Factor Automata of Automata and Applications

Mehryar Mohri[1,2], Pedro Moreno[2], and Eugene Weinstein[1,2]

[1] Courant Institute of Mathematical Sciences
251 Mercer Street, New York, NY 10012.
[2] Google Research
76 Ninth Avenue, New York, NY 10011.

Abstract. An efficient data structure for representing the full index of a set of strings is the *factor automaton*, the minimal deterministic automaton representing the set of all factors or substrings of these strings. This paper presents a novel analysis of the size of the *factor automaton of an automaton*, that is the minimal deterministic automaton accepting the set of factors of a finite set of strings, itself represented by a finite automaton. It shows that the factor automaton of a set of strings U has at most $2|Q| - 2$ states, where Q is the number of nodes of a prefix-tree representing the strings in U, a bound that significantly improves over $2\|U\|-1$, the bound given by Blumer et al. (1987), where $\|U\|$ is the sum of the lengths of all strings in U. It also gives novel and general bounds for the size of the factor automaton of an automaton as a function of the size of the original automaton and the maximal length of a suffix shared by the strings it accepts. Our analysis suggests that the use of factor automata of automata can be practical for large-scale applications, a fact that is further supported by the results of our experiments applying factor automata to a music identification task with more than 15,000 songs.

1 Introduction

Pattern matching in strings is a fundamental problem in computer science and has been extensively studied in the past [6, 5]. With the renewed interest in search within the massive amounts of natural language texts, biological sequences, and other widely accessible digitized sequences, this problem has gained further attention and significance.

This paper considers the problem of constructing a full index, or inverted file, for a large set of strings represented by a finite automaton. This problem arises in a number of different contexts, such as those where the set of strings is directly given as an automaton produced by an information extraction or speech recognition system, or when the set of strings is compactly stored as an automaton to reduce storage and increase the efficiency of their use [1, 9].

An efficient and compact data structure for representing a full index of a set of strings is a *factor automaton*, that is a minimal deterministic automaton representing the set of all factors of a set of strings. Since it is deterministic, the factor automaton can be used to determine if a string x is a factor in time linear in its length $O(|x|)$, which is optimal.

The construction and the size of a factor automaton have been specifically analyzed in the case of a single string [3, 4]. These authors demonstrated the remarkable result that the size of the factor automaton of a string x is linear, and that, more precisely, for strings x of length more than three, it has at most $2|x| - 2$ states and $3|x| - 4$ transitions. They also gave on-line linear-time algorithms for constructing a factor automaton from x. Similar results were given for the suffix automata, the minimal deterministic automata accepting exactly the set of suffixes of a string.

The construction and the size of the factor automata has also been previously studied in the case of a finite set of strings $U = \{x_1, \ldots, x_m\}$ [2]. These authors showed that an automaton accepting all factors of U can be constructed that has at most $2\|U\| - 1$ states and $3\|U\| - 3$ transitions, where $\|U\|$ is the sum of the lengths of all strings in U, that is $\|U\| = \sum_{i=1}^{m} |x_i|$.

This paper proves a significantly better bound on the size of the suffix automaton or factor automaton of a set of strings. It shows that the factor automaton of a set of strings U has at most $2|Q| - 2$ states, where Q is the number of nodes of a prefix-tree representation of the strings in U. The number of nodes $|Q|$ can be dramatically smaller than $\|U\|$, the sum of the lengths of all strings. Thus, our space bound clearly improves on previous work [2]. Also, although we are not demonstrating this here, our analysis leads to an algorithm for constructing the factor automaton of U in time $O(|Q|)$. More generally, we give novel bounds for the size of the factor automaton of an acyclic finite automaton as as a function of the size of the original automaton and the maximal length of a suffix shared by the strings accepted by the original automaton.

The original motivation for this work was the design of a large-scale music identification system [9], where we represented the songs found in the database by a compact finite automaton, as we shall briefly describe later in this paper. To facilitate an efficient search of song snippets, we constructed the minimal deterministic factor automaton of the song automaton. Empirically, the size of the factor automaton was not prohibitive. But, to ensure the scalability of our approach to a larger set of songs, e.g., several million songs, we wished to derive a bound on the size of the factor automata of automata. One characteristic of the strings considered in this application as in many others is that the original strings do not share long suffixes. This motivated our analysis of the size of the factor automata with respect to the length of the common suffixes in the original automaton.

