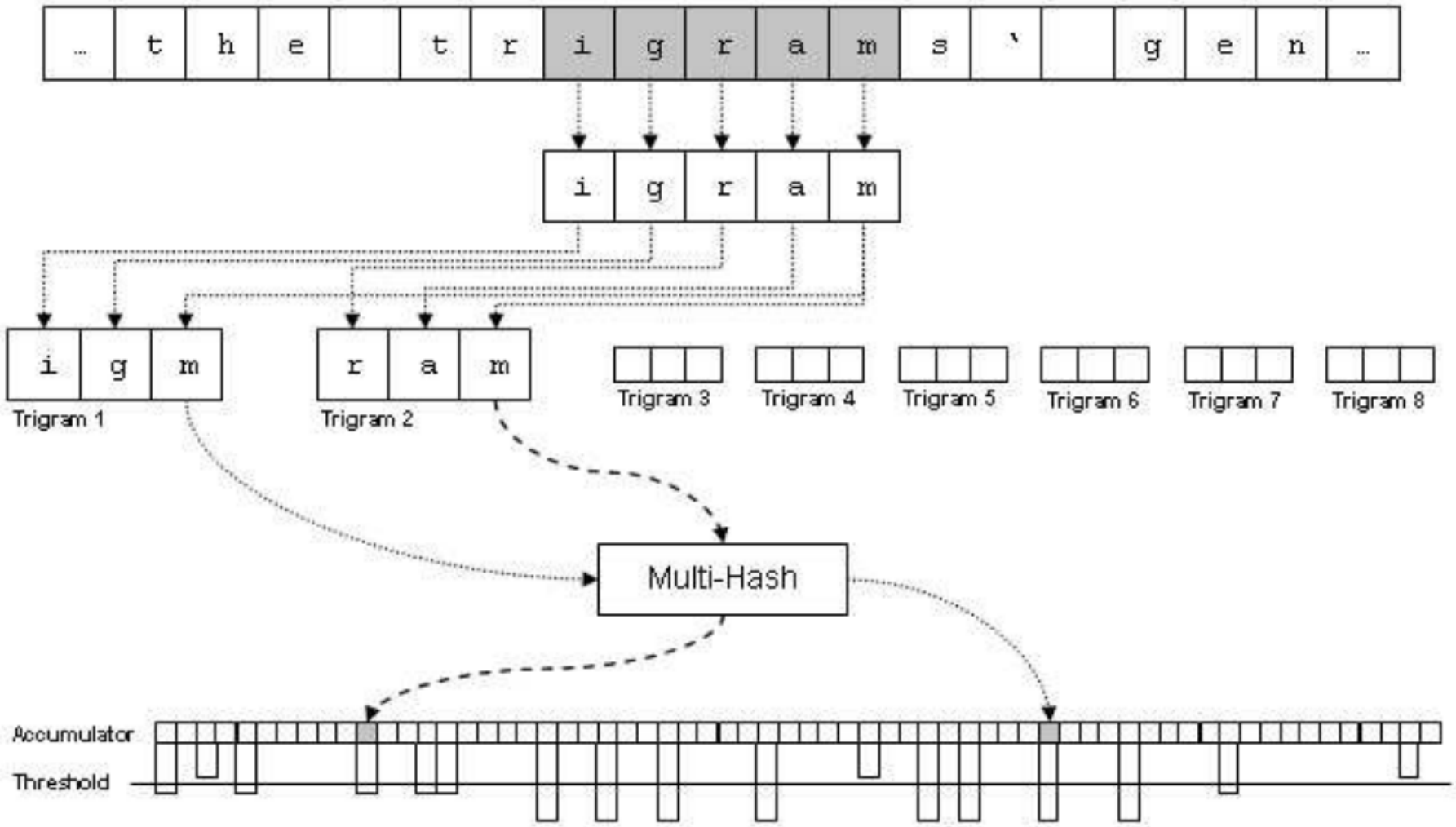
this is the second string

- word1 ="this is the first string", word2 ="this is the first string" Try!
- word1 ="this is the first string", word2 ="this is the string first" Try!
- word1 ="this is the first string", word2 ="this is the first help" Try!
- word1 ="this is the first string", word2 ="this keep the first help" Try!
- word1 ="this is the first string", word2 ="a totally different sentence" Try!

```
String 1:       this is the first string
String 2:       this is the second string

==== 8-bit hash ====
Hash1:          0xea
Hash2:          0xca
Similarity:     0.875

==== 24 -bit hash ====
Hash1:          0x9cc9ea
Hash2:          0xc81ca
Similarity:     0.791666666667

====256-bit hash ====
Hash1:          0x582200201d9f29269cc9eaL
Hash2:          0x20080204a02f0b69270c81caL
Similarity:     0.92578125

====64-bit Nilsimsa hash ====
Hash1:          0xffffffff00000000000000000000000L
Hash2:          0xffffffff00000000000000000000000L
Similarity:     1.0
```

Code

Nilsimsa



Code

cyber & data

"From bits to information"

Regular Expressions

Regular Expressions

[character_group]	Match any single character in character_group Example: gr[ae]y – gray, grey
[^character_group]	Match any single character in character_group Example: gr[^ae]y – grby, grcy
[a-z]	Character range Example a, b, c … z
{n}	Matches previous character repeated n times
a{n,m}	Matches between n and m or a
\d	Matches a digit
.	Single character
(a | b)	Matches a or b
a?	Zero or one match of a
a*	Zero or more match of a
a+	One or more match of a
$	Match at the end
Escape:	\s (space)

Data Formats

Telephone: \\d{3}[-.]?\\d{3}[-.]?\\d{4}

444.444.2312

Year: [0-9]{4}

1961

test@home.com

Email: [a-zA-Z0-9._%+-]+@[a-zA-Z0-9._%+-]

