Minimal Absent Words in Rooted and Unrooted Trees

Gabriele Fici[1(✉)] and Paweł Gawrychowski[2]

[1] Dipartimento di Matematica e Informatica, Università di Palermo, Palermo, Italy
gabriele.fici@unipa.it
[2] Institute of Computer Science, University of Wrocław, Wrocław, Poland
gawry@cs.uni.wroc.pl

Abstract. We extend the theory of minimal absent words to (rooted and unrooted) trees, having edges labeled by letters from an alphabet Σ of cardinality σ. We show that the set $\mathrm{MAW}(T)$ of minimal absent words of a rooted (resp. unrooted) tree T with n nodes has cardinality $O(n\sigma)$ (resp. $O(n^2\sigma)$), and we show that these bounds are realized. Then, we exhibit algorithms to compute all minimal absent words in a rooted (resp. unrooted) tree in output-sensitive time $O(n + |\mathrm{MAW}(T)|)$ (resp. $O(n^2 + |\mathrm{MAW}(T)|)$ assuming an integer alphabet of size polynomial in n.

1 Introduction

Minimal absent words (a.k.a. minimal forbidden words or minimal forbidden factors) are a useful combinatorial tool for investigating words (strings). A word u is absent from a word w if u does not occur (as a factor) in w, and it is minimal if all its proper factors occur in w. This definition naturally extends to languages of words closed under taking factors.

The theory of minimal absent words has been developed in a series of papers [3,5,14,25,27] (the reader is pointed to [18] for a survey on these results). Minimal absent words have then found applications in several areas, e.g., data compression [15–17,28], on-line pattern matching [13], sequence comparison [10,11], sequence assembly [20,26], bioinformatics [9,19,31], musical data extraction [12].

Bounds on the number of minimal absent words have been extensively investigated. The upper bound on the number of minimal absent words of a word of length n over an alphabet of size σ is $O(n\sigma)$ [14,27], and this is tight for integer alphabets [10]; in fact, for large alphabets, such as when $\sigma \geq \sqrt{n}$, this bound is also tight even for minimal absent words having the same length [1].

Several algorithms are known to compute the set of minimal absent words of a word. State-of-the-art algorithms compute all minimal absent words of a word of length n over an alphabet of size σ in time $O(n\sigma)$ [2,14] or in output-sensitive $O(n + |\mathrm{MAW}(w)|)$ time [11,22] for integer alphabets. Space-efficient data structures based on the Burrows-Wheeler transform can also be applied for this computation [6,7].

© Springer Nature Switzerland AG 2019
N. R. Brisaboa and S. J. Puglisi (Eds.): SPIRE 2019, LNCS 11811, pp. 152–161, 2019.
https://doi.org/10.1007/978-3-030-32686-9_11

For a finite set of words P over an alphabet of size σ, the minimal absent words of the factorial closure of P can be computed in $O(|P|^2\sigma)$ [3], where $|P|$ is the sum of the lengths of the words of P. Generalizations of minimal absent words have been considered for circular words [10,21] and multi-dimensional shifts [4].

In this paper, we extend the theory of minimal absent words to trees. We consider trees with edges labeled by letters from an integer alphabet Σ of cardinality σ polynomial in n. In the case of a rooted tree T, every node v is associated with a word $\mathsf{str}(v)$, defined as the sequence of edge labels from v to the root. A rooted tree T can therefore be seen as a set of words $L_{\mathsf{T}} = \{\mathsf{str}(v) \mid v \text{ in } \mathsf{T}\}$, that we call the *language* of T. If T has n nodes, then L_{T} contains at most n distinct words, each of which has length at most n. We call a rooted tree T *proper* when the edges from a node to its children are labeled by pairwise distinct letters. Throughout the paper we will assume that all rooted trees are proper, which can be ensured without losing the generality thank to the following lemma.

