2007 Conference on Analysis of Algorithms, AofA 07 DMTCS proc. **AH**, 2007, 29–44

Counting occurrences for a finite set of words: an inclusion-exclusion approach

F. Bassino[1], J. Clément[2], J. Fayolle[3], and P. Nicodème[4]

[1]*IGM, Université de Marne la Vallée, 77454 Marne-la-Vallée Cedex 2, France.* Frederique.Bassino@univ-mlv.fr
[2]*GREYC, CNRS-UMR 6072, Université de Caen, 14032 Caen, France.* Julien.Clement@info.unicaen.fr
[3]*LRI; Univ. Paris-Sud, CNRS ; Bât 490, 91405 Orsay, France.* Julien.Fayolle@lri.fr
[4]*LIX, CNRS-UMR 7161, École polytechnique, 91128, Palaiseau, France.* nicodeme@lix.polytechnique.fr

In this paper, we give the multivariate generating function counting texts according to their length and to the number of occurrences of words from a finite set. The application of the inclusion-exclusion principle to word counting due to Goulden and Jackson (1979, 1983) is used to derive the result. Unlike some other techniques which suppose that the set of words is *reduced* (*i.e.*, where no two words are factor of one another), the finite set can be chosen arbitrarily. Noonan and Zeilberger (1999) already provided a MAPLE package treating the non-reduced case, without giving an expression of the generating function or a detailed proof. We give a complete proof validating the use of the inclusion-exclusion principle and compare the complexity of the method proposed here with the one using automata for solving the problem.

Keywords: word statistics, inclusion-exclusion, generating functions

1 Introduction

Enumerating sequences with given combinatorial properties is rigorously formalized since the end of the seventies and the beginning of the eighties by Goulden and Jackson (GJ79; GJ83) and by Guibas and Odlyzko (GO81a; GO81b).

The former (GJ79; GJ83) introduce a very powerful method of inclusion-exclusion to count occurrences of words from a *reduced* set of words (*i.e.*, where no word is factor of another word of the set) in texts; this method is characterized by counting texts where some occurrences are marked (other terms are pointed or anchored) and then removing multiple count of the same text (text counted several times with different markings). We refer later to this by *inclusion-exclusion* method. Goulden-Jackson counting is typically multivariate, a formal parameter being associated to each word.

The latter (GO81a; GO81b) introduce the notion of auto-correlation of a word that generalizes to correlation between words. Formal non-ambiguous manipulations over languages translates to generating functions. We refer later to this by *formal language* method. Unlike Goulden and Jackson, Guibas and Odlyzko consider univariate cases, like enumerating sequences avoiding a pattern, or sequences terminating with a first occurrence of a pattern in a text (see also (SF96)). Régnier and Szpankowski (RS98) generalize the formal language approach by a bivariate analysis for counting the number of matches of a word in random texts (handling also a Markovian source on the symbol emission) and prove a normal limit law. Régnier (R0̂0) extends this further to multivariate analysis and simultaneous counting of several words. See also the books of Szpankowski (Szp01) and Lothaire (Lot05). Bourdon and Vallée (BV02; BV06) apply the previous analysis to dynamical sources. Prum *et al.* (PRdT95) follow a more probabilistic approach.

Noonan and Zeilberger (NZ99) extend the inclusion-exclusion method of Goulden and Jackson and solve the general non-reduced case (words may be factor of other words), implementing corresponding MAPLE programs, without however completely publishing the explicit result formulæ. Recently Kong (Kon05) applies the results of Noonan and Zeilberger for the reduced case to an asymmetrical Bernoulli (also called memoryless) model for the generation of symbols. He also compares the Goulden and Jackson method to the Régnier and Szpankowski method, emphasizing the conceptual simplicity of the inclusion-exclusion approach. It is however useful to note that the formal language approach provides access to information that the inclusion-exclusion method does not, such as the waiting time for a first match of a word or the time separating two matches of the same word or of two different words (in both case eventually forbidding matches with other words).

A third approach is possible by use of automata. Nicodème *et al.* (NSF02) use classical algorithms to (1) build a marked deterministic automaton recognizing a regular expression and (2) translate into generating function (Chomsky-Schützenberger algorithm (CS63)); this provides the bivariate generating function counting the matches. A variation of the method extends the results to Markovian sources. This result applies immediately to a set of words considered as a regular expression. Nicodème (Nic03) extends this to multivariate counting by taking the product of marked automata (with an automaton and a mark associated to a word) and to set of words with possible errors [(i)]. Notice that step (1) of this approach may be directly done by building the Aho-Corasick automaton, designed for pattern-matching.

Each of the three above-mentioned approaches did develop quite independently and partially unaware of each other.

Let $\mathcal{A}$ be the alphabet on which the words are written and $\mathcal{U} = \{u_1, u_2, \ldots, u_r\}$ be a finite set of distinct words on the alphabet $\mathcal{A}$. We note $\pi(w)$ the weight of the word w. The weight could be a formal weight over the commutative monoid $\mathcal{A}^*$ (*i.e.*, $\pi(ababab) = \alpha^3\beta^3$) or, the probability generating function in the Bernoulli (also called *memoryless*) setting, $\pi(w) = \Pr(w)$, or even $\pi(w) = 1$ for a uniform weighted model over all words.

We set some more notations: given a r-row vector $\mathbf{x} = (x_1, \ldots, x_r)$ of formal variables and a r-row vector $\mathbf{j} = (j_1, \ldots, j_r)$ of integers, we will denote by $\mathbf{x}^{\mathbf{j}}$ the product $\prod_{i=1}^{r} x_i^{j_i}$.

In this article we describe two approaches to compute the multivariate generating function $F_{\mathcal{U}}$ counting texts according to their length and to their number of occurrences of words from the set $\mathcal{U}$:

$$F_{\mathcal{U}}(z, \mathbf{x}) = F(z, \mathbf{x}) := \sum_{w \in \mathcal{A}^*} \pi(w) z^{|w|} \mathbf{x}^{\boldsymbol{\tau}(w)}, \tag{1}$$

where $\boldsymbol{\tau}(w) = (|w|_1, \ldots, |w|_r)$, and $|w|_i$ is the total number of occurrences of u_i in w (with possible overlaps). We focus on methods which solve the problem fully without making any assumption on the set itself (for instance on its reduction, hence $\mathcal{U}$ can contain $u_1 = abbababa$ and $u_2 = baba$ although u_2 is a factor of u_1). We aim at presenting a novel approach and a full proof of results partially in Noonan and Zeilberger.

(i) Algorithms implemented in the package `regexpcount` of `algolib`, Algorithms Project, INRIA

In Section 2 we present an approach using the Aho-Corasick automaton that solves the general (non-reduced) problem; we also consider the complexity of this method. We describe and prove our results in Section 3 using the inclusion-exclusion principle. Algorithmic aspects are also considered in this section. Appendix A is devoted, as a case study, to the comparison of complexity of the two methods when computing the covariance of the number of occurrences of two words, the inclusion-exclusion approach being more efficient both for exact and asymptotic computations than the automaton approach.

