Artem M. Sokolov

Talk Plan (∼25 min.)

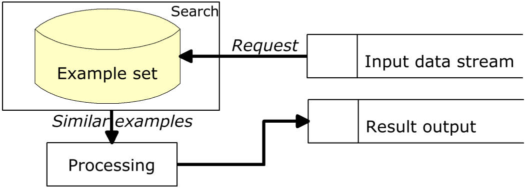
1. **Sequence embeddings (18 slides)**
   - Sequence processing, case-based reasoning and space embeddings
   - Deterministic embedding of edit distance into vector space
   - Randomized embedding and nearest neighbour search
   - Applications
2. **Markov chains with variable memory length (3 slides)**
   - Suffix automaton
   - Modification
   - Applications
3. **Other projects (1 slide)**

Sequence Processing and Case-Based Reasoning

Tasks

- duplicate detection
- spam filtering
- gene finding
- intrusion detection

Case-Based Reasoning



Searching for similar examples is the basic operation

Let $x, y \in \Sigma^n$ be symbol strings of length n over a finite alphabet Σ.

Edit Distance $ed(x, y)$

Minimum number of symbol **changes, deletions and inserts** to transform x into y.

Calculating $ed(x, y)$

- Dynamic programming – $O(n^2)$
- Best result – $O(\frac{n^2}{\log n})$
- Still too bad for large n

Space embeddings

How to compute a «difficult» metric

Idea – embed into a «simpler» space: $(X, \rho_1) \xrightarrow{v} (Y, \rho_2)$

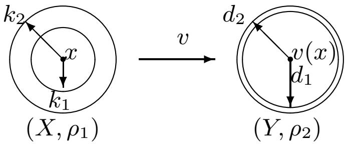
- usually vector space (preferably of small dimension)
- with simple metrics ρ_2 (e.g., ℓ_1, ℓ_2, ℓ_∞, $Hamming$)

Embedding quality (possible definition) — (k_1, k_2, d_1, d_2)-embedding

There exist $k_1 \leq k_2$ and $d_1 \leq d_2$, such that

$$\text{if } \rho_1(x,y) \leq k_1, \text{ then } \rho_2(v(x), v(y)) \leq d_1,$$
$$\text{if } \rho_1(x,y) > k_2, \text{ then } \rho_2(v(x), v(y)) > d_2.$$

The less is $(k_2 - k_1)$, the better is the approximation

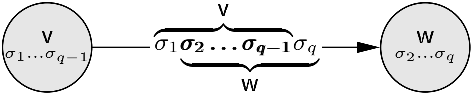


De Bruijn graphs

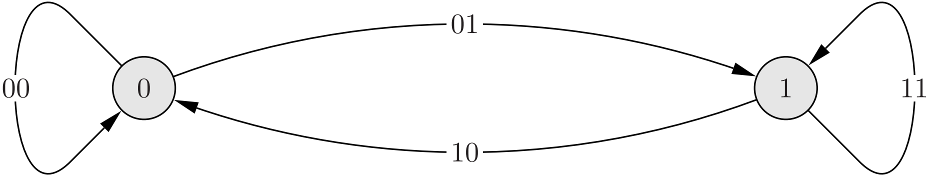
- $x[i, i+q-1]$ q-gram
- $(v_q(x))_j = \sum_{i=1}^{n-q+1} [[x[i, i+q-1] = \sigma_j]], \sigma_j \in \Sigma^q$ q-gram vector
- $d_q(x, y) = \sum_j |(v_q(x))_j - (v_q(y))_j|$ q-gram distance

De Bruijn graph

- $B(\Sigma; q) = G(V, E)$
  $V = \Sigma^{q-1}$
  $E = \Sigma^q$



Example $B(\{0,1\}, 2)$

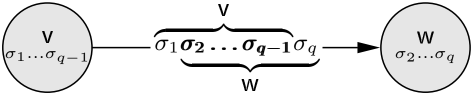


De Bruijn graphs

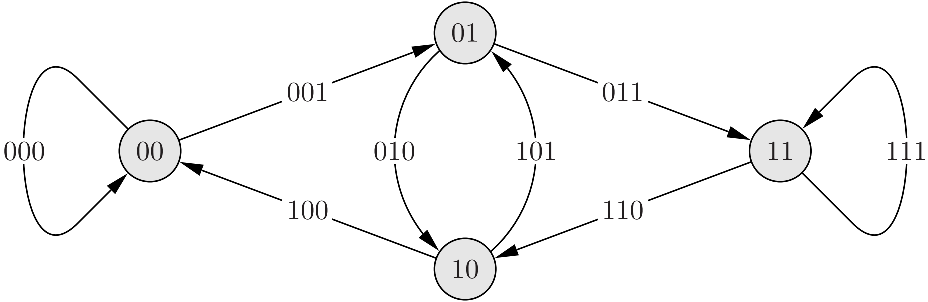
- $x[i, i+q-1]$ — q-gram
- $(v_q(x))_j = \sum_{i=1}^{n-q+1} [[x[i, i+q-1] = \sigma_j]], \sigma_j \in \Sigma^q$ — q-gram vector
- $d_q(x, y) = \sum_j |(v_q(x))_j - (v_q(y))_j|$ — q-gram distance