The remainder of the paper is organized as follows. Section 2 introduces the string and automata definitions and terminology used throughout the paper. In Section 3, we describe a novel analysis of factor automata and present new bounds on the size of the factor automaton and suffix automaton of an automaton. Section 4 briefly describes the use of factor automata in music identification and reports several empirical results related to their size.

2 Factors of a Finite Automaton

This section reviews some key properties of factors of a fixed finite automaton, generalizing similar observations made for strings by Blumer et al. (1985).

We denote by Σ a finite alphabet. The length of a string $x \in \Sigma^*$ over that alphabet is denoted by $|x|$. A *factor*, or *substring*, of a string $x \in \Sigma^*$ is a sequence of symbols appearing consecutively in x. Thus, y is a factor of x iff there exist $u, v \in \Sigma^*$ such that $x = uyv$. A *suffix* of a string $x \in \Sigma^*$ is a factor that appears at the end of x. Put otherwise, y is a suffix of x iff there exists $u \in \Sigma^*$ such that $x = uy$. Analogously, y is a *prefix* of x iff there exists $u \in \Sigma^*$ such that $x = yu$. More generally, a factor, suffix, or prefix of a set of strings X or an automaton A, is a factor, suffix, or prefix of a string in X or a string accepted by A, respectively.

In some applications such as music identification the strings considered may be long, e.g., sequences of music sounds, but with relatively short common suffixes. This motivates the following definition.

Definition 1. *Let k be a non-negative integer. We will say that a finite automaton A is* k-suffix unique *if no two strings accepted by A share a suffix of length k. A is said to be* suffix-unique *when it is k-suffix unique with $k = 1$.*

We denote by $F(A)$ the minimal deterministic automaton accepting the set of factors of a finite automaton A, that is the set of factors of the strings accepted by A. Similarly, we denote by $S(A)$ the minimal deterministic automaton accepting the set of suffixes of an automaton A.

The following generalizes to automata some definitions and results given by Blumer et al. (1985) for a single string.

Definition 2. *Let A be a finite automaton. For any string $x \in \Sigma^*$, we define end-set(x) as the set of states of A reached by the paths in A labeled with x. We say that two strings x and y in Σ^* are equivalent and denote this by $x \equiv y$, when end-set(x) = end-set(y). This defines a right-invariant equivalence relation on Σ^*. We denote by $[x]$ the equivalence class of $x \in \Sigma^*$.*

Lemma 1. *Assume that A is suffix-unique. Then, a non-suffix factor x of the automaton A is the longest member of $[x]$ iff it is either a prefix of A, or both ax and bx are factors of A for distinct $a, b \in \Sigma$.*

Proof. Let x be a non-suffix factor of A. Clearly, if x is not a prefix, then there must be distinct a and b such that ax and bx are factors of A, otherwise $[x]$ would admit a longer member. Conversely, assume that ax and bx are both factors of A with $a \neq b$. Let y be the longest member of $[x]$. Let q be a state in *end-set*(x) = *end-set*(y). Since x is not a suffix, q is not a final state, and there exists a non-empty string z labeling a path from q to a final state. Since A is suffix-unique, both xz and yz are suffixes of the same string. Since y is the longest member of $[x]$, x must be a suffix of y. Since ax and bx are both factors of A with $a \neq b$, we must have $y = x$. Finally, if x is a prefix, then clearly it is the longest member of $[x]$. □

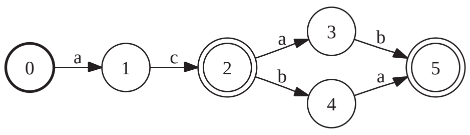


Fig. 1. Finite automaton A accepting the strings $ac, acab, acba$.