2 Automaton approach

We resort in this section to the well-known Aho-Corasick algorithm (AC75; CR02) which builds from a finite set of words $\mathcal{U}$ a deterministic complete automaton (not necessarily minimal) recognizing the language $\mathcal{A}^*\mathcal{U}$. This automaton denoted by $\mathcal{A}_\mathcal{U}$ is the basis of many efficient algorithms on string matching problems and is often called the *string matching automaton*. This automaton is usually described by the trie of the set of words together with a failure function. Let $\mathcal{T}_\mathcal{U}$ be the ordinary trie representing the set $\mathcal{U}$, seen as a finite deterministic automaton $(Q, \delta, \varepsilon, T)$ where the set of states is $Q = \mathrm{Pref}(\mathcal{U})$ (prefixes of words in $\mathcal{U}$), the initial state is ε, the set of final states is $T = \mathcal{A}^*\mathcal{U} \cap \mathrm{Pref}(\mathcal{U})$ and the transition function δ is defined on $\mathrm{Pref}(\mathcal{U}) \times \mathcal{A}$ by

$$\delta(p, x) = \begin{cases} px & \text{if } px \in \mathrm{Pref}(\mathcal{U}), \\ \mathrm{Border}(px) & \text{otherwise,} \end{cases}$$

where the failure function Border is defined by

$$\mathrm{Border}(v) = \text{the longest proper suffix of } v \text{ which belongs to } \mathrm{Pref}(\mathcal{U}) \text{ if defined, or } \varepsilon \text{ otherwise.}$$

In the following we identify a word $v \in \mathrm{Pref}(\mathcal{U})$ with the node at the end of the branch of the tree labeled by v, so that Border defines also a map on the nodes of the tree. There are efficient $O(|\mathcal{U}|)$ algorithms (AC75; CR02) linear both in time and space to build such a tree structure and the auxiliary Border function.

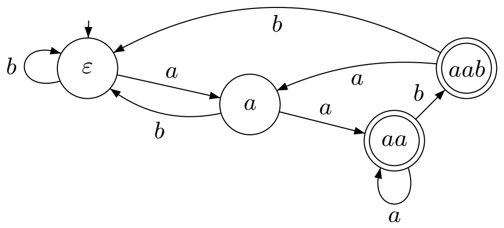
The matrix $\mathbb{T}(\mathbf{x})$ (with $\mathbf{x}$ a r-vector of formal variables) denotes the transition matrix of the Aho-Corasick automaton where the variable x_i marks the states accepting the word u_i. The generating function is expressed as

$$F(z, \mathbf{x}) = \sum_{w \in \mathcal{A}^*} \pi(w) z^{|w|} \mathbf{x}^{\boldsymbol{\tau}(w)} = \begin{pmatrix} 1, & 0, & \cdots, & 0 \end{pmatrix} (\mathbb{I} - z\mathbb{T}(\mathbf{x}))^{-1} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \tag{2}$$

where $\pi(w)$ can be viewed as the weight of word w.

Example 1 *Let $\mathcal{U} = \{aab, aa\}$. We have*

$$\mathbb{T}(x_1, x_2) = \begin{pmatrix} b & a & 0 & 0 \\ b & 0 & ax_2 & 0 \\ 0 & 0 & ax_2 & bx_1 \\ b & a & 0 & 0 \end{pmatrix},$$



and

$$F(z, x_1, x_2) = \frac{1 - a(x_2 - 1)z}{1 - z(ax_2 + b - ab(x_2 - 1)z + a^2bx_2(x_1 - 1)z^2)}.$$

Complexity. Let $L = \sum_{u \in \mathcal{U}} |u|$ be the sum of the lengths of the words from $\mathcal{U}$. We first have to compute the Aho-Corasick automaton and this can be done classically in time $O(L)$ for a finite alphabet. The automaton can have up to L states. Denoting by N the number of states of the Aho-Corasick automaton, the transitions matrix $\mathbb{T}$ is of size N^2, but in general this matrix is sparse: only $N \times \operatorname{Card} \mathcal{A}$ entries are non-zero (since the automaton is complete and deterministic with $\operatorname{Card} \mathcal{A}$ transitions from each state).

So the complexity to obtain the counting multivariate generating function by this approach is basically the one of inverting a relatively sparse matrix of the form $\mathbb{I} - z\mathbb{T}(\mathbf{x})$ whose all terms are monomials of the form $\alpha \prod x_i^{\varepsilon_i}$ (with $\alpha \in \mathcal{A}$ and the ε_i's in $\{0, 1\}$) corresponding to the transition matrix of the automaton. The limit of this approach is the fact that the size of the transition matrix L^2 can grow rapidly if we consider many rather long words. In the next section, we adopt another approach which leads also to solve a system of equations, but then the size of the system is $r \times r$ (where r is the number of words in $\mathcal{U}$). We there present a detailed way to compute the generating function of occurrences using the Goulden and Jackson method.

3 Inclusion-exclusion method applied to word counting

This section presents an approach exactly along the same line as in (GJ83) but extended to the non-reduced case. In (NZ99) the authors provide the main ideas to treat the non-reduced case and a MAPLE package, neither giving explicit expressions nor detailed proofs. We consider it important to give a more formal presentation of the Goulden and Jackson method for an arbitrary finite set of words as it can be of interest to a broad audience and it is the first step to the generalization of the underlying probabilistic model. The complexity of such an approach is also examined from a computational point of view. Indeed, statistics on words occurrences are useful in many fields (in fact each time unusual events in sequences are looked at); moreover, in many applications, it is necessary to compute the corresponding statistics as fast as possible.

We aim to count texts according to their length and to their number of occurrences of words from a set $\mathcal{U}$. A text where some occurrences of words from $\mathcal{U}$ are marked is decomposed combinatorically as a sequence of letters from $\mathcal{A}$ and clusters (set of overlapping and marked occurrences of $\mathcal{U}$, noted $\mathcal{L}_{\mathcal{U}}$; see Definitions (2) and (3) in the next section). Each text is counted several times depending on which occurrences are marked (each text is counted as many times as the number of possible marking of occurrences). This multiple counting is eliminated by use of the inclusion-exclusion principle (see among others (GJ83), (Szp01), and (FS07, III.6.4) for details).

3.1 Preliminaries

First we formally state the generating function in terms of occurrence positions.

