A Word Counting Graph

Mireille Regnier, Zara Kirakossian, Eugenia Furletova, Mikhail Roytberg

A Word Counting Graph

Mireille Régnier[1] and Zara Kirakosyan[2] and Evgenia Furletova[3,4] and Mikhail Roytberg[3,4,5]

[1] INRIA, 78153 Le Chesnay, France
mireille.regnier@inria.fr
[2] Yerevan State University, 377 200 Yerevan, Armenia
zarulinka@yahoo.com
[3] Institute of Mathematical Problems of Biology, 142290, Institutskaya, 4, Pushchino, Russia
janny51@rambler.ru, mroytberg@gmail.com
[4] Pushchino State University, 142290, Prospect Nauki,5, Pushchino, Russia
[5] Laboratoire J.-V. Poncelet (UMI 2615), 119002, Bolshoy Vlasyevskiy Pereulok 11, Moscow, Russia

Abstract. We study methods for counting occurrences of words from a given set $\mathcal{H}$ over an alphabet V in a given text. All words have the same length m. Our goal is the computation of the probability to find p occurrences of words from a set $\mathcal{H}$ in a random text of size n, assuming that the text is generated by a Bernoulli or Markov model. We have designed an algorithm solving the problem; the algorithm relies on traversals of a graph, whose set of vertices is associated with the overlaps of words from $\mathcal{H}$. Edges define two oriented subgraphs that can be interpreted as equivalence relations on words of $\mathcal{H}$. Let $\mathcal{P}(\mathcal{H})$ be the set of equivalence classes and S be the set of other vertices. The run time for the Bernoulli model is $O(np(|\mathcal{P}(\mathcal{H})|+|S|))$ time and the space complexity is $O(pm|S|+|\mathcal{P}(\mathcal{H})|)$. In a Markov model of order K, additional space complexity is $O(pm|V|^K)$ and additional time complexity is $O(npm|V|^K)$. Our preprocessing uses a variant of Aho-Corasick automaton and achieves $O(m|\mathcal{H}|)$ time complexity. Our algorithm is implemented and provides a significant space improvement in practice. We compare its complexity to the additional improvement due to AhoCorasick minimization.

1 Introduction

Studies on word probabilities started as early as Eighties with the seed paper [GO81]. A recent interest arose from applications to computational biology [GKM00,Lat04,GK97,PMG00,VM03]. Numerous statistical softwares have been designed recently [CFG$^+$05,TML$^+$02,BPL$^+$04] to extract "In Silico" exceptional words , i.e. words that are either overrepresented or underrepresented in genomes. An international competition was organized by M. Tompa [TLB$^+$05] to evaluate their capabilities and weaknesses. All softwares combine an algorithm to extract candidate motifs and a statistical criterium to evaluate overrepresented or underrepresented motifs. Sensitivity and selectivity of such criteria turn out to be

crucial, as well as the speed and easiness of computation. A survey on motif searching can be found in [DD07].

We address here the following problem, that is fundamental to assess significance. Given the alphabet V and a set $\mathcal{H}$ of words on alphabet V, all words in $\mathcal{H}$ have the same length m. Our aim is to compute the probability to find exactly p occurrences of a word from a set $\mathcal{H}$ in a text of size n. Below, $t_n^{[p]}(\mathcal{H})$ denotes this probability. One assumes the text is randomly generated according to a Bernoulli or Markov model of order K. Naive solutions to this problem when $p = 1$ [CP90] lead to a $O(n|\mathrm{V}|^K|\mathcal{H}|^2)$ time complexity. Recent states of the art of various improvements can be found in [Szp01,Lot05]. On the one hand, the language approach defined in [RS97] allows for an elimination of $|\mathrm{V}|^K$ multiplicative factor in [Rég00]. On the other hand, several approximations can be derived for $t_n^{[1]}(\mathcal{H})$. They imply, at some stage, the computation of all possible overlaps of two words of $\mathcal{H}$ or a formal inversion of a $|\mathcal{H}| \times |\mathcal{H}|$ matrix of polynoms. Therefore, time complexity is $O(|\mathcal{H}|^2)$. A recent algorithm [HZGD05] allows for the computation of the generating function $\sum_{n\geq 0} t_n^{[1]}(\mathcal{H})z^n$ in $O(nm|\mathcal{H}|)$ time and $O(m|\mathcal{H}|)$ space complexity. Interestingly, $O(nm|\mathcal{H}|)$ time complexity outperforms $O(|\mathcal{H}|^2)$ complexity for most practical values of n. Indeed, sets may be rather big [TV07], especially when they are defined through a Position Specific Scoring Matrix or PSSM [Sto00]. Other algorithms simulate an automaton. This automaton is derived from a transition study [CS03] or language equations in [NSF02]; it is embedded in a Markov chain, using sparse matrices in [FL03,Nue06]. Algorithm AhoPro [BCR+07] addresses a more general problem. It computes the probability to find several occurrences (up to fixed bound p_i for set $\mathcal{H}_i$) in several sets $\mathcal{H}_1, \cdots, \mathcal{H}_q$. It simulates an Aho-Corasick like automaton. Its time and space complexity are $O(n(|S| + |\mathcal{H}|)|\mathrm{V}| \prod p_i)$ and $O((|S| + |\mathcal{H}|) \times \prod p_i)$, where $|S|$ is the size of the automaton. A rough upper bound for this size is $(m-1)|\mathcal{H}|$. This reduces to $O(pn(|S|+|\mathcal{H}|))$ and $O(p(|S|+|\mathcal{H}|))$ for p occurrences in a single set.

One common feature of these approaches is their easy extension to the Markov model. The multiplicative factor $|V|^K$ in time complexity [CP90] becomes an *additive* factor, except for [NSF02]. The main difference comes from the *space* complexity. As mentioned in [RR08], Markov chain embeddings may be costly, due to matrices sparsity. Although automaton implementations are less sparse, space complexity remains an actual drawback [BCR+07].

One possible way to reduce space requirement is the classical minimization of the underlying automaton. This is realized, for p occurrences in a single set $\mathcal{H}$, in [Nue08], with an $O(np|S||V|)$ time complexity. A recent publication [RR08] simulates the minimized automaton and implements a $\log n$ time complexity improvement, for a single set. In this paper, we propose an alternative modelling of word counting based on graphs that are related to overlaps of words of the set. We define a partition of set $\mathcal{H}$ into equivalence overlap classes that we represent in graphs and we show that the set of equivalence overlap classes, denoted $\mathcal{P}(\mathcal{H})$, is efficiently computed in a preprocessing step from a variant of classic Aho-Corasick tree, that can be built in $O(m|\mathcal{H}|)$ time. Let S denote the set of internal nodes, that are associated to proper overlaps. We also show

that probabilities $t_n^{[p]}(\mathcal{H})$ satisfy induction equations that only depend on these overlap classes. A simple algorithm follows, that relies on classical tree traversals with simple additional data structures that optimize memory management. As a whole, this algorithm improves on [NSF02,CS03,HZGD05,Nue06,BCR+07] as it achieves a $O(np(|\mathcal{P}(\mathcal{H})|+|S|))$ time computation and $O(pm|S|)$ space complexity with a *smaller* set S. Here, S is the set of vertices of the overlap graph that are not in $\mathcal{P}(\mathcal{H})$. Finally, this algorithm can be extended to address the same features as AhoPro. This drastic space improvement on space and time complexity is discussed in 5. We also compare it with the alternative minimization approach [Nue08]. Finally, we suggest on one example a possible combination with Aho-Corasick minimization.

2 Overlap Graphs

We will use terms *word* and *text* as synonyms; informally, a text is long, and a word is short. A pattern $\mathcal{H}$ of length m is a set $\mathcal{H} = \{\mathrm{H}_1, \cdots, \mathrm{H}_q\}$ of words such that all words have the same length m.

Given a word w, let $|w|$ denote its length. Given two words w and t, one notes $w \prec t$ iff w is a proper prefix of t and $w \subset t$ iff w is a proper suffix of t.

Definition 1. *Given a pattern $\mathcal{H}$ over an alphabet* V*, a word w is a* suffixprefix *word for $\mathcal{H}$* **iff** *exists* H, F *in $\mathcal{H}$ such as*

$$w \subset \mathrm{H} \text{ and } w \prec \mathrm{F} \ . \tag{1}$$

The set of suffixprefix words of a set $\mathcal{H}$ is called its overlap set *and denoted $OV(\mathcal{H})$.*

In all examples, we use DNA alphabet $V = \{A, C, G, T\}$.

Example 1. Let $\mathcal{H}$ be the set

$$\mathcal{H} = \{\ \mathrm{H}_1 = ACATATA, \mathrm{H}_2 = \mathrm{AGACACA}, \mathrm{H}_3 = \mathrm{ATACACA}, \mathrm{H}_4 = \mathrm{ATAGATA}\ ,$$
$$\mathrm{H}_5 = \mathrm{CATTATA}, \mathrm{H}_6 = \mathrm{CTTTCAC}, \mathrm{H}_7 = \mathrm{CTTTCCA}, \mathrm{H}_8 = \mathrm{TACCACA}\}\ .$$

Overlap set is $OV(\mathcal{H}) = \{ATA, ACA, TA, CA, A, AC, C, \epsilon\}$. One has, for example, $ACA \subset \mathrm{H}_2$ and $ACA \prec \mathrm{H}_1$.

One observes that prefix relation and suffix relation define partial orders on $OV(\mathcal{H}) \cup \mathcal{H}$. Clearly, the empty sequence ϵ is a prefix (respectively a suffix) of any suffixprefix and any word from $\mathcal{H}$. Moreover, the set of prefixes (respectively suffixes) of a word H in $\mathcal{H}$ is totally ordered. Thus, it admits a maximal element in $OV(\mathcal{H})$. Therefore, these relations naturally define equivalence relations between words in set $\mathcal{H}$ and, consequently, a partition of set $\mathcal{H}$.

Definition 2. *Given a pattern $\mathcal{H}$ over an alphabet* V*, the* left predecessor *of a word* H*, noted lpred*(H)*, is its longest prefix belonging to the overlap set $OV(\mathcal{H})$. In other words,*

$$lpred(\mathrm{H}) = \max\{w \prec \mathrm{H}, w \in OV(\mathcal{H})\}\ . \tag{2}$$

Analogously, the right predecessor *of* H*, noted rpred*(H)*, is its longest suffix belonging to the overlap set* $OV(\mathcal{H})$*. In other words,*

$$rpred(\mathrm{H}) = \max\{w \subset \mathrm{H}, w \in OV(\mathcal{H})\} \ . \tag{3}$$

Two words H *and* F *are said* left *(respectively* right*)* equivalent *iff they have the same left (respectively right) predecessor. For any pattern* H*, its left class is denoted* $\bar{\mathrm{H}}$ *and its right class is denoted* $\tilde{\mathrm{H}}$*.*

Two words H *and* F *are said* overlap equivalent *if they are both left and right equivalent. The* overlap class *(i.e. the class of overlap equivalence) of a word* H *is denoted* $\dot{H}$*. The set of all overlap classes is denoted* $\mathcal{P}(\mathcal{H})$*. One denotes lpred*($\dot{\mathrm{H}}$) *(respectively rpred*($\dot{\mathrm{H}}$)*) the common left (respectively right) predecessor of the patterns in* $\dot{\mathrm{H}}$*.*

Example 2. In Example 1, $\tilde{\mathrm{H}}_2 = \tilde{\mathrm{H}}_3 = \tilde{\mathrm{H}}_8$ as ACA is their largest *suffix* in $OV(\mathcal{H})$. There are four right classes $\tilde{C}_{ACA} = \{\mathrm{H}_2, \mathrm{H}_3, \mathrm{H}_8\}$, $\tilde{C}_{ATA} = \{\mathrm{H}_1, \mathrm{H}_4, \mathrm{H}_5\}$, $\tilde{C}_{AC} = \{\mathrm{H}_6\}$ and $\tilde{C}_{CA} = \{\mathrm{H}_7\}$.