Proposition 1. *Assume that A is suffix-unique. Let $S_A = (Q_A, I_A, F_A, E_A)$ be the deterministic automaton whose states are the equivalence classes $Q_A = \{[x] \neq \emptyset : x \in \Sigma^*\}$, its initial state $I_A = \{[\epsilon]\}$, its final states $F_A = \{[x] : \text{end-set}(x) \cap F \neq \emptyset\}$ where F is the set of final states of A, and its transition set $E = \{([x], a, [xa]) : [x], [xa] \in Q_A\}$. Then, S_A is the minimal deterministic suffix of A: $S_A = S(A)$.*

Proof. By construction, S_A is deterministic and accepts exactly the set of suffixes of A. Let $[x]$ and $[y]$ be two equivalent states of S_A. Then, for all $z \in \Sigma^*$, $[xz] \in F_A$ iff $[yz] \in F_A$, that is z is a suffix of A iff yz is a suffix of A. Since A is suffix-unique, this implies that either x is a suffix of y or vice-versa, and thus that $[x] = [y]$. Thus, S_A is minimal. □

In much of what follows, we will be interested in the case where the automaton A is acyclic. We denote by $|A|_Q$ the number of states of A, by $|A|_E$ the number of transitions of A, and by $|A|$ the *size of A* defined as the sum of the number of states and transitions of A.

3 Space Bounds for Factor Automata

The objective of this section is to derive new bounds on the size of $S(A)$ and $F(A)$ in the case of interest for our applications where A is an acyclic automaton, typically deterministic and minimal, representing a set of strings.

When A represents a single string, there are standard algorithms for constructing $S(A)$ and $F(A)$ from A in linear time [3, 4]. In the general case, $S(A)$ can be constructed from A as follows: add an ϵ-transition from the initial state of A to each state of A, then apply an ϵ-removal algorithm, followed by determinization and minimization to obtain $S(A)$. $F(A)$ can be obtained similarly by further making all states final before applying ϵ-removal, determinization, and minimization. It can also be obtained from $S(A)$ by making all states of $S(A)$ final and applying minimization. Figure 1 shows a simple automaton A accepting three strings and Figure 2 its suffix automaton $S(A)$.

When A represents a single string x, the size of the automata $S(A)$ and $F(A)$ can be proved to be linear in $|x|$. More precisely, the following bounds hold for $|S(A)|$ and $|F(A)|$ [4, 3]:

$$\begin{array}{ll} |S(A)|_Q \leq 2|x| - 1 & |S(A)|_E \leq 3|x| - 4 \\ |F(A)|_Q \leq 2|x| - 2 & |F(A)|_E \leq 3|x| - 4. \end{array} \tag{1}$$

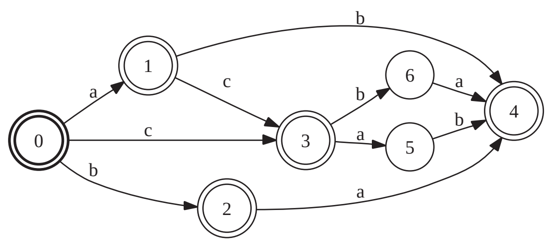


Fig. 2. Suffix automaton $S(A)$ of the automaton A of Figure 1.

These bounds are tight for strings of length more than three. [2] gave similar results for the case of a set of strings U by showing that the size of the factor automaton $F(U)$ representing this set is bounded as follows

$$|F(U)|_Q \leq 2\|U\| - 1 \quad |F(U)|_E \leq 3\|U\|_E - 3, \tag{2}$$

where $\|U\|$ denotes the sum of the lengths of all strings in U.

In general, the size of an acyclic automaton A representing a finite set of strings U can be substantially smaller than $\|U\|$. In fact, $|A|$ can be exponentially smaller than $\|U\|$. Thus, we are interested in bounding the size of $S(A)$ or $F(A)$ in terms of the size of A, rather than the sum of the lengths of all strings accepted by A.

For any state q of $S(A)$, we denote by $\text{suff}(q)$ the set of strings labeling the paths from q to a final state. We also denote by $N(q)$ the set of states in A from which a non-empty string in $\text{suff}(q)$ can be read to reach a final state.

Lemma 2. *Let A be a suffix-unique automaton and let q and q' be two states of $S(A)$ such that $N(q) \cap N(q') \neq \emptyset$, then*

$$\text{suff}(q) \subseteq \text{suff}(q') \textit{ and } N(q) \subseteq N(q') \quad \textit{or} \quad \text{suff}(q') \subseteq \text{suff}(q) \textit{ and } N(q') \subseteq N(q)\ . \tag{3}$$

Proof. Since $S(A)$ is a minimal automaton, its states are accessible from the initial state. Let u be the label of a path from the initial I of $S(A)$ to q and similarly u' the label of a path from I to q'.