Definition 1 (Occurrence positions set) *The* occurrence positions set *of a word u in a word w is the set of final positions of occurrences of u in w:*

$$\operatorname{Occ}(u, w) = \big\{ p \in \{1, \dots, |w|\} \quad \big| \quad w[(p - |u| + 1) \dots p] = u \big\}.$$

With this definition, we can rewrite the counting generating function of Equation (1)

$$F(z,\mathbf{x}) = \sum_{w\in\mathcal{A}^*} \pi(w) z^{|w|} \prod_{i=1}^{r} x_i^{\mathrm{Card}(\mathrm{Occ}(u_i,w))}.$$

Definition 2 (Clustering-word) *A* clustering-word *for the set $\mathcal{U} = \{u_1,\dots,u_r\}$ is a word $w \in \mathcal{A}^*$ such that any two consecutive positions in w are* covered *by the same occurrence in w of a word $u \in \mathcal{U}$. The position i of the word w is* covered *by a word u if $u = w[(j-|u|+1)\dots j]$ for some $j \in \{|u|,\dots,n\}$ and $j-|u|+1 \le i \le j$. The language of all clustering-words for a given set $\mathcal{U}$ is noted $\mathcal{K}_{\mathcal{U}}$.*

Definition 3 (Cluster) *A* cluster *of a clustering-word w in $\mathcal{K}_{\mathcal{U}}$ is a set of occurrence positions subsets $\{\,\mathcal{S}_u \subset \mathrm{Occ}(u,w) \mid u \in \mathcal{U}\,\}$ which covers exactly w, that is, every two consecutive positions i and $i+1$ in w are covered by at least one same occurrence of some $u \in \mathcal{U}$. More formally*

$$\forall i \in \{1,\dots,|w|-1\} \quad \exists\, u \in \mathcal{U},\, \exists\, p \in \mathcal{S}_u \quad \text{such that} \quad p-|u|+1 < i+1 \le p.$$

The set of clusters with respect to clustering-words built from some finite set of words $\mathcal{U}$ is noted $\mathcal{L}_{\mathcal{U}}$. We note $\mathcal{L}_{\mathcal{U}}(w)$ the subset of $\mathcal{L}_{\mathcal{U}}$ corresponding to the clustering-word $w \in \mathcal{K}_{\mathcal{U}}$. For a cluster $\mathfrak{C} = \{\mathcal{S}_u \mid u \in \mathcal{U}\}$, we also define $w(\mathfrak{C})$ the corresponding (unique) clustering-word and $|\mathfrak{C}|_u$ the number of marked occurrences of the word u in the cluster, i.e.,

$$|\mathfrak{C}|_u = \mathrm{Card}\,\mathcal{S}_u.$$

Example 2 *Let $\mathcal{U} = \{baba, ab\}$ and $w = abababa$, so that $w \in \mathcal{K}_{\mathcal{U}}$. We have*

$$\begin{aligned}\mathcal{L}_{\mathcal{U}}(w) = \Big\{ &\big\{\mathcal{S}_{ab} = \{2,4,6\}, \mathcal{S}_{baba} = \{5,7\}\big\}, \big\{\mathcal{S}_{ab} = \{2,6\}, \mathcal{S}_{baba} = \{5,7\}\big\},\\ &\big\{\mathcal{S}_{ab} = \{2,4\}, \mathcal{S}_{baba} = \{5,7\}\big\}, \big\{\mathcal{S}_{ab} = \{2\}, \mathcal{S}_{baba} = \{5,7\}\big\}\Big\}.\end{aligned}$$

In the non-reduced case, a word u_i may occur within some other word from $\mathcal{U}$. In order to properly generate the clusters we introduce the notion of *right extension* of a pair of words (h_1,h_2). This notion is a generalization of the correlation set of two words h_1 and h_2 but differs in that:

(i) overlapping is not allowed to occur at the beginning of h_1.
(ii) extension has to add some letters to the right of h_1.

More formally we have

Definition 4 (Right extension set) *The* right extension set *of a pair of words (h_1,h_2) is*

$$\mathcal{E}_{h_1,h_2} = \{\; e \quad | \quad \textit{there exists } e' \in \mathcal{A}^+ \quad \textit{such that} \quad h_1 e = e' h_2 \quad \textit{with } 0 < |e| < |h_2| \}.$$

Note that, when h_1 and h_2 have no factor relation, the right extension set $\mathcal{E}_{h_1,h_2}$ is the correlation set of h_1 to h_2. Moreover, when $h_1 = h_2$, the set $\mathcal{E}_{h_1,h_2}$ is the strict auto-correlation set of h_1 (the empty word does not belong to $\mathcal{E}_{h_1,h_2}$).

One can also define the right extension matrix of a vector of words $\mathbf{u} = (u_1,\dots,u_r)$

$$\mathcal{E}_{\mathbf{u}} = \left(\mathcal{E}_{u_i,u_j}\right)_{1\le i,j\le r}.$$

As examples, we have
$\mathbf{u}_1 = (aba, ab)$ gives $\mathcal{E}_{\mathbf{u}_1} = \begin{pmatrix} ba & b \\ \emptyset & \emptyset \end{pmatrix}$, and $\mathbf{u}_2 = (aaaa, aaa)$ gives $\mathcal{E}_{\mathbf{u}_2} = \begin{pmatrix} a+a^2+a^3 & a+a^2 \\ a^2+a^3 & a+a^2 \end{pmatrix}$.

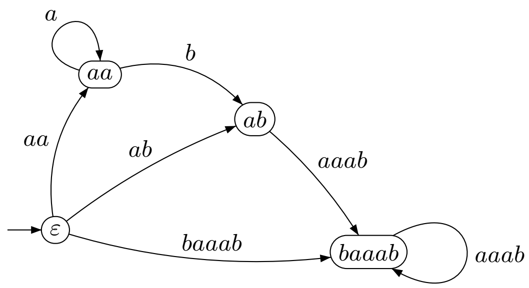


Fig. 1: Graph $\mathcal{G}$ for $\mathcal{U} = \{baaab, aa, ab\}$.

3.2 Generating function of clusters

We define the generating function $\xi(z, \mathbf{t})$ of the set of clusters $\mathcal{L}_\mathcal{U}$ on $\mathcal{U}$ where the length of a cluster is marked by the formal variable z and each marked occurrence of u_i in clusters is marked by the formal variable t_i. The set of all possible clusters is the disjoint union over all clustering-words w of the set of all the clusters built from w, hence

$$\xi(z, \mathbf{t}) = \sum_{w \in \mathcal{K}_\mathcal{U}} \sum_{\mathfrak{C} \in \mathcal{L}_\mathcal{U}(w)} z^{|w|} \pi(w) t_1^{|\mathfrak{C}|_{u_1}} \dots t_r^{|\mathfrak{C}|_{u_r}}.$$

3.2.1 Basic decomposition

We use a bijection between clusters and paths in a graph to derive an expression for the generating function $\xi(z, \mathbf{t})$ of clusters in $\mathcal{L}_\mathcal{U}$.

Let $\mathcal{G} = (V, E)$ be a directed labeled graph such that:

(a) the set of vertices is $V = \{\varepsilon\} \cup \mathcal{U}$;

(b) the set of edges is $E = \{\varepsilon \xrightarrow{u} u \mid u \in \mathcal{U}\} \cup \{u \xrightarrow{y} u' \mid u, u' \in \mathcal{U}$ and $y \in \mathcal{E}(u, u')\}$.