Lemma 1. *Given a rooted T on n nodes we can construct in $O(n)$ time a proper rooted tree T' with the same set of corresponding words.*

Proof. The depth of a node of T is its distance from the root. We start with sorting, for every $d = 1, 2, ..$, the set of nodes $S(d)$ at depth d according to the labels of the edges leading to their parents. This can be done in $O(n)$ total time with counting sort. Then, we construct T' by processing $S(0), S(1), S(2), ...$ Assuming that we have already identified, for every node $u \in S(d)$, its corresponding node $f(u)$ of T', we need to construct and identify the nodes $f(u')$ for every $u' \in S(d+1)$. We process all nodes $u' \in S(d+1)$ in groups corresponding to the same letter a on the edge leading to their parent (because of the initial sorting we already have these groups available). Denoting by u the parent of u' in T, we check if $f(u)$ has been already accessed while processing the group of a, and if so we set $f(u')$ to be the already created node of T'. Otherwise, we create a new edge outgoing from $f(u)$ to a new node v in T' and labeled with a, and set $f(u')$ to be v. To check if $f(u)$ has been already accessed while processing the current group (and retrieve the corresponding $f(u')$ if this is the case) we simply allocate an array A of size n indexed by nodes of T' identified by number from $\{1, 2, \ldots, n\}$. For every entry of A we additionally store a timestamp denoting the most recent group for which the corresponding entry has been modified, and increase the timestamp after having processed the current group. □

One could also define the set of words corresponding to a rooted tree T by considering a set of words from the root to every node v (in the literature this is sometimes called a *forward trie*, as opposed to a *backward trie*, cf. [24]). In our context, this distinction is meaningless, as the obtained languages are the same up to reversing all the words.

We say that a word aub, with $a, b \in \Sigma$, is a minimal absent word of a rooted tree T if aub is not a factor of any word $\mathsf{str}(v)$ in L_{T} but there exist words $\mathsf{str}(v_1)$ and $\mathsf{str}(v_2)$ in L_{T} (not necessarily distinct) such that au is a factor of $\mathsf{str}(v_1)$ and ub is a factor of $\mathsf{str}(v_2)$. That is, the set $\mathrm{MAW}(\mathsf{T})$ of minimal absent words of T

is the set of minimal absent words of the factorial closure of the language L_{T}. Since any word of length n can be transformed into a unary rooted tree with $n+1$ nodes, some of the properties of minimal absent words for usual words can be transferred to rooted trees. Indeed, rooted trees are a strict generalization of words.

For unrooted trees, the definition of minimal absent words is analogous: We identify an unrooted tree T with the language of words $L(\mathsf{T})$ corresponding to all (concatenations of labels of) simple paths that can be read in T from any of its nodes. The language $L(\mathsf{T})$ contains $O(n^2)$ words, each of which has length at most n. We therefore define the set $\mathrm{MAW}(\mathsf{T})$ of minimal absent words of T as the set of minimal absent words of the language $L(\mathsf{T})$, which in this case is already closed under taking factors by definition.

Our Results. We prove that for any rooted tree with n nodes there are $O(n\sigma)$ minimal absent words, and we show that this bound is tight. For unrooted trees, we prove that the previous bound becomes $O(n^2\sigma)$, and we give an explicit construction that achieves this bound. We also consider the case of minimal absent words of fixed length and generalize a previously-known construction.

Furthermore, we present an algorithm that computes all the minimal absent words in a rooted tree T with n nodes in output-sensitive time $O(n+|\mathrm{MAW}(\mathsf{T})|)$. This also yields an algorithm that computes all the minimal absent words in an unrooted tree T with n nodes in time $O(n^2 + |\mathrm{MAW}(\mathsf{T})|)$. Note that while it is plausible that an efficient algorithm could have been designed, as in the case of words, from a DAWG [22], the size of the DAWG of a backward/forward tree is superlinear [24], so it is not immediately clear if such an approach would lead to an optimal algorithm. Excluding the space necessary to store all the results, our algorithms need $O(n)$ and $O(n^2)$ space, respectively.

Our algorithms are designed in the word-RAM model with $\Omega(\log n)$-bit words.

2 Bounds on the Number of Minimal Absent Words

Let T be a rooted tree with n nodes and edges labeled by letters from an integer alphabet Σ of cardinality σ polynomial in n. Let the language of T be $L_{\mathsf{T}} = \{\mathsf{str}(v) \mid v \text{ in } \mathsf{T}\}$, where $\mathsf{str}(v)$ is the sequence of edge labels from node v to the root.

