Analytic Pattern Matching: From DNA to Twitter
(With Open Problems)

Wojciech Szpankowski*
Purdue University
W. Lafayette, IN 47907

June 28, 2016

Center for Science of Information
NSF Science & Technology Center

AofA, Kraków, 2016

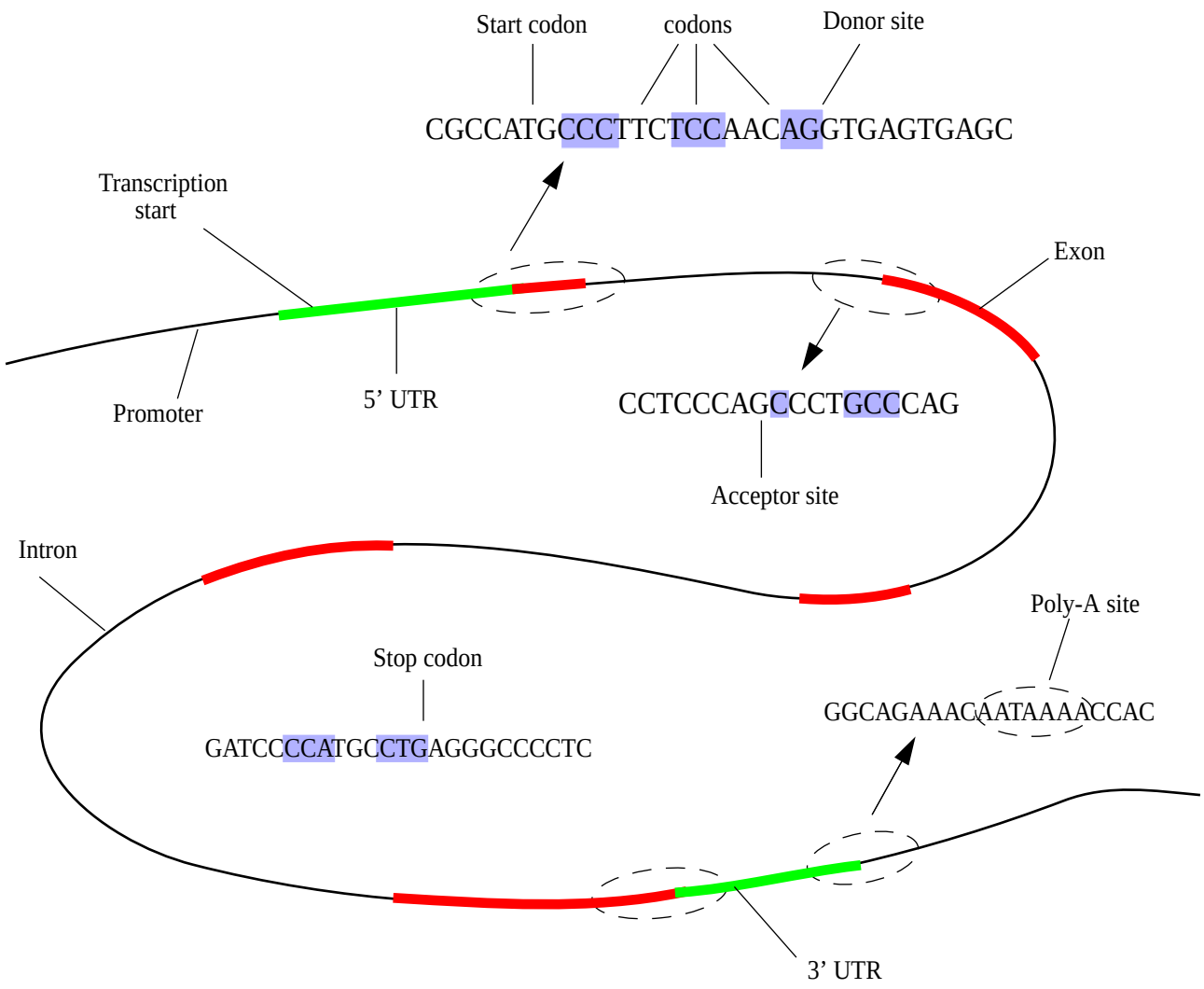
*Joint work with Philippe Jacquet.

Outline

1. Motivations
   - Finding Biological Signals
   - Searching Google
   - Classifying Twitter
2. Pattern Matching Problems
   - Exact String Matching
   - Generalized String Matching
   - Subsequence String Matching (Hidden Patterns)
3. Analysis & Applications
   - Exact String Matching (**Warmup**)
   - Generalized String Matching & Biological Motifs (**Open Problem**)
   - Hidden Patterns & Ranking Google Pages (**Open Problems**)
   - Joint String Complexity & Classification of Twitter (**Open problems**)

Motivation – Biology & String Matching

Biological world is highly stochasticand inhomogeneous (S. Salzberg).



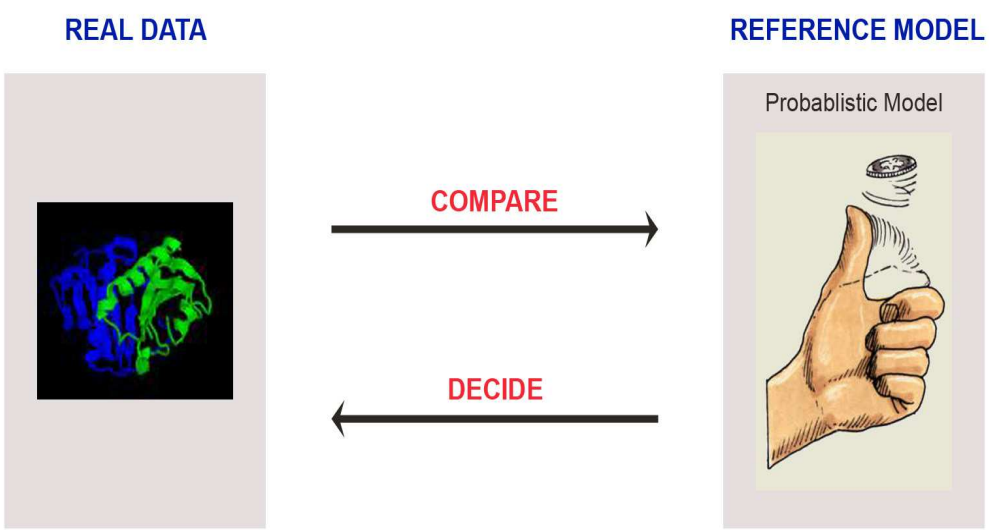
Pattern Matching

Let $\mathcal{W} = w_1 \ldots w_m$ and T be strings over a finite alphabet $\mathcal{A}$.

Basic question: **how many times $\mathcal{W}$ occurs in T**.

Define $O_n(\mathcal{W})$ — the number of times $\mathcal{W}$ occurs in T, that is,

$$O_n(\mathcal{W}) = \#\{i : \; T_{i-m+1}^{i} = \mathcal{W}, \; m \leq i \leq n\}.$$



Basic Thrust of our Approach

When searching for over-represented or under-represented patterns we must assure that such a pattern is not generated by randomness itself (to avoid too many false positives).

Motivation – Google & Subsequence Matching

Google

tree probability estimation compression

All Images Shopping Videos News More Search tools

About 22,800,000 results (0.96 seconds)

Scholarly articles for **tree probability estimation compression**

The context-**tree** weighting method: basic properties - Willems - Cited by 808

... arithmetic coding in the H. 264/AVC video **compression** ... - Marpe - Cited by 1228

... with posterior **estimation** for image **compression** and ... - Perlmutter - Cited by 93

Smoothing in Probability Estimation Trees - Semantic Scholar

https://www.semanticscholar.org/...**Probability-Estimation-Trees**.../e6892...

Smoothing is a technique used to improve the **probability estimates**. There are several ... This thesis migrates smoothing methods from text **compression** to PETs.

Smoothing in Probability Estimation Trees

researchcommons.waikato.ac.nz/handle/.../5701 University of Waikato

Smoothing in **Probability Estimation Trees**. Research ... In the field of text **compression**, PPM in particular, smoothing methods play a important role. This thesis ...

Huffman coding - Wikipedia, the free encyclopedia

https://en.wikipedia.org/wiki/Huffman_coding Wikipedia

Huffman **tree** generated from the exact frequencies of the text "this is an example of a ... type of optimal prefix code that is commonly used for lossless data **compression**. ..... the coding **tree** structure to match the updated **probability estimates**.