$\bar{\mathrm{H}}_3 = \bar{\mathrm{H}}_4$ as ATA is their largest *prefix* in $OV(\mathcal{H})$. There are six left classes $\bar{C}_{ACA} = \{\mathrm{H}_1\}$, $\bar{C}_{ATA} = \{\mathrm{H}_3, \mathrm{H}_4\}$, $\bar{C}_{CA} = \{\mathrm{H}_5\}$, $\bar{C}_{TA} = \{\mathrm{H}_8\}$, $\bar{C}_C = \{\mathrm{H}_6, \mathrm{H}_7\}$, $\bar{C}_A = \{\mathrm{H}_2\}$.

There are eight overlap classes in $\mathcal{P}(\mathcal{H})$, that are the eight singletons of $\mathcal{H}$. In other words, $\mathcal{P}(\mathcal{H}) = \mathcal{H}$ and the partition is trivial.

Definition 3. *The* left overlap graph *is the oriented graph* $LOG_{\mathcal{H}}$ *built on the set of vertices* $OV(\mathcal{H}) \cup \mathcal{P}(\mathcal{H})$ *with an edge from vertex* s *to vertex* t *iff:*

$$s = lpred(t) = \max\{w \prec t, w \in OV(\mathcal{H})\} \ . \tag{4}$$

The right overlap graph *is the oriented graph* $ROG_{\mathcal{H}}$ *built on the set of vertices* $OV(\mathcal{H}) \cup \mathcal{P}(\mathcal{H})$ *with an edge from vertex* s *to vertex* t *iff*

$$s = rpred(t) = \max\{w \subset t, w \in OV(\mathcal{H})\} \ . \tag{5}$$

The overlap graph $GOV_{\mathcal{H}}$ *is the oriented graph* $LOG_{\mathcal{H}} \cup ROG_{\mathcal{H}}$*.*

As all words in an overlap class have the same predecessor in the prefix and suffix relations, this definition makes sense for vertices t that represent overlap classes. Moreover, one observes that all predecessors of a word are totally ordered and that the empty string is a minimal element for this order. Therefore, both left overlap graph and right overlap graphs are *trees* where the root is associated to the empty string ϵ that has no predecessor. The leaves are the elements of $\mathcal{P}(\mathcal{H})$. This definition implies the property below.

Property 1. Given a word H in $\mathcal{H}$, or a class $\dot{\mathrm{H}}$ in $\mathcal{P}(\mathcal{H})$, its ancestors in the *left* (respectively *right*) *overlap graph* are its prefixes (respectively suffixes) that belong to $OV(\mathcal{H})$.

Example 3. In Example 1, ancestors of H_1 and H_6 in left overlap graph, depicted in Figure 1, distinct from the root are $\{ACA, AC, A\}$ and $\{C\}$ respectively. In the right overlap graph, depicted in Figure 2, their ancestors distinct from the root, are $\{ATA, TA, A\}$ and $\{AC, C\}$. The paths associated to a node in $LOG_{\mathcal{H}}$ (respectively $ROG_{\mathcal{H}}$) are read from the top to the node (respectively, from the node to the top). [htb]

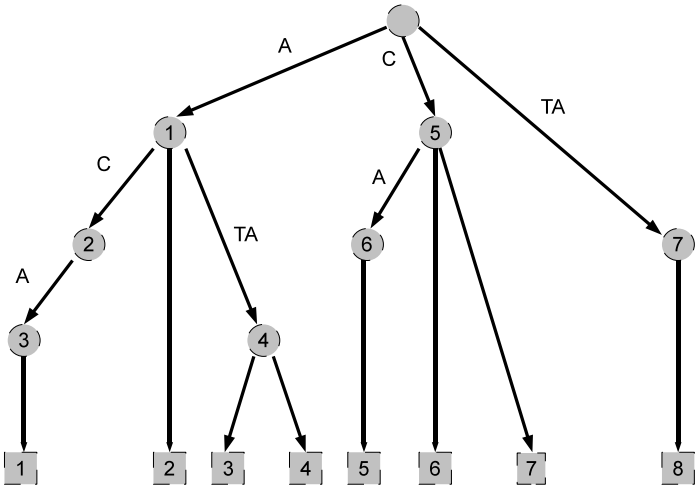

Fig. 1. Left overlap graphs for Example 1. Round nodes are the elements of $OV(\mathcal{H})$ and leaves are the elements of $\mathcal{P}(\mathcal{H})$. In that particular case, since $\mathcal{P}(\mathcal{H}) = \mathcal{H}$, each leaf i is even simply the word H_i.

Remark 1. Let us call *deep nodes* are the predecessors of overlap classes. In automaton theory, deep nodes represent left and right quotients of the recognized language, e.g. $\mathcal{H}$.

One observes that AC represents a right class and does not represent a left class. Nevertheless, the largest words in $OV(\mathcal{H})$, e.g. ACA and ATA, represent a right class and a left class.

Example 4. Let $\mathcal{G}$ be $\{ACNNNTA\}$ where N stands for any symbol in alphabet V. Pattern $\mathcal{G}$ contains $4^3 = 64$ words and its overlap set contains

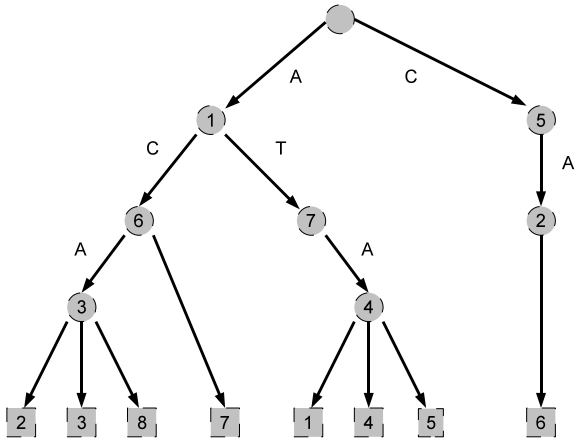

Fig. 2. Right overlap graphs for Example 1.Subwords are read bottom-up

6 words: $OV(\mathcal{G}) = \{A, ACTA, ACATA, ACCTA, ACGTA, ACTTA, \epsilon\}$. There are 12 overlap classes in $\mathcal{P}(\mathcal{G})$. For each class, the left predecessor is underlined and the right predecessor is given in bold. Nine overlap classes reduce to one element: $\dot{\mathrm{G}}_3 = \{\underline{A}C\mathbf{ACATA}\}$, $\dot{\mathrm{G}}_4 = \{\underline{A}C\mathbf{ACCTA}\}$, $\dot{\mathrm{G}}_5 = \{\underline{A}C\mathbf{ACGTA}\}$, $\dot{\mathrm{G}}_6 = \{\underline{A}C\mathbf{ACTTA}\}$, $\dot{\mathrm{G}}_7 = \{\underline{ACTA}\mathbf{CTA}\}$, $\dot{\mathrm{G}}_9 = \{\underline{ACATA}T\mathbf{A}\}$, $\dot{\mathrm{G}}_{10} = \{\underline{ACCTA}T\mathbf{A}\}$, $\dot{\mathrm{G}}_{11} = \{\underline{ACGTA}T\mathbf{A}\}$, and $\dot{\mathrm{G}}_{12} = \{\underline{ACTTA}T\mathbf{A}\}$. Three overlap classes contain several words: $\dot{\mathrm{G}}_2 = \{\underline{A}C[ACG]\mathbf{ACTA}\}$ and $\dot{\mathrm{G}}_8 = \{\underline{ACTA}[AGT]T\mathbf{A}\}$. The remaining 49 words of the pattern $\mathcal{G}$ form the last overlap class $\dot{\mathrm{G}}_1$: left and right predecessors of the class are equal to A. Interestingly, the minimal automaton that recognizes $\mathcal{G}$ has a larger number of states, namely 19.

Below, we define unions of overlap classes, as they appear in equations of words occurrences probabilities.

Definition 4. *Given a set $\mathcal{H}$ over an alphabet* V *and a word w in $OV(\mathcal{H})$, let $\bar{T}_w$ and $\tilde{T}_w$ be the subtrees rooted in w in the left and right overlap graphs, respectively. One defines the subsets of $\mathcal{P}(\mathcal{H})$:*

$$\tilde{C}_w = \tilde{T}_w \cap \mathcal{P}(\mathcal{H}) \ , \tag{6}$$

$$\bar{C}_w = \bar{T}_w \cap \mathcal{P}(\mathcal{H}) \ . \tag{7}$$

A set $\tilde{C}_w$ is called a right union *and a set $\bar{C}_w$ is called a* left union.

Remark 2. Given a right union $\tilde{C}_w$ and an overlap class $\dot{\mathrm{H}}$ in $\mathcal{P}(\mathcal{H})$, the set $\dot{\mathrm{H}} \cap \tilde{C}_w$ is either empty, or equal to $\dot{\mathrm{H}}$ when w is a suffix of any word in $\dot{\mathrm{H}}$. When w is a maximal suffixprefix, $\bar{C}_w$ and $\tilde{C}_w$ are overlap classes.

It follows from these definitions that a right (respectively left) union contains the leaves of tree $\tilde{T}_w$ (respectively $\bar{T}_w$). Therefore, Equations (8) and (9) follow immediately.

$$\tilde{C}_w = \cup_{x \in ROG_{\mathcal{H}}, w \subseteq x} \tilde{C}_x = \cup_{w \subset \mathrm{H}} (\dot{\mathrm{H}}) \ , \tag{8}$$

$$\bar{C}_w = \cup_{x \in LOG_{\mathcal{H}}, w \preceq x} \bar{C}_s = \cup_{w \prec \mathrm{H}} (\dot{\mathrm{H}}) \ . \tag{9}$$

In Example 1, $\tilde{C}_A = \tilde{C}_{ACA} \cup \tilde{C}_{ATA} \cup \tilde{C}_{CA} = \{\mathrm{H}_1, \mathrm{H}_2, \mathrm{H}_3, \mathrm{H}_4, \mathrm{H}_5, \mathrm{H}_7, \mathrm{H}_8\}$ and $\bar{C}_A = \bar{C}_{ACA} \cup \bar{C}_{ATA} \cup \bar{C}_A = \{\mathrm{H}_1, \mathrm{H}_2, \mathrm{H}_3, \mathrm{H}_4\}$. In Example 4, one has $\tilde{C}_{ACTA} = \dot{\mathrm{G}}_2 \cup \dot{\mathrm{G}}_7$.

Our bottom-up computations in 4.1 and 4.2 rely on these union properties.

Remark 3. The relation $\sim$ defined as $x \sim y$ iff $xt \in \mathcal{H}$ is equivalent to $yt \in \mathcal{H}$ is well-known in automata theory as right $\mathcal{H}$-equivalence. There is a bijection between the states of the minimal automaton that recognizes $\mathcal{H}$ and classes of right equivalence.

3 Equations for words occurrences probabilities

The aim of this section is to establish our master Theorems 1 and 2 which state induction relations for word occurrence probabilities, in the Bernoulli and Markov model. They rely on a factorization lemma to achieve announced time and space complexities to be discussed in 5.1.

3.1 Basic Notations and Equations

One assumes the text **T** may be generated according to either one of the two models below.

(B) BERNOULLI MODEL: every symbol s of a finite alphabet V is created independently of the other symbols, with probability p_s.

(M) MARKOV MODEL: in a Markov model of order K, the probability of a symbol occurrence at some position in the text depends on the K previous symbols.