5555-1234-3456-4312

DLP

```
Master:    5\\d{3}(\\s|-)?\\d{4}(\\s|-)?\\d{4}(\\s|-)?\d{4}
Am Ex:     3\\d{3}(\\s|-)?\\d{6}(\\s|-)?\\d{5}
Visa:      4\\d{3}(\\s|-)?\\d{4}(\\s|-)?\\d{4}(\\s|-)?\d{4}
```

1.2.3.4

IP: [0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}.[0-9]{1,3}

RegEx

Regular Expressions

Python

main.py

```
1   import re
2
3   st="There is not much we can do apart from contacting
    There is not much we can do apart from contacting
    f.smith@home.net to see if he would like to reboot the
    server at 192.168.0.1. If he can do this then I will
    call him on 444.3212.5431. My credit card details are
    4321-4444-5412-2310 and 5430-5411-4333-5123 and my name
    on the card is Fred Smith (fred@home.com). I really like
    the name domain fred@home. Overall our target areas are
    SW1 7AF and EH105DT. I tested the server last night, and
    I think the IP address is 10.0.0.1 and there are two MAC
    addresses which are 01:23:45:67:89:ab or it might be
    00.11.22.33.44.55. The book we will use is “At Home” and
    it can be bought on amazon.com or google.com, if you
    search for 978-1-4302-1998-9. My password is: a1b2c3
    Best regards, Bert. EH14 1DJ +44 (960) 000 00 00
    1/1/2009"
4
5   # reg="[a-zA-Z0-9._%+-]+@[a-zA-Z0-9._%+-]"
6   # reg="[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}.[0-9]{1,3}"
7   # reg="\d{3}[-.]?\d{4}[-.]?\d{4}"
8   # reg="[A-Z]{1,2}[0-9]{1,2}[A-Z]?\s[0-9][A-Z][A-Z]"
9   reg ="4\d{3}(\s|-)?\d{4}(\s|-)?\d{4}(\s|-)?\d{4}"
10
11  result = re.search(reg, st)
12
13  print (result)
```

https://regex.billbuchanan.repl.run

```
<re.Match object; span=(262, 281), match='4321-4444-5412-2310'>
```

cyber & data

"From bits to information"

Similarity and Matching