De Bruijn graph

- $B(\Sigma; q) = G(V, E)$
  $V = \Sigma^{q-1}$
  $E = \Sigma^q$



Example $B(\{0,1\}, 3)$

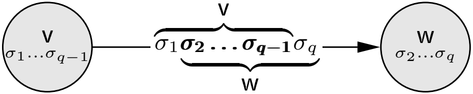


De Bruijn graphs

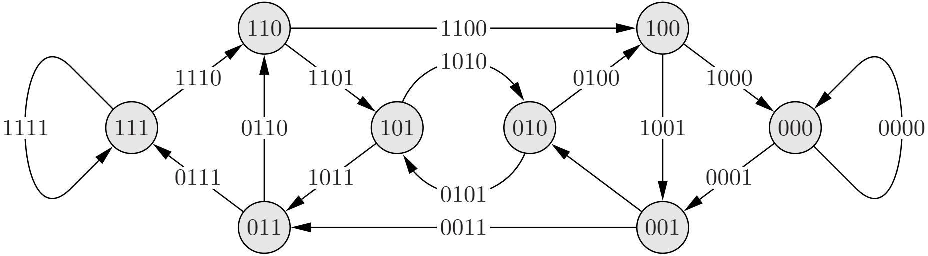
- $x[i, i+q-1]$ — q-gram
- $(v_q(x))_j = \sum_{i=1}^{n-q+1} [[x[i,i+q-1] = \sigma_j]], \sigma_j \in \Sigma^q$ — q-gram vector
- $d_q(x,y) = \sum_j |(v_q(x))_j - (v_q(y))_j|$ — q-gram distance

De Bruijn graph

- $B(\Sigma; q) = G(V, E)$
  $V = \Sigma^{q-1}$
  $E = \Sigma^q$



Example $B(\{0,1\}, 4)$

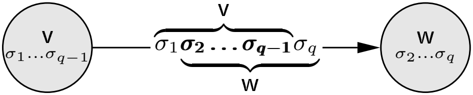


De Bruijn graphs

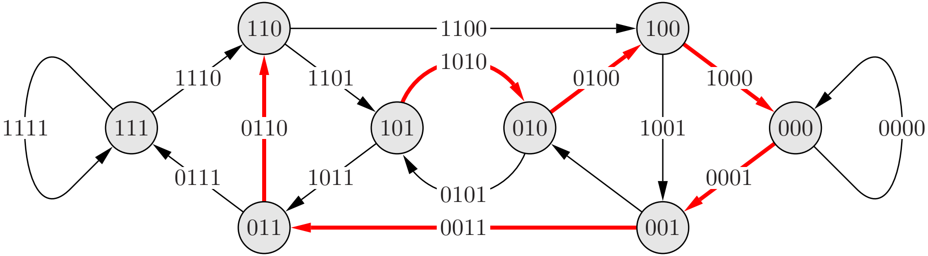
- $x[i, i+q-1]$ — q-gram
- $(v_q(x))_j = \sum_{i=1}^{n-q+1} [[x[i, i+q-1] = \sigma_j]], \sigma_j \in \Sigma^q$ — q-gram vector
- $d_q(x, y) = \sum_j |(v_q(x))_j - (v_q(y))_j|$ — q-gram distance