In a Markov model of order K, let $\mathrm{P}_w(\mathrm{H})$ denote the probability to find H right of a given word w of length K. A Bernoulli model is viewed as a Markov model where K is 0. Given a word H, one denotes P(H) its probability in the Bernoulli model. Given a set of words $\mathcal{F}$, one denotes $\mathrm{P}(\mathcal{F}) = \sum_{\mathrm{H} \in \mathcal{F}} \mathrm{P}(\mathrm{H})$ and $\mathrm{P}_w(\mathcal{F}) = \sum_{\mathrm{H} \in \mathcal{F}} \mathrm{P}_w(\mathrm{H})$ its probability in the Bernoulli and the Markov models, respectively.

Definition 5. *Let $\mathcal{H} = \{\mathrm{H}_1, \mathrm{H}_2, \ldots, \mathrm{H}_\mathrm{q}\}$ be a given set of words with the same length m over an alphabet* V.

One denotes $t_n^{[p]}(\mathcal{H})$ (respectively $f_n^{[p]}(\mathcal{H})$) the probability to find exactly (respectively at least) p occurrences of $\mathcal{H}$ words in a text of size n.

One denotes $r_n^{[p]}(\mathrm{H})$ the probability to find word H *in text* **T**, *with its end at position n, under the condition that exactly $(p-1)$ other patterns from $\mathcal{H}$ appeared among positions $1, \cdots, n-1$.*

For a subset $\mathcal{F}$ of $\mathcal{H}$, one denotes $r_n^{[p]}(\mathcal{F})$ the probability to find a word of $\mathcal{F}$ ending at position n in text **T** *with exactly $(p-1)$ $\mathcal{F}$-words before it, i.e. $r_n^{[p]}(\mathcal{F}) = \sum_{\mathrm{H}\in\mathcal{F}} r_n^{[p]}(\mathrm{H})$.*

Property 2. For any integers n and p, one has

$$t_n^{[p]}(\mathcal{H}) = f_n^{[p]}(\mathcal{H}) - f_n^{[p+1]}(\mathcal{H}) \ ,$$
$$f_n^{[p]}(\mathcal{H}) = \sum_{k=0}^{n} r_k^{[p]}(\mathcal{H})$$

Therefore, the computation of $t_n^{[i]}$ follows from the computation of $(r_n^{[i]}(\mathrm{H}))$ where H ranges over $\mathcal{H}$.

Definition 6. *Given a word t and a prefix w of t, one denotes $\Phi_w(t)$ the suffix of t that satisfies $t = w\Phi_w(t)$. One denotes $\phi_w(t)$ the probability of $\Phi_w(t)$ in the Bernoulli model. By convention, $\phi_w(t) = 0$ if w is not a prefix of t.*

Given a set $\mathcal{F}$ of words with a left common ancestor w, one defines

$$\phi_w(\mathcal{F}) = \sum_{\mathrm{F}\in\mathcal{F}} \phi_w(\mathrm{F}) \ . \tag{10}$$

Example 5. For word H_1 from set $\mathcal{H}$ in Example 1, one has $\Phi_{ACA}(\mathrm{H}_1) = TATA$; then, in the Bernoulli model, $\phi_{ACA}(\mathrm{H}_1) = p_T^2 p_A^2$.

Proposition 1 below rewrites, using unions in overlap graphs defined above, a classical equation for words in $\mathcal{H}$ derived for $p = 1$ in [Szp01,Lot05,HZGD05] for the Bernoulli model. This rewriting will incorporate classes of overlap equivalence in Theorems 1 and allows for a further generalization to Markov models in 2. These two theorems allow for an implementation based on overlap graphs with announced complexities.

Proposition 1. *Assume a Bernoulli model. Let $\mathcal{H}$ be a given set of words and p be an integer. The probabilities $(r_n^{[1]}(\mathrm{H}))_{n\geq m}$ where* H *ranges in $\mathcal{H}$ satisfy the induction relation:*

$$r_n^{[1]}(\mathrm{H}) = \left[1 - \sum_{k=1}^{n-m} r_k^{[1]}(\mathcal{H})\right] \mathrm{P}(\mathrm{H}) - \sum_{w\in LOG_{\mathcal{H}}\setminus\{\epsilon\}, w\prec \mathrm{H}} r_{n-m+|w|}^{[1]}(\tilde{C}_w)\phi_w(\mathrm{H}) \ . \tag{11}$$

For any integer i satisfying $2 \leq i \leq p$, the probabilities $(r_n^{[i]}(\mathrm{H}))_{n \geq m}$ where H *ranges in $\mathcal{H}$ satisfy the induction relations:*

$$r_n^{[i]}(\mathrm{H}) = \sum_{k=1}^{n-m} \left[r_k^{[i-1]}(\mathcal{H}) - r_k^{[i]}(\mathcal{H}) \right] \mathrm{P}(\mathrm{H}) \tag{12}$$
$$+ \sum_{w \in LOG_{\mathcal{H}} \setminus \{\epsilon\}, w \prec \mathrm{H}} \left[r_{n-m+|w|}^{[i-1]}(\tilde{C}_w) - r_{n-m+|w|}^{[i]}(\tilde{C}_w) \right] \phi_w(\mathrm{H}) \ ,$$

with initial conditions:

$$r_1^{[i]}(\mathrm{H}) = \cdots = r_{m-1}^{[i]}(\mathrm{H}) = 0 \ , \quad 1 \leq i \leq p \ ,$$
$$r_m^{[1]}(\mathrm{H}) = P(\mathrm{H}) \ ,$$
$$r_m^{[i]}(\mathrm{H}) = 0 \ , \quad 2 \leq i \leq p \ .$$

Proof. We propose a proof of this Proposition in the Bernoulli model, when $p = 1$, that extends to p occurrences, $p \geq 1$. This Proposition is extended to overlap classes and to Markov model in Theorems 1 and 2. We study events that lead to a first occurrence and establish their probabilities. A word H is the first occurrence of $\mathcal{H}$ in text $\mathbf{T}$ at position n iff

(i) word H occurs, ending at position n;
(ii) no $\mathcal{H}$ word occurred up to position $n - m$;
(iii) no $\mathcal{H}$ word occurred in positions $n - m + 1, \cdots, n - 1$.

The probability of (ii) is $1 - \sum_{k=1}^{n-m} r_k^{[1]}(\mathcal{H})$. Term $(1 - \sum_{k=1}^{n-m} r_k^{[1]}(\mathcal{H}))P(\mathrm{H})$ is the probability of the simultaneous occurrence of (i) and (ii), as these events are independent in the Bernoulli model. One must substract the probability of the complementary event of (iii), conditioned by (i) and (ii). Complementary event of (iii) conditioned by (ii) is the union of $(m-1)$ disjoint events, $(E_l)_{1 \leq l \leq m-1}$, where each event E_l represents a first occurrence of a $\mathcal{H}$-word ending at position $n - m + l$. Each event E_l contains (i). Now, event E_l conditioned by (i) occurs iff a word F of $\mathcal{H}$ occurs at position $n - m + l$ when H ends at position n. This is satisfied iff exists w, with $|w| = l$ and $l \neq 0$, that is a prefix of H and a suffix of F. Equivalently, w is a left ancestor of H different from the root and F belongs to $\tilde{C}_w$.

This reasoning steadily extends. To find an i-th occurrence ending at position n, $(i-1)$ occurrences, are needed before $n - m$; an i-th occurrence is forbidden before position $n-m$ (condition (ii)), and between $n-m+1$ and $n-1$ (condition (iii)).

Example 6. Equation for word $\mathrm{H}_1 \in \mathcal{H}$ is:

$$r_n^{[1]}(\mathrm{H}_1) = [1 - \sum_{k=1}^{n-7} r_k^{[1]}(\mathcal{H}))]P(\mathrm{H}_1) - r_{n-6}^{[1]}(\tilde{C}_A)P(CATATA) -$$
$$- r_{n-5}^{[1]}(\tilde{C}_{AC})P(ATATA) - r_{n-4}^{[1]}(\tilde{C}_{ACA})P(TATA) \ .$$

As each word has no more than m ancestors, the recursive equations (11) and (12) provide an easy way to compute $(r_n^{[p]}(\mathrm{H}))_{\mathrm{H}\in\mathcal{H}}$ in $O(npm|\mathcal{H}|)$ time complexity. This complexity is achieved in [HZGD05] with a matrix and in [BCR+07] with an Aho-Corasick-like automaton. In [Nue08], the automaton is minimized, which yields a complexity $O(np|V||S|)$, where the size of the automaton is smaller. Overlap graphs provide an alternative improvement.

3.2 Overlap Graph Traversal

Our time complexity improvement, i.e. $O(np(|\mathcal{P}(\mathcal{H})|+|S|))$, where $S = OV(\mathcal{H})$, relies on a factorization property introduced below that reduces the computation to two graph traversals in Lemmas 1, 2 and 3. Space complexity issues will be discussed in Section 4 below.

Graph traversal Equations (11) and (12) are the same for all words H in $\mathcal{H}$, up to the coefficients $\mathrm{P}(\mathrm{H})$ and $\phi_w(\mathrm{H})$. Lemma 1 establishes that the computation of the second term in (11) or(12) for each *word* H or *left class* $\bar{\mathrm{H}}$ only requires *one* parameter $(\psi_n^{[p]}(t))$ defined for its left predecessor t in $LOG_{\mathcal{H}}$ and that this parameter can be computed in a traversal of the graph.

Definition 7. *In the Bernoulli model, one defines* $\{(\psi_n^{[i]}(w))_{w\in OV(\mathcal{H})\setminus\{\epsilon\}}\}_{1\le i\le p}$ *by the* p *top-down inductions*

$$\psi_n^{[i]}(w) = r^{[i]}_{n-m+|w|}(\tilde{C}_w), \text{ if } lpred(w) = \epsilon$$
$$\psi_n^{[i]}(w) = \psi_n^{[i]}(lpred(w))\cdot\phi_{lpred(w)}(w) + r^{[i]}_{n-m+|w|}(\tilde{C}_w), \text{otherwise } . \quad (13)$$

Lemma 1 establishes that $\psi_n^{[i]}(w)$ represents the information on the past that is shared by all descendants of w.

Lemma 1. *For any word* H *in* $\mathcal{H}$*, and any left ancestor* w *of* H *such that* $w \neq \epsilon$*, one has:*

$$\sum_{x\in OV(\mathcal{H})\setminus\{\epsilon\}, x\preceq w} r^{[i]}_{n-m+|x|}(\tilde{C}_x)\phi_x(\mathrm{H}) = \psi_n^{[i]}(w)\cdot\phi_w(\mathrm{H}) \ . \quad (14)$$

Any subset $\mathcal{T}$ *of a left class* $\bar{\mathrm{H}}$ *in* $\mathcal{H}$ *satisfies*

$$\sum_{\mathrm{F}\in\mathcal{T}}\sum_{w\in OV(\mathcal{H})\setminus\{\epsilon\}, w\prec\mathrm{F}} r^{[i]}_{n-m+|w|}(\tilde{C}_w)\phi_w(\mathrm{F}) = \psi_n^{[i]}(t)\cdot\phi_t(\mathcal{T}) \ , \quad (15)$$

where t *is the common left predecessor.*

Proof. Indeed, $\phi_x(\mathrm{H})$ rewrites $\phi_x(w)\phi_w(\mathrm{H})$. Therefore, $\psi_n^{[i]}$ inductive definition allow to prove inductively, for any i in $[1..p]$, that

$$\sum_{w\in OV(\mathcal{H})\setminus\{\epsilon\}, w\prec\mathrm{H}} r^{[i]}_{n-m+|w|}(\tilde{C}_w)\phi_w(\mathrm{H}) = \psi_n^{[i]}(t)\cdot\phi_t(\mathrm{H}) \ , \quad (16)$$

where t is the left predecessor of H in $LOG_{\mathcal{H}}$. A summation over all words in a subset T of left class $\bar{\mathrm{H}}$ and a factorization yields (15).