By assumption, there exists $p \in N(q) \cap N(q')$. Thus, there exist non-empty strings $v \in \text{suff}(q)$ and $v' \in \text{suff}(q')$ such that both v and v' label paths from p to a final state.

By definition of u and u', both uv and $u'v'$ are suffixes of A. Since A is suffix-unique and v is non-empty, there exists a unique string accepted by A and ending with v. There exists also a unique string accepted by A and ending with uv. Thus, these two strings must coincide.

This implies that any string accepted by A and admitting v as suffix also admits uv as suffix. In particular, the label of any path from an initial state to p must admit u as suffix. Reasoning in the same way for v' let us conclude that the label of any path from an initial state to p must also admit u' as suffix. Thus,

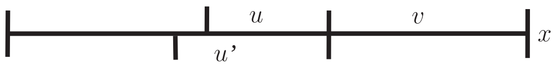


Fig. 3. Illustration of the situation described in Lemma 2. uv and $u'v$ are suffixes of the same string x. Thus, u and u' are also suffixes of the same string. Thus, u is a suffix of u' or vice-versa.

u and u' are suffixes of the same string. Thus, u is a suffix of u' or vice-versa. Figure 3 illustrates this situation.

Assume without loss of generality that u is a suffix of u'. Then, for any string w, if $u'w$ is a suffix of A so is uw. Thus, $\text{suff}(q') \subseteq \text{suff}(q)$, which implies $N(q') \subseteq N(q)$. When u' is a suffix of u, we obtain similarly the other case of the statement of the lemma. □

Note that Lemma 2 holds even when A is a non-deterministic automaton.

Lemma 3. *Let A be a suffix-unique deterministic automaton and let q and q' be two distinct states of $S(A)$ such that $N(q) = N(q')$, then either q is a final state and q' is not, or q' is a final state and q is not.*

Proof. Assume that $N(q) = N(q')$. By Lemma 2, this implies $\text{suff}(q) = \text{suff}(q')$. Thus, the same non-empty strings label the paths from q to a final state or the paths from q' to a final state. Since $S(A)$ is a minimal automaton, the distinct states q and q' are not equivalent. Thus, one must admit an empty path to a final state and not the other. □

The following proposition extends the results of [3] which hold for a set of strings, to the case where A is an automaton.

Proposition 2. *Let A be a suffix-unique deterministic and minimal automaton accepting strings of length more than three. Then, the number of states of the suffix automaton of A is bounded as follows*

$$|S(A)|_Q \leq 2|A|_Q - 3. \tag{4}$$

Proof. If the strings accepted by A are all of the form a^n, $S(A)$ can be derived from A simply by making all its states final and the bound is trivially achieved. In the remaining of the proof, we can thus assume that not all strings accepted by A are of this form.

Let F be the unique final state of $S(A)$ with no outgoing transitions. Lemmas 2-3 help define a tree T associated to all states of $S(A)$ other than F by using the ordering:

$$N(q) \sqsubseteq N(q') \text{ iff} \begin{cases} N(q) \subset N(q') \text{ or} \\ N(q) = N(q') \text{ and } q' \text{ final}, q \text{ non-final.} \end{cases} \tag{5}$$

We will identify each node of T with its corresponding state in $S(A)$. By Proposition 1, each state q of $S(A)$ can also be identified with an equivalence class

$[x]$. Let q be a state of $S(A)$ distinct from F, and let $[x]$ be its corresponding equivalence class. Observe that since A is suffix-unique, *end-set*(x) coincides with $N(q)$.

We will show that the number of nodes of T is at most $2|A|_Q - 4$, which will yield the desired bound on the number of states of $S(A)$. To do so, we bound separately the number of non-branching and branching nodes of T.

Let q be a node of T and let $[x]$ be the corresponding equivalence class, with x its longest member. The children of q are the nodes corresponding to the equivalence classes $[ax]$ where $a \in \Sigma$ and ax is a factor of A.

By Lemma 1, if x is a non-suffix and non-prefix factor, then there exist factors ax and bx with $a \neq b$. Thus, q admits at least two children corresponding to $[ax]$ and $[bx]$ and is thus a branching node. Thus non-branching nodes can only be either nodes q where x is a prefix, or those where x is a suffix, that is when q is a final state of $S(A)$.