See an example on Figure 1 with $\mathcal{U} = \{baaab, aa, ab\}$.

If the set $\mathcal{U}$ is reduced (*i.e.*, without factor relations) then a cluster is completely described by a path in this graph starting at ε. When the set is not reduced, this is no longer true. We need to associate along the path the possible occurrences of $\mathcal{U}$ within the last label read.

Thus we define a bijection between a cluster $\mathfrak{C}$ and a pair $(c, \mathcal{F}_c)$ where c is a path in $\mathcal{G}$ (starting at ε) and $\mathcal{F}_c$ is a k-tuple (k is the length of the path c) of sets of positions of occurrences. Each set in $\mathcal{F}_c$ is made of position occurrences of words from $\mathcal{U}$ that end within the label of the corresponding edge of the path.

Let $\mathfrak{C} = \{\mathcal{S}_u \mid u \in \mathcal{U}\}$ be a cluster for a clustering-word w (each set $\mathcal{S}_u$ is composed of some end positions of occurrences of u inside the clustering-word w). We partition each occurrence positions set of $\mathfrak{C}$ as $\mathcal{S}_u = \mathcal{S}'_u \cup \mathcal{S}''_u$ where $\mathcal{S}'_u$ contains positions of the occurrences of u that are not factor of any another occurrence of $\mathcal{U}$. We are then assured that $\mathfrak{C}' = \{\mathcal{S}'_u \mid u \in \mathcal{U}\}$ is a cluster (with no factor occurrences) for the same clustering-word $w(\mathfrak{C})$. Then we build from $\mathfrak{C}'$ a sequence $((u_{i_1}, p_{i_1}), (u_{i_2}, p_{i_2}), \dots, (u_{i_k}, p_{i_k}))$, where p_{i_j} is the ending position of u_{i_j} (a word from $\mathcal{U}$). This sequence is sorted by increasing position: $p_{i_1} = |u_{i_1}| < p_{i_2} < \cdots < p_{i_k} = |w|$. Each word $w[1 \dots p_{i_j}]$ for $j \in \{1, \dots, r\}$ is a clustering-word.

We set $y_1 = u_{i_1}$; then each y_j for $j \in \{2, \dots, k\}$ is the word such that $w[1 \dots p_{i_{j-1}}] \cdot y_j = w[1 \dots p_{i_j}]$. By definition of the right extension sets, $y_j \in \mathcal{E}_{u_{i_{j-1}}, u_{i_j}}$ for each j. We therefore get a unique path $c = y_1.y_2 \dots y_k$ in the graph $\mathcal{G}$

$$\varepsilon \xrightarrow{y_1} u_{i_1} \xrightarrow{y_2} u_{i_2} \xrightarrow{y_3} \dots \xrightarrow{y_k} u_{i_k}.$$

To take into account the factor occurrences in the cluster, we associate to each step $u_{i_{j-1}} \xrightarrow{y_j} u_{i_j}$ of the path a set $\mathcal{F}^j = \{\mathcal{F}_u^j \mid u \in \mathcal{U} - \{u_{i_j}\}\}$ where $\mathcal{F}_u^j$ is the set of occurrence positions in the word u_{i_j} of words ending within y_j, more precisely

$$\mathcal{F}_u^j = \left\{ p - |y_1 \dots y_j| + \left|u_{i_j}\right| \quad \middle| \quad p \in \mathcal{S}_u'' \quad \text{and} \quad |y_1 \dots y_{j-1}| < p \le |y_1 \dots y_j| \right\}.$$

By construction, we have an application mapping a cluster $\mathfrak{C}$ to a unique pair $(c, (\mathcal{F}^1, \dots, \mathcal{F}^k))$ and this application is clearly injective.

Conversely, let us consider a path $c = \varepsilon \xrightarrow{y_1} u_{i_1} \xrightarrow{y_2} u_{i_2} \xrightarrow{y_3} \dots \xrightarrow{y_k} u_{i_k}$, a k-tuple $(\mathcal{F}^1, \dots, \mathcal{F}^k)$ with $\mathcal{F}^j = \{\mathcal{F}_u^j \mid u \neq u_{i_j}\}$ and

$$\mathcal{F}_u^j \subset \left\{ l \quad \middle| \quad l \in \mathrm{Occ}(u, u_{i_j}) \text{ and } \left|u_{i_j}\right| - l < |y_j| \right\}.$$

This defines a unique cluster $\mathfrak{C} = \{\mathcal{S}_u \mid u \in \mathcal{U}\}$ as follows: we start with $\mathcal{S}_u = \emptyset$ for all $u \in \mathcal{U}$; we then build the clustering-word $w = y_1 \cdot y_2 \dots \cdot y_k$ by reading the labels along the path and, at step j, we put position $|y_1 \dots y_j|$ into $\mathcal{S}_{u_{i_j}}$; finally, for all $u \neq u_{i_j}$, we add to $\mathcal{S}_u$ the factor occurrences, *i.e.*, the set of positions

$$\left\{p + |y_1 \dots y_j| - \left|u_{i_j}\right| \quad \middle| \quad p \in \mathcal{F}_u^j\right\}.$$

We hence have built a bijection.

We introduce some notations to translate this construction to generating functions. Let $N_{i,j}(k)$ count the number of occurrences of u_j in u_i ending in the last k positions

$$N_{i,j}(k) = \left|u_i\right|_j - \left|u_i[1 \dots |u_i| - k]\right|_j. \tag{3}$$

For a suffix s of u_i, we introduce a formal weight $\langle s \rangle_i$ where each possible occurrence of u_m in u_i ending within s can be marked (or not) by t_m (hence marked by $1 + t_m$)

$$\langle s \rangle_i = \pi(s) z^{|s|} \prod_{m \neq i} (t_m + 1)^{N_{i,m}(|s|)}. \tag{4}$$

The notation $\langle \cdot \rangle_i$ extends readily to a set of words S which are suffixes of u_i, which gives

$$\langle S \rangle_i = \sum_{s \in S} \langle s \rangle_i.$$

Finally we define

$$\langle \mathbf{u} \rangle = (\langle u_1 \rangle_1, \dots, \langle u_r \rangle_r) \quad \text{and} \quad \langle \mathcal{E}_\mathbf{u} \rangle = \begin{pmatrix} \langle \mathcal{E}_{1,1} \rangle_1 & \langle \mathcal{E}_{1,2} \rangle_2 & \dots & \langle \mathcal{E}_{1,r} \rangle_r \\ \langle \mathcal{E}_{2,1} \rangle_1 & \langle \mathcal{E}_{2,2} \rangle_2 & \dots & \langle \mathcal{E}_{2,r} \rangle_r \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathcal{E}_{r,1} \rangle_1 & \langle \mathcal{E}_{r,2} \rangle_2 & \dots & \langle \mathcal{E}_{r,r} \rangle_r \end{pmatrix}. \tag{5}$$

We get to the following proposition.