For convenience, we add a new root to T and an edge labeled by a new letter \$ not belonging to Σ from the new root to the old root. This corresponds to appending \$ at the end of each word of L_{T}. We then arrange all the words of L_{T} in a trie. Each node u of this trie corresponds to a word obtained by concatenating the edges from the root of the trie to node u, so in this paper we will implicitly identify a node of the trie with the corresponding word in the set of prefixes of L_{T}. Following a standard approach, if we compact this trie by collapsing maximal chains of edges with every inner node having exactly one child and edges labeled by words, we obtain the suffix tree ST of T. The nodes in ST (the branching nodes) are called explicit nodes, while the nodes of the trie

that have been collapsed (the non-branching nodes) are called implicit. Because \$ does not belong to the original alphabet, the leaves of ST are in one-to-one correspondence with the nodes of T.

A word aub, with $a, b \in \Sigma$, is a minimal absent word for T if it is a minimal absent word for the factorial closure of L_{T}, that is, if both au and ub but not aub are factors of some words in L_{T}. The set of minimal absent words of T is denoted by MAW(T).

If $aub \in$ MAW(T), then au occurs as a factor in some word of L_{T} but never followed by letter b, hence there exists a letter $b' \in \Sigma \cup \{\$\}$ such that ub and ub' can be read in ST spelled from the root (possibly terminating in an implicit node). This implies that u corresponds to an explicit node in ST, and b is the first letter on its outgoing edge. Consequently, ub can be identified with an edge of ST, so the number of minimal absent words of T is upper-bounded by the product of σ and the number of edges of ST.

Theorem 1. *The number of minimal absent words of a rooted tree with n nodes whose edges are labeled by letters from an alphabet of size σ is $O(n\sigma)$.*

Therefore, the same upper bound that holds for words also holds for rooted trees. As a consequence, we have that all known upper bounds for words, and constructions that realize them, are still valid for rooted trees.

In particular, one question that has been studied is whether the upper bound $O(n\sigma)$ is still tight when one considers minimal absent words of a fixed length. Almirantis et al. [1, Lemma 2] showed that the upper bound $O(n\sigma)$ for a fixed length of minimal absent words is tight if $\sqrt{n} < \sigma \leq n$. Actually, they showed that it is possible to construct words of any length n, with $\sigma \leq n \leq \sigma(\sigma - 1)$, having $\Omega(n\sigma)$ minimal absent words of length 3. We now give a construction that generalizes this result.

Let $\Sigma = \{1, 2, \ldots, \sigma\}$. For every n, let $k > 1$ be such that $\sigma^k \leq n < \sigma^{k+1}$. Let $\Sigma^k = \{s_1, s_2, \ldots, s_{\sigma^k}\}$. For every $1 \leq i \leq \sigma^k$ we define the word

$$w_i = \$1s_i\$s_i1\$2s_i\$s_i2\$ \cdots \$\sigma s_i\$s_i\sigma\$,$$

where \$ is a new symbol not belonging to Σ. The length of each word w_i is $2\sigma(k+2)+1$, which is smaller than n up to excluding small cases [1].

Let $\ell = \lfloor n/|w_i| \rfloor$ and set $w = w_1 w_2 \cdots w_\ell$, so that $|w| > n/2$. We have that as_ib is a minimal absent word of w for every $a, b \in \Sigma$ and $1 \leq i \leq \ell$. So, w has length $\Theta(k\sigma\ell)$ and there are $\Theta(\sigma^2\ell)$ minimal absent words of w of length $k+2$.

Thus, we have proved the following theorem.

Theorem 2. *A word of length n over an alphabet of size σ can have $\Omega(n\sigma/\log_\sigma n)$ minimal absent words all of the same length.*

Observe that for $\sqrt{n} < \sigma \leq n$, $\log_\sigma n = \Theta(1)$, therefore Theorem 2 strictly generalizes Almirantis et al.'s result.

[1] The reader may verify that for $k = 2$, $|w_i| \leq \sigma^k$ as soon as $\sigma \geq 9$; for $k > 2$, $|w_i| \leq \sigma^k$ as soon as $\sigma + k \geq 7$.

Let now T be an unrooted tree. Then the number of distinct simple paths in T is $O(n^2\sigma)$, and this is thus an upper bound on the number of minimal absent words of T.