De Bruijn graph

- $B(\Sigma; q) = G(V, E)$
  $V = \Sigma^{q-1}$
  $E = \Sigma^q$



Example $B(\{0, 1\}, 4)$, $x = 101000110$

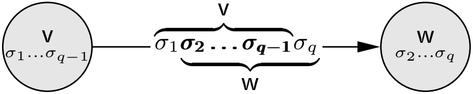


De Bruijn graphs

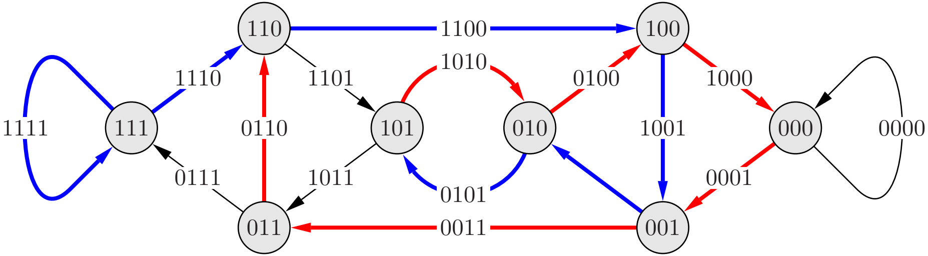
- $x[i, i+q-1]$ — q-gram
- $(v_q(x))_j = \sum_{i=1}^{n-q+1} [[x[i, i+q-1] = \sigma_j]], \sigma_j \in \Sigma^q$ — q-gram vector
- $d_q(x, y) = \sum_j |(v_q(x))_j - (v_q(y))_j|$ — q-gram distance

De Bruijn graph

- $B(\Sigma; q) = G(V, E)$
  $V = \Sigma^{q-1}$
  $E = \Sigma^q$

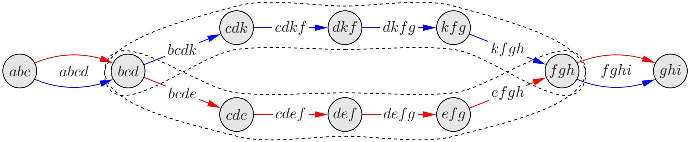


Example $B(\{0, 1\}, 4)$, $x = 101000110$, $y = 111100101$

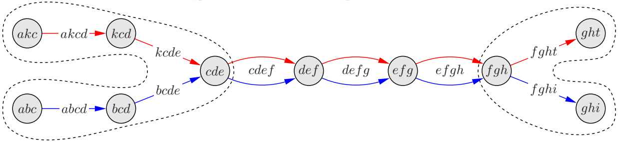


Types of path configurations on a de Bruijn graph

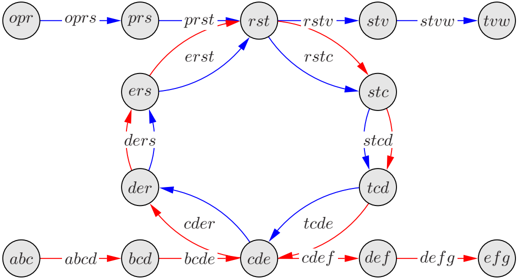
Loop *abcd***e***fghi*
*abcd***k***fght*



Fork *ak***cdefgh***t*
*ab***cdefgh***i*



Cycle *ab***cderstcde***fg*
*op***rstcderst***vw*

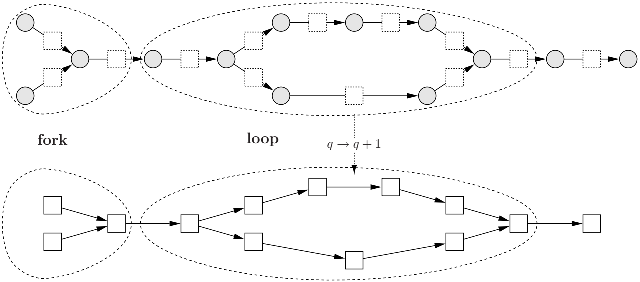


Idea: evolution of the configurations with q

- Take two paths on $B(\Sigma, q)$ corresponding to strings x and y
- Increment q by 1 and get $B(\Sigma, q+1)$
- The number of distinct edges changes differently depending on the initial configuration.

q-gram distance changes:

- Loop – $d_{q+1}(x,y) = d_q(x,y) + 2$
- Fork – $d_{q+1}(x,y) = d_q(x,y)$
- Cycle is more complicated



Intuition for the embedding

- too many different edges (loop) $\longrightarrow$ likely large edit distance
- few different edges (fork) $\longrightarrow$ likely small edit distance (just edit the forking parts)

Deterministic embedding of ed into ℓ_1

Embedding construction

- q_1, q_2 — **min** and **max** lengths of q-grams
- w — width of sliding window
- $\Sigma^n \longrightarrow (\mathbb{N} \cup 0)^{|\Sigma^n|(n-w+1)(q_2-q_1+1)}$ — embedding
  $x \mapsto (v_{q_1}(x[1,w]), v_{q_1+1}(x[1,w]), .., v_{q_2}(x[1,w]), v_{q_1}(x[2,w+1], .., v_{q_2}(x[n-w+1,n]))$
  all q-gram vectors concatenated for all q's and all windows of width w
- $D(x,y) = \frac{\sum_{i=1}^{n-w+1} \sum_{q_1}^{q_2} d_q(x[i,i+w-1], y[i,i+w-1])}{(n-w+1)(\Delta q+1)}$ — new metrics
  just normalized q-gram distance