Motivation – Twitter & String Complexity

"allow users to download an entire movie in one second." I need this http://t.co/3fbNfKEkah
Green energy boss accuses Govt of obstructing renewable energy development http://t.co/v5Lq2Jx1GQ

Figure 1: Two similar twitter texts have many common words

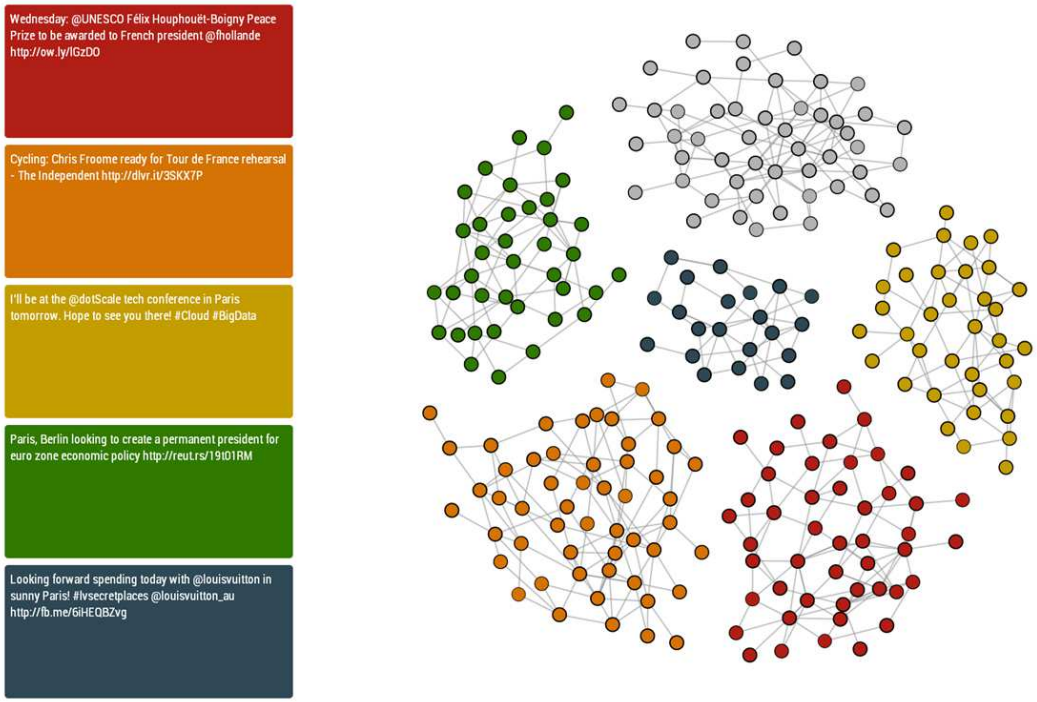


Figure 2: Twitters Classification

Outline Update

1. Motivations
2. Pattern Matching Problems
   - Exact String Matching
   - Generalized String Matching
   - Subsequence String Matching
3. Analysis & Applications

Pattern Matching

Let $\mathcal{W}$ and T be (set of) strings generated over a finite alphabet $\mathcal{A}$.

We call $\mathcal{W}$ the pattern and T the text. The text T is of length n and is generated by a probabilistic source.

Text: $T_m^n = T_m \dots T_n$.
Pattern: $\mathcal{W} = w_1 \dots w_m, \quad w_i \in \mathcal{A}$;
Set of patterns: $\mathcal{W} = \{\mathcal{W}_1, \dots, \mathcal{W}_d\}$ with $\mathcal{W}_i \in \mathcal{A}^{m_i}$.

Basic question:

how many times $\mathcal{W}$ occurs in T (or how long to wait until $\mathcal{W}$ occurs in T).

Define

$$O_n(\mathcal{W}) = \#\{i : \ T_{i-m+1}^i = \mathcal{W}, \ m \le i \le n\}$$

as the number of w occurrences in the text T_1^n.

Our goal: Study probabilistic behavior of $O_n(\mathcal{W})$ for various pattern matching problems using tools of analytic combinatorics.

Variations on Pattern Matching

(Exact) String Matching: In the exact string matching the pattern

$$\mathcal{W} = w_1 \dots w_m$$

is a given string (i.e., consecutive sequence of symbols).

Generalized String Matching: In the generalized pattern matching a set of patterns (rather than a single pattern) is given, that is,

$$\mathcal{W} = (\mathcal{W}_0, \mathcal{W}_1, \dots, \mathcal{W}_d), \quad \mathcal{W}_i \in \mathcal{A}^{m_i}$$

where $\mathcal{W}_i$ itself for $i \geq 1$ is a subset of $\mathcal{A}^{m_i}$.
The set $\mathcal{W}_0$ is called the forbidden set.

Three cases to be considered:

$\mathcal{W}_0 = \emptyset$ — interest in the number of patterns from $\mathcal{W}$ occurring in the text.

$\mathcal{W}_0 \neq \emptyset$ — we study the number of $\mathcal{W}_i$, $i \geq 1$ pattern occurrences under the condition that no pattern from $\mathcal{W}_0$ occurs in the text.

$\mathcal{W}_i = \emptyset$, $i \geq 1$, $\mathcal{W}_0 \neq \emptyset$ — restricted pattern matching.

Pattern Matching Problems

Hidden Words or Subsequence Pattern Matching: We search for a subsequence $\mathcal{W} = w_1 \ldots w_m$ rather than a string in a text.
That is, there are indices $1 \leq i_1 < i_2 < \cdots < i_m \leq n$ such that

$$T_{i_1} = w_1,\ T_{i_2} = w_2, \cdots,\ T_{i_m} = w_m.$$

We also say that the word $\mathcal{W}$ is "hidden" in the text.

For example:

$$\begin{aligned} \mathcal{W} &= \mathbf{date} \\ T &= \mathbf{hidden\ pattern} \end{aligned}$$

occurs four times as a subsequence in the text as hidden pattern.

Joint String Complexity: For a given string X, we ask how many distinct subwords it contains. This is called string complexity.

For two strings X and Y, we want to know how many common and distinct subwords they contain. This is called joint string complexity.

Outline Update

1. Motivations
2. Pattern Matching Problems
3. Analysis & Applications
   - Exact String Matching
   - Generalized String Matching & Finding Biological Motifs
   - Hidden Patterns & Ranking Google Pages
   - Joint String Complexity & Classification of Twitter

Warmup: Analysis of Exact String Matching

For Language $\mathcal{L}$, its generating function (GF) $L(z)$ is

$$L(z) = \sum_{u \in \mathcal{L}} P(u) z^{|u|}.$$

Concatenation of languages translates into product of GFs.
Autocorrelation Polynomial: For $\mathcal{W}$ define the autocorrelation set $\mathcal{S}$ as:

$$\mathcal{S} = \{w_{k+1}^m : \ w_1^k = w_{m-k+1}^m\},$$

and $\mathcal{WW}$ is the set of positions k satisfying $w_1^k = w_{m-k+1}^m$, and

$$S(z) = \sum_{k \in \mathcal{WW}} P(w_{k+1}^m) z^{m-k}.$$

Example: For $\mathcal{W} = abab$, we have $\mathcal{WW} = \{1, 3\}$ and $\mathcal{S} = \{\epsilon, ab\}$.

w:	a	b	a	b		
w:	a	b	a	b	ϵ	
$w_1^2 = w_2^4$			a	b	a	b

Warmup: Analysis of Exact String Matching

For Language $\mathcal{L}$, its generating function (GF) $L(z)$ is

$$L(z) = \sum_{u \in \mathcal{L}} P(u) z^{|u|}.$$

Concatenation of languages translates into product of GFs.
Autocorrelation Polynomial: For $\mathcal{W}$ define the autocorrelation set $\mathcal{S}$ as:

$$\mathcal{S} = \{w_{k+1}^m : \; w_1^k = w_{m-k+1}^m\},$$

and $\mathcal{WW}$ is the set of positions k satisfying $w_1^k = w_{m-k+1}^m$, and

$$S(z) = \sum_{k \in \mathcal{WW}} P(w_{k+1}^m) z^{m-k}.$$

Example: For $\mathcal{W} = abab$, we have $\mathcal{WW} = \{1, 3\}$ and $\mathcal{S} = \{\epsilon, ab\}$.

w:	a	b	a	b		
w:	a	b	a	b	ϵ	
$w_1^2 = w_2^4$			a	b	a	b

Define $\mathcal{T}_r$ as set of words containing exactly $r \geq 1$ occurrences of $\mathcal{W}$:

$$\mathcal{T}_r = \mathcal{R} \cdot \mathcal{M}^{r-1} \cdot \mathcal{U}.$$

which can be illustrated for $\mathcal{T}_4$ as follows

Language Relations & Generating Functions

Lemma 1. (i) *The languages* $\mathcal{M}, \mathcal{U}$ *and* $\mathcal{R}$ *satisfy:*

$$\begin{aligned} \mathcal{U} \cdot \mathcal{A} &= \mathcal{M} + \mathcal{U} - \{\epsilon\}, \\ \bigcup_{k \geq 1} \mathcal{M}^k &= \mathcal{A}^* \cdot \mathcal{W} + \mathcal{S} - \{\epsilon\}, \quad \mathcal{W} \cdot \mathcal{M} = \mathcal{A} \cdot \mathcal{R} - (\mathcal{R} - \mathcal{W}). \end{aligned}$$

Language Relations & Generating Functions

Lemma 1. (i) *The languages* $\mathcal{M}, \mathcal{U}$ *and* $\mathcal{R}$ *satisfy:*

$$\begin{aligned} \mathcal{U} \cdot \mathcal{A} &= \mathcal{M} + \mathcal{U} - \{\epsilon\}, \\ \bigcup_{k \geq 1} \mathcal{M}^k &= \mathcal{A}^* \cdot \mathcal{W} + \mathcal{S} - \{\epsilon\}, \quad \mathcal{W} \cdot \mathcal{M} = \mathcal{A} \cdot \mathcal{R} - (\mathcal{R} - \mathcal{W}). \end{aligned}$$