Proposition 1 *The generating function $\xi(z, \mathbf{t})$ of clusters built from the set $\mathcal{U} = \{u_1, \ldots, u_r\}$ is given by*

$$\xi(z, \mathbf{t}) = \langle \mathbf{u} \rangle \Delta(\mathbf{t}) \cdot \Big(\mathbb{I} - \langle \mathcal{E}_{\mathbf{u}} \rangle \Delta(\mathbf{t}) \Big)^{-1} \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \tag{6}$$

where $\mathbf{u} = (u_1, \ldots, u_r)$, $\mathbf{t} = (t_1, \ldots, t_r)$, and the matrix $\Delta(\mathbf{t})$ is the $r \times r$ diagonal matrix with entries $t_1, \ldots, t_r$.

Proof: The matrix $\langle \mathcal{E}_{\mathbf{u}} \rangle$ is the transition matrix of the graph $\mathcal{G}$ where the vertex ε and its corresponding edges have been removed. Some occurrences of the word u_i (for each $i \in \{1, \ldots, n\}$) are marked with the formal variables t_i in the labels of $\mathcal{G}$. More precisely, a word occurrence u_i obtained when visiting a vertex u_i is marked by the formal variable t_i (and appears in the calculus through the diagonal matrix $\Delta(\mathbf{t})$ in (6)); in contrary, a factor occurrence can be marked or not (this does not change the path in the graph), hence providing a term of the form $\prod_{m \neq i} (t_m + 1)^{N_{i,m}(|y|)}$ (see Eq. (4)). The first transition from ε to any $u \in \mathcal{U}$ is handled similarly. So the paths with $k+1$ transitions in $\mathcal{G}$ starting from ε have generating function

$$\langle \mathbf{u} \rangle \Delta(\mathbf{t}) \cdot \Big(\langle \mathcal{E}_{\mathbf{u}} \rangle \Delta(\mathbf{t}) \Big)^{k} \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Finally we use the quasi-inverse notation $\sum_{j=0}^{\infty} \langle \mathcal{E}_{\mathbf{u}} \rangle^j \Delta(\mathbf{t}) = \big(\mathbb{I} - \langle \mathcal{E}_{\mathbf{u}} \rangle \Delta(\mathbf{t}) \big)^{-1}$ to get the result. □

3.2.2 Applications

Reduced set. When the set $\mathcal{U}$ is reduced, that is, no word of $\mathcal{U}$ is factor of another, the clusters are uniquely defined by a path in the previous graph $\mathcal{G}$. So $\langle \mathbf{u} \rangle$ and $\langle \mathcal{E}_{\mathbf{u}} \rangle$ do not depend on any of the variables t_i's. Hence in Eq. (6), variables t_i's are gathered inside $\Delta(\mathbf{t})$. This is another formulation of the result of Goulden and Jackson (GJ83).

One word. For $\mathcal{U} = \{u\}$, we get

$$\xi(z, t) = \frac{t \langle u \rangle}{1 - t \langle \mathcal{E}_u \rangle} = \frac{t \pi(u) z^{|u|}}{1 - t \hat{c}(z)} = \frac{t \pi(u) z^{|u|}}{1 - t(c(z) - 1)}, \tag{7}$$

where $\hat{c}(z)$ is the generating function of the strict autocorrelation set of word u (empty word ε omitted), and $c(z)$ is the auto-correlation polynomial of u.

Two words. For a set of two words $\{u_1, u_2\}$, one can compute explicitly $\xi(z, t_1, t_2)$ by the Cramer's rule

$$\xi(z, t_1, t_2) = \frac{t_1 \langle u_1 \rangle_1 + t_2 \langle u_2 \rangle_2 - t_1 t_2 \big(\langle u_1 \rangle_1 \big[\langle \mathcal{E}_{2,2} \rangle_2 - \langle \mathcal{E}_{1,2} \rangle_2 \big] + \langle u_2 \rangle_2 \big[\langle \mathcal{E}_{1,1} \rangle_1 - \langle \mathcal{E}_{2,1} \rangle_1 \big] \big)}{1 - t_2 \langle \mathcal{E}_{2,2} \rangle_2 - t_1 \langle \mathcal{E}_{1,1} \rangle_1 + t_1 t_2 \big(\langle \mathcal{E}_{1,1} \rangle_1 \langle \mathcal{E}_{2,2} \rangle_2 - \langle \mathcal{E}_{2,1} \rangle_1 \langle \mathcal{E}_{1,2} \rangle_2 \big)}, \tag{8}$$

and this expression is computable from the right extension matrix of $\{u_1, u_2\}$.

Example 3 *Let $\mathbf{u} = (a^7, a^3)$. The right extension matrix is:*

$$\mathcal{E}_{\mathbf{u}} = \begin{pmatrix} a + a^2 + a^3 + a^4 + a^5 + a^6 & a + a^2 \\ a^5 + a^6 & a + a^2 \end{pmatrix}.$$

We have $\langle \mathbf{u} \rangle = \big((1+t_2)^5 z^7 \pi(a^7), z\pi(a^3)\big)$, if we note $p = \pi(a)$ and use the property $p^k = \pi(a^k)$, then

$$\langle \mathcal{E}_{1,1} \rangle_1 = (1+t_2)zp + (1+t_2)^2(zp)^2 + (1+t_2)^3(zp)^3 + (1+t_2)^4(zp)^4 + (1+t_2)^5\big((zp)^5 + (zp)^6\big),$$
$$\langle \mathcal{E}_{1,2} \rangle_2 = zp + (zp)^2, \quad \langle \mathcal{E}_{2,1} \rangle_1 = (1+t_2)^5((zp)^5 + (zp)^6), \quad \langle \mathcal{E}_{2,2} \rangle_2 = zp + (zp)^2.$$

By substituting these values in Eq. (8) we get

$$\xi(z, t_1, t_2) = \frac{-(pz)^7 t_1 (t_2+1)^4 + (pz)^6 t_2 t_1 (t_2+1)^3 + (pz)^5 t_2 t_1 (t_2+1)^2 + (pz)^4 t_2 t_1 (t_2+1) - (pz)^3 t_2}{-1 + t_1 (pz)^6 (t_2+1)^4 + t_1 (pz)^5 (t_2+1)^3 + (pz)^4 t_1 (t_2+1)^2 + (pz)^3 t_1 (t_2+1) + (pz)^2 (t_1+t_2+t_1 t_2) + pz(t_1+t_2+t_1 t_2)}.$$

3.3 Generating function of texts

A text is decomposed combinatorically as a sequence of letters from $\mathcal{A}$ (of generating function $A(z)$) and clusters (or more rigorously clustering words) from $\mathcal{L}_{\mathcal{U}}$ (of generating function $\xi(z, \mathbf{t})$). The multivariate generating function F of Equation (1) is derived by substituting $t_i \mapsto x_i - 1$ for $i \in \{1, \ldots, r\}$ in each $(A(z) + \xi(z, \mathbf{t}))^k$, where k is the number of combinatorial objects in the decomposition.