Analysis Flow

$$\text{new distance } D(x,y) = \frac{\sum_{i=1}^{n-w+1} \sum_{q_1}^{q_2} d_q(x[i,i+w-1], y[i,i+w-1])}{(n-w+1)(\Delta q+1)}$$

Lower bound

strings	one interval of width w		strings of length $n > w$
repetitive	**if** q **is "large enough"** $\Rightarrow$ **only 1 cycle on** $B(x,q)$ $\downarrow$ **if** q **is "large enough"** **&** $\exists$ **cycle on** $B(x,y,q)$ $\Rightarrow$ **no "non-cycle" common edges** $\downarrow$ **(under some conditions)** $D(x,y) < (\Delta q+1)(\Delta q+2)$ $\Rightarrow ed(x,y) \leq 2(\Delta q+1)$	**(under united conditions)** $D(x,y) < (\Delta q+1)(\Delta q+2)$ $\Rightarrow ed(x,y) \leq 2(\Delta q+1)$	**if for each consecutive interval holds** $D(x_i, y_i) < (\Delta q+1)(\Delta q+2)$ $\Rightarrow ed(x,y) \leq 2(\Delta q+1)$ $\downarrow$ $\downarrow$ **bound** $ed(x,y)$ **in terms of number** N **of "bad" intervals** $ed(x,y) \geq k_2 \Rightarrow$ $N > (w - \Delta q + 1)(\frac{k_2}{2(\Delta q+1)} - 2)$
non repetitive	**(under some conditions)** $D(x,y) < (\Delta q+1)(\Delta q+2)$ $\Rightarrow ed(x,y) \leq 2(\Delta q+1)$		

Upper bound

Each edit operation changes at most w intervals, so

$$ed(x,y) \leq k_1 \Rightarrow D(x,y) \leq \frac{2k_1[w^2+n+1]}{n-w+1}$$

Deterministic embedding of ed into ℓ_1

Recall: (k_1, k_2, d_1, d_2)-**embedding**

There exist $k_1 \le k_2$ and $d_1 \le d_2$, such that

$$\text{if } \rho_1(x,y) \le k_1, \text{ then } \rho_2(v(x), v(y)) \le d_1,$$
$$\text{if } \rho_1(x,y) > k_2, \text{ then } \rho_2(v(x), v(y)) > d_2.$$

The less is $(k_2 - k_1)$, the better is the approximation

The result can be formulated as:

Theorem

For $w \ge 6$, $k_1 \ge 1$, $q_1 = 2w/3$, $n > w(k_1+1)+1$, $\Delta q = \frac{1}{2}(-7 + \sqrt{57 + 16(w - q_1)})$, $Q = (\Delta q + 1)(\Delta q + 2)$, $t = w - \Delta q + 1$

$$\text{If } ed(x,y) \le k_1, \text{ then } D(x,y) \le \frac{2k_1[w^2 + (n+1)]}{n - w + 1}$$

$$\text{If } ed(x,y) > k_2, \text{ then } D(x,y) \ge \frac{Qt(\frac{k_2}{2(\Delta q+1)} - 2)}{(n-w+1)(\Delta q + 1)}.$$

Deterministic embedding of ed into ℓ_1

Recall: (k_1, k_2, d_1, d_2)**-embedding**

There exist $k_1 \le k_2$ and $d_1 \le d_2$, such that

$$\text{if } \rho_1(x,y) \le k_1, \text{ then } \rho_2(v(x), v(y)) \le d_1,$$
$$\text{if } \rho_1(x,y) > k_2, \text{ then } \rho_2(v(x), v(y)) > d_2.$$

The less is $(k_2 - k_1)$, the better is the approximation

The result can be formulated as:

Theorem

For $w \ge 6$, $k_1 \ge 1$, $q_1 = 2w/3$, $n > w(k_1+1)+1$, $\Delta q = \frac{1}{2}(-7 + \sqrt{57 + 16(w - q_1)})$, $Q = (\Delta q + 1)(\Delta q + 2)$, $t = w - \Delta q + 1$

$$\text{If } ed(x,y) \le k_1, \text{ then } D(x,y) \le \frac{2k_1[w^2 + (n+1)]}{n - w + 1}$$

$$\text{If } ed(x,y) > k_2, \text{ then } D(x,y) \ge \frac{Qt(\frac{k_2}{2(\Delta q+1)} - 2)}{(n-w+1)(\Delta q + 1)}.$$

Deterministic embedding of ed into ℓ_1

Recall: **(k_1, k_2, d_1, d_2)-embedding**

There exist $k_1 \leq k_2$ and $d_1 \leq d_2$, such that

$$\text{if } \rho_1(x,y) \leq k_1, \text{ then } \rho_2(v(x), v(y)) \leq d_1,$$
$$\text{if } \rho_1(x,y) > k_2, \text{ then } \rho_2(v(x), v(y)) > d_2.$$