Since the strings accepted by A are not all of the form a^n for some $a \in \Sigma$, the empty prefix ϵ occurs at least in two distinct left contexts a and b with $a \neq b$. Thus, the prefix ϵ, which corresponds to the root of T, is necessarily branching. Also, let f be the unique final state of A with no outgoing transitions. The equivalence class of the longest factor ending in f, that is the longest string accepted by A corresponds to the state F in $S(A)$ which is not included in the tree T. Thus, there are at most $|A|_Q - 2$ non-branching prefixes.

There can be at most one non-branching node for each string accepted by A. Let N_{str} denote the number of strings accepted by A, then, the number of non-branching nodes N_{nb} of T is at most $N_{nb} \leq |A|_Q - 2 + N_{str}$.

To bound the number of branching nodes N_b of T, observe that since A is suffix-unique, each string accepted by A must end with a distinct symbol a_i, $i = 1, \ldots, N_{str}$. Each a_i represents a distinct left context for the empty factor ϵ, thus the root node $[\epsilon]$ admits all $[a_i]$s, $i = 1, \ldots, N_{str}$, as children. Let T_{a_i} represent the sub-tree rooted at $[a_i]$ and let n_{a_i} represent the number of leaves of T_{a_i}. Let a_j, $j = N_{str} + 1, \ldots, N_{str} + k$ denote the other children of the root and let T_{a_j} denote each of the corresponding sub-tree. A tree with n_{a_i} leaves has less than n_{a_i} branching nodes. Thus, the number of branching nodes of T_{a_i} is at most $n_{a_i} - 1$. The total number of leaves of T is at most the number of disjoint subsets of Q excluding the initial state and f.

Note however that when the root node $[\epsilon]$ admits only $[a_i]$s, $i = 1, \ldots, N_{str}$, as children, that is when $k = 0$, then there is at least one a_i, say a_1, that is also a prefix of A since any other symbol would have been the root node's child. The node a_1 will then have also a child since it corresponds to a suffix or final state of $S(A)$. Thus, a_1 cannot be a leaf in that case. Thus, there are at most as many as $\sum_{i=1}^{N_{str}+k} n_{a_i} \leq |A|_Q - 2 - \min\{1, k\}$ leaves and the total number of branching nodes of T, including the root is at most $N_b \leq \sum_{i=1}^{N_{str}+k}(n_{a_i} - 1) + 1 \leq |A|_Q - 2 - \min\{1, k\} - (N_{str} + k) + 1 \leq |A|_Q - 2 - N_{str}$. The total number of nodes of the tree T is thus at most $N_{nb} + N_b \leq 2|A|_Q - 4$. □

In the specific case where A represents a single string x, the bound of Proposition 2 matches that of [4] or [3] since $|A|_Q = |x| + 1$. The bound of Proposition 2 is tight for strings of length more than three and thus is also tight for automata accepting strings of length more than three. Note that the automaton of Figure 1 is suffix-unique, deterministic, and minimal and has $|A|_Q = 6$ states. The number of states of the minimal suffix automaton of A is $|S(A)|_Q = 7 < 2|A|_Q - 3$.

Corollary 1. *Let A be a suffix-unique deterministic and minimal automaton accepting strings of length more than three. Then, the number of states of the factor automaton of A is bounded as follows*

$$|F(A)|_Q \leq 2|A|_Q - 3. \tag{6}$$

Proof. As mentioned earlier, a factor automaton $F(A)$ can be obtained from a suffix automaton $S(A)$ by making all states final and applying minimization. Thus, $|F(A)| \leq |S(A)|$. The result follows Proposition 2. □

Blumer et al. (1987) showed that an automaton accepting all factors of a set of strings U has at most $2\|U\|-1$ states, where $\|U\|$ is the sum of the lengths of all strings in U. The following gives a significantly better bound on the size of the factor automaton of a set of strings U as a function of the number of nodes of a prefix-tree representing U, which is typically substantially smaller than $\|U\|$.