Theorem 3. *The number of minimal absent words of an unrooted tree with n nodes whose edges are labeled by letters from an alphabet of size σ is $O(n^2\sigma)$.*

We now provide an example of an unrooted tree realizing this bound. Let $\Sigma = \{0, \bar{0}, 1, \ldots, s\}$. Our unrooted tree T is built as follows:

- We first build a sequence of $2N+1$ nodes such that every other node is connected to s terminal nodes with edges labeled by $1, 2, \ldots, s$ and is connected to the next node of the sequence with an edge labeled by 0 and to the previous node of the sequence with an edge labeled by $\bar{0}$;
- Then, we attach to each of the last nodes of the previous sequence s simple paths composed of $2N$ nodes with edges labeled by alternating 0 and $\bar{0}$.

See Fig. 1 for an illustration.

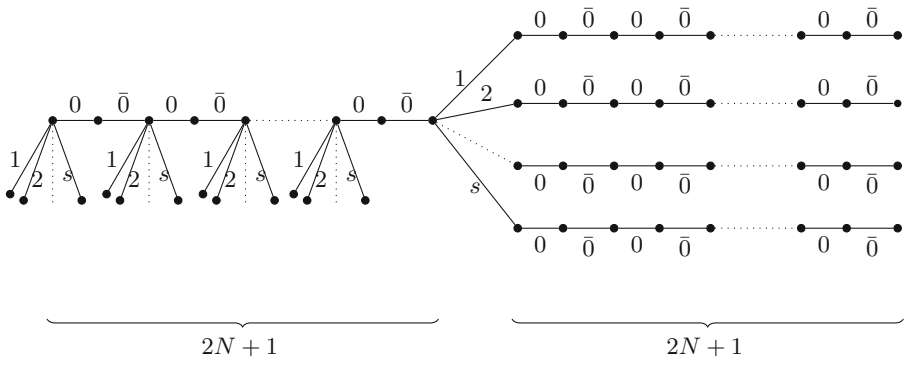

Fig. 1. An unrooted tree realizing the upper bound on the number of minimal absent words.

In total, T has $(s+1)(2N+1)+sN$ nodes. We therefore set $n = (s+1)(2N+1)+sN$, so that $n = \Theta(sN)$.

It is readily verified that for every a, b, c in $\Sigma \setminus \{0, \bar{0}\}$ and for every $0 < j, k \leq N$, there is a minimal absent word of the form $a(0\bar{0})^j b(0\bar{0})^k c$ (the prefix $a(0\bar{0})^j b(0\bar{0})^k$ can be found reading from the left part to the right part of the figure, while the suffix $(0\bar{0})^j b(0\bar{0})^k c$ can be found reading from the right part to the left part, the letter b being one of the labels of the edges joining the left and the right part). Hence, the number of minimal absent words of T is $\Omega(s^3N^2) = \Omega(n^2\sigma)$.

Remark 1. The previous construction can be simplified by merging adjacent edges labeled by 0 and $\bar{0}$ into one edge labeled by 0, if one does not require the condition that edges adjacent to a node must have distinct labels.

3 Algorithms for Computing Minimal Absent Words

We now present an algorithm that computes the set MAW(T) of all minimal absent words of a rooted tree T with n nodes in output-sensitive time $O(n + |\text{MAW}(\mathsf{T})|)$.

We construct the suffix tree ST of T in time $O(n)$ [30]. Recall that the leaves of ST are in one-to-one correspondence with the nodes of T and we can assume that every node u of T stores a pointer to the leaf of ST corresponding to $\mathsf{str}(u)$.

Definition 1. *For every (implicit or explicit) node u of ST, we define the set $A(u)$ as the set of all letters $a \in \Sigma$ such that au can be spelled from the root of ST, i.e., there exists a node v of T such that $\mathsf{str}(v) = auz$ for some (possibly empty) word z.*

As already noted before, if aub is a minimal absent word of T, then au occurs as a factor in some word of L_{T} followed by a letter $b' \in \Sigma \cup \{\$\}$ different from b, hence u is an explicit node of ST.

Lemma 2. *Let u be an explicit node of the suffix tree ST of the tree T. Let $u_1, u_2, \ldots, u_k$ be the children of u in the non-compacted trie from which we obtained ST, and let $b_1, b_2, \ldots, b_k$ be the labels of the corresponding edges. Then, for every $1 \leq i \leq k$ and every letter*

$$a_j \in (A(u_1) \cup \ldots \cup A(u_k)) \setminus A(u_i),$$

the word $a_j u b_i$ is a minimal absent word of T.