(ii) *For* **memoryless source** *the generating functions associated with languages* $\mathcal{M}, \mathcal{U}$ *and* $\mathcal{R}$ *satisfy*

$$\begin{aligned} U(z) &= \frac{M(z) - 1}{z - 1} \\ \frac{1}{1 - M(z)} &= S_{\mathcal{W}}(z) + P(\mathcal{W}) \frac{z^m}{1 - z}, \quad R(z) = P(\mathcal{W}) z^m \cdot U(z) \end{aligned}$$

Language Relations & Generating Functions

Lemma 1. (i) *The languages* $\mathcal{M}, \mathcal{U}$ *and* $\mathcal{R}$ *satisfy:*

$$\begin{aligned} \mathcal{U} \cdot \mathcal{A} &= \mathcal{M} + \mathcal{U} - \{\epsilon\}, \\ \bigcup_{k \geq 1} \mathcal{M}^k &= \mathcal{A}^* \cdot \mathcal{W} + \mathcal{S} - \{\epsilon\}, \quad \mathcal{W} \cdot \mathcal{M} = \mathcal{A} \cdot \mathcal{R} - (\mathcal{R} - \mathcal{W}). \end{aligned}$$

(ii) *For* **memoryless source** *the generating functions associated with languages* $\mathcal{M}, \mathcal{U}$ *and* $\mathcal{R}$ *satisfy*

$$\begin{aligned} U(z) &= \frac{M(z) - 1}{z - 1} \\ \frac{1}{1 - M(z)} &= S_{\mathcal{W}}(z) + P(\mathcal{W}) \frac{z^m}{1 - z}, \quad R(z) = P(\mathcal{W}) z^m \cdot U(z) \end{aligned}$$

(iii) *The generating functions* $T_r(z) = \sum_{n \geq 0} \Pr\{O_n(\mathcal{W}) = r\} z^n$ *and* $T(z, u) = \sum_{r=1}^{\infty} T_r(z) u^r$ *satisfy*

$$\begin{aligned} T_r(z) &= R(z) M_{\mathcal{W}}^{r-1}(z) U(z), \quad r \geq 1, \quad T_0(z) = \frac{S(z)}{(1 - z) S(z) + z^m P(w)}, \\ T(z, u) &= R(z) \frac{u}{1 - u M(z)} U(z). \end{aligned}$$

Main Results: Moments and Limiting Distributions

Moments: We have

$$\mathbf{E}[O_n(\mathcal{W})] = P(\mathcal{W})(n-m+1), \quad \mathbf{Var}[O_n(\mathcal{W})] = nc_1 + c_2$$

with

$$c_1 \quad = \quad P(\mathcal{W})(2S(1) - 1 - (2m-1)P(\mathcal{W}). - (m-1)(2S(1)-1) - 2S'(1)).$$

Limiting Distributions: The limiting distribution of $P(O_n(\mathcal{W}) = k)$ depends on the relation between n and $m = |w|$.

$$P(O_n(\mathcal{W}) = k) = \begin{cases} \text{CLT, LLC, LD} & nP(w) \to \infty, \ w \text{ is given}, \\ \mathrm{Po}(\tilde{\lambda}) \star \mathrm{Geom}(\theta) & nP(w) \to \lambda > 0, \ mP(w) \to 0, \\ \text{(Polýa-Aeppli)} & \tilde{\lambda} = nP(w)/S(1), \ \theta = (S(1)-1)/S(1) \\ \frac{nP(w)}{S^2(1)} \left(\frac{S(1)-1}{S(1)}\right)^k & nP(w) \to 0, \ m = o(n), \\ \frac{n(1-\alpha)P(w)}{S^2(1)} \left(\frac{S(1)-1}{S(1)}\right)^k & m = \alpha n. \end{cases}$$

Sketch of Proofs

1. By Cauchy formula we have

$$\mathbf{E}[u^{O_n}] = [z^n]T(z,u) = \frac{1}{2\pi i}\oint \frac{R(z)U(z)}{(1-uM(z))}\frac{dz}{z^n}.$$

2. For CLT and LD we apply residue theorem to find for large $R > 1$

$$\mathbf{E}[u^{O_n}] = C(u)\rho(u)^{n-m+1} + O(R^{-n})$$

where $\rho(u)$ is the root of $1 - uM(z) = 0$.

3. For Poisson law we assume $nP(w) \to \lambda$ and $mP(w) \to 0$ to find that $\rho(u) \approx 1$. Hence, using

$$\mathbf{E}[u^{O_n}] = [z^n]\left(T(z,u) + T_0(z)\right)$$

we are led to

$$\begin{aligned}\mathbf{E}[u^{O_n}] &= \exp\left(\frac{nP(w)}{S(1)}\cdot\frac{z-1}{1-z\frac{S(1)-1}{S(1)}}\right)(1+O(mP(w))\,, \quad nP(w)\to\lambda \\ &= 1+\frac{nP(w)}{S(1)}\frac{z-1}{1-z\frac{S(1)-1}{S(1)}} + o(nP(w)), \quad nP(w)\to 0.\end{aligned}$$

Open Problem

Open Problem 1:
Extend String Pattern Matching to Graph Pattern Matching, that is, replace the text T by a graph G and find the number of occurrences of a given subgraph/pattern g in G for different models of graph G generation.

Attention: Watch out for group automorphism of g!

Open Problem

Open Problem 1:
Extend String Pattern Matching to Graph Pattern Matching, that is, replace the text T by a graph G and find the number of occurrences of a given subgraph/pattern g in G for different models of graph G generation.

Attention: Watch out for group automorphism of g!

Exact graph matching was analyzed in:
Picard F, Daudin JJ, Schbath S, Robin S, "Assessing the Exceptionality of Network Motifs". *J. Comp. Bio.*, 2008.

Open Problem

Open Problem 1:
Extend String Pattern Matching to Graph Pattern Matching, that is, replace the text T by a graph G and find the number of occurrences of a given subgraph/pattern g in G for different models of graph G generation.

Attention: Watch out for group automorphism of g!

Exact graph matching was analyzed in:
Picard F, Daudin JJ, Schbath S, Robin S, "Assessing the Exceptionality of Network Motifs". *J. Comp. Bio.*, 2008.

Open Problem 2:
Extend String Pattern Matching to other Adanced Structural Pattern Matching, such as trees, sets, and multimodal data structures.

Outline Update

1. Motivations
2. Pattern Matching Problems
3. Analysis & Applications
   - Exact String Matching
   - Generalized String Matching & Finding Biological Motifs
   - Hidden Patterns & Ranking Google Pages
   - Joint String Complexity & Classification of Twitter

Biology: Weak Signals and Generalized Pattern Matching

Denise and Regnier (2002) observed that in biological sequence whenever a word is overrepresented, then its subwords are also overrepresented.
For example, if $\mathcal{W}_1 = AATAAA$, then

$$\mathcal{W}_2 = ATAAAN$$

is also overrepresented.
Overrepresented subwords are called artifact, and it is important to disregard automatically noise created by artifacts.

New Approach:

Once a dominating signal has been detected, we look for a weaker signal by comparing the number of observed occurrences of patterns to the conditional expectations **not** the regular expectations.

Biology: Weak Signals and Generalized Pattern Matching

Denise and Regnier (2002) observed that in biological sequence whenever a word is overrepresented, then its subwords are also overrepresented.
For example, if $\mathcal{W}_1 = AATAAA$, then

$$\mathcal{W}_2 = ATAAAN$$

is also overrepresented.
Overrepresented subwords are called artifact, and it is important to disregard automatically noise created by artifacts.

New Approach:

> Once a dominating signal has been detected, we look for a weaker signal by comparing the number of observed occurrences of patterns to the conditional expectations **not** the regular expectations.

This harder question needs a new approach thru Generalized Pattern Matching to show that

$$\mathbf{E}[O_n(\mathcal{W}_2)|O_n(\mathcal{W}_1) = k] \sim \alpha n.$$

To evaluate this we need to study **generalized pattern matching**.

Polyadenylation Signals in Human Genes

Beaudoing et al. (2000) studied several variants of the well known AAUAAA polyadenylation signal in mRNA of humans genes.

Using our approach Denise and Regnier (2002) discovered/eliminated all artifacts and found new signals in a much simpler and reliable way.