The less is $(k_2 - k_1)$, the better is the approximation

The result can be formulated as:

Theorem

For $w \geq 6$, $k_1 \geq 1$, $q_1 = 2w/3$, $n > w(k_1+1)+1$, $\Delta q = \frac{1}{2}(-7 + \sqrt{57 + 16(w - q_1)})$, $Q = (\Delta q + 1)(\Delta q + 2)$, $t = w - \Delta q + 1$

$$\text{If } ed(x,y) \leq k_1, \text{ then } D(x,y) \leq \frac{2k_1[w^2 + (n+1)]}{n - w + 1}$$

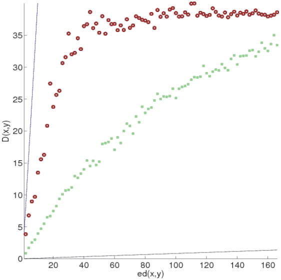
$$\text{If } ed(x,y) > k_2, \text{ then } D(x,y) \geq \frac{Qt(\frac{k_2}{2(\Delta q+1)} - 2)}{(n-w+1)(\Delta q + 1)}.$$

Comparison and experimental illustration

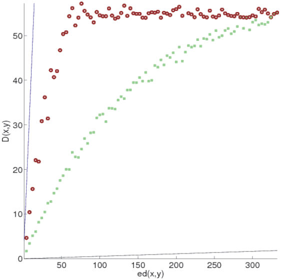
n – sequence length k – approximation parameter

ref	spaces	k_1	k_2	size
[Andoni et al, 03]	$ed \to \ell_1$	distortion $> \frac{3}{2}$		–
[Batu et al, 03]	$ed \to ed$	$O(n^\alpha)$	$\Omega(n)$	$\tilde{O}(n^{\max(\frac{\alpha}{2}, 2\alpha-1)})$
[Bar-Yossef et al, 04]	$ed \to$ Hamm.	k	$(kn)^{2/3}$	$O(1)$
[Ostrovsky et al, 05]	$ed \to \ell_1$	k	$k2^{O(\sqrt{\log n \log\log n})}$	$O(n^2)$
this talk	$ed \to \ell_1$	k	$k\sqrt{n}$	$O(n^{5/4})$

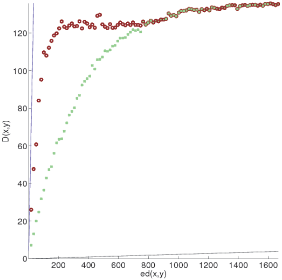
Numerical experiment



$n = 5000$



$n = 10000$



$n = 50000$

Randomized embedding and NN-search

Approximate nearest neighbours

- approximate neighbours are often enough in applications
- data is often known with some accuracy
- «curse of dimensionality» for exact nearest neighbours

(k_1, k_2)-nearest neighbour task

(k_1, k_2)-NN

Given:

- $P \subset \Sigma^n$ – string set
- $k_1 < k_2$ – parameters
- $z \in \Sigma^n$ – query

Task:
If $\exists x \in P$, such that $ed(x, z) \leq k_1$, then return any $y \in P$, such that $ed(y, z) \leq k_2$

Locality-sensitive hash function for ed

Definition [Indyk, Motwani, 98]

A family $H = \{h : (X, \rho) \rightarrow Y\}$ is locality-sensitive for metrics ρ, if for $\forall x, y \in X$ and any i.i.d. $h \in H$ holds:

$$\text{if } \rho(x,y) \leq k_1, \text{ than } Prob[h(x) = h(y)] > p_1,$$
$$\text{if } \rho(x,y) > k_2, \text{ than } Prob[h(x) = h(y)] < p_2,$$

$$k_1 < k_2, p_1 > p_2$$

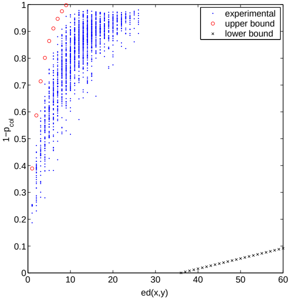
Construction of the locality-sensitive hash function for ed:

- i – independent uniform random value from $[1, \ldots, n - w + 1]$
- v_{q_1,q_2} – concatenation of q-gram vectors from window $x[i, i + w - 1]$ for $q = q_1, \ldots, q_2$
- ϕ – random Cauchy vector, $p(x) = \frac{1}{\pi(1+x^2)}$
- $b \in \mathbb{R}$ – uniform random value from $[0, r]$.