Conversely, every minimal absent word of T is of the form $a_j u b_i$ described above.

Proof. Since a_j does not belong to $A(u_i)$, then by definition the word $a_j u b_i$ does not belong to L_{T}, but there exists $\ell \neq i$ such that $a_j \in A(u_\ell)$, that is, $a_j u b_\ell$ is a factor of a word in L_{T}. Hence, $a_j u$ is a factor of a word in L_{T}. Since $u b_i$ is also a factor of a word in L_{T} by construction, we have that $a_j u b_i$ is a minimal absent word of T.

Conversely, if $a_j u b_i$ is a minimal absent word of T, then u occurs as a factor in some word of L_{T} followed by different letters in $\Sigma \cup \{\$\}$, hence it corresponds to an explicit node in ST, so all minimal absent words of T are found in this way. □

Definition 2. *For every leaf u of ST we define the set $B(u)$ as the set of all letters $a \in \Sigma$ such that $au = \mathsf{str}(v)$ for some node v in T.*

Lemma 3. *For every (implicit or explicit) node u of ST, we have $A(u) = \bigcup\{B(u') \mid u' \text{ is a leaf in the subtree of } ST \text{ rooted at } u\}$.*

Proof. Let u' be a leaf in the subtree of ST rooted at u. Thus, the word u is a prefix of the word u', i.e., $u' = uz$ for some word z. By definition, $B(u')$ is the set of all letters $a \in \Sigma$ such that $au' = \mathsf{str}(v')$ for some node v' in T. That is,

the set of all letters $a \in \Sigma$ such that $au' = auz$ is a word in L_{T}. On the other hand, by definition, $A(u)$ is the set of all letters $a \in \Sigma$ such that $\mathsf{str}(v) = auz$ for some node v of T and word z. That is, the set of all letters $a \in \Sigma$ such that auz is a word in L_{T} for some word z. □

We now show how to compute, in time proportional to the output size, the set MAW(T).

We start with creating, for every letter $a \in \Sigma$, a list $L(a)$ of all leaves u such that $a \in B(u)$ sorted in preorder. The lists can be obtained in linear time by traversing all the non-root nodes $v \in \mathsf{T}$, following the edge labeled by a from v to its parent v', and finally following the pointer from v' to the leaf v'' of ST corresponding to $\mathsf{str}(v')$ and adding v'' to $L(a)$. Finally, because the preorder numbers are from $[n]$ the lists can be sorted in linear time with counting sort.

Now we iterate over all letters $a \in \Sigma$. Due to Lemma 2, the goal is to extract all explicit nodes $u \in ST$ such that, for some child u_i of u such that the b_i is the first letter on the edge from u to u_i, aub_i is a minimal absent word. By Lemma 3, this is equivalent to u having a descendant $u' \in L(a)$ (where possibly $u = u'$) and u_i not having any such descendant. This suggests that we should work with the subtree of ST, denoted $ST(a)$, induced by all leaves $v \in L(a)$. Formally, $u \in ST(a)$ when $u' \in L(a)$ for some leaf u' in the subtree of u. Even though ST does not contain nodes with just one child, this is no longer the case for $ST(a)$. Thus, we actually work with its compact version, denoted $ST(a)$. Every node of $ST(a)$ stores a pointer to its corresponding node of ST. Assuming that ST has been preprocessed for constant-time Lowest Common Ancestor queries (which can be done in linear time and space [8,29]), we can construct $ST(a)$ efficiently due to the following lemma.

Lemma 4. *Given $L(a)$, we can construct $ST(a)$ in $O(|L(a)|)$ time.*

Proof. The procedure follows the general idea used in the well-known linear time procedure for creating a Cartesian tree [23]. We process the nodes $u \in L(a)$ in preorder and maintain a compact version of the subtree of ST induced by all the already-processed nodes. Additionally, we maintain a stack storing the edges on its rightmost path. Processing $u \in L(a)$ requires popping a number of edges from the stack, possibly splitting the topmost edge into two (with one immediately popped as well), and pushing a new edge ending at u. Checking if an edge should be popped, and also determining if (and how) an edge should be split, can be implemented with LCA queries on ST, assuming that we maintain pointers to the corresponding nodes of ST. □

