Journal of Computer and System Sciences 80 (2014) 450–456

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

# Deciding unique decodability of bigram counts via finite automata

Aryeh Kontorovich [a,*], Ari Trachtenberg [b]

[a] Department of Computer Science, Ben-Gurion University, Beer Sheva 84105, Israel
[b] Department of Electrical & Computer Engineering, Boston University, 8 Saint Mary's Street, Boston, MA 02215, United States

ARTICLE INFO

*Article history:*
Received 28 November 2011
Received in revised form 23 August 2013
Accepted 18 September 2013
Available online 26 September 2013

*Keywords:*
Uniqueness
Sequence reconstruction
Eulerian graph
Finite-state automata

ABSTRACT

We revisit the problem of deciding by means of a finite automaton whether a string is uniquely decodable from its bigram counts. An efficient algorithm for constructing a polynomial-size Nondeterministic Finite Automaton (NFA) that decides unique decodability is given. This NFA may be simulated efficiently in time and space. Conversely, we show that the minimum deterministic finite automaton for deciding unique decodability has exponentially many states in alphabet size, and compute the correct order of magnitude of the exponent.



## 1. Introduction

Reconstructing a string from its snippets is a problem of fundamental importance in many areas of computing. In a biological context this problem amounts to sequencing of DNA from short reads [7] and reconstruction of protein sequences from K-peptides [10]. Communications protocols [3,9] recombine snippets from related documents to identify differences between them, and fuzzy extractors [11] use similar techniques for producing keys from noise-prone biometric data. Computational linguistics also makes occasional use of this snippet representation (under the name Wickelfeatures [1]), as a means to learn transformations on varying-length sequences.

In general, there may be a large number of possible string reconstructions from a given collection of overlapping snippets; for example, the snippets {at, an, ka, na, ta} can be combined into katana or kanata. In order to keep the decoding complexity and ambiguity low, it is desirable in practice to choose a snippet length that allows only a few distinct reconstructions — the ideal number being exactly one.

*Main results.* We consider the problem of *efficiently* determining whether a collection of snippets has a unique reconstruction. More precisely, we construct a nondeterministic finite automaton (NFA) on $O(|\Sigma|^3)$ states that recognizes the language of strings that admit a unique reconstruction from bigram counts over the alphabet $\Sigma$. Our NFA has a particularly simple form that provides for an easy and efficient implementation, and runs on a string of length $\ell$ in time $O(\ell|\Sigma|^3)$ and constant memory.[1] We further show that the minimum equivalent deterministic finite automaton has at least $|\Sigma|!$ states.[2] Together

* Corresponding author. Fax: +972 8 647 7650.
*E-mail addresses:* karyeh@cs.bgu.ac.il (A. Kontorovich), trachten@bu.edu (A. Trachtenberg).

[1] We have since reduced the time to linear via an entirely different approach that sidesteps automata [4]. See also the algorithm sketched but not formally analyzed in [12].

[2] In a personal communication, Q. Li improved our previous bound of $2^{|\Sigma|-1}$.

0022-0000/$ – see front matter 
http://dx.doi.org/10.1016/j.jcss.2013.09.003

with the upper bound $2^{O(|\Sigma|\log|\Sigma|)}$ implicit in [12], this pegs the size of the canonical DFA at $\exp(\Theta(|\Sigma|\log|\Sigma|))$; closing this gap is an intriguing open problem.

*Related work.* It was shown in [8] that the collection of strings having a unique reconstruction from the snippet representation is a regular language. An explicit construction of a deterministic finite-state automaton (DFA) recognizing this language was given in by Li and Xie [12]. Unfortunately, this DFA has

$$2^{|\Sigma|}\big(|\Sigma|+1\big)\big(|\Sigma|+1\big)^{(|\Sigma|+1)} \in 2^{O(|\Sigma|\log|\Sigma|)}$$

states, and thus is not practical except for very small alphabets.[3] As we show in this paper, there is no DFA of subexponential size for recognizing this language; however, we exhibit an equivalent NFA with $O(|\Sigma|^3)$ states.

*Outline.* We proceed in Section 2 with some preliminary definitions and notation. In Section 3 we present our construction of an NFA recognizing uniquely decodable strings, and we prove its correctness in Section 4. Finally, we present a new lower bound on the size of a DFA accepting uniquely decodable strings in Section 5, and conclude in Section 6 with discussion and an open problem.