To summarize, we have the following proposition:

Proposition 2 *Let $\mathbf{u} = (u_1, \ldots, u_r)$ be a finite vector of words in $\mathcal{A}^*$ and $\mathcal{E}_{\mathbf{u}}$ the associated right extension matrix. The multivariate generating function F counting texts where length is marked by the variable z and occurrences of u_i are marked by the vector of formal variables $\mathbf{x} = (x_1, \ldots, x_r)$ is*

$$F(z, \mathbf{x}) = \frac{1}{1 - A(z) - \xi(z, \mathbf{x} - \mathbf{1})}, \tag{9}$$

where $A(z) = \sum_{\sigma \in \mathcal{A}} \pi(\sigma) z$ is the generating function of the alphabet and $\xi(z, \mathbf{t})$ is defined in Eq. (6).

Proof: The proof relies on two main points. On one hand, the generating function $\xi(z, \mathbf{t})$ counts all the clusters (see Proposition 1 in Section 3.2.1). On the other hand, the inclusion-exclusion principle yields the final result by the substitutions $t_i \mapsto x_i - 1$. □

The application of the standard techniques of analytic combinatorics (see (FS07)) to the multivariate generating function F gives access to many statistics (*e.g.* mean, variance, covariance,...).

3.4 Algorithmic point of view

We present here an general method in order to compute the generating function $\xi(z, t)$. This is a two-step approach:

(i) we compute the $r \times r$ matrix $\langle \mathcal{E}_{\mathbf{u}} \rangle$ (where r is the number of words in $\mathcal{U}$); coefficients are polynomials whose degree (in any variable) is bounded by $\max_{u \in \mathcal{U}} |u| - 1$; we provide next an algorithm computing the extension sets with the help of the Aho-Corasick automaton $\mathcal{A}_{\mathcal{U}}$;

(ii) we have to invert this matrix.

With the inclusion-exclusion approach, the $r \times r$ matrix is smaller and more compact than the linear system obtained by applying Chomsky-Schützenberger on the Aho-Corasick automaton of Section 2 which has size $O((\sum_{u \in \mathcal{U}} |u|)^2))$ since there are $O\left(\sum_{u \in \mathcal{U}} |u|\right)$ states in the automaton.

We exhibit an algorithm computing from the Aho-Corasick automaton (represented by a failure function) the multivariate matrix $\langle \mathcal{E} \rangle$ and the vector $\langle \mathbf{u} \rangle$ in time $O(r^2 \times s + \sum_{u \in \mathcal{U}} |u|)$ where r is the cardinality of $\mathcal{U}$ and s is the size of the longest suffix chain[(ii)] of a word $u \in \mathcal{U}$.

First we compute an auxiliary function which associates to any prefix w of the set $\mathcal{U}$ a vector $(f_i(w))_{i=1}^r$ defined by

$$f_i(w) = \langle v \rangle_i, \text{ for } v \in \mathcal{A}^* \text{ and } w \cdot v = u_i$$

We remark that $\langle \mathbf{u} \rangle = (f_1(\varepsilon), \ldots, f_r(\varepsilon))$. The "time complexity" (measured as the number of updates of the $f_i(w)$'s) of the following algorithm is $O(r \times \sum_{u \in \mathcal{U}} |u|)$.

INIT($\mathcal{A}_{\mathcal{U}}$)

for $i \leftarrow 1$ **to** r **do**
    $f_i(u_i) \leftarrow 1$
for $w \in \mathrm{Pref}(\mathcal{U})$ by a postorder traversal of the tree **do**
    for $i \leftarrow 1$ **to** r **do**
        for $\alpha \in \mathcal{A}$ such that $w \cdot \alpha \in \mathrm{Pref}(u_i)$ **do**
            $f_i(w) \leftarrow \pi(\alpha) z f_i(w \cdot \alpha) \prod_{j \neq i} (1 + t_j)^{[\![u_j \text{ suffix of } w \cdot \alpha]\!]}$
return $(f_i)_{1 \leq i \leq r}$

The matrix $\langle \mathcal{E}_{\mathbf{u}} \rangle$ is computed by the following algorithm. The time complexity of the main loop is $O(s \times r^2)$ where r is the number of words and s is the length of the longest suffix chain.

BUILD-EXTENSION-MATRIX($\mathcal{A}_{\mathcal{U}}$)

$\triangleright$ Initialize the matrix $(\mathcal{E}_{i,j})_{1 \leq i,j \leq r}$
for $i \leftarrow 1$ **to** r **do**
    for $j \leftarrow 1$ **to** r **do**
        $\mathcal{E}_{i,j} \leftarrow 0$
$\triangleright$ Compute the maps $(f_i(w))$ for $i = 1..r$ and $w \in \mathrm{Pref}(\mathcal{U})$
$(f_i)_{1 \leq i \leq r} \leftarrow$ INIT($\mathcal{A}_{\mathcal{U}}$)
$\triangleright$ Main loop
for $i \leftarrow 1$ **to** r **do**
    $v \leftarrow u_i$
    do   **for** $j \leftarrow 1$ **to** r **do**
            $\mathcal{E}_{i,j} \leftarrow \mathcal{E}_{i,j} + f_j(v)$
            $v \leftarrow \mathrm{Border}(v)$
    while $v \neq \varepsilon$
return E

From an algorithmic perspective we point the reader to Appendix A for a comparison of the automaton construction and the inclusion-exclusion method on a specific example: the covariance of the number of occurrences of two words.

(ii) The suffix chain of $u \in \mathcal{U}$ is the sequence $(u_1 = u, u_2 = \mathrm{Border}(u_1), u_3 = \mathrm{Border}(u_2), \ldots, u_s = \mathrm{Border}(u_{s-1}) = \varepsilon)$.

Conclusion and perspectives

We obtained a detailed proof and an explicit expression of the multivariate generating function counting texts according to their length and to their number of occurrences of words from a finite set. This result facilitates access to various moments and and may facilitate access to limiting distributions. From Bender and Kochman (BK93), we expect to find mostly a multivariate normal law for word counts. Our approach can possibly provide simpler criteria to decide if such a limiting law holds or not. Another nice aspect to the inclusion-exclusion approach is that it provides explicit formulae like Eq. (8), whereas the Aho-Corasick construction does not preserve the structure: even for a single pattern the autocorrelation polynomial does not come out easily and visibly.

We plan to extend the analysis to more complex sources, such as Markovian or dynamical sources (see Vallée (Val01)). We can probably improve on the complexity of computing the auxiliary functions f_i.