Hexamer	Obs.	Rk	Exp.	Z-sc.	Rk	Cd.Exp.	Cd.Z-sc.	Rk
AAUAAA	3456	1	363.16	167.03	1			1
AAAUAA	1721	2	363.16	71.25	2	1678.53	1.04	1300
AUAAAA	1530	3	363.16	61.23	3	1311.03	6.05	404
UUUUUU	1105	4	416.36	33.75	8	373 .30	37.87	2
AUAAAU	1043	5	373.23	34.67	6	1529.15	12.43	4078
AAAAUA	1019	6	363.16	34.41	7	848.76	5.84	420
UAAAAU	1017	7	373.23	33.32	9	780.18	8.48	211
AUUAAA	1013	I	373.23	33.12	10	385.85	31.93	3
AUAAAG	972	9	184.27	58.03	4	593.90	15.51	34
UAAUAA	922	10	373.23	28.41	13	1233.24	–8.86	4034
UAAAAA	922	11	363.16	29.32	12	922.67	9.79	155
UUAAAA	863	12	373.23	25.35	15	374.81	25.21	4
CAAUAA	847	13	185.59	48.55	5	613.24	9.44	167
AAAAAA	841	14	353.37	25.94	14	496.38	15.47	36
UAAAUA	805	15	373.23	22.35	21	1143.73	–10.02	4068

Outline Update

1. Motivations
2. Pattern Matching Problems
3. Analysis & Applications
   - Exact String Matching
   - Generalized String Matching & Finding Biological Motifs
   - Hidden Patterns & Ranking Google Pages
   - Joint String Complexity & Classification of Twitter

Hidden Patterns

In the subsequence pattern or a hidden word $\mathcal{W}$ occurs as a subsequence:

$$T_{i_1} = w_1, \; T_{i_2} = w_2, \; \ldots, \; T_{i_m} = w_m.$$

where we put additional constraints that

$$i_{j+1} - i_j \leq d_j.$$

The $I = (i_1, \ldots, i_m)$-tuple is called a position and $\mathcal{D} = (d_1, \ldots, d_m$ constitutes the constraints.

Let $O_n(\mathcal{W})$ be the number of $\mathcal{W}$ occurrences in T. Observe that

$$O_n(\mathcal{W}) = \sum_I X_I$$

where

$$X_I := 1 \;\; if \;\; \mathcal{W} \text{ occurs at position } I \text{ in } T_n$$

and 0 otherwise.

Hidden Patterns

In the subsequence pattern or a hidden word $\mathcal{W}$ occurs as a subsequence:

$$T_{i_1} = w_1, \; T_{i_2} = w_2, \; \ldots, \; T_{i_m} = w_m.$$

where we put additional constraints that

$$i_{j+1} - i_j \leq d_j.$$

The $I = (i_1, \ldots, i_m)$-tuple is called a position and $\mathcal{D} = (d_1, \ldots, d_m$ constitutes the constraints.

Let $O_n(\mathcal{W})$ be the number of $\mathcal{W}$ occurrences in T. Observe that

$$O_n(\mathcal{W}) = \sum_I X_I$$

where

$$X_I := 1 \;\; if \;\; \mathcal{W} \text{ occurs at position } I \text{ in } T_n$$

and 0 otherwise.

If all d_j are finite, then we have the **constrained problem**, which is a **generalized pattern matching**.

Below analysis is based on: P. Flajolet, W.S., and B. Vallee, ICALP 2001 & JACM 2005. (Extension to dynamaic sources by Bourdon and Vallee.)

Constrained HPM and Generalized Pattern Matching

1. The $(\mathcal{W}, \mathcal{D})$ **constrained subsequence problem** is viewed as the generalized string matching.

Example: If $(\mathcal{W}, \mathcal{D}) = a\#_2 b$, then $\mathcal{W} = \{ab, aab, abb\}$.

Constrained HPM and Generalized Pattern Matching

1. The $(\mathcal{W}, \mathcal{D})$ **constrained subsequence problem** is viewed as the generalized string matching.
Example: If $(\mathcal{W}, \mathcal{D}) = a\#_2 b$, then $\mathcal{W} = \{ab, aab, abb\}$.

2. de Bruijn Automaton. Let $M = \max\{length(\mathcal{W})\} - 1$. Define a de Bruijn automaton over $\mathcal{B}$ where

$$\mathcal{B} = \mathcal{A}^M.$$

De Bruijn automaton is built over $\mathcal{B}$.

Constrained HPM and Generalized Pattern Matching

1. The $(\mathcal{W}, \mathcal{D})$ **constrained subsequence problem** is viewed as the generalized string matching.
Example: If $(\mathcal{W}, \mathcal{D}) = a\#_2 b$, then $\mathcal{W} = \{ab, aab, abb\}$.

2. de Bruijn Automaton. Let $M = \max\{length(\mathcal{W})\} - 1$. Define a de Bruijn automaton over $\mathcal{B}$ where

$$\mathcal{B} = \mathcal{A}^M.$$

De Bruijn automaton is built over $\mathcal{B}$.

3. Let $b \in \mathcal{B}$ and $a \in \mathcal{A}$. Then the transition from the state b upon scanning symbol a of the text is to $\hat{b} \in \mathcal{B}$ such that

$$ba \mapsto \hat{b} = b_2 b_3 \cdots b_M a.$$

For example

$$\underbrace{abb}_{\mathcal{B}}\underbrace{a}_{\mathcal{A}} \mapsto \underbrace{bba}_{\mathcal{B}}.$$

4. The Transition Matrix: $\mathbf{T}(u)$ is a complex-valued transition matrix defined as:

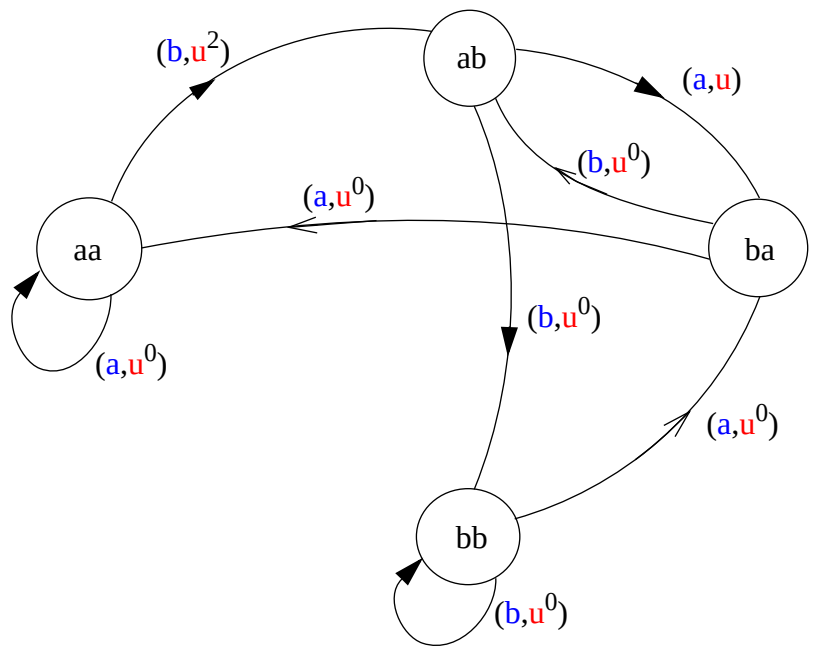
$$[\mathbf{T}(u)]_{b,\hat{b}} := P(a) u^{O_{M+1}(ba) - O_M(b)} [\![\hat{b} = b_2 b_3 \cdots b_M a]\!]$$

where $O_M(b)$ is the number of pattern occurrences $\mathcal{W}$ in the text b.

Example

5. **Example**. Let $\mathcal{W} = \{ab, aab, aba\}$. Then $M = 2$, and the the de Bruijn graph and matrix $\mathbf{T}(u)$ are shown below

$$\mathbf{T}(u) = \begin{array}{c} \\ aa \\ ab \\ ba \\ bb \end{array} \begin{array}{c} \begin{array}{cccc} aa & ab & ba & bb \end{array} \\ \begin{pmatrix} P(a) & P(b)\,u^2 & 0 & 0 \\ 0 & 0 & P(a)\,u & P(b) \\ P(a) & P(b) & 0 & 0 \\ 0 & 0 & P(a) & P(b) \end{pmatrix} \end{array} .$$



For a great analysis of **generalized pattern matching** by symbolic calculus see Clement, Bassimo, and Nicodeme, TALG, 2012.

Generating Functions

6. Using properties of product of matrices we conclude that

$$O_n(u) = \mathbf{E}[u^{O_n(\mathcal{W})}] = \mathbf{b}^t(u)\mathbf{T}^n(u)\vec{1}$$

where $\mathbf{b}^t(u)$ is an initial vector and $\vec{1} = (1, \ldots, 1)$.

7. Spectral Decomposition
Let $\lambda(u)$ be the largest eigenvalue of $\mathbf{T}(u)$. Then for some $A < 1$:

$$O_n(u) = c(u)\lambda^n(u)(1 + O(A^n))$$

Theorem 1 (Flajolet, Vallee, WS). *For fixed $\mathcal{W}$ (i.e., $m = O(1)$):*
Central Limit Theorem:

$$\Pr\left\{\frac{O_n - nP(\mathcal{W})}{\sigma(\mathcal{W})\sqrt{n}} \leq x\right\} \sim \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-t^2/2}$$

Large Deviations: *If $\mathbf{T}(u)$ is primitive, then*

$$\Pr\{O_n(\mathcal{W}) = a\mathbf{E}[O_n]\} \sim \frac{1}{\sigma_a\sqrt{2\pi n}}e^{-nI(a)+\theta_a}$$

where $I(a)$ can be explicitly computed, and θ_a is a known constant.