Second term in (11) or (12) may be computed once for any subset of words in a left class. Yet, probabilities computation for larger n implies to memorize separately right classes. Therefore, we split left classes with respect to right equivalence, which yields classes of $\mathcal{P}(\mathcal{H})$. A rewriting of Equation (8) leads to Lemma 2 below.

Lemma 2. *Let w be a node in $ROG_{\mathcal{H}}$. For any integer i or n, probabilities $r_n^{[i]}(\tilde{C}_w)$ satisfy:*

$$r_n^{[i]}(\tilde{C}_w) = \sum_{x \in OV(\mathcal{H}), w \subseteq x} r_n^{[i]}(\tilde{C}_x) = \sum_{\dot{\mathrm{H}} \subseteq \tilde{C}_w} r_n^{[i]}(\dot{\mathrm{H}}) \ . \tag{17}$$

Corollary 1. *The set $\{r_n^{[i]}(\tilde{C}_w)\}_{w \in OV(\mathcal{H})}$ can be computed from $\{r_n^{[i]}(\tilde{C}_x)\}_{x \in D_r}$ by a bottom-up traversal of $ROG_{\mathcal{H}}$.*

Theorem 1. *Let $\dot{\mathrm{H}}$ be a class in $\mathcal{P}(\mathcal{H})$ and denote t the left predecessor of $\dot{\mathrm{H}}$. In the Bernoulli model, probabilities $r_n^{[i]}(\dot{\mathrm{H}})$ satisfy*

$$r_n^{[1]}(\dot{\mathrm{H}}) = \left[1 - \sum_{k=1}^{n-m} r_k^{[1]}(\mathcal{H})\right] P(\dot{\mathrm{H}}) - \psi_n^{[1]}(t) \cdot \phi_t(\dot{\mathrm{H}}), \tag{18}$$

and, for $i \geq 2$,

$$r_n^{[i]}(\dot{\mathrm{H}}) = \sum_{k=1}^{n-m} \left[r_k^{[i-1]}(\mathcal{H}) - r_k^{[i]}(\mathcal{H})\right] P(\dot{\mathrm{H}}) + \left[\psi_n^{[i-1]}(t) - \psi_n^{[i]}(t)\right] \cdot \phi_t(\dot{\mathrm{H}}), \tag{19}$$

with initial conditions

$$\begin{aligned} r_1^{[i]}(\mathrm{H}) = \cdots = r_{m-1}^{[i]}(\mathrm{H}) = 0 \ , \quad 1 \leq i \leq p \ , \\ r_m^{[1]}(\mathrm{H}) = P(\mathrm{H}) \ , \\ r_m^{[i]}(\mathrm{H}) = 0 \ , \quad 2 \leq i \leq p \ . \end{aligned}$$

3.3 Markov extension

Unlike the algorithms based on finite automata accepting the corresponding texts (see e.g. [NSF02,CS03,HZGD05,Nue08,BCR+07,RR08]), our computation on the Markov model is a generalization of (11) that relies onto overlap graphs introduced above. An extension of the induction to the Markov model requires to extend the $(\psi_n^{[i]})$ family for nodes w with a depth smaller than K. This is realized with a suitable *partition* of union $\tilde{C}_w$. We use the notations of Definition 6 and restrict the presentation to the case $K \leq m$.

Definition 8. *Given a word w such that $K \leq |w|$, one denotes $suf_K(w)$ its suffix of size K. Given a word α in V^K, one denotes*

$$\mathcal{H}(\alpha) = \{\mathrm{H} \in \mathcal{H}; \alpha = suf_K(\mathrm{H})\} \ .$$

Given a word t such that w is a prefix of t, one defines parameter $\phi_w(t)$ by

$$\phi_w(t) = P_{suf_K(w)}(\Phi_w(t)) \ . \tag{20}$$

Remark 4. $\phi_w(t)$ represents the probability to find $\Phi_w(t)$ right after a w-occurrence. In Example 1, $\phi_{ACA}(\mathrm{H}_1) = P_{CA}(T)P_{AT}(A)P_{TA}(T)P_{AT}(A)$ in the Markov model of order 2.

Definition 9. *Given a set $\mathcal{H}$ and a Markov model of order K, with $K \leq m$, one defines the* right K-frontier *of $\mathcal{H}$, noted $RF_K(\mathcal{H})$, and the* left K-frontier*, noted $LF_K(\mathcal{H})$, by:*

$$RF_K(\mathcal{H}) = \{w \in OV(\mathcal{H}); |rpred(w)| < K \leq |w|\} \ , \tag{21}$$

$$LF_K(\mathcal{H}) = \{w \in OV(\mathcal{H}); |lpred(w)| < K \leq |w|\} \ . \tag{22}$$

The set of right K-leaves*, noted $RL_K(\mathcal{H})$, and the set of* left K-leaves*, noted $LL_K(\mathcal{H})$, are defined as:*

$$RL_K(\mathcal{H}) = \{\dot{\mathrm{H}} \in \mathcal{P}(\mathcal{H}); |rpred(\dot{\mathrm{H}})| < K\} \ , \tag{23}$$

$$LL_K(\mathcal{H}) = \{\dot{\mathrm{H}} \in \mathcal{P}(\mathcal{H}); |lpred(\dot{\mathrm{H}})| < K\} \ . \tag{24}$$

Example 7. In Example 1, the right 3-frontier is $\{ACA, ATA\}$ and the right 3-leaves are $\dot{\mathrm{H}}_6$ and $\dot{\mathrm{H}}_7$.

The right K-frontier and the set of right K-leaves allow to define for every word w such that $|w| < K$, a natural partition of each union $\tilde{C}_w$ defined by the successors of w in $ROG_{\mathcal{H}}$ that belong to these two sets.

Definition 10. *Given a word w in $OV(\mathcal{H}) \setminus \{\epsilon\}$ such that $|w| < K$, one defines a subset $S_K(w)$ of V^K as*

$$S_K(w) = \bigcup_{t \in RF_K(\mathcal{H}) \cup RL_K(\mathcal{H}), w \subset t} \{suf_K(t)\} \ . \tag{25}$$

Given a word w in $RF_K(\mathcal{H}) \cup RL_K(\mathcal{H})$, one denotes

$$S_K(w) = \{suf_K(w)\} \ .$$

Example 8. Let w be A in Example 1. Word t will range in $\{ACA, ATA, \mathrm{H}_7\}$. Therefore $S_3(A) = \{ACA, ATA, CCA\}$. Let w be AC. Then word t ranges in $\{\mathrm{H}_6\}$ and $S_3(AC) = \{CAC\}$.

Remark 5. With the notations above, one gets

$$\tilde{C}_w = (\cup_{\{t \in RF_K(\mathcal{H}); w \subseteq \mathrm{rpred}(t)\}} \tilde{C}_t) \quad \bigcup \quad (\cup_{\{\dot{\mathrm{H}} \in RL_K(\mathcal{H}); w \subseteq \mathrm{rpred}(\dot{\mathrm{H}})\}} (\dot{\mathrm{H}})) \ .$$

For instance, it follows from Example 8 that $\tilde{C}_{CA} = \tilde{C}_{ACA} \cup \{\mathrm{H}_7\}$.

Definition 11. *Given a word w in $OV(\mathcal{H}) \setminus \{\epsilon\}$ satisfying $|w| < K$, one defines for each α in $S_K(w)$:*

$$r^{[i]}_{n,\alpha}(w) = \sum_{\{t \in RF_K(\mathcal{H}); w \subseteq rpred(t), \alpha \subseteq t\}} r^{[i]}_n(\tilde{C}_t) \ + \sum_{\{\dot{\mathrm{H}} \subset RL_K(\mathcal{H}); w \subseteq rpred(\dot{\mathrm{H}}), \alpha \subseteq \dot{\mathrm{H}}\}} r^{[i]}_n(\dot{\mathrm{H}}) \ . \tag{26}$$

By convention, $r^{[i]}_{n,\alpha}(w) = 0$ if $\alpha \in V^K \setminus S_K(w)$.

Remark 6. It is guaranteed that at least one of the two sums contribute, and that $w \subset \alpha$.

We now show how this knowledge on the past allows for the computation of Equations (11) and (12).

Definition 12. *Given a word w in $OV(\mathcal{H})\setminus\{\epsilon\}$ satisfying $|w| < K$, one denotes $SP_K(w)$ the subset of V^K defined inductively as follows.*

$$SP_K(w) = \begin{cases} S_K(w) & \text{if } lpred(w) = \epsilon \\ S_K(w) \cup \{\alpha \in \mathrm{V}^K; S^{[\alpha]}(w) \neq \emptyset\} & \text{otherwise} \ , \end{cases} \tag{27}$$

where

$$S^{[\alpha]}(w) = \{\beta \in SP_K(lpred(w)); \alpha = suf_K(\beta \cdot \Phi_{lpred(w)}(w))\} \ . \tag{28}$$

Example 9. In Example 1, one has $SP_3(A) = S_3(A) = \{ACA, ATA, CCA\}$. Let w be AC. One has $\mathrm{lpred}(AC) = A$, and α ranges over $\{CAC, TAC\}$ when β ranges over $SP_3(A)$. Finally, $S^{[CAC]}(AC) = \{ACA, CCA\}$ and $S^{[TAC]}(AC) = \{ATA\}$ that are a partition of $SP_3(A)$. Finally, $SP_3(AC) = \{CAC, TAC\}$.

Definition 13. *Let w be any word in $OV(\mathcal{H}) \setminus \{\epsilon\}$.*

If $|w| < K$, one defines a family $(\psi^{[p]}_{n,\alpha}(w))_{\alpha \in SP_K(w)}$.

$$\psi^{[p]}_{n,\alpha}(w) = \begin{cases} r^{[p]}_{n-m+|w|,\alpha}(w) & \text{if } lpred(w) = \epsilon \\ \sum_{\beta \in S^{[\alpha]}(w)} \psi^{[p]}_{n,\beta}(x) P_\beta(\Phi_x(w)) + r^{[p]}_{n-m+|w|,\alpha}(w) & \text{otherwise} \end{cases} \tag{29}$$

If $|w| \geq K$, one defines a single function $\psi^{[p]}_n(w)$.

$$\psi^{[p]}_n(w) = r^{[p]}_{n-m+|w|}(\tilde{C}_w) + \begin{cases} \sum_{\beta \in S^{[suf_K(w)]}(w)} \psi^{[p]}_{n,\beta}(x) P_\beta(\Phi_x(w)) & \text{if } w \in LF_K(\mathcal{H}) \\ \psi^{[p]}_n(x) \cdot \phi_x(w) & \text{otherwise} \ . \end{cases} \tag{30}$$

Example 10. Let us use (26), (29) and (30) in Example 1.

(i) Let w be A. As $\mathrm{lpred}(A) = \epsilon$, one applies the first equation in (29) for α ranging in $S_3(A)$. Right members are computed through (26) which yields $\psi^{[p]}_{n,ACA}(A) = r^{[p]}_{n-6}(\mathrm{H}_2) + \mathrm{r}^{[\mathrm{p}]}_{\mathrm{n-6}}(\mathrm{H}_3) + \mathrm{r}^{[\mathrm{p}]}_{\mathrm{n-6}}(\mathrm{H}_8)$, $\psi^{[p]}_{n,ATA}(A) = r^{[p]}_{n-6}(\mathrm{H}_1) + r^{[p]}_{n-6}(\mathrm{H}_4) + \mathrm{r}^{[\mathrm{p}]}_{\mathrm{n-6}}(\mathrm{H}_5)$ and $\psi^{[p]}_{n,CCA}(A) = r^{[p]}_{n-6}(\mathrm{H}_7)$.