Acknowledgements

The authors thank Jérémie Bourdon and Bruno Salvy for fruitful discussions and for providing important feedback to this paper.

References

[AC75] Alfred Aho and Margaret Corasick. Efficient String Matching: An Aid to Bibliographic Search. *Communications of the ACM*, 18:333–340, 1975.

[BK93] Edward Bender and Fred Kochman. The distribution of subword counts is usually normal. *European Journal of Combinatorics*, 14:265–275, 1993.

[BV02] Jérémie Bourdon and Brigitte Vallée. Generalized pattern matching statistics. In *Proc. Colloquium on Mathematics and Computer Science: Algorithms, Trees, Combinatorics and Probabilities*, Birkhauser, Trends in Mathematics, pages 249–265, 2002.

[BV06] Jérémie Bourdon and Brigitte Vallée. Pattern matching statistics on correlated sources. In *Proc. of LATIN'06*, volume 3887 of *LNCS*, pages 224–237, 2006.

[CR02] Maxime Crochemore and Wojciech Rytter. *Jewels of Stringology*. World Scientific Publishing, Hong-Kong, 2002. 310 pages.

[CS63] Noam Chomsky and Marcel Schützenberger. The algebraic theory of context-free languages. *Computer Programming and Formal Languages,*, pages 118–161, 1963. P. Braffort and D. Hirschberg, eds, North Holland.

[FS07] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. xxx, 2007. In preparation (http://algo.inria.fr/flajolet/Publications/books.html).

[GJ79] Ian Goulden and David Jackson. An inversion theorem for clusters decompositions of sequences with distinguished subsequences. *J. London Math. Soc.*, 2(20):567–576, 1979.

[GJ83] Ian Goulden and David Jackson. *Combinatorial Enumeration.* John Wiley, 1983. New-York.

[GO81a] Leo Guibas and Andrew Odlyzko. Periods in strings. *J. Combin. Theory*, A(30):19–42, 1981.

[GO81b] Leo Guibas and Andrew Odlyzko. Strings overlaps, pattern matching, and non-transitive games. *J. Combin. Theory*, A(30):108–203, 1981.

[Kon05] Yong Kong. Extension of Goulden-Jackson cluster method on pattern occurrences in random sequences and comparison with Régnier Szpankowski method. *J. of Difference Equations and Applications*, 11(15):1265–1271, 2005.

[Lot05] M. Lothaire. *Applied Combinatorics on Words*. Encyclopedia of Mathematics. Cambridge University Press, 2005.

[Nic03] Pierre Nicodème. Regexpcount, a symbolic package for counting problems on regular expressions and words. *Fundamenta Informaticae*, 56(1-2):71–88, 2003.

[NSF02] Pierre Nicodème, Bruno Salvy, and Philippe Flajolet. Motif statistics. *Theoretical Computer Science*, 287(2):593–618, 2002.

[NZ99] John Noonan and Doron Zeilberger. The Goulden-Jackson Method: Extensions, Applications and Implementations. *J. of Difference Equations and Applications*, 5(4-5):355–377, 1999.

[PRdT95] Bernard Prum, F. Rodolphe, and E. de Turckheim. Finding words with unexpected frequencies in deoxyribonucleic acid sequences. *J. R. Statist. Soc. B*, 57(1):205–220, 1995.

[R0́0] Mireille Régnier. A unified approach to word occurrences probabilities. *Discrete Applied Mathematics*, 104(1):259–280, 2000. Special issue on Computational Biology.

[RS98] Mireille Régnier and Wojciech Szpankowski. On Pattern Frequency Occurrences in a Markovian Sequence. *Algorithmica*, 22(4):631–649, 1998.

[SF96] Robert Sedgewick and Philippe Flajolet. *An Introduction to the Analysis of Algorithms*. Addison-Wesley Publishing Company, 1996.

[Szp01] Wojciech Szpankowski. *Average Case Analysis of Algorithms on Sequences*. Series in Discrete Mathematics and Optimization. John Wiley & Sons, 2001.

[Val01] Brigitte Vallée. Dynamical sources in information theory: Fundamental intervals and word prefixes. *Algorithmica*, 29(1):262–306, 2001.

[vzG99] Joachim von zur Gathen. *Modern Computer Algebra*. Cambridge University Press, 1999. 768 pages.

[Wie86] Douglas Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1):54–62, January 1986.

A Complexity of computing the covariance of number of occurrences of two words

We provide in this appendix a case study, focusing on the computation of the covariance of the number of occurrences of two words. We place ourselves in the Bernoulli model so that the weight $\pi(w)$ given to a word is the product of probabilities of individual letters. This entails for instance for the alphabet $\mathcal{A}$ that $A(z) = \sum_{i\in\mathcal{A}} p_i z = z$.

We consider here two words u_1 and u_2 with $|u_1| = |u_2| = \ell$, and two random variables $X_1^{(n)}$ and $X_2^{(n)}$ counting the number of occurrences of u_1 and u_2 in random texts of size n. Since this covariance is equal to

$$\mathbf{Cov}\left(X_1^{(n)}, X_2^{(n)}\right) = \mathbf{E}\left(X_1^{(n)} X_2^{(n)}\right) - \mathbf{E}\left(X_1^{(n)}\right)\mathbf{E}\left(X_2^{(n)}\right),$$

and that we have easy access to $\mathbf{E}\left(X_1^{(n)}\right)$ and $\mathbf{E}\left(X_2^{(n)}\right)$, it remains to evaluate the joint moment $M_{1,2}^{(n)} = \mathbf{E}\left(X_1^{(n)} X_2^{(n)}\right)$ of the number of occurrences of two words. We compare both for the inclusion-exclusion method and for the automaton method the complexity of computing $M_{1,2}^{(n)}$. The complexity is expressed in terms of number of operations on real or rational numbers.

Some possible following steps of computation are summarized in Tables 1 and 2 (with the option to compute the joint moment either asymptotically or exactly). We refer to von zur Gathen and Gerhard (vzG99) for algorithms used in this section.

We have basically the same complexity for Step 1 for both methods, which requires $O(\ell)$ operations.

We give in the following two paragraphs some elements to justify the complexities stated in Tables 1 and 2.