Locality-sensitive family of hash functions

$$h(x) = \left\lfloor \frac{(v_{q_1,q_2}(x[i, i + w - 1]), \phi) + b}{r} \right\rfloor$$

Collision probability



$n = 1000$

Searching for (k_1, k_2)-nearest neighbours

Using

- results from deterministic embedding
- Cauchy distribution properties,

it is possible to show that $h(x)$ is a locality-sensitive function for ed

NN search algorithm

1. For $\forall x \in P$ create L vectors $h^j(x) = (h_{1j}(x), h_{2j}(x), \ldots, h_{Kj}(x))$
2. Memorize string x in all cells with «addresses» $h^j(x)$
3. For a given query z select up to $2L$ strings from cells $h^j(z), j = 1, \ldots, L$
4. If for some corresponding string x_i in the selected cells holds, $ed(x_i, z) < k_2$
   $\Rightarrow$ this is a neighbour

Theorem

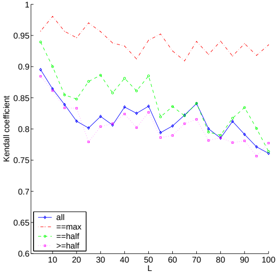
If $K = \log_{1/p_2} |P|, L = |P|^{\frac{\ln p_1}{\ln p_2}}$, then with probability $> 1/2$ this algorithm finds a $(k_1, O(\alpha k_1 n^{1/3} \ln n))$-nearest neighbour using time $O(|P|^{\frac{1}{1+\alpha}}), \alpha > 1$.

Numeric experiments on a random dataset

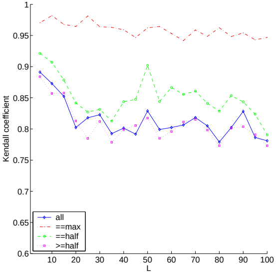
Precision vs. L and $|S|$ (number of retrieved NN-candidates)

L	$\lvert S\rvert = \frac{1}{2}L$	σ_p	$\lvert S\rvert = L$	σ_p	$\lvert S\rvert = 2L$	σ_p	$\lvert S\rvert = 3L$	σ_p	$\lvert S\rvert = 4L$	σ_p
1			0.950	0.048	0.945	0.029	0.927	0.025	0.893	0.031
2	0.930	0.065	0.895	0.056	0.853	0.054	0.825	0.032	0.810	0.028
3			0.885	0.050	0.816	0.035	0.784	0.030	0.770	0.025
4	0.855	0.071	0.842	0.041	0.810	0.031	0.777	0.023	0.757	0.021
5			0.824	0.039	0.786	0.021	0.759	0.014	0.735	0.014
6	0.853	0.052	0.797	0.040	0.782	0.025	0.760	0.019	0.730	0.014
8	0.846	0.038	0.791	0.024	0.755	0.016	0.729	0.013	0.724	0.010
10	0.860	0.030	0.787	0.024	0.749	0.015	0.722	0.009	0.689	0.008
20	0.811	0.024	0.762	0.014	0.708	0.006	0.682	0.005	0.658	0.004

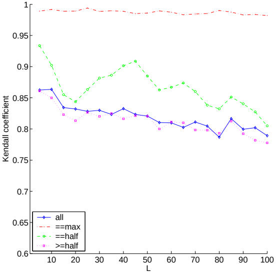
Quality of ordering in S



$|S| = 50$



$|S| = 100$



$|S| = 500$

Web-page duplicate detection

Yandex.ru dataset

- ~ 800000 web-pages $\simeq$ 0.3% of the Russian segment of the Internet
- 10 mln. pairs of duplicates – «ground truth»
- recall $r = \frac{\text{num. of found duplicates}}{\text{num. of existent duplicates}}$
- precision $p = \frac{\text{num. of found duplicates}}{\text{num. of found documents}}$

Results

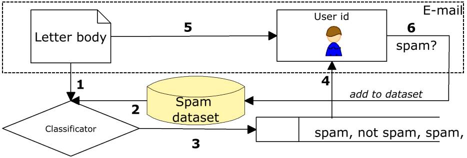
reference	document similarity	precision p	recall r	$F = \frac{2rp}{r+p}$
[Kuznetsov, 05]				0.14-0.49
[Kosinov, 07]	0.85	0.92	0.37	0.53
	0.90	0.92	0.42	0.58
	0.95	0.87	0.50	0.64
	1.00	0.48	0.91	0.63
this talk	0.85	0.78	0.57	0.66
	0.90	0.82	0.69	0.75
	0.95	0.81	0.83	0.82
	1.00	0.87	0.91	0.89