Corollary 2. *Let $U = \{x_1, \ldots, x_m\}$ be a set of strings of length more than three and let A be a prefix-tree representing U. Then, the number of states of the factor automaton $F(U)$ and that of the suffix tree $S(U)$ of the strings of U are bounded as follows*

$$|F(U)|_Q \leq 2|A|_Q - 2 \quad |S(U)|_Q \leq 2|A|_Q - 2. \tag{7}$$

Proof. Let B be a prefix-tree representing the set $U' = \{x_1\$_1, \ldots, x_m\$_m\}$, obtained by appending to each string of U a new symbol $\$_i$, $i = 1, \ldots, m$, to make their suffixes distinct and let B' be the automaton obtained by minimization of B. By construction, B has m more states than A, but since all final states of B are equivalent and merged after minimization, B' has at most one more state than A.

By construction, B' is a suffix-unique automaton and by Proposition 2, $|S(B')|_Q \leq 2|B'|_Q - 3$. Removing from $S(B')$ the transitions labeled with the extra symbols $\$_i$ and connecting the resulting automaton yields the minimal suffix automaton $S(U)$. In $S(B')$, there must be a final state reachable by the transitions labeled with $\$_i$ and only such transitions, which becomes non-accessible after removal of the extra symbols. Thus, $S(U)$ has at least one state less than $S(B')$, which gives:

$$|S(U)|_Q \leq |S(B')|_Q - 1 \leq 2|B'|_Q - 4 = 2|A|_Q - 2. \tag{8}$$

A similar bound holds for the factor automaton $F(U)$ following the argument given in the proof of Corollary 1. □

When A is k-suffix-unique with a relatively small k as in our applications of interest, the following proposition provides a convenient bound on the size of the suffix automaton.

Proposition 3. *Let A be a k-suffix-unique deterministic automaton accepting strings of length more than three and let n be the number of strings accepted by A. Then, the following bound holds for the number of states of the suffix automaton of A:*

$$|S(A)|_Q \leq 2|A_k|_Q + 2kn - 3, \tag{9}$$

where A_k is the part of the automaton of A obtained by removing the states and transitions of all suffixes of length k.

Proof. Let A be a k-suffix-unique deterministic automaton accepting strings of length more than three and let the alphabet Σ be augmented with n temporary symbols $\$_1, \ldots, \$_n$. By marking each string accepted by A with a distinct symbol $\$_i$, we can turn A into a suffix-unique deterministic automaton A'.

To do that, we first unfold all k-length suffixes of A. In the worst case, all these (distinct) suffixes were sharing the same $(k-1)$-length suffix. Unfolding can thus increase the number of states of A by as many as $kn - n$ states in the worst case. Marking the end of each suffix with a distinct \$-sign further increases the size by n. The resulting automaton A' is deterministic and $|A'|_Q \leq |A_k|_Q + kn$. By Proposition 2, the size of the suffix automaton of A' is bounded as follows: $|S(A')| \leq 2|A'| - 3$. Since transitions labeled with a \$-sign can only appear at the end of successful paths in $S(A')$, we can remove these transitions and make their origin state final, and minimize the resulting automaton to derive a deterministic automaton A'' accepting the set of suffixes of A. The statement of the proposition follows the fact that $|A''| \leq |S(A')|$. $\square$

Since the size of $F(A)$ is always less than or equal to that of $S(A)$, we obtain directly the following result.

Corollary 3. *Let A be a k-suffix-unique automaton accepting strings of length more than three. Then, the following bound holds for the factor automaton of A:*

$$|F(A)|_Q \leq 2|A_k|_Q + 2kn - 3. \tag{10}$$

The bound given by the corollary is not tight for relatively small values of k in the sense that in practice, the size of the factor automaton does not depend on kn, the sum of the lengths of suffixes of length k, but rather on the number of states of A used for their representation, which for a minimal automaton can be substantially less. However, for large k, e.g., when all strings are of the same length and k is as long as the length of the strings accepted by A, our bound coincides with that of [2].

Similar results can be obtained for the number of transitions of the suffix automaton or factor automaton of a suffix-unique automaton ($|S(A)|_E \leq 3|A|_E - 4$) and k-suffix-unique automaton ($|S(A)|_E \leq 3|A_k|_E + 3kn - 3k - 1$), as in the string case.