Inclusion-Exclusion approach. By Equation (8), we write $1/(1-z-\xi(z, x_1-1, x_2-1))$ as a rational function $P(z, x_1, x_2)/Q(z, x_1, x_2)$ where $P(z, x_1, x_2)$ and $Q(z, x_1, x_2)$ are polynomials. This leads to consider

$$M_{1,2}(z) = \sum_{n\geq 0} M_{1,2}^{(n)} z^n = \left.\frac{\partial^2}{\partial x_1 \partial x_2} \frac{P(z, x_1, x_2)}{Q(z, x_1, x_2)}\right|_{x_1=x_2=1}. \qquad (10)$$

Introducing some polynomials to alleviate notations, i.e, $P(z) = P(z, 1, 1)$, $Q(z) = Q(z, 1, 1)$ and for any polynomial $U(z, x_1, x_2)$, we define

$$U_j(z) = \left.\tfrac{\partial}{\partial x_j} U(z, x_1, x_2)\right|_{x_1=x_2=1} \text{ for } j = 1, 2, \quad \text{and } U_{1,2}(z) = \left.\tfrac{\partial^2}{\partial x_1 \partial x_2} U(z, x_1, x_2)\right|_{x_1=x_2=1}.$$

Then one has the exact expression

$$M_{1,2}(z) = \frac{P_{1,2}}{Q(z)} - \frac{P_1(z)Q_2(z) + P(z)Q_{1,2}(z) + P_2(z)Q_1(z)}{Q(z)^2} + 2\frac{P(z)Q_1(z)Q_2(z)}{Q(z)^3}.$$

We claim that $P(z, x_1, x_2)$ and $Q(z, x_1, x_2)$ have at most $2\ell + 1$ terms (see Example 3 to be convinced). Therefore the formal differentiations of Step 2 require $O(\ell)$ operations on monomials. By Perron-Frobenius, we have a dominant pole for x_1 and x_2 real in a neighborhood of 1. Moreover we have $P(z, 1, 1)/Q(z, 1, 1) = 1/(1-z)$. This implies that Equation (10) can be expanded locally as a Laurent series

$$M_{1,2}(z) = \frac{a_3}{(1-z)^3} + \frac{a_2}{(1-z)^2} + \frac{a_1}{1-z} + a_0 + O(1-z),$$

giving access to an asymptotic expression for $M_{1,2}^{(n)} = [z^n]M_{1,2}(z)$. Therefore computing the joint moment $\mathbf{E}(X_1^{(n)} X_2^{(n)})$ asymptotically requires only to perform a finite number of product of polynomials in the variable z the degrees of which is $O(\ell)$; this corresponds to a complexity of order $O(\ell \log(\ell) \log\log(\ell))$ by using Fast Fourier Transforms for the polynomials multiplications.

Automaton approach. As noticed in Nicodème *et al.* (NSF02), when considering asymptotic computation of $M_{1,2}^{(n)}$, it is possible to avoid the inversion of a linear system of size ℓ with polynomial entries by expanding the system in a neighborhood of the dominant singularity $z = 1$ after differentiating with respect to x_1 and x_2 and substituting $x_1 = x_2 = 1$. Doing this avoids the computation of the multivariate generating function $F(z, x_1, x_2)$. This leads to handle a finite set of sparse linear systems of size ℓ, where the number of non-zero terms is $O(\ell)$. Using the Wiedemann algorithm (Wie86) it is therefore possible to compute the moment $M_{1,2}^{(n)}$ in $O(\ell^2)$ operations.

When exact computation is needed, it is necessary to compute the rational generating function $\frac{P(z,x_1,x_2)}{Q(z,x_1,x_2)}$. Note that to get to the result, the Aho-Corasick automaton may be minimized in (here negligible) time $O(\ell \log(\ell))$.

Starting from Equation (2), we remark that $\boldsymbol{F}(z, x_1, x_2) = (\mathbb{I} - z\mathbb{T}(x_1, x_2))^{-1}\mathbf{1}$, where $\mathbf{1}$ is the vector $(1, \dots, 1)^t$, is a vector of rational functions in z, x_1 and x_2. We write the $2\ell + 1$ first terms of the Taylor expansion of $\boldsymbol{F}(z, x_1, x_2)$ in the neighborhood of $z = 0$, which gives

$$\boldsymbol{F}(z, x_1, x_2) = \mathbb{I}\mathbf{1} + z\mathbb{T}\mathbf{1} + z^2\mathbb{T}\mathbb{T}\mathbf{1} + \cdots + z^{i+1}\mathbb{T}\mathbb{T}^i\mathbf{1} + \cdots + z^{2l+1}\mathbb{T}\mathbb{T}^{2l}\mathbf{1} + \dots$$

The principle is to benefit from the fact that $\mathbb{T}$ is a sparse matrix with $O(\ell)$ non null entries. Each $\mathbb{T}^i\mathbf{1}$ is a vector, entries of which are polynomials in x_1 and x_2 of at most i^2 terms. The cost of computing the $2\ell + 1$ first terms of the expansion is therefore $O(\ell^4)$. Multiplying to the left this expansion by the vector $(1, 0, \dots, 0)$ provide the $2\ell + 1$ first terms of the expansion of the rational function $F(z, x_1, x_2)$ which can be computed by using a Padé approximant with a cost $O(\ell^2) \times O(\ell^2 \log(\ell) \log\log(\ell)) = O(\ell^4 \log(\ell) \log\log(\ell))$ where the first term corresponds to the number of operations of computation of the Padé approximant and the second term to the multiplication of polynomials of the variable x_1 and x_2 of degree at most $2\ell + 1$ in the two variables (univariate polynomials are multiplied by FFT).

Binary Powering. For both approaches, the exact computation of $M_{1,2}^{(n)}$ follows by computing the recurrence associated to the rational fraction $M_{1,2}(z)$ (computed exactly), rewriting it as a matricial equation and using binary powering to compute the relevant powers of the matrix in $O(\log(n))$ operations.

Automaton approach (asympt.)	Complexity
1) Build the Aho Corasick automaton	$O(\ell)$
2) Inverse a linear system with constant coefficients	$O(\ell^2)$
Overall Cost	$O(\ell^2)$

Inclusion-exclusion (asympt.)	Complexity
1) Compute the right extension sets	$O(\ell)$
2) Differentiate and get first terms of Laurent series	$O(\ell \log(\ell) \log\log(\ell))$
Overall Cost	$O(\ell \log(\ell) \log\log(\ell))$

Tab. 1: Asymptotic computation of $M_{1,2}^{(n)}$ with the automaton approach (left) and inclusion-exclusion method (right) for two words of length ℓ and a text of length n.

Automaton approach (exact)	Complexity
1) Build the Aho Corasick automaton	$O(\ell)$
2a) Inverse a linear system with polynomial coefficients	$O(\ell^4 \log(\ell) \log\log(\ell))$
2b) Differentiate	$O(\ell \log(\ell) \log\log(\ell))$
3) Binary powering	$O(\log(n))$
Overall Cost	$O(\log(n) + \ell^4 \log(\ell) \log\log(\ell))$

Inclusion-exclusion (exact)	Complexity
1) Compute the right extension sets	$O(\ell)$
2) Differentiate	$O(\ell \log(\ell) \log\log(\ell))$
3) Binary powering	$O(\log(n))$
Overall Cost	$O(\log(n) + \ell \log(\ell) \log\log(\ell))$

Tab. 2: Exact computation of $M_{1,2}^{(n)}$ with the automaton approach (left) and inclusion-exclusion method (right) for two words of length ℓ and a text of length n.