Spam volume assessment

Spam

- 80-85% of all e-mail
- abundancy of similar/identical spam
- word distortions to fool stat. filters:
  'mortgage' 'buy viagra'
  'm0rtg@ge' '6uy v1agraa'

Classification



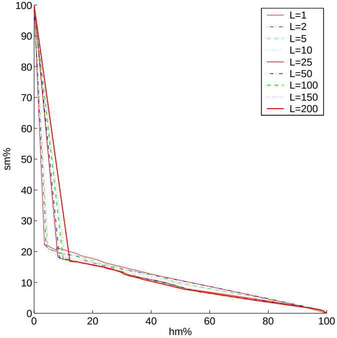
Result

About 80% of correctly classified spam with 5% misclassified ham

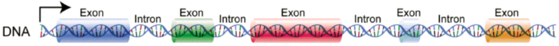
TREC Spam Track 2006 dataset

- ~ 38000 letters (189Mb)
- spam/ham ratio – 66%/33%
- sm% – false negative
- hm% – false positive

sm% vs hm%



Coding regions in DNA



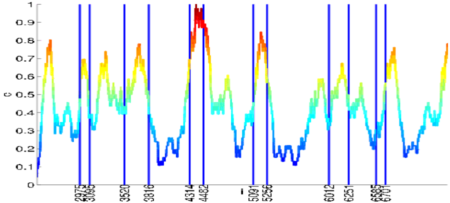
$\Sigma = \{A, T, G, C\}$ – nucleotides

Genetic data

- 10^{10} Mb/year
- GenBank doubles annually
- up to $3.2 \cdot 10^9$ symbols in a sequence

DNA datasets

- training data HMR195 (948 exons)
- test data BursetGuigo (570 seq.)



Coding regions search method

- z — query, training exon
- t — test sequence
- $c_j, \ j = 1, \ldots, |t|$ — counters $t[j]$
- $P = \{t[i, i + |z| - 1]\}$ — database
- S — nearest neighbours candidate set
- if $ed(t[i', i' + |z| - 1], z) = \min_{x \in S} ed(x, z)$, then $c_i = c_i + 1$, $i = i', \ldots, i' + |z| - 1$
- T – threshold on the value of c_i — if $c_i \geq T$, then $t[i]$ belongs to an exon

Quality measure

$$AC = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TP}{TP + FP} + \frac{TN}{TN + FP} + \frac{TN}{TN + FN}\right) - 1$$

Result

reference	AC	time on one PC
[Costello, 03]	0.49	~ 6 years (est. class. alg.)
this talk	0.47	70 hours

User session classification

Intrusion detection

- $\Sigma = \{\text{'ls'}, \text{'mail'}, \text{'rm'}, \ldots\}$, $|\Sigma| \sim 10^3$
- $\sim 10^3$ processes/hour
- anomaly – unusual behaviour

FreeBSD audit-session dataset

- collect time – ~ 3 year
- >500 users
- ~ 20 mln. commands

Method

- U user set
- u^* real user
- t his test session
- $c_u,\ u \in U$ counters

Result

n	K	L		
		1	5	10
10	5	0.470	0.984	0.997
	7	0.431	0.945	0.987
20	5	0.440	0.942	0.983
	7	0.403	0.741	0.942
40	5	0.354	0.848	0.967
	7	0.230	0.390	0.836

- P_u datasets for $\forall u \in U$ (all sliding windows of their sessions)
- $z = t[i, i+n-1]$ query, window contests of the current session
- S_u set of nearest windows found in P_u
- if $S_u \neq \emptyset$, then $c_u = c_u + 1$, if $c_{u^*} = \max_u c_u$, then there is no anomaly

Markov chains with variable memory length

Ron, Singer, Tishby, 95

Σ – alphabet, Q – states, $s \in \Sigma^*$ – state label.

Probabilistic Suffix Automaton (PSA) –
$< Q, \Sigma, \tau, \gamma, \pi >$, where

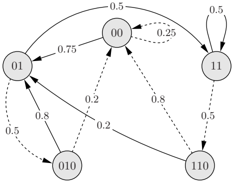
- $\tau \colon Q \times \Sigma \to Q$ – transition function,
- $\gamma \colon Q \times \Sigma \to [0, 1]$ – symbol emission probability,
- $\pi \colon Q \to [0, 1]$ – initial state distribution.

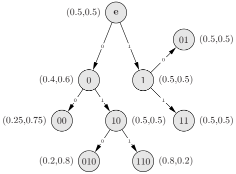
For $q^1, q^2 \in Q$, $\forall \sigma \in \Sigma$, if $\tau(q^1, \sigma) = q^2$ i q^1 has label s^1, then q^2 has label s^2, which is a suffix of $s^1\sigma$.