Generating Functions

6. Using properties of product of matrices we conclude that

$$O_n(u) = \mathbf{E}[u^{O_n(\mathcal{W})}] = \mathbf{b}^t(u)\mathbf{T}^n(u)\vec{1}$$

where $\mathbf{b}^t(u)$ is an initial vector and $\vec{1} = (1, \ldots, 1)$.

7. Spectral Decomposition
Let $\lambda(u)$ be the largest eigenvalue of $\mathbf{T}(u)$. Then for some $A < 1$:

$$O_n(u) = c(u)\lambda^n(u)(1 + O(A^n))$$

Theorem 1 (Flajolet, Vallee, WS). *For fixed $\mathcal{W}$ (i.e., $m = O(1)$):*
Central Limit Theorem:

$$\Pr\left\{\frac{O_n - nP(\mathcal{W})}{\sigma(\mathcal{W})\sqrt{n}} \leq x\right\} \sim \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-t^2/2}$$

Large Deviations: *If $\mathbf{T}(u)$ is primitive, then*

$$\Pr\{O_n(\mathcal{W}) = a\mathbf{E}[O_n]\} \sim \frac{1}{\sigma_a\sqrt{2\pi n}}e^{-nI(a)+\theta_a}$$

where $I(a)$ can be explicitly computed, and θ_a is a known constant.

Question: Extension to large $m \to \infty$? How?

Deletion Channel – Why HPM for Large m

A deletion channel with parameter d:

$$x_1^n = 0010101 \rightarrow \boxed{\text{DELETION CHANNEL}} \rightarrow Y(x_1^n) = x_{i_1} \ldots x_{i_M} \ldots = 0011$$

$$\uparrow_{d}^{\text{deletion}} \qquad M \sim Binom(n, 1-d)$$

input: a binary sequence $x := x_1^n = x_1 \cdots x_n$,
channel: deletes each symbol independently with probability d ,
output: a *subsequence* $Y = Y(x) = x_{i_1} \ldots x_{i_M}$ of x;
M follows the binomial distribution $\mathrm{Bi}(n, (1-d))$ and indices $i_1, \ldots, i_M$ correspond to undeleted bits,
$I(X_1^n, Y(X_1^n))$: mutual information betwen the ouput and the input,
$C = \max_P I(X;Y)$: channel capacity.

Deletion Channel – Why HPM for Large m

A deletion channel with parameter d:

$$x_1^n = 0010101 \rightarrow \boxed{\text{DELETION CHANNEL}} \rightarrow Y(x_1^n) = x_{i_1} \ldots x_{i_M} \ldots = 0011$$

$$\uparrow_{d}^{\text{deletion}} \qquad M \sim Binom(n, 1-d)$$

input: a binary sequence $x := x_1^n = x_1 \cdots x_n$,
channel: deletes each symbol independently with probability d ,
output: a *subsequence* $Y = Y(x) = x_{i_1} \ldots x_{i_M}$ of x;
M follows the binomial distribution $\mathrm{Bi}(n, (1-d))$ and indices $i_1, \ldots, i_M$ correspond to undeleted bits,
$I(X_1^n, Y(X_1^n))$: mutual information betwen the ouput and the input,
$C = \max_P I(X;Y)$: channel capacity.

Theorem 2. *For any random input X_1^n, the mutual information satisfies*

$$I(X_1^n; Y_1^M)) = \sum_{w \in \mathcal{A}^*} d^{n-|w|}(1-d)^{|w|}(\mathbf{E}[O_n(w) \log O_n(w)] - \mathbf{E}[O_n(w)] \log \mathbf{E}[O_n(w)]) .$$

where $O_n(w)$ is the number of hidden patterns w in X_1^n.

Open Problems

Open Problem 3:
Extend the analysis of HPM and GPM to large m; in particular $m = \Theta(n)$.
Open Problem 4:
Analyze HPM and GPM for large alphabet size $|\mathcal{A}|$; e.g., $|\mathcal{A}| = \Theta(n)$.

Open Problems

Open Problem 3:
Extend the analysis of HPM and GPM to large m; in particular $m = \Theta(n)$.
Open Problem 4:
Analyze HPM and GPM for large alphabet size $|\mathcal{A}|$; e.g., $|\mathcal{A}| = \Theta(n)$.

Open Problem 5:
Find mutual information and capacity of the deletion channel.

Open Problems

Open Problem 3:
Extend the analysis of HPM and GPM to large m; in particular $m = \Theta(n)$.
Open Problem 4:
Analyze HPM and GPM for large alphabet size $|\mathcal{A}|$; e.g., $|\mathcal{A}| = \Theta(n)$.

Open Problem 5:
Find mutual information and capacity of the deletion channel.

Some Observations. **1**. Assume $m = \theta n$, $\theta = 1 - d$, and $p = P(1)$:

$$\mathbf{E}[O_n(\mathcal{W})] = \binom{n}{m} P(\mathcal{W})$$

and for a typical $\mathcal{W}$ we have: $\boxed{\mathbf{E}[O_n(\mathcal{W})] \sim 2^{n(H(\theta)-\theta H(p))}}$,

where $H(x)$ is the binary entropy.
2. Let now $\mathcal{W} = 0^m$. Then we can prove

$$P\left(O_n(0^m) = \binom{n-k}{m}\right) = \binom{n}{k} p^k (1-p)^{n-k}$$

and then $\mathbf{Var}[O_n(0^m)] \sim \mathbf{E}[O_n^2(0^m)] \sim 2^{n\beta((1-d),p)}$

$$\beta(\theta, p) = (2(q+\theta p-\delta)H(\theta/(q+\theta p-\delta))+H((1-\theta)p+\delta)+((1-\theta)p+\delta)\log p+(q+\theta p-\delta)$$

Outline Update

1. Motivations
2. Pattern Matching Problems
3. Analysis & Applications
   - Exact String Matching
   - Generalized String Matching & Finding Biological Motifs
   - Hidden Patterns & Intrusion Detection
   - Joint String Complexity & Classification of Twitter

Some Definitions

String Complexity of a single sequence is the number of distinct substrings.

Throughout, we write X for the string and denote by $I(X)$ the set of *distinct substrings* of X over alphabet $\mathcal{A}$.

Example. If $X = aabaa$, then

$$I(X) = \{\epsilon, a, b, aa, ab, ba, aab, aba, baa, aaba, abaa, aabaa\},$$

so $|I(X)| = 12$. But if $X = aaaaa$, then

$$I(X) = \{\epsilon, a, aa, aaa, aaaa, aaaaa\},$$

so $|I(X)| = 6$.

Some Definitions

String Complexity of a single sequence is the number of distinct substrings.

Throughout, we write X for the string and denote by $I(X)$ the set of *distinct substrings* of X over alphabet $\mathcal{A}$.

Example. If $X = aabaa$, then

$$I(X) = \{\epsilon, a, b, aa, ab, ba, aab, aba, baa, aaba, abaa, aabaa\},$$

so $|I(X)| = 12$. But if $X = aaaaa$, then

$$I(X) = \{\epsilon, a, aa, aaa, aaaa, aaaaa\},$$

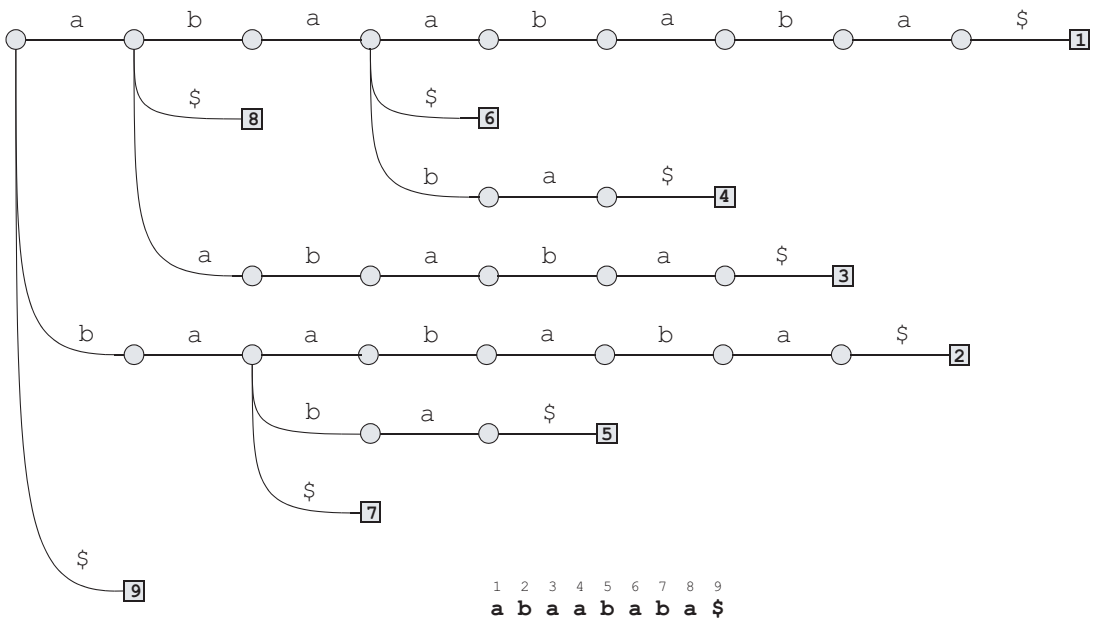
so $|I(X)| = 6$.