(ii) One uses (29) to compute $\psi^{[i]}_{n,TAC}(AC)$ and $\psi^{[i]}_{n,CAC}(AC)$. For instance,

$$\psi^{[i]}_{n,CAC}(AC) = \psi^{[i]}_{n,ACA}(A) \cdot P_{ACA}(C) + \psi^{[i]}_{n,CCA}(A) \cdot P_{CCA}(C) + r^{[i]}_{n-5,CAC}(AC) \ ,$$

where, using (26), $r^{[i]}_{n-5,CAC}(AC) = r^{[i]}_{n-5}(\dot{\mathrm{H}}_6)$.

(iii) Word A is the left predecessor of ATA that belongs to the right 3-frontier. Therefore, (30) yields

$$\psi_n^{[p]}(ATA) = \psi_{n,ACA}^{[p]}(A)P_{ACA}(TA) + \psi_{n,ATA}^{[p]}(A)P_{ATA}(TA) + \psi_{n,CCA}^{[p]}(A)P_{CCA}(TA) + r_{n-4}^{[p]}(\mathrm{H}_1) + r_{n-4}^{[p]}(\mathrm{H}_4) + \mathrm{r}_{\mathrm{n-4}}^{[\mathrm{p}]}(\mathrm{H}_5) \ .$$

Lemma 1 can now be rewritten into Lemma 3 below that steadily yields Theorem 2.

Lemma 3. *Any leaf class* $\dot{\mathrm{H}}$ *in* $\mathcal{P}(\mathcal{H})$ *with left predecessor* t *satisfies*

$$\sum_{w\in LOG_{\mathcal{H}}\setminus\{\epsilon\}, w\prec \mathrm{H}} r_{n-m+|w|}^{[p]}(\tilde{C}_w)\phi_w(\dot{\mathrm{H}}) = \psi_n^{[p]}(t)\cdot\phi_t(\dot{\mathrm{H}}) \ \textit{if}\ |t|\geq K \ , \tag{31}$$

$$= \sum_{\alpha\in V^K, t\subset\alpha} \psi_{n,\alpha}^{[p]}(t)\cdot\phi_t(\dot{\mathrm{H}}) \ \textit{if}\ |t|<K \tag{32}$$

Definition 14. *Let* $f_k^{[p]}(\alpha)$ *be the probability that a text of size* k, $k\geq m$, *ending with suffix* α *in* V^K, *contains at least* p *occurrences of words from* $\mathcal{H}$.

Lemma 4. *The probabilities* $f_k^{[p]}(\alpha)$ *satisfy the following induction*

$$f_k^{[p]}(\alpha) = \sum_{\beta\in V^K} f_{k-1}^{[p]}(\beta)\cdot P_\beta(\alpha) + r_k^{[p]}(\mathcal{H}(\alpha)) \ , \tag{33}$$

with the initial condition

$$f_m^{[p]}(\alpha) = r_m^{[p]}(\mathcal{H}(\alpha)) \ .$$

Theorem 2. *Let* $\dot{\mathrm{H}}$ *be a class in* $\mathcal{P}(\mathcal{H})$ *and* t *be its left predecessor. In the Markov model, probabilities* $r_n^{[i]}(\dot{\mathrm{H}}(\beta))$ *are equal to*

$$\sum_{\alpha\in V^K}(f_{n-m}^{[i-1]}(\alpha) - f_{n-m}^{[i]}(\alpha))P_\alpha(\dot{\mathrm{H}}(\beta)) + \begin{cases} (\psi_n^{[i-1]}(t) - \psi_n^{[i]}(t))\cdot\phi_t(\dot{\mathrm{H}}) & if\ |t|\geq K \\ \sum_{\alpha\in V^K}(\psi_{n,\alpha}^{[i-1]}(t) - \psi_{n,\alpha}^{[i]}(t))\cdot\phi_t(\dot{\mathrm{H}}) & if\ \dot{\mathrm{H}}\subset LL_K(\mathcal{H}) \end{cases} \tag{34}$$

with initial conditions

$$r_1^{[i]}(\mathrm{H}) = \cdots = r_{m-1}^{[i]}(\mathrm{H}) = 0 \ , \quad 1\leq i\leq p \ ,$$
$$r_m^{[1]}(\mathrm{H}) = P(\mathrm{H}) \ ,$$
$$r_m^{[i]}(\mathrm{H}) = 0 \ , \quad 2\leq i\leq p \ ,$$

and convention $\psi_n^{[0]}(t) = 0$ *and* $\psi_{n,\alpha}^{[0]}(t) = 0$.

4 Basic Algorithm

We describe a procedure to compute the probabilities of words occurrences in texts. Our input is a set of words $\mathcal{H}$. Our output is the set of probabilities $(r_n^{[i]}(\mathcal{H}))_{1\leq i\leq p}$ for a given position n. Overlap graph modelling provides a straighforward but efficient algorithm.

4.1 Overlap Graphs Construction

Our preprocessing step builds right and left overlap graphs and partition $\mathcal{P}(\mathcal{H})$. It relies on a tree $\mathcal{T}_\mathcal{H}$ built with Aho-Corasick algorithm [AC75]. A basic feature of Aho-Corasick is the on-line construction of so-called *suffix links*. Given a word s in V^*, suf(s) denotes the largest suffix of s that is a proper prefix of some word from the set $\mathcal{H}$. Below, we identify each node or leaf with the word defined by the path from the root. A suffix link is a link from a node s to node suf(s).

Right overlap graph We start with the right overlap graph. According to Definition 1, a node in $\mathcal{T}_\mathcal{H}$ belongs to $OV(\mathcal{H})$ iff there exists a sequence of suffix links from a leaf to this node. Therefore, a bottom-up traversal of $\mathcal{T}_\mathcal{H}$ according to the suffix links allows one to find out the nodes from $OV(\mathcal{H})$ and to construct $ROG_\mathcal{H}$. See Figure 3.

Left overlap graph To link the nodes in $OV(\mathcal{H})$ with the links for $LOG_\mathcal{H}$, one can use a depth-first traversal of $\mathcal{T}_\mathcal{H}$.

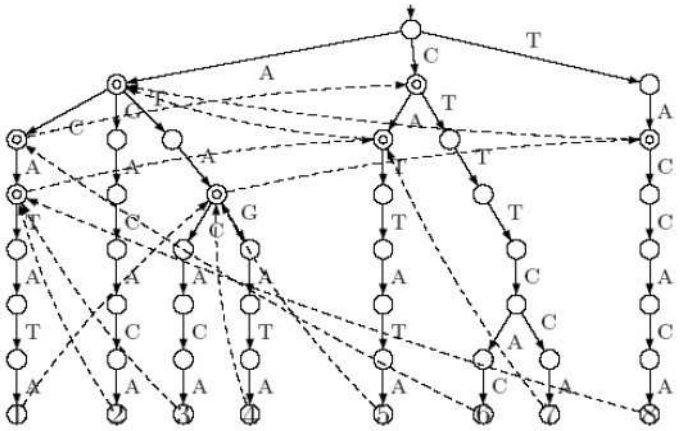

Fig. 3. Aho-Corasick tree $\mathcal{T}_\mathcal{H}$ for set $\mathcal{H}$ in Example 1 used to build $ROG_\mathcal{H}$ and $LOG_\mathcal{H}$ in Figures 1 and 2. The nodes corresponding to $OV(\mathcal{H})$ are marked with double circles. A leaf corresponding to the word H_i in $\mathcal{H}$ is labeled with the number $i, 1 \leq i \leq 8$. The dashed lines depicts suffix links needed to construct $ROG_\mathcal{H}$; the other Aho-Corasicks suffix links are omitted.

Partition Thanks to the order of AC tree, a depth-first traversal of $LOG_\mathcal{H}$, *MaxSufPref*, achieves a partition of $\mathcal{H}$ into $\mathcal{P}(\mathcal{H})$. As all words of a class are left equivalent, it is enough to identify below each deep node x all words that are right equivalent and number the designed classes. To achieve this linearly, one maintains in each internal node s the class number i(s) of the last visited leaf H

in $\mathcal{H}$ satisfying suf(H) $= s$. One achieves an increasing numbering; let j denote the number of the last class created before the visit of leaves below x. When a leaf H is visited, an equivalent leaf has been previously visited iff $i(\text{suf(H)}) > j$. In this case, leaf H is addressed to class $i(\text{suf(H)})$. Otherwise, it is given the next available class number k and $i(\text{suf(H)})$ is updated to k.

Markov model Main preprocessing step is the computation of the right K-frontier and the right K-leaves, which is $O(|V|^K)$.

Aho-Corasick construction of the tree, including the suffix links, can be achieved in time and space $O(m|\mathcal{H}|)$ [CR02]. Subgraph extraction algorithms are classical graph traversals. They run with $O(|\mathcal{T}_\mathcal{H}|)$ time and space complexity and $O(|\mathcal{T}_\mathcal{H}|)$ is upper bounded by $O(m|\mathcal{H}|)$ [AC75]. Once these steps have been performed, tree $\mathcal{T}_\mathcal{H}$ is not needed any more and space can be released.

4.2 Algorithm and basic data structures

Our algorithm computes inductions (19) and (34) in a depth-first traversal of $LOG_\mathcal{H}$. Bottom-up traversals of $ROG_\mathcal{H}$ allow to manage memory: an extraction of suitable information is performed at the beginning of each cycle and an update is realized at the end of each cycle. For sake of clarity, we first describe the Bernoulli model.

Memory management One observes that results of previous computations that are used in (19) are $\sum_{k=1}^{n-m}(r_k^{[i-1]}(\mathcal{H}) - r_k^{[i]}(\mathcal{H}))$ and $(\psi_n^{[i]}(t))$ that depend on $(r^{[i]}_{n-m+|w|}(\tilde{C}_w))_{w\in LOG_\mathcal{H}\setminus\{\epsilon\}}$.

1. Values $\sum_{k=1}^{n-m} r_k^{[i]}(\mathcal{H})$ where i ranges in $\{1\cdots p\}$ are memorized in p global variables, e.g. an array $SumProb$ of p integers.
2. For each node w at depth $|w|$ in $LOG_\mathcal{H} \setminus \{\epsilon\}$, one needs, to perform computation at cycle n, values $(r^{[i]}_{n-m+|w|}(\tilde{C}_w))_{1\le i\le p}$. Therefore, one memorizes the $p(m-|w|)$ values $\{r^{[i]}_{n-m+|w|}(\tilde{C}_w), \cdots, r^{[i]}_{n-1}(\tilde{C}_w)\}_{1\le i\le p}$ in an array $ProbMark$ of size $p(m-|w|)$.
3. In each node w and in each leaf class $\dot{\text{H}}$ in $LOG_\mathcal{H} \setminus \{\epsilon\}$, one memorizes probability $\phi_x(w)$ or $\phi_x(\dot{\text{H}})$, where x is its father in $LOG_\mathcal{H}$.

Algorithm For any inductive step n, our algorithm executes:

1. In a bottom-up traversal of $ROG_\mathcal{H}$, including the root, $ProbMainCalc$ extracts from $ProbMark$, for each internal node, the p values $(r^{[i]}_{n-m+|w|}(\tilde{C}_w))_{1\le i\le p}$. They are addressed in a field of each node. At the root, $(r_n^{[i]}(\mathcal{H}))$ are extracted.
2. In a depth-first traversal of $LOG_\mathcal{H}$:
   (a) when one visits an internal node, $(\psi_n^{[i]}(w))_{1\le i\le p}$ are computed according to (13);

(b) when a leaf class $\dot{\mathrm{H}}$ is visited:
   (i) compute $r_n^{[i]}(\dot{\mathrm{H}})$ using (19), for i in $\{1, \cdots, p\}$;
   (ii) Add $r_n^{[i]}(\dot{\mathrm{H}})$ to a temporary variable, $FirstTemp[i]$, in its predecessor in $ROG_{\mathcal{H}}$ (a deep node).