4 Factor Automata for Music Identification

We have verified the above insights into factor automata in the context of a music identification system [9]. Music identification is the task of matching an audio

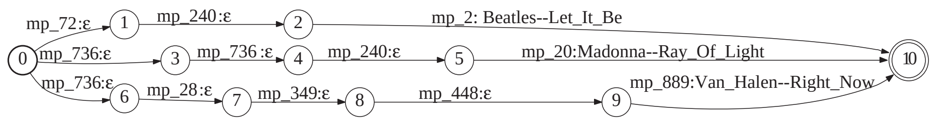


Fig. 4. Finite-state transducer T_0 mapping each song to its identifier.

stream to a particular song. In our system, we learn an inventory of music phone units similar to phonemes in speech and a unique sequence of music phones characterizing each song. We view the music phone set as our alphabet and the music phone sequences as a set of strings, transforming the task into a factor recognition problem. Our approach is to construct a compact transducer mapping music phone sequences to corresponding song identifiers.

4.1 Factor Transducer Construction

Let Σ denote the set of music phones and let the set of music phone sequences describing m songs be $U = \{x_1, \ldots, x_m\}, x_i \in \Sigma^*$ for $i \in \{1, \ldots, m\}$. In our experiments, $m = 15{,}455$, $|\Sigma| = 1{,}024$ and the average length of a transcription x_i is more than 1,700. Thus, in the worst case, there can be as many as $15{,}455 \times 1{,}700^2 \approx 45 \times 10^9$ factors. The size of a naive prefix-tree-based representation would thus be prohibitive, and hence we endeavor to represent the set of factors with a much more compact factor automaton. We construct a deterministic and minimal automaton representing the sequences in U and subsequently a deterministic and minimal finite-state transducer mapping each song to its identifier using transducer determinization and minimization algorithms [7, 8].

Let T_0 be the transducer mapping phone sequences to song identifiers before determinization and minimization. Figure 4 shows T_0 when U is reduced to three short songs. Let A be the acceptor obtained by omitting the output labels of T_0. The factor automaton $F(A)$ is constructed as described in Section 3: by creating ϵ-transitions from the initial state of A to all other states, making all states final, and applying epsilon-removal, determinization, and minimization. This leads to a compact automaton representing the factors of U (see Figure 5(a)), but it does not allow us to output the song identifier associated with each factor.

To accomplish this, we create a compact weighted acceptor over the tropical semiring accepting the factors of U that associates the total weight s_x to each factor x. A crucial advantage of this representation is the use of weighted determinization and minimization [7] during which the song identifier is treated as a weight that can be distributed along a path. The property that the sum of the weights along the path labeled with x is s_x is preserved by these operations. Let $F_w(A)$ be the automaton constructed analogously to $F(A)$, but where each ϵ-transition added is weighted with the song identifier corresponding to that transition. The weighted acceptor $F_w(A)$, obtained after determinization and minimization over the tropical semiring, is transformed into a song recognition transducer T by treating each output weight integer as a regular output symbol.

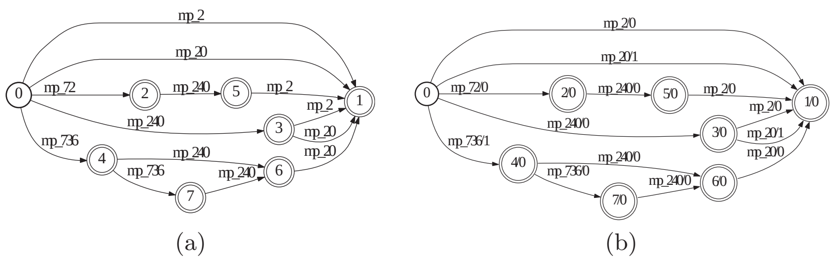


Fig. 5. (a) Deterministic and minimal unweighted factor acceptor $F(A)$ for two songs. (b) Deterministic and minimal weighted factor acceptor $F_w(A)$ for two songs.

Given a music phone sequence as input, the associated song identifier is obtained by summing the outputs yielded by T.