The **string complexity** is the cardinality of $I(X)$ and we study here the *average* string complexity

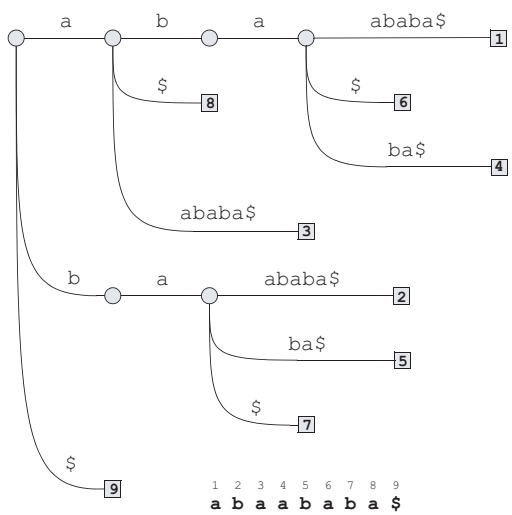
$$\mathbf{E}[|I(X)|] = \sum_{X \in \mathcal{A}^n} P(X)|I(X)|.$$

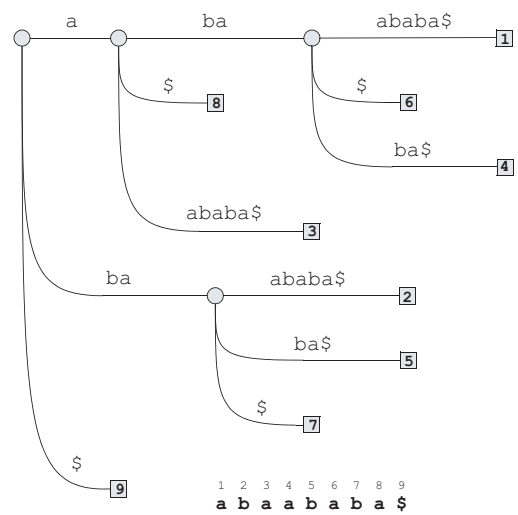
where X is generated by a memoryless/Markov source.

Suffix Trees and String Complexity



Non-compact suffix trie for $X = \texttt{abaababa}$ and string complexity $I(X) = 24$.





String Complexity = # internal nodes in a non-compact suffix tree.

Some Simple Facts

Let $O(w)$ denote the number of times that the word w occurs in X. Then

$$|I(X)| = \sum_{w \in \mathcal{A}^*} \min\{1, O_X(w)\}.$$

Since between any two positions in X there is one and only one substring:

$$\sum_{w \in \mathcal{A}^*} O_X(w) = \frac{(|X|+1)|X|}{2}.$$

Hence

$$|I(X)| = \frac{(|X|+1)|X|}{2} - \sum_{w \in \mathcal{A}^*} \max\{0, O_X(w) - 1\}.$$

Define: $C_n := \mathbf{E}[|I(X)| \mid |X| = n]$. Then

$$\begin{aligned} C_n &= \frac{(n+1)n}{2} - \sum_{w \in \mathcal{A}^*} \sum_{k \geq 2} (k-1) P(O_n(w) = k) \\ &= \frac{(n+1)n}{2} - \sum_{w \in \mathcal{A}^*} \sum_{k \geq 2} k P(O_n(w) = k) + \sum_{w \in \mathcal{A}^*} P(O_n(w) \geq 2). \end{aligned}$$

We need to study probabilistically $O_n(w)$: that is:

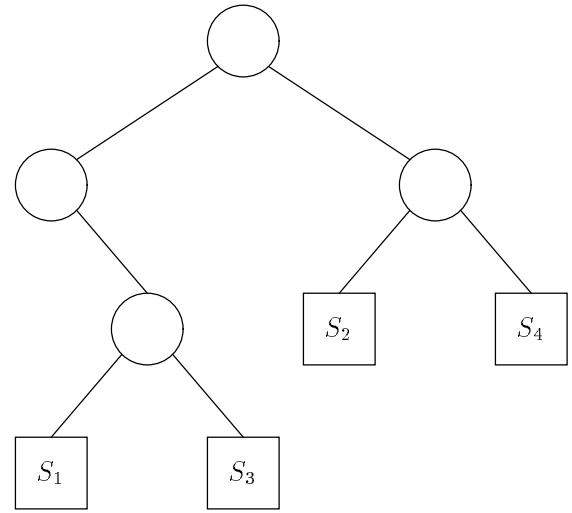
number of w occurrences in a text X generated a probabilistic source.

Back to Suffix Trees

Size S_n and path length L_n:

$$\mathbf{E}[S_n] = \sum_{w \in \mathcal{A}^*} P(O_{n-|w|+1}(w) \geq 2)$$

$$\mathbf{E}[L_n] = \sum_{w \in \mathcal{A}^*} \sum_{k \geq 2} k P(O_n(w) = k)$$



Suffix tree built from the first four suffixes of $X = 0101101110$, i.e. 0101101110, 101101110, 01101110, 1101110.

Random suffix tree resembles random independent trie:
(Jacquet & WS, 1994, Ward & Fayolle, 2005):

$$\begin{aligned} \mathbf{E}[S_n] - \mathbf{E}[S_n^T] &= O(n^{1-\varepsilon}) \\ \mathbf{E}[L_n] - \mathbf{E}[L_n^T] &= O(n^{1-\varepsilon}), \end{aligned}$$

since we can prove that

$$P(O_n(w) \geq 2) - P(\Omega_n(w) \geq 2) = O\left(n^{-\varepsilon} P^{1-\varepsilon}(w)\right)$$

where $\Omega_n(w)$ indicates how many of n indepdendent strings share the same prefix w.

Back to String Complexity

Last expression allows us to write

$$C_n = \frac{(n+1)n}{2} + \mathbf{E}[S_n] - \mathbf{E}[L_n]$$

where $\mathbf{E}[S_n] \sim \mathbf{E}[S_n^T]$ and $\mathbf{E}[L_n] \sim \mathbf{E}[L_n^T]$ are, respectively, the average size and path length in the associated indepdendent tries.

Back to String Complexity

Last expression allows us to write

$$C_n = \frac{(n+1)n}{2} + \mathbf{E}[S_n] - \mathbf{E}[L_n]$$

where $\mathbf{E}[S_n] \sim \mathbf{E}[S_n^T]$ and $\mathbf{E}[L_n] \sim \mathbf{E}[L_n^T]$ are, respectively, the average size and path length in the associated indepdendent tries.

Poissonization → Mellin → Inverse Mellin (SP) → De-Poissonization

Back to String Complexity

Last expression allows us to write

$$C_n = \frac{(n+1)n}{2} + \mathbf{E}[S_n] - \mathbf{E}[L_n]$$

where $\mathbf{E}[S_n] \sim \mathbf{E}[S_n^T]$ and $\mathbf{E}[L_n] \sim \mathbf{E}[L_n^T]$ are, respectively, the average size and path length in the associated indepdendent tries.

Poissonization → Mellin → Inverse Mellin (SP) → De-Poissonization

We know that (Jacquet & Regnier, 1989; W.S., 2001)

$$\mathbf{E}[{S_n}^T] = \frac{1}{h}(n + \Psi(\log n)) + o(n), \quad \mathbf{E}[{L_n}^T] = \frac{n \log n}{h} + n\Psi_2(\log n) + o(n),$$

where $\Psi(\log n)$ and $\Psi_2(\log n)$ are periodic functions. Therefore,

$$C_n = \frac{(n+1)n}{2} - \frac{n}{h}(\log n - 1 + Q_0(\log n) + o(1))$$

Back to String Complexity

Last expression allows us to write

$$C_n = \frac{(n+1)n}{2} + \mathbf{E}[S_n] - \mathbf{E}[L_n]$$

where $\mathbf{E}[S_n] \sim \mathbf{E}[S_n^T]$ and $\mathbf{E}[L_n] \sim \mathbf{E}[L_n^T]$ are, respectively, the average size and path length in the associated indepdendent tries.

Poissonization → Mellin → Inverse Mellin (SP) → De-Poissonization

We know that (Jacquet & Regnier, 1989; W.S., 2001)

$$\mathbf{E}[{S_n}^T] = \frac{1}{h}(n + \Psi(\log n)) + o(n), \quad \mathbf{E}[{L_n}^T] = \frac{n \log n}{h} + n\Psi_2(\log n) + o(n),$$

where $\Psi(\log n)$ and $\Psi_2(\log n)$ are periodic functions. Therefore,

$$C_n = \frac{(n+1)n}{2} - \frac{n}{h}(\log n - 1 + Q_0(\log n) + o(1))$$

Theorem 3 (Janson, Lonardi, W.S., 2004). *For unbiased memoryless source:*

$$C_n = \binom{n+1}{2} - n \log_{|\mathcal{A}|} n + \left(\frac{1}{2} + \frac{1-\gamma}{\ln |\mathcal{A}|} + Q_1(\log_{|\mathcal{A}|} n)\right) n + O(\sqrt{n \log n})$$

where $\gamma \approx 0.577$ is Euler's constant and Q_1 is a periodic function.