3. In a bottom-up traversal of $ROG_{\mathcal{H}}$, update $ProbMark$ for internal nodes, using $FirstTemp$ according to (17). At the root ϵ, $r_n^{[i]}(C_\epsilon) = r_n^{[i]}(\mathcal{H})$ is added to $SumProb$.

Markov model For memory management, it is now necessary to store values $\sum r_k^{[i]}(\mathcal{H}(\beta))$ for all words β in V^K. Moreover, for a node w at depth smaller than K, one stores, for each i, non-zero values $r_{n,\alpha}^{[i]}(w)$. For a node w in the left K-frontier or above it, and for a leaf class $\dot{\mathrm{H}}$ in the left K-leaves, one must memorize several values $P_\beta(\Phi_x(w))$ or $P_\beta(\Phi_t(\dot{\mathrm{H}}))$.

Procedures are slightly modified. In nodes above the right K-frontier, one extracts additional values, namely $r_{n,\alpha}^{[i]}(w)$, in step 1. In step 2, families $(\psi_n^{[i]}(w))$ computation relies on (29) and necessitates a preprocessing to memorize all $\phi_x(w)$.

If a leaf class $\dot{\mathrm{H}}$ is a left K-leaf, one computes $r_n^{[i]}(\dot{\mathrm{H}})$ through (34) instead of (19). In step 3, the bottom-up traversal stops at the right K-frontier.

5 Complexity and experimental results

5.1 Complexity

Overlap graph complexity Our algorithm achieves an overall $O(np(|S|+|\mathcal{P}(\mathcal{H})|))$ time complexity, where $S = OV(\mathcal{H})$. Indeed, depth-first traversal at step 2 and bottom-up traversals at steps 1 and 3 yield a $O(p|S|)$ time complexity. Computation of (19) is done for each overlap class in constant time. This yields an $O(|\mathcal{P}(\mathcal{H})|)$ time complexity in the Bernoulli model.

Temporary memory for $\psi_n^{[p]}(w)$ computation or probability update is $O(p|S|)$. In our current implementation, $ProbMark$ is built for all internal nodes, which yields a $O(mp|S|)$ space complexity. This already improves on recent algorithms [HZGD05,BCR+07]. This upper bound can be improved. Indeed, given a node w at depth k, one needs only memorize $(r_{n-l}^{[i]}(\tilde{C}_w))$ when for $1 \leq i \leq p$ and $1 \leq l \leq m-k$. Therefore, space complexity is pD with

$$D = \sum_{w \in S} [m - |w|] \ . \tag{35}$$

When a Markov model is used, additional space is needed. First, m values $f_k^{[i]}(\alpha)$ use $O(pm|V|^K)$ space. Then, for a node w, constraint $w \subset \alpha$ yields at most $|V|^{K-|w|}$ non-zero values $r_{n,\alpha}^{[i]}$. Additional space is $O(pK|V|^K)$. Finally, computation of (29) needs values $P_\beta(\Phi_x(w))$. A node x needs at most $|V|^{K-|x|}$ values $P_\beta(\Phi_x(w))$. Summing over at most $|V|^{|x|}$ nodes at level $|x|$ yields

an upper bound $O(pK|V|^K)$. Therefore, overall additional space complexity is $O(pm|V|^K)$. As each term is computed once at each inductive step, in $O(1)$ time, additional time complexity is $O(npm|V|^K)$.

Previous algorithms Exact computation of p occurrences probability depends on the text size n as a linear function $O(n)$ or a logarithmic function $O(\log n)$. Let us mention that there also exist *approximate* computations that may be realized in constant time with some drawbacks on the space complexity or the tightness of the approximation. They are beyond the scope of this paper.

We now discuss exact computation. All approaches, including automata or Markov Chain embedding [NSF02,CS03,FL03,Nue08], matrices [HZGD05] or languages [Rég00] need to compute a set of *linear* equations of order m with *constant* coefficients. Therefore, theoretical complexity is known. Algorithms that are linear in n, such as RegExpCount, AhoPro or Spatt achieve a complexity $p|S|n$ where $|S|$ is the size of the structure used for the *computation*. Using the classical system reduction [Rég00], it is theoretically possible to achieve a $O(m^p \log n)$ algorithm, with a constant factor that depends of the data structure to be used. This optimization has been recently implemented in [RR08]. Nevertheless, one observes that the m^p factor may represent a significant drawback. Therefore, counting algorithms are compared below on the basis of the two data structures used in the computation and the memorization.

[HZGD05] data structure is $O(|\mathcal{H}| \log |\mathcal{H}|)$. Markov Chain Embedding algorithms use very sparse matrices. Therefore, RegExpCount and AhoPro automata outperform them, and the minimization achieved in [Nue08] is a further improvement. Overlap graph is an alternative approach. In both cases, there is no non-trivial upper bound for the ratio between $|S|$ and the size of Aho-Corasick automata. Remark that one must take into account the alphabet size V in automata time complexity and a multiplicative factor m in overlap graph space complexity. When a minimization is very efficient, (pattern $ANNNNNA$), an overlap graph is less significantly reduced. Our last example suggests that building an overlap graph on a minimized automaton is an improvement and might be optimal. Indeed, for set $\mathcal{G}$ in Example 4, minimized Aho-Corasick automaton has 3 final states and 19 internal nodes. Overlap graph for this minimized automaton contains 4 classes and 4 internal nodes, associated to prefixes $\epsilon, C, CC = GG$ and $CCCC$. This difference is mainly due to backward links, that are necessary for a searching algorithm, but are unnecessary for counting algorithms, and therefore erased by overlap graphs.

5.2 Experiments

We have performed computer experiments to compare our approach and the approach using finite automaton representation of the corresponding set of text (e.g. [BCR$^+$07,Nue08]). We used a Intel Celeron 2.26 GHz with Windows XP. The complexity of automaton based methods can be described with the size $|\Sigma|$ of minimal automaton accepting the set of texts containing the desired number of patterns. The proposed approach leads to other characteristics of the pattern:

(i) $|S|$ is the number of overlaps;
(ii) $|\mathcal{P}(\mathcal{H})|$ is the number of *overlap classes*, i.e. equivalence classes in the overlap-based partition of the given word set;
(iii) D is the sum of differences $m - |w|$ over all suffix-prefixes w defined in (35).

In our case, it was stated above in section 5.1) that the run-time is proportional to $|S| + |\mathcal{P}(\mathcal{H})|$ and the required space is proportional to D.

First, we studied the dependence of the size $|S|$ of overlap graph on the size and length of the pattern $\mathcal{H}$. Data are given in Table 2 and depicted in Figure 4 Surprisingly, the average value of $|S|$ for random patterns does not depend on the pattern length m while the size of the minimal automaton grows almost linearly with the growth of the pattern length (the number of words in the pattern being fixed). One can see that for reasonable pattern lengths (up to 24) and sizes (up to 1500 words) the value $|S|$ (and even $|S| + |\mathcal{P}(\mathcal{H})|$) is less than the number $|\Sigma|$ of states of the corresponding minimized automaton. The independence of value $|S|$ of the pattern length can be justified theoretically (see Theorem 3 in Section 6).

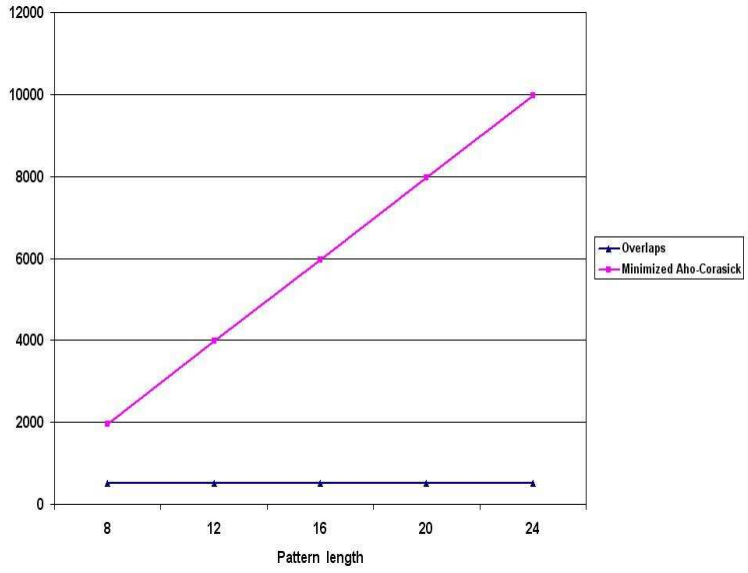

Fig. 4. Overlap graph size $|S|$ and minimal automaton size $|\Sigma|$. Average values for random patterns of size 500. Blue diamond shapes represent $|S|$ and purple squares represent $|\Sigma|$.

Figure 5 demonstrates the impact of usage of overlap classes instead of individual words (cf. equations (11) and (19)). We have generated 30 sets of $50k$ random words ($k = 1, 2, \cdots, 30$) of length 8 and 12; the random words were generated in 4-letter alphabet $\{A, C, G, T\}$ using Bernoulli model with equal probabilities of all letters. In the vast majority of cases, the number of overlap classes coincides with the initial number of words. We give the line only for the random patterns of length 8; the line for random patterns of length 12 coincides with it. Two lower lines corresponds to the patterns based on two position-specific scoring matrices (PSSM). Given a PSSM $\mathcal{M}$, a threshold or cutoff s defines a set $\mathcal{F}(\mathcal{M}, s)$, that consists of all the words in V^m with a score greater than s. Different cutoffs were used to obtain patterns of different sizes. The PSSMs were created by Dan Pollard for Drosophila genes (http://www.danielpollard.com/matrices.html) and are given in Table 1. The PSSMs are referred to as PSSM-08 and PSSM-12; their lengths are 8 and 12, respectively. Cutoffs are chosen to ensure that the set sizes range from 1 to 1500. One can see that in case of non-random patterns usage of overlap classes may bring significant improvement.

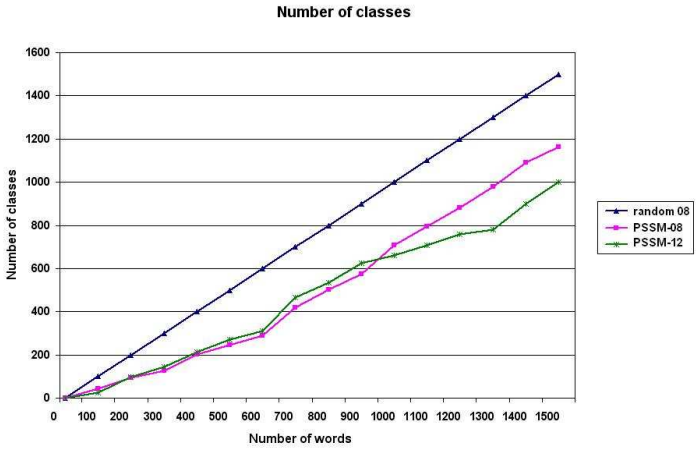

Fig. 5. Number of overlap classes as a function of the number of words in the pattern. The data are given for three series of patterns: (1) random patterns of length 8; (2) patterns corresponding to two PSSMs with different cutoffs. The upper line (blue triangles) corresponds to random patterns.