Probabilistic Suffix Tree (PST):

- edges correspond to symbols of Σ,
- each node has (s, γ_s), where s is a «descend label»,
- $\gamma_s \colon \Sigma \to [0, 1]$ – symbol probability.

Examples





Modified learning algorithm

Empiric probabilities

$$\chi_j(s) = [[r_{j-|s|+1} \dots r_j = s]]$$

$$\tilde{P}(s) = \frac{1}{n-L+1} \sum_{j=L}^{n-1} \chi_j(s),$$

$$\tilde{P}(\sigma|s) = \frac{\sum_{j=L}^{n-1} \chi_{j+1}(s\sigma)}{\sum_{j=L}^{n-1} \chi_j(s)}.$$

1. Take tree $\bar{T}_{t-1}$, current sequence r_t and update:
   $$\tilde{P}_t(s) = \alpha \tilde{P}_{t-1}(s) + (1-\alpha) P'_t(s),$$
   where $P'_t(s)$ – empiric probability of s in r_t.
2. Delete all states such that: $\tilde{P}_t(s) < (1-\epsilon_1)\epsilon_0$
3. $\bar{S} = \{s | s \in \Sigma^*, suffix(s) \in \mathcal{L}(\hat{T}_{t-1}), \tilde{P}_t(\sigma) \geq (1-\epsilon_1)\epsilon_0\}$, where, $\mathcal{L}(\hat{T}_{t-1})$ is the set of leaves of $\hat{T}_{t-1}$.
4. While $\bar{S} \neq \varnothing$ choose any $s \in \bar{S}$ and
   1. delete s from $\bar{S}$
   2. if $\exists \sigma \in \Sigma$ such that $\tilde{P}_t(\sigma|s) \geq (1+\epsilon_2)\gamma_{min}$, and $\frac{\tilde{P}_t(\sigma|s)}{\tilde{P}_t(\sigma|suffix(s))} > 1 + 3\epsilon_2$, add node s to the tree.
   3. if $|s| < L$, then for $\forall \sigma' \in \Sigma$, if $\tilde{P}(\sigma' s) \geq (1-\epsilon_1)\epsilon_0$, add $\sigma' s$ to $\bar{S}$.

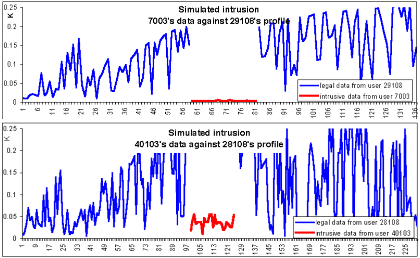
For $\forall \epsilon > 0$, $0 < \delta < 1$, $\exists \alpha(\epsilon, \delta) : 0 < \alpha < 1$ and sufficiently large $t \geq t_0$, such that with the update rule

$$S_t = \alpha^t X_0 + (1-\alpha) \sum_{\tau=1}^{t} \alpha^{t-\tau} X_\tau,$$

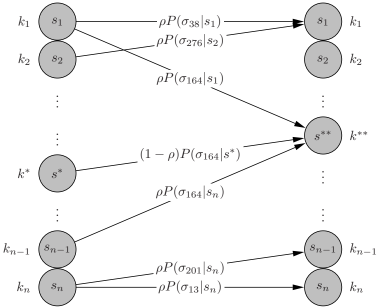
the following holds:

$$P\{|S_t - MX_t| \leq \epsilon\} \geq 1 - \delta.$$

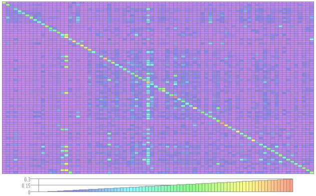
User substitution

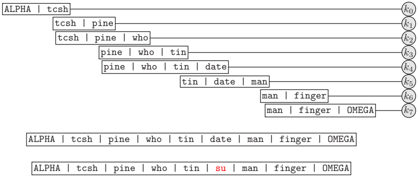


Replay-attacks



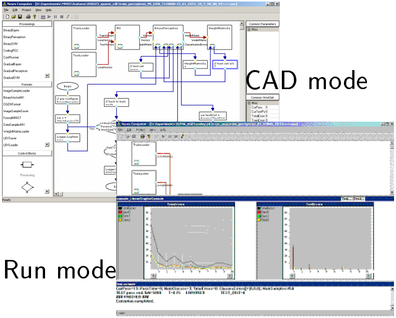
Cross test





Other projects

Software NeuroComputer



CAD mode

Run mode

Context vectors

Text classification, semantic search, TOEFL,...

Dynamic Routing Server

Quality-of-Service routing in a network

The End

Thank you!