Open Problems

There are still many open problems in digital trees and string complexity.

Open Problem 6 :
Analyze suffix trees and tries when inserted strings are faulty. In particular, study noisy digital trees built from strings at the output of a noisy channel fed by a Markov source (hidden Markov process).

Open Problems

There are still many open problems in digital trees and string complexity.

Open Problem 6 :
Analyze suffix trees and tries when inserted strings are faulty. In particular, study noisy digital trees built from strings at the output of a noisy channel fed by a Markov source (hidden Markov process).

Open Problem 7 :
Analyze suffix trees built from, say 30 (initially idential) faulty strings.

Open Problems

There are still many open problems in digital trees and string complexity.

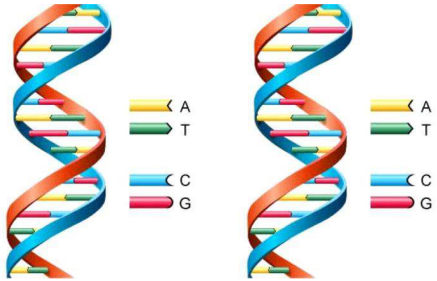
Open Problem 6 :
Analyze suffix trees and tries when inserted strings are faulty. In particular, study noisy digital trees built from strings at the output of a noisy channel fed by a Markov source (hidden Markov process).

Open Problem 7 :
Analyze suffix trees built from, say 30 (initially idential) faulty strings.

Open Problem 8 : Noisy Renyí Problem:
Find the number of random queries necessary to recover a hidden bijective labeling of n distinct objects when in each query one selects a random subset of labels and asks which objects have those labels. **BUT** answers are contaminated with some errors. In other words, reconstruct a PATRICIA trie from faulty strings!

Joint String Complexity

For X and Y, let $J(X,Y)$ be the set of common words between X and Y.



The joint string complexity is
$|J(X,Y)| = |I(X) \cap I(Y)|$

Example. If $X = aabaa$ and $Y = abbba$, then $J(X,Y) = \{\varepsilon, a, b, ab, ba\}$.

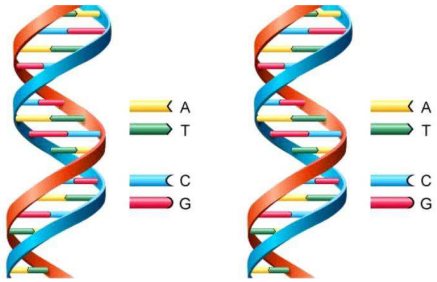
Goal. Estimate

$$J_{n,m} = \mathbf{E}[|J(X,Y)|]$$

when $|X| = n$ and $|Y| = m$.

Joint String Complexity

For X and Y, let $J(X,Y)$ be the set of common words between X and Y.



The joint string complexity is
$|J(X,Y)| = |I(X) \cap I(Y)|$

Example. If $X = aabaa$ and $Y = abbba$, then $J(X,Y) = \{\varepsilon, a, b, ab, ba\}$.

Goal. Estimate

$$J_{n,m} = \mathbf{E}[|J(X,Y)|]$$

when $|X| = n$ and $|Y| = m$.

Some Observations. For any word $w \in \mathcal{A}^*$

$$|J(X,Y)| = \sum_{w \in \mathcal{A}^*} \min\{1, O_X(w)\} \cdot \min\{1, O_Y(w)\}.$$

When $|X| = n$ and $|Y| = m$, we have

$$J_{n,m} = \mathbf{E}[|J(X,Y)|] - 1 = \sum_{w \in \mathcal{A}^* - \{\varepsilon\}} P(O_n^1(w) \geq 1) P(O_m^2(w) \geq 1)$$

where $O_n^i(w)$ is the number of w-occurrences in a string of generated by source $i = 1, 2$ (i.e., X and Y) which we assume to be memoryless sources.

Independent Joint String Complexity

Consider two sets of n independently generated (memoryless) strings.

Let ${\Omega_n}^i(w)$ be the number of strings for which w is a prefix when the n strings are generated by a source $i = 1, 2$ define

$$C_{n,m} = \sum_{w \in \mathcal{A}^* - \{\varepsilon\}} P(\Omega_n^1(w) \geq 1) P(\Omega_m^2(w) \geq 1)$$

Theorem 4. *There exists $\varepsilon > 0$ such that*

$$J_{n,m} - C_{n,m} = O(\min\{n, m\}^{-\varepsilon})$$

for large n.

Independent Joint String Complexity

Consider two sets of n independently generated (memoryless) strings.

Let ${\Omega_n}^i(w)$ be the number of strings for which w is a prefix when the n strings are generated by a source $i = 1, 2$ define

$$C_{n,m} = \sum_{w \in \mathcal{A}^* - \{\varepsilon\}} P(\Omega_n^1(w) \geq 1) P(\Omega_m^2(w) \geq 1)$$

Theorem 4. *There exists $\varepsilon > 0$ such that*

$$J_{n,m} - C_{n,m} = O(\min\{n, m\}^{-\varepsilon})$$

for large n.

Recurrence for $C_{n,m}$

$$C_{n,m} = 1 + \sum_{a \in \mathcal{A}} \sum_{k,\ell \geq 0} \binom{n}{k} P_1(a)^k (1 - P_1(a))^{n-k} \binom{m}{\ell} P_2(a)^\ell (1 - P_2(a))^{m-\ell} C_{k,\ell}$$

with $C_{0,m} = C_{n,0} = 0$.

Independent Joint String Complexity

Consider two sets of n independently generated (memoryless) strings.

Let ${\Omega_n}^i(w)$ be the number of strings for which w is a prefix when the n strings are generated by a source $i = 1, 2$ define

$$C_{n,m} = \sum_{w \in \mathcal{A}^* - \{\varepsilon\}} P(\Omega_n^1(w) \geq 1) P(\Omega_m^2(w) \geq 1)$$

Theorem 4. *There exists $\varepsilon > 0$ such that*

$$J_{n,m} - C_{n,m} = O(\min\{n, m\}^{-\varepsilon})$$

for large n.

Recurrence for $C_{n,m}$

$$C_{n,m} = 1 + \sum_{a \in \mathcal{A}} \sum_{k,\ell \geq 0} \binom{n}{k} P_1(a)^k (1 - P_1(a))^{n-k} \binom{m}{\ell} P_2(a)^\ell (1 - P_2(a))^{m-\ell} C_{k,\ell}$$

with $C_{0,m} = C_{n,0} = 0$.

Poissonization → Double Mellin → Inverse Mellin (SP) → Double dePoissonization

Main Results

Assume that $\forall a \in \mathcal{A}$ we have $P_1(a) = P_2(a) = p_a$.

Theorem 5. *For a biased memoryless source, the joint complexity is asymptotically*

$$C_{n,n} = n\frac{2\log 2}{h} + Q(\log n)n + o(n),$$

where $Q(x)$ is a small periodic function (with amplitude smaller than 10^{-6}) which is nonzero only when the $\log p_a$, $a \in \mathcal{A}$, are rationally related, that is, $\log p_a / \log p_b \in \mathbb{Q}$.

Main Results

Assume that $\forall a \in \mathcal{A}$ we have $P_1(a) = P_2(a) = p_a$.

Theorem 5. *For a biased memoryless source, the joint complexity is asymptotically*

$$C_{n,n} = n\frac{2\log 2}{h} + Q(\log n)n + o(n),$$

where $Q(x)$ is a small periodic function (with amplitude smaller than 10^{-6}) which is nonzero only when the $\log p_a$, $a \in \mathcal{A}$, are rationally related, that is, $\log p_a / \log p_b \in \mathbb{Q}$.

Assume that $P_1(a) \neq P_2(a)$.

Theorem 6. *Define $\kappa = \min_{(s_1,s_2)\in\mathcal{K}\cap\mathbb{R}^2}\{(-s_1 - s_2)\} < 1$, where s_1 and s_2 are roots of*

$$H(s_1, s_2) = 1 - \sum_{a\in\mathcal{A}} (P_1(a))^{-s_1}(P_2(a))^{-s_2} = 0.$$

Then

$$C_{n,n} = \frac{n^{\kappa}}{\sqrt{\log n}}\left(\frac{\Gamma(c_1)\Gamma(c_2)}{\sqrt{\pi\Delta H(c_1, c_2)\nabla H(c_1, c_2)}} + Q(\log n) + O(1/\log n)\right),$$

where Q is a double periodic function.

Classification of Sources

The growth of $C_{n,n}$ is:

- $\Theta(n)$ for identical sources;
- $\Theta(n^{\kappa}/\sqrt{\log n})$ for non identical sources with $\kappa < 1$.