Tables 3, 4, 5 and 6 demonstrate results of experiments with the above four series of patterns. For each pattern we show its length and size, as well as above complexity measures related both to automaton-based and overlap-based approaches. Columns *Time* and *Space* represent time and space complexity of algorithms. For the automaton-based approach the run-time is proportional to

-	A	C	G	T
1	0.405	-1.424	0.301	-0.097
2	-0.944	0.122	-0.097	0.454
3	-0.622	-0.622	-3.989	1.067
4	1.041	-3.989	-1.156	-0.182
5	1.353	-3.989	-3.989	-2.380
6	-3.989	-1.792	-1.792	1.294
7	-3.989	-0.770	0.054	0.901
8	0.702	-3.989	0.665	-3.989

-	A	C	G	T
1	0.368	-2.197	-0.588	0.636
2	0.000	0.000	0.000	0.000
3	0.636	-2.197	-2.197	0.636
4	-2.197	-2.197	-2.197	1.299
5	1.299	-2.197	-2.197	-2.197
6	-2.197	-2.197	-2.197	1.299
7	-2.197	1.299	-2.197	-2.197
8	-2.197	-2.197	1.299	-2.197
9	1.022	0.000	-2.197	-2.197
10	0.000	-0.588	-2.197	0.847
11	0.847	-0.588	-0.588	-0.588
12	0.636	-0.588	0.368	-2.197

Table 1. Position-specific scoring matrices (PSSM) built to study Drosophila genes (http://www.danielpollard.com/matrices.html) of lengths 8 and 12.

$n \cdot p \cdot |V| \cdot |\Sigma|$ and the required space is proportional to $p \cdot |\Sigma|$ (see [Nue08]); here n is the length of the text and p is the desired number of occurrences. As we have shown in section 5.1, in our approach we get analogous formulas, namely $O(n \cdot p \cdot (|S| + |\mathcal{P}(\mathcal{H})|))$ for time and $O(p \cdot D)$ for space. Thus, in our comparison we use $|V| \cdot |\Sigma|$ and $|\Sigma|$ as time and space coefficients for automaton-based algorithm; we use $|S| + |\mathcal{P}(\mathcal{H})|$ and D as analogous coefficients for the overlap-based approach.

In Figures 6, 7, 8 and 9, one can see that in the vast majority of cases the coefficients for overlap approach are less than ones for the automaton approach. Time coefficients and space coefficients for PSSM-08 patterns are depicted in Figures 6 and 7, respectively. The average ratios of time coefficients are 4.64 (for patterns of length less than 500); 2.12 (for patterns with lengths between 500 and 1000); 1.65 (for patterns of length between 1000 and 15004). The plot for PSSM-12 is similar (the plot is not shown, see Tables 3 and 4). The advantage of overlap approach for space coefficients (in case of PSSM-08 patterns) is not so high compared to time coefficients, but still significant (see Figure 7). However for the PSSM-12 patterns, the space coefficients for the automaton approach are in average slightly better (the plot is not shown, see Tables 3 and 4).

The data on random patterns (see Figures 8 and 9) demonstrate that in this case the overlap-based approach works better. The lines depicting the behavior of the overlap approach for random patterns of different lengths almost coincide. This corresponds to the results given in Table 2 and Figure 4. In contrast, the automaton approach coefficients for the patterns of length 12 are significantly greater than ones for the patterns of length 8.

6 Discussion

The majority of the algorithms [NSF02,CS03,FL03,Nue06,BCR+07,Nue08] computing probability to find the given number of occurrences of a pattern in a ran-

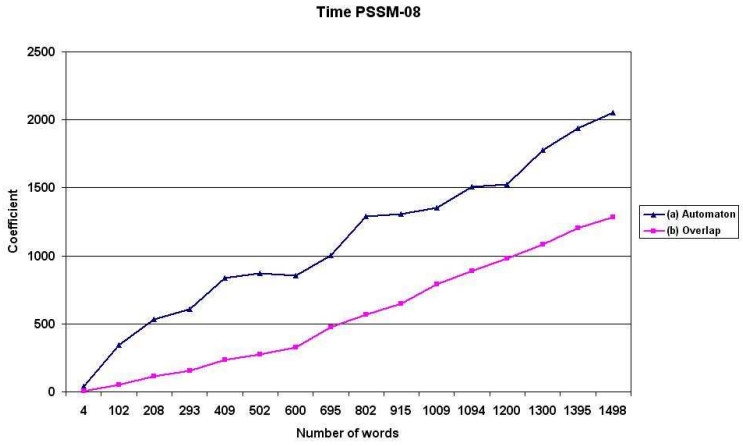

Fig. 6. Time coefficients of the patterns determined by PSSM-08

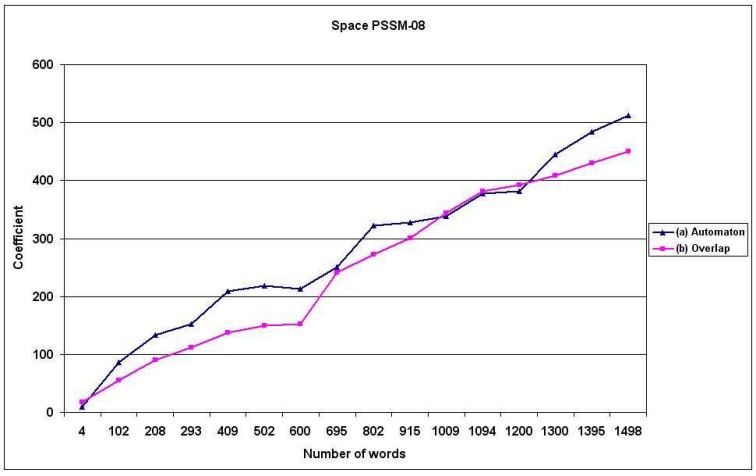

Fig. 7. Space coefficients of the patterns determined by PSSM-08

dom text are based on the usage of the automaton accepting the appropriate set of texts. The structure of the automaton reflects the set of suffix-prefixes (overlaps) of the given pattern but relation between states of the minimal automaton and overlaps can be quite complicated.

In this paper we present another approach based on explicit usage of partition of the given word set according to the structure of its overlaps. Recursive equations, (11) used here and in [HZGD05], (12) and (19) employ subtraction of probabilities and therefore cannot be reduced to the automaton-based equations for any automaton.

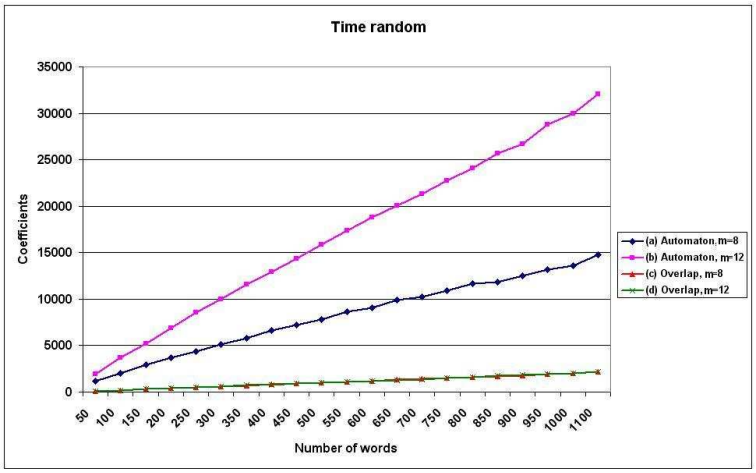

Fig. 8. Time coefficients for the random patterns of lengths 8 and 12. The graphs for overlap coefficients of lengths 8 and 12 (the lowest lines) almost coincide

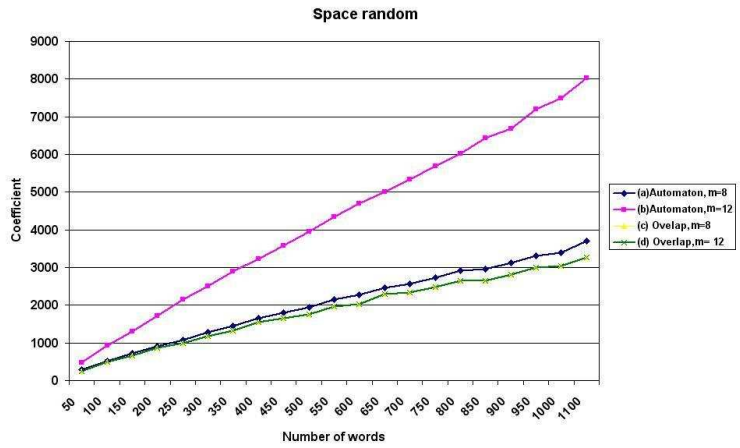

Fig. 9. Space coefficients for the random patterns of lengths 8 and 12. The graphs for overlap coefficients of lengths 8 and 12 (the lowest lines) almost coincide

The series of computer experiments allowed one to learn and compare the advantages of the approaches. For the overlap approach, the important characteristics of the pattern is number $|S|$ of its suffix-prefixes; the characteristics is also important for various pattern matching methods. The experiments demonstrate that for random patterns the value $|S|$ depend only of patterns size (i.e. number of words) but does not depend on pattern length (see Table 2 and Figure 4). The following Theorem justifies this observation.

Theorem 3. *Let V be an alphabet and $U(r;m)$ be a set of all sets (*patterns*) of r words with the same length m over an alphabet V; all elements of $U(r;m)$ are considered as having the same probability. Let further $E(r,m)$ be the average number of suffix-prefixes (*overlaps*) of the patterns from $U(r;m)$. Then*

$$E(r,m) \leq C \cdot r \tag{36}$$

where C does not depend on r and m.

The proof of the theorem is not given here due to lack of space and will be presented in a companion paper.

The above feature predetermines that the proposed approach exceeds the automaton one in case of random patterns (Figures 8 and 9). For non-random patterns, the automaton approach due to general minimization procedure get benefits from the structure of pattern and in these cases the advantage of the overlap approach is not so clear; moreover, in some cases the automaton approach works better. It seems to be interesting to provide deeper investigation of the dependence of the methods behavior on the structure of patterns and to develop a new method combining the advantages of the approaches.

7 Conclusion

In this paper, we introduced a new concept of overlap graphs to count word occurrences and their probabilities. The concept led to a recursive equation that differs from the classical one based on the finite automaton accepting the proper set of texts. In case of many occurrences, our approach yields the same order of time and space complexity as the approach based on minimized automaton.

For random patterns, our approach has asymptotically better constants and this advantage is achieved for relatively small pattern length, e.g. for lengths around 10 (see Figures 8 and 9). For non-random patterns, our approach gives results compatible with S-Patt [Nue08]; the results depend on the structure of the pattern. Because of this, we here restricted ourselves with comparison of theoretical constants rather than run-times. The description of the algorithm in Sections 4 and 5 justifies this way of algorithms comparison. Remark that the constants well describe the space complexity of the algorithms. In the same time, our results are significantly better than the results of methods that do not implement the automaton minimization, e.g. [NSF02,CS03,HZGD05,Nue06,BCR$^+$07].

In the future, we plan to design a novel recursion equation combining advantages of overlap graphs and minimal automata. Other directions of improvement are extensions of the scheme allowing to count differently various patterns and implement probability distributions described with Hidden Markov Models.

Acknowledgments: The authors are very grateful to anonymous referees whose remarks greatly helped to improve this paper.

E. Furletova and M. Roytberg were supported by grants 08-01-92496-NCNIL-a and 09-04-01053- from RFBR (Russia). M. Régnier and M. Roytberg were supported by Intas grant 05-1000008-8028 and Migec-Inria associate team.

References

[AC75] A.V. Aho and M. Corasick. Efficient String Matching. *CACM*, 18(6):333–340, 1975.

[BCR+07] V. Boeva, J. Clément, M. Régnier, M. Roytberg, and V. Makeev. Exact p-value calculation for heterotypic clusters of regulatory motifs and its application in computational annotation of cis-regulatory modules. *Algorithms for molecular biology*, 2(13):25 pages, 2007.

[BPL+04] B.P. Berman, B.D. Pfeiffer, T.R. Laverty, S.L. Salzberg, G.M. Rubin, M.B. Eisen, and S.E. Celniker. Computational identification of developmental enhancers: conservation and function of transcription factor binding-site clusters in drosophila melanogaster and drosophila pseudoobscura. *Genome Biol.*, 5(9), 2004. R61.