## 2. Preliminaries

We assume a finite alphabet $\Sigma$ along with a special delimiter character $\$ \notin \Sigma$, and define $\Sigma_\$ = \Sigma \cup \{\$\}$. For $k \geqslant 1$, the $k$-gram map $\Phi$ takes string $x \in \$\Sigma^*\$$ to a vector $\xi \in \mathbb{N}^{\Sigma_\$^k}$, where $\xi_{i_1,\ldots,i_k} \in \mathbb{N}$ is the number of times the string $i_1 \ldots i_k \in \Sigma^k$ occurred in $x$ as a contiguous subsequence, counting overlaps. In this paper we will focus on the *bigram* case $k = 2$, and leave a detailed study of higher $k$-grams for future work. As we have seen, the bigram map $\Phi : \$\Sigma^*\$ \to \mathbb{N}^{\Sigma_\$^2}$ is not injective; for example, $\Phi(\$\text{katana}\$) = \Phi(\$\text{kanata}\$)$.

We denote by $L_{\text{UNIQ}} \subseteq \Sigma^*$ the collection of all strings $w$ for which

$$\Phi^{-1}\big(\Phi(\$w\$)\big) = \{\$w\$\}$$

and refer to these strings as *uniquely decodable*, meaning that there is exactly one way to reconstruct them from their bigram snippets. The examples \$katan\$ and \$katana\$ show that $\emptyset \neq L_{\text{UNIQ}} \neq \Sigma^*$ for $|\Sigma| > 1$. The induced *bigram graph* of a string $w \in \Sigma^*$ is a weighted directed graph $G = (V, E)$, with $V = \Sigma_\$$ and $E = \{e(a,b)\colon\ a, b \in \Sigma_\$\}$, where the edge weight $e(a,b) \geqslant 0$ records the number of times $a$ occurs immediately before $b$ in the string $\$w\$$.

We also follow the standard conventions for sets, languages, regular expressions, and automata [2,5,6]. As such, a *factor* of a string (colloquially a *snippet*) is any of its contiguous substrings. The term $\Sigma^*$ denotes the free monoid over the alphabet $\Sigma$, and, for $S \subseteq \Sigma$, the term $S^*$ has the usual regular-expression interpretation; the language defined by a regular expression $\mathbf{R}$ will be denoted $L(\mathbf{R})$. In addition, we will denote the omission of a symbol from the alphabet by $\Sigma_{\bar{x}} := \Sigma \setminus \{x\}$ for $x \in \Sigma$.

Finally, we shall use the standard five-tuple [5] notation $(\Sigma, Q, q_0, \delta, F)$ to specify a given DFA, where $\Sigma$ is the input alphabet, $Q$ is the set of states, $q_0$ is the initial state, $\delta$ is the transition function, and $F$ are the final states; an analogous notation is used for NFAs. We use the notation $|\cdot|$ both to denote the size of an automaton (measured by the number of states) and the length of a string.

## 3. Construction and simulation of the NFA

### 3.1. Obstruction languages and their DFAs

Our starting point is the observation, also made in [12], that $L_{\text{UNIQ}}$ is a *factorial* language, meaning that it is closed under taking factors. From here, Li and Xie [12] proceed to characterize $L_{\text{UNIQ}}$ in terms of its minimal forbidden words. We shall likewise consider obstructions in the form of simple regular languages. Indeed, our Theorem 2 bears a superficial similarity to [12, Theorem 3]. However, since we deal with forbidden *languages* rather than words, we are able to obtain a much more compact description of $L_{\text{UNIQ}}$.

For $x \in \Sigma$ and $a, b \in \Sigma_{\bar{x}}$, define

$$I_{x,a,b} = L\big(\Sigma^* a \Sigma_b^* x \Sigma_a^* b \Sigma^*\big).$$

In particular, note that for every $w \in I_{x,a,b}$, its induced bigram graph has a directed path from $a$ to $x$ avoiding $b$ and a directed path from $x$ to $b$ avoiding $a$ — but this is not a characterization of $I_{x,a,b}$, as the example $w = xbax$ shows. Similarly, for $x \in \Sigma$ and $a, b \in \Sigma_{\bar{x}}$, define

$$J_{x,a,b} = L\big(\Sigma^* a \Sigma_{\bar{x}}^* b \Sigma^*\big).$$

[3] Li and Xie [12] apparently have an efficient algorithm for simulating their exponentially large DFA, but they do not clearly provide all the details.