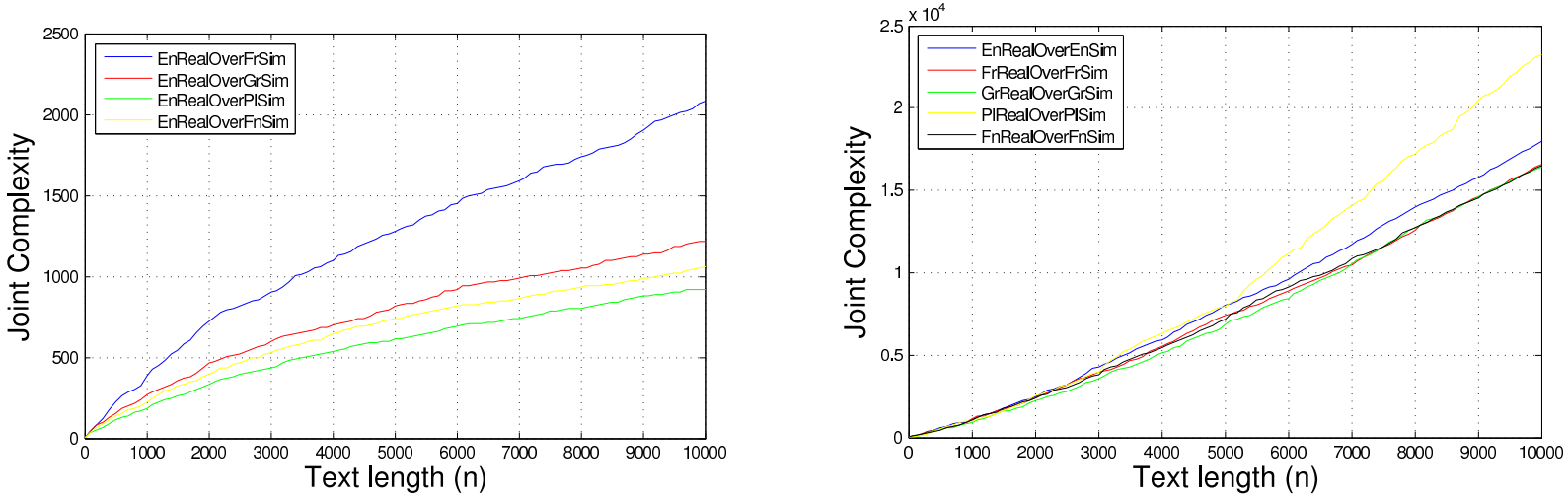
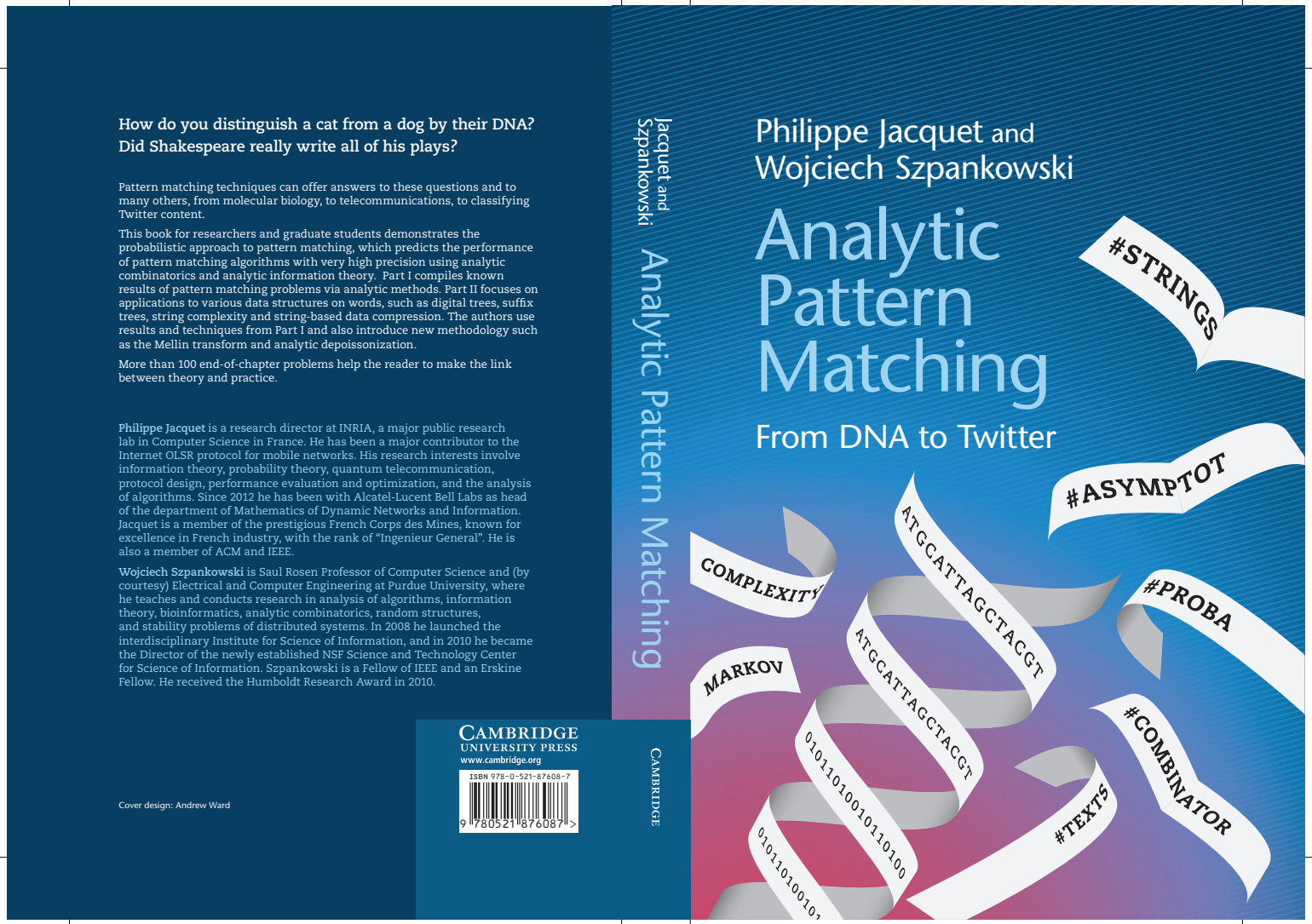


Figure 3: Joint complexity: (a) English text vs French, Greek, Polish, and Finnish texts; (b) real and simulated texts (3rd Markov order) of English, French, Greek, Polish and Finnish language.

You can find more in the book following . . .



How do you distinguish a cat from a dog by their DNA? Did Shakespeare really write all of his plays?

Pattern matching techniques can offer answers to these questions and to many others, from molecular biology, to telecommunications, to classifying Twitter content.

This book for researchers and graduate students demonstrates the probabilistic approach to pattern matching, which predicts the performance of pattern matching algorithms with very high precision using analytic combinatorics and analytic information theory. Part I compiles known results of pattern matching problems via analytic methods. Part II focuses on applications to various data structures on words, such as digital trees, suffix trees, string complexity and string-based data compression. The authors use results and techniques from Part I and also introduce new methodology such as the Mellin transform and analytic depoissonization.

More than 100 end-of-chapter problems help the reader to make the link between theory and practice.

Philippe Jacquet is a research director at INRIA, a major public research lab in Computer Science in France. He has been a major contributor to the Internet OLSR protocol for mobile networks. His research interests involve information theory, probability theory, quantum telecommunication, protocol design, performance evaluation and optimization, and the analysis of algorithms. Since 2012 he has been with Alcatel-Lucent Bell Labs as head of the department of Mathematics of Dynamic Networks and Information. Jacquet is a member of the prestigious French Corps des Mines, known for excellence in French industry, with the rank of "Ingenieur General". He is also a member of ACM and IEEE.

Wojciech Szpankowski is Saul Rosen Professor of Computer Science and (by courtesy) Electrical and Computer Engineering at Purdue University, where he teaches and conducts research in analysis of algorithms, information theory, bioinformatics, analytic combinatorics, random structures, and stability problems of distributed systems. In 2008 he launched the interdisciplinary Institute for Science of Information, and in 2010 he became the Director of the newly established NSF Science and Technology Center for Science of Information. Szpankowski is a Fellow of IEEE and an Erskine Fellow. He received the Humboldt Research Award in 2010.

Cover design: Andrew Ward

9780521876087; Jacquet & Szpankowski; PPC: C M Y K

CAMBRIDGE
UNIVERSITY PRESS
www.cambridge.org

ISBN 978-0-521-87608-7

Jacquet and Szpankowski Analytic Pattern Matching

CAMBRIDGE

Philippe Jacquet and Wojciech Szpankowski

Analytic Pattern Matching

From DNA to Twitter

#STRINGS

#ASYMPTOT

#PROBA

#COMBINATOR

#TEXTS

COMPLEXITY

MARKOV

ATGCATTAGCTACGT

ATGCATTAGCTACGT

0101101001011010 0

0101101001 01

Acknowledgments

My French Connection:

Philippe Flajolet (1948-2011)
Analytic Combinatorics

Acknowledgments

My French Connection:

Philippe Flajolet (1948-2011)
Analytic Combinatorics

My Italian Connection:

Alberto Apostolico:
The Master from whom I learned Stringology and More!

Acknowledgments

My French Connection:

Philippe Flajolet (1948-2011)
Analytic Combinatorics

My Italian Connection:

Alberto Apostolico:
The Master from whom I learned Stringology and More!

. . . and **ALL** my co-authors.

That's It

THANK YOU