4.2 Automata Size

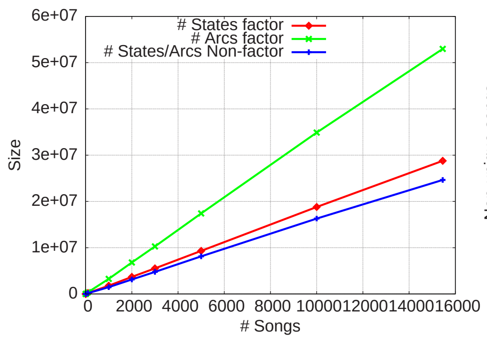
Figure 5(b) shows the weighted automaton $F_w(A)$ corresponding to the unweighted automaton $F(A)$ of Figure 5(a). Note that $F_w(A)$ is no larger than $F(A)$. Remarkably, even in the case of 15,455 songs, the total number of transitions of $F_w(A)$ was 53.0M, only about 0.004% more than $F(A)$. We also have $|F(A)|_E \approx 2.1|A|_E$. As is illustrated in Figure 6(a), this multiplicative relationship is maintained as the song set size is varied between 1 and 15 ,455. Furthermore, for the case of 15 ,455 songs, U is 45-suffix-unique. Figure 6(b) demonstrates that the number of suffix "collisions" drops rapidly as the suffix size is increased. We also have $|F_w(A)|_Q \approx 28.8M \approx 1.2|A|_Q$, meaning the bound of Corollary 3 is verified in this empirical context.

5 Conclusion

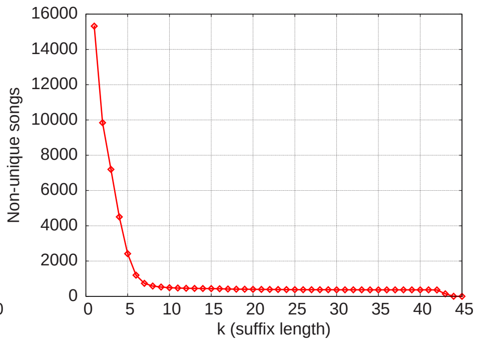
We presented a novel analysis of the size of the factor automaton of an automaton in terms of the size of the original automaton and described the use of factor automata in a large-scale application. Our analysis shows that factor automata can be practical for a large number of strings. Our application to a large-scale music identification task further demonstrates this fact. Factor automata of automata are likely to form a useful and compact index for very large-scale tasks.

Acknowledgments

We thank Cyril Allauzen for several discussions about the material presented. The research of Mehryar Mohri and Eugene Weinstein was partially supported by the New York State Office of Science Technology and Academic Research (NYSTAR). This project was also sponsored in part by the Department of the Army Award Number W81XWH-04-1-0307. The U.S. Army Medical Research Acquisition Activity, 820



(a)



(b)

Fig. 6. (a) Comparison of automaton sizes for different numbers of songs. "#States/Arcs Non-factor" is the size of the automaton A accepting the entire song transcriptions. "# States factor" and "# Arcs factor" is the number of states and transitions in the weighted factor acceptor $F_w(A)$, respectively. (b) Number of strings in U for which the suffix of length k is also a suffix of another string in U.

Chandler Street, Fort Detrick MD 21702-5014 is the awarding and administering acquisition office. The content of this material does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

References

1. Cyril Allauzen, Mehryar Mohri, and Murat Saraclar. General Indexation of Weighted Automata – Application to Spoken Utterance Retrieval. In *Proceedings of the Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval (HLT/NAACL 2004)*, pages 33–40, Boston, Massachusetts, May 2004.
2. A. Blumer, J. Blumer, A. Ehrenfeucht, D. Haussler, and R. McConnell. Complete inverted files for efficient text retrieval and analysis. *Journal of the ACM*, 34:578–589, 1987.
3. A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht, M.T. Chen, and J. Seiferas. The smallest automaton recognizing the subwords of a text. *Theoretical Computer Science*, 40:31–55, 1985.
4. M. Crochemore. Transducers and repetitions. *Theoretical Computer Science*, 45:63–86, 1986.
5. M. Crochemore and W. Rytter. *Jewels of Stringology*. World Scientific, 2002.
6. Dan Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, Cambridge, UK., 1997.
7. M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
8. M. Mohri. Statistical Natural Language Processing. In M. Lothaire, editor, *Applied Combinatorics on Words*. Cambridge University Press, 2005.
9. E. Weinstein and P. Moreno. Music Identification with Weighted Finite-State Transducers. In *Proceedings of ICASSP 2007*, Honolulu, Hawaii, 2007.