[CFG+05] K. Cartharius, K. Frech, K. Grote, B. Klocke, M. Haltmeier, A. Klingenhoff, M. Frisch, M. Bayerlein, and T. Werner. MatInspector and beyond: promoter analysis based on transcription factor binding sites. *Bioinformatics*, pages 2933–2942, 2005.

[CP90] C. Chrysaphinou and S. Papastavridis. The occurrence of sequence of patterns in repeated dependent experiments. *Theory of Probability and Applications*, 79:167–173, 1990.

[CR02] M. Crochemore and W. Rytter. *Jewels in Stringology.* World Scientific Publishing, Hong Kong, 2002.

[CS03] M. Crochemore and V. Stefanov. Waiting time and complexity for matching patterns with automata. *Information Processing Letters*, 87(3):119–125, 2003.

[DD07] M.K. Das and H-K Dai. A survey of DNA motif finding algorithms. *BMC Bioinformatics*, 8(52):247, 2007.

[FL03] J.C. Fu and W. Lou. *Distribution theory of runs and patterns and its applications. A finite Markov chain imbedding approach.* World Scientific, Singapore, 2003. 162p., ISBN 981-02-4587-4.

[GK97] M.S. Gelfand and E.V. Koonin. Avoidance of Palindromic Words in Bacterial and Archaeal Genomes: a Close Connection with Restriction Enzymess. *Nucleic Acids Research*, 25(12):2430–2439, 1997.

[GKM00] M. Gelfand, E. Koonin, and A. Mironov. Prediction of Transcription Regulatory Sites in A*rchaea* by a Comparative Genome Approach. *Nucleic Acids Research*, 28:695–705, 2000.

[GO81] L. Guibas and A.M. Odlyzko. String Overlaps, Pattern Matching and Nontransitive Games. *Journal of Combinatorial Theory,* Series A, 30:183–208, 1981.

[HZGD05] L. Hertzberg, O. Zuk, G. Getz, and E. Domany. Finding motifs in promoter regions. *Journal of Computational Biology*, 12(3):314–330, 2005.

[Lat04] D. S. Latchman. *Eukaryotic Transcription Factors.* Elsevier Academic Press, New York, 2004. 4-th edition, 360 pp., ISBN 0-12-437178-7.

[Lot05] Lothaire. *Applied Combinatorics on Words.* Cambridge University Press, Reading, Mass., 2005.

[NSF02] Pierre Nicodème, Bruno Salvy, and Philippe Flajolet. Motif statistics. *Theoretical Computer Science*, 287(2):593–618, 2002. preliminary version at ESA'99.

[Nue06] G. Nuel. Effective p-value computations using finite markov chain imbedding (fmci): application to local score and to pattern statistics. *Algorithms for molecular biology*, 1(5):14 pages, 2006.

[Nue08] G. Nuel. Pattern markov chains: optimal markov chain embedding through deterministic finite automata. *J. Appl. Prob.*, 45:226–243, 2008.

[PMG00] E. Panina, A. Mironov, and M. Gelfand. Statistical analysis of complete bacterial genomes:Avoidance of palindromes and restriction-modification systems. *Mol. Biol.*, 34:215–221, 2000.

[Rég00] M. Régnier. A Unified Approach to Word Occurrences Probabilities. *Discrete Applied Mathematics*, 104(1):259–280, 2000. Special issue on Computational Biology;preliminary version at RECOMB'98.

[RR08] P. Ribeca and E. Raineri. Faster exact Markovian probability functions for motif occurrences: a DFA-only approach. *Bioinformatics*, 24(24):2839–2848, 2008.

[RS97] M. Régnier and W. Szpankowski. On Pattern Frequency Occurrences in a Markovian Sequence. *Algorithmica*, 22(4):631–649, 1997. preliminary draft at ISIT'97.

[Sto00] G. D. Stormo. DNA binding sites: representation and discovery. *Bioinformatics*, 16:16–23, 2000.

[Szp01] W. Szpankowski. *Average Case Analysis of Algorithms on Sequences*. John Wiley and Sons, New York, 2001.

[TLB+05] M. Tompa, N. Li, T.L. Bailey, G.M. Church, B. De Moor, E. Eskin, A.V. Favorov, M.C. Frith, Y. Fu, J.W. Kent, V.J. Makeev, A.A. Mironov, W.S. Noble, G. Pavesi, G. Pesole, M. Régnier, N. Simonis, S. Sinha, G. Thijs, J. van Helden, M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, and Z. Zhu. An assessment of computational tools for the discovery of transcription factor binding sites. *Nature Biotechnology*, 23(1):137 – 144, January 2005.

[TML+02] G. Thijs, K. Marchal, M. Lescot, S. Rombauts, B. De Moor, P. Rouze, and Y. Moreau. A Gibbs sampling method to detect overrepresented motifs in the upstream regions of coexpressed genes. *Journal of Computational Biology*, 9:447–464, 2002.

[TV07] H. Touzet and J.-S. Varré. Efficient and accurate p-value computation for position weight matrices. *Algorithms for Molecular Biology*, 15(2), 2007. 12 pages.

[VM03] M. Vandenbogaert and V. Makeev. Analysis of Bacterial RM-systems through Genome-scale Analysis and Related Taxonomic Issues. *In Silico Biology*, 3:12, 2003. preliminary version at BGRS'02.

	100		200		500		1000	
m	$\|\Sigma\|$	$\|S\|$	$\|\Sigma\|$	$\|S\|$	$\|\Sigma\|$	$\|S\|$	$\|\Sigma\|$	$\|S\|$
8	510	96	929	199	1960	515	3427	988
12	906	98	1713	192	3976	518	7358	1006
16	1302	107	2533	207	5956	503	11368	961
20	1698	95	3307	190	7958	516	15415	987
24	2105	92	4114	201	9983	512	19442	991

Table 2. Average number of overlaps and size of minimized Aho-Corasick automaton for random patterns of different lengths (from 8 to 24) and sizes (from 100 to 1000 words). The number of overlaps does not depend on the pattern's length while the size of the automaton grows linearly with the length of the pattern

	Automaton approach			Overlap approach			
$\|\mathcal{H}\|$	N_{AC}	$\|\Sigma\|$	Time	$\|\mathcal{P}(\mathcal{H})\|$	$\|S\|$	Time	Space
4	19	10	40	4	4	8	18
102	305	86	344	42	11	53	55
208	550	134	536	94	19	113	90
293	733	152	608	128	24	152	112
409	971	209	836	204	31	235	138
502	1159	218	872	244	34	278	149
600	1351	213	852	289	36	325	153
695	1528	251	1004	419	55	474	241
802	1740	322	1288	501	65	566	273
915	1940	327	1308	576	73	649	301
1009	2130	339	1356	707	85	792	344
1094	2288	377	1508	793	94	887	381
1200	2499	381	1524	883	97	980	392
1300	2689	445	1780	978	106	1084	409
1395	2860	484	1936	1089	113	1202	430
1498	3039	513	2052	1164	122	1286	450

Table 3. Characteristics of patterns related to PSSM-08 containing different numbers $|\mathcal{H}|$ of words. For the automaton approach, N_{AC} is the size of Aho-Corasick automaton; Σ is the size of minimal automaton that represents the coefficient for space complexity; coefficient for time complexity $Time$ equals $|\Sigma| \times |V|$. For the overlap approach we give the number of overlap classes, $\mathcal{P}(\mathcal{H})$; the number of overlaps S; time coefficient that is equal to $|S| + |\mathcal{P}(\mathcal{H})|$ and space coefficient $Space$ from (35)

	Automaton approach			Overlap approach			
$\|\mathcal{H}\|$	N_{AC}	$\|\Sigma\|$	Time	$\|\mathcal{P}(\mathcal{H})\|$	$\|S\|$	Time	Space
48	243	57	228	14	6	20	18
104	416	103	412	24	8	32	55
160	512	108	432	74	14	88	90
232	640	119	476	125	16	141	112
296	736	119	476	144	16	160	138
352	840	129	516	198	20	218	149
400	928	131	524	213	21	234	153
464	1016	133	532	237	21	258	241
528	1120	145	580	304	23	327	273
592	1224	148	592	312	23	335	301
664	1469	207	828	446	44	490	344
728	1533	205	820	488	44	532	381
768	1589	201	804	496	45	541	382
816	1725	234	936	563	50	613	383
880	1861	246	984	611	50	661	384
952	1973	219	876	638	51	689	385
984	2005	214	856	662	51	713	386
1184	2997	344	1376	754	58	812	392
1240	3077	345	1380	762	58	820	409
1328	3237	404	1616	792	60	852	430
1584	4213	561	2244	1111	83	1194	450

Table 4. Characteristics of patterns related to PSSM-12 containing different numbers of words.

	Automaton approach			Overlap approach			
$\|\mathcal{H}\|$	N_{AC}	$\|\Sigma\|$	Time	$\|\mathcal{P}(\mathcal{H})\|$	$\|S\|$	Time	Space
50	299	284	1136	50	51	101	256
100	549	512	2048	100	108	208	501
150	783	729	2916	149	150	299	660
200	1003	919	3676	200	208	408	868
250	1198	1085	4340	250	243	493	985
300	1410	1290	5160	300	301	601	1175
350	1592	1448	5792	350	341	691	1319
400	1814	1658	6632	400	417	817	1561
450	1993	1808	7232	450	454	904	1648
500	2156	1941	7764	500	488	988	1766
550	2362	2161	8644	550	561	1111	1973
600	2528	2276	9104	600	584	1184	2034
650	2722	2469	9876	650	674	1324	2291
700	2855	2564	10256	700	697	1397	2346
750	3050	2738	10952	750	746	1496	2483
800	3217	2914	11656	800	815	1615	2656
850	3336	2955	11820	850	799	1649	2641
900	3519	3130	12520	900	879	1779	2807
950	3693	3303	13212	950	947	1897	3002
1000	3823	3398	13592	1000	976	1976	3044
1100	4156	3699	14796	1100	1059	2159	3262
1200	4452	3970	15880	1200	1195	2395	3564
1300	4750	4202	16808	1300	1231	2531	3693
1400	5063	4502	18008	1400	1388	2788	4027
1500	5356	4790	19160	1500	1509	3009	4284

Table 5. Characteristics of random patterns of length 8 containing different numbers of words.

	Automaton approach			Overlap approach			
$\|\mathcal{H}\|$	N_{AC}	$\|\Sigma\|$	Time	$\|\mathcal{P}(\mathcal{H})\|$	$\|S\|$	Time	Space
50	496	476	1904	50	51	101	256
100	959	921	3684	99	95	194	501
150	1367	1311	5244	150	153	303	660
200	1801	1722	6888	199	200	399	868
250	2224	2142	8568	250	265	515	985
300	2613	2494	9976	300	288	588	1175
350	3007	2888	11552	349	370	719	1319
400	3396	3232	12928	400	395	795	1561
450	3756	3579	14316	450	452	902	1648
500	4170	3959	15836	500	489	989	1766
550	4552	4335	17340	550	567	1117	1973
600	4938	4694	18776	600	609	1209	2034
650	5294	5013	20052	650	627	1277	2291
700	5650	5339	21356	699	677	1376	2346
750	5985	5686	22744	750	736	1486	2483
800	6368	6022	24088	800	803	1603	2656
850	6776	6431	25724	850	887	1737	2641
900	7061	6675	26700	900	908	1808	2807
950	7514	7194	28776	950	994	1944	3002
1000	7854	7495	29980	1000	1021	2021	3044
1100	8499	8019	32076	1099	1060	2159	3262

Table 6. Characteristics of random patterns of length 12 containing different numbers of words.