Having constructed $ST(a)$, we need to consider two cases corresponding to u being an explicit or an implicit node of $ST(a)$. In the former case, we need to extract the edges outgoing from u in ST such that there is no edge outgoing from the corresponding node in $ST(a)$ starting with the same letter b, and output aub as a minimal absent word. Assuming that the outgoing edges are sorted by their first letters, this can be easily done in time proportional to the degree of u plus the number of extracted letters. In the latter case, let the implicit node belong

to an edge connecting u to v in $ST(a)$, and let u' and v' be their corresponding nodes in ST with u' being an ancestor of v'. We iterate through all explicit nodes between u' and v' in ST and for each such node we extract all of its outgoing edges. For each such edge we check if v' belongs to the subtree rooted at its endpoint other than u, and if not, extract its first letter b to output aub as a minimal absent word.

The overall time for every letter $a \in \Sigma$ can be bounded by the sum of the size of $ST(a)$ and the number of generated minimal absent words. Because $\sum_{a\in\Sigma} |L(a)| = O(n)$ and the size of $ST(a)$ can be bounded by $O(|L(a)|)$, the total time complexity is $O(n + |\mathrm{MAW}(\mathsf{T})|)$.

The previous algorithm can be used to design an algorithm that outputs all the minimal absent words of an unrooted tree T with n nodes in time $O(n^2 + |\mathrm{MAW}(\mathsf{T})|)$ as follows. For every node u of T, we create a rooted tree T_u by fixing u as the root. Then we merge all trees T_u into a single tree T of size $O(n^2)$ by identifying their roots. Finally, we apply Lemma 1 to make T proper and apply our algorithm for rooted trees in $O(n^2)$ total time.

Acknowledgments. This research was carried out during a visit of the first author to the Institute of Computer Science of the University of Wrocław, supported by grant CORI-2018-D-D11-010133 of the University of Palermo. The first author is also supported by MIUR project PRIN 2017K7XPAN "Algorithms, Data Structures and Combinatorics for Machine Learning".

We thank anonymous reviewers for helpful comments.

References

1. Almirantis, Y., et al.: On avoided words, absent words, and their application to biological sequence analysis. Algorithms Mol. Biol. **12**(1), 51–512 (2017)
2. Barton, C., Héliou, A., Mouchard, L., Pissis, S.P.: Linear-time computation of minimal absent words using suffix array. BMC Bioinform. **15**, 388 (2014)
3. Béal, M., Crochemore, M., Mignosi, F., Restivo, A., Sciortino, M.: Computing forbidden words of regular languages. Fundam. Inform. **56**(1–2), 121–135 (2003)
4. Béal, M., Fiorenzi, F., Mignosi, F.: Minimal forbidden patterns of multi-dimensional shifts. IJAC **15**(1), 73–93 (2005)
5. Béal, M.-P., Mignosi, F., Restivo, A.: Minimal forbidden words and symbolic dynamics. In: Puech, C., Reischuk, R. (eds.) STACS 1996. LNCS, vol. 1046, pp. 555–566. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-60922-9_45
6. Belazzougui, D., Cunial, F.: A framework for space-efficient string kernels. Algorithmica **79**(3), 857–883 (2017)
7. Belazzougui, D., Cunial, F., Kärkkäinen, J., Mäkinen, V.: Versatile succinct representations of the bidirectional burrows-wheeler transform. In: Bodlaender, H.L., Italiano, G.F. (eds.) ESA 2013. LNCS, vol. 8125, pp. 133–144. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40450-4_12
8. Bender, M.A., Farach-Colton, M.: The LCA problem revisited. In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 88–94. Springer, Heidelberg (2000). https://doi.org/10.1007/10719839_9
9. Chairungsee, S., Crochemore, M.: Using minimal absent words to build phylogeny. Theoret. Comput. Sci. **450**, 109–116 (2012)

10. Charalampopoulos, P., Crochemore, M., Fici, G., Mercaş, R., Pissis, S.P.: Alignment-free sequence comparison using absent words. Inf. Comput. **262**(1), 57–68 (2018)
11. Charalampopoulos, P., Crochemore, M., Pissis, S.P.: On extended special factors of a word. In: Gagie, T., Moffat, A., Navarro, G., Cuadros-Vargas, E. (eds.) SPIRE 2018. LNCS, vol. 11147, pp. 131–138. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00479-8_11
12. Crawford, T., Badkobeh, G., Lewis, D.: Searching page-images of early music scanned with OMR: a scalable solution using minimal absent words. In: ISMIR, pp. 233–239 (2018)
13. Crochemore, M., Héliou, A., Kucherov, G., Mouchard, L., Pissis, S.P., Ramusat, Y.: Minimal absent words in a sliding window and applications to on-line pattern matching. In: Klasing, R., Zeitoun, M. (eds.) FCT 2017. LNCS, vol. 10472, pp. 164–176. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-55751-8_14
14. Crochemore, M., Mignosi, F., Restivo, A.: Automata and forbidden words. Inf. Process. Lett. **67**(3), 111–117 (1998)
15. Crochemore, M., Mignosi, F., Restivo, A., Salemi, S.: Data compression using antidictionaries. Proc. IEEE **88**(11), 1756–1768 (2000)
16. Crochemore, M., Navarro, G.: Improved antidictionary based compression. In: 22nd International Conference of the Chilean Computer Science Society (SCCC 2002), 6–8 November 2002, Copiapo, Chile, pp. 7–13 (2002)
17. Fiala, M., Holub, J.: DCA using suffix arrays. In: 2008 Data Compression Conference (DCC 2008), 25–27 March 2008, Snowbird, UT, USA, p. 516. IEEE Computer Society (2008)
18. Fici, G.: Minimal Forbidden Words and Applications. Ph.D. thesis, Université de Marne-la-Vallée (2006)
19. Fici, G., Langiu, A., Lo Bosco, G., Rizzo, R.: Bacteria classification using minimal absent words. AIMS Med. Sci. **5**(1), 23–32 (2018)
20. Fici, G., Mignosi, F., Restivo, A., Sciortino, M.: Word assembly through minimal forbidden words. Theoret. Comput. Sci. **359**(1), 214–230 (2006)
21. Fici, G., Restivo, A., Rizzo, L.: Minimal forbidden factors of circular words. Theoret. Comput. Sci., to appear
22. Fujishige, Y., Tsujimaru, Y., Inenaga, S., Bannai, H., Takeda, M.: Computing DAWGs and minimal absent words in linear time for integer alphabets. In: Faliszewski, P., Muscholl, A., Niedermeier, R. (eds.) 41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22–26, 2016 - Kraków, Poland. LIPIcs, vol. 58, pp. 38:1–38:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
23. Gabow, H.N., Bentley, J.L., Tarjan, R.E.: Scaling and related techniques for geometry problems. In: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing, STOC 1984, pp. 135–143. ACM, New York (1984)
24. Inenaga, S.: Suffix Trees, DAWGs and CDAWGs for Forward and Backward Tries. CoRR, abs/1904.04513 (2019)
25. Mignosi, F., Restivo, A., Sciortino, M.: Forbidden factors in finite and infinite words. In: Karhumäki, J., Maurer, H., Păun, G., Rozenberg, G. (eds.) Jewels are Forever, pp. 339–350. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-642-60207-8_30
26. Mignosi, F., Restivo, A., Sciortino, M.: Forbidden factors and fragment assembly. ITA **35**(6), 565–577 (2001)

27. Mignosi, F., Restivo, A., Sciortino, M.: Words and forbidden factors. Theor. Comput. Sci. **273**(1–2), 99–117 (2002)
28. Ota, T., Morita, H.: On the adaptive antidictionary code using minimal forbidden words with constant lengths. In: Proceedings of the International Symposium on Information Theory and its Applications, ISITA 2010, 17–20 October 2010, Taichung, Taiwan, pp. 72–77. IEEE (2010)
29. Schieber, B., Vishkin, U.: On finding lowest common ancestors: simplification and parallelization. SIAM J. Comput. **17**(6), 1253–1262 (1988)
30. Shibuya, T.: Constructing the suffix tree of a tree with a large alphabet. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E86–A**(5), 1061–1066 (2003)
31. Silva, R.M., Pratas, D., Castro, L., Pinho, A.J., Ferreira, P.J.S.G.: Three minimal sequences found in Ebola virus genomes and absent from human DNA. Bioinformatics **31**(15), 2421–2425 (2015)