Comparing nondeterministic and quasideterministic finite-state transducers built from morphological dictionaries*

Alicia Garrido-Alenda and Mikel L. Forcada
Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant
E-03071 Alacant, Spain
{alicia, mlf}@dlsi.ua.es

Resumen: Se presenta una comparación entre transductores de estados finitos cuasideterministas y no deterministas generados a partir de diccionarios morfológicos que contienen el vocabulario (lemas) y la información sobre flexión morfológica de una aplicación de procesamiento del lenguaje natural tal como el analizador morfológico en un sistema de traducción automática. Los resultados indican que los transductores no deterministas son más compactos que los cuasideterministas sin ser mucho más lentos.
Palabras clave: Transductores de estados finitos, diccionarios morfológicos

Abstract: This paper describes a comparison between quasideterministic and non-deterministic finite-state transducers generated from morphological dictionaries containing the vocabulary (lemmas) and the morphological inflection information of a natural language processing application such as the morphological analyser of a machine translation system. Results show that non-deterministic transducers are more compact than their quasideterministic counterparts while being almost as fast.
Keywords: Finite-state transducers, morphological dictionaries

1 Introduction

This paper describes a comparison between quasideterministic and non-deterministic finite-state transducers generated from morphological dictionaries containing the vocabulary (lemmas) and the morphological inflection information of a natural language processing (NLP) application such as the morphological analyser of a machine translation (MT) system. Results show that non-deterministic transducers are more compact than their quasideterministic counterparts while being almost as fast, especially when linguists have found a good alignment between the strings representing surface and lexical forms in the morphological dictionary, and justify their current use in real applications such as the MT system `www.interNOSTRUM.com` (Canals-Marote et al., 2001). The paper is organized as follows: section 2 describes morphological dictionaries, analysers and generators; section 3 reviews subsequential tranductions and transducers, and their problems are stated in section 4. Letter transducers are introduced as an alternative in section 5; the comparison is described in section 6 and some concluding remarks are given in section 7.

2 Morphological dictionaries, analysers and generators

A *morphological dictionary* (MD) is a file containing correspondences between *surface forms* (SFs), that is, the (possibly inflected) forms of words and lexical units as found in texts, and *lexical forms* (LFs), consisting of the *lemma* or base form of the word as found in a printed dictionary, its lexical category, and inflection information. For example, the Spanish SF *cantaríamos* corresponds to the LF with lemma *cantar*, lexical category (*verb*), and inflection information *conditional*, *1st person*, *plural*. Although MDs are usually written in a compact way by encoding the regularities observed in LF-SF correspondences (usually as inflection *paradigms*), we will, consider them as lists of SF–LF pairs (that is, after paradigm expansion). The information in a MD is used to describe the behavior of *morphological analysers* and *mor-*

* Work supported by the Spanish Comisión Interministerial de Ciencia y Tecnología through grant TIC2000-1599-CO2-02, and by the Universitat d'Alacant and Caja de Ahorros del Mediterráneo through project `internostrum.com`.

phological generators, two important components of MT systems and many other NLP applications. Morphological analysers are the first to deal with the source text and identify and classify lexically relevant units; they are ideally compact and read text left-to-right, quickly, and only once, divide the text into SFs, and incrementally output the corresponding LFs (one or more per SF). Morphological generators are one of the the last steps in the generation of the target text and generate SFs from LFs.

We will distinguish two kinds of morphological dictionaries: *aligned* MDs (AMDs) and *unaligned* MDs. AMDs, the specify SF–LF correspondences at the character level, in an explanatory attempt; a possible alignment (not the only possible one) for the previous example would be: (c, c) (a, a) (n, n) (t, t) (θ, a) (θ, r) (θ, <vb>) (a, θ) (r, θ) (í, <cnd>) (a, θ) (m, <1>) (o, <pl>) (s, θ), Where θ is an empty symbol, and <vb>, <cnd>, <1>, <pl> are specialized one-symbol entities which stand for *verb*, *conditional*, *1st person* and *plural*. Since MDs can be used to specify both analysers and generators, we will talk of *input* and *output*: in the example, the input is the SF and the output is the LF, as in an analyser. Formally, if Σ is the alphabet of input symbols and Γ the alphabet of output symbols, an AMD is a subset $A \subset L^*$ where

$$L = (\Sigma \cup \{\theta\}) \times \Gamma \cup \Sigma \times (\Gamma \cup \{\theta\}) \qquad (1)$$

is the "alphabet" of all possible alignment pairs,[1] including pairs with no input (θ, γ) or no output (σ, θ) (θ is the *empty symbol*).

An unaligned MD (UMD) is simply a set $U \subset \Gamma^* \times \Sigma^*$ of input–output string pairs such as (cantaríamos, cantar<vb><cnd><1><pl>), without alignment information. An UMD may be trivially obtained from an AMD.

The domain or set $E \subset \Sigma^*$ of input strings of an MD is $E = \{w : (w, w') \in U\}$. In morphological analysis, some SFs may be ambiguous (homographs) and have more than one LF. A function $\tau : E \to 2^{\Gamma^*}$ mapping input strings in E to the corresponding sets of output strings may be defined as follows: $\tau(w) = \{w' : (w, w') \in U\}$ for all $w \in E$. Usually, there exists a $p = \max_{w \in E} |\tau(w)|$ (no SF w has more than p LFs).

[1] These alignment pairs correspond the edit operations specified in string edit distances (Wagner and Fischer, 1974): substitutions ($\sigma : \gamma$), deletions ($\sigma : \theta$), and insertions ($\theta : \gamma$).

3 Subsequential processing

3.1 Subsequential transductions

Many lexical transformations, such as morphological analysis and generation, or even the lookup of a bilingual dictionary may be formulated in a sequential manner: output prefixes may be incrementally built as input is read from left to right. For many languages (e.g., Indoeuropean languages), this is because usually, a set of SFs sharing a prefix corresponds to a set of LFs sharing a nontrivial prefix (morphological variations mainly affect the suffixes of forms). The same may occur between the lemmas of a bilingual dictionary. This is formalized through the concepts of *sequential* and *subsequential* transductions.

A transformation $\eta : E \to \Gamma^*$, where E is the subset of Σ^* where η is defined, and Γ^* is the output language, is called *sequential* if

$$\eta(\epsilon) = \mu_0$$
$$\eta(w\sigma) = \eta(w)\zeta(w, \sigma) \quad \forall w, w\sigma \in \Pr(E),\ \sigma \in \Sigma$$

where $\mu_0 \in \Gamma^*$ is the initial output string (usually empty), $\zeta(w, \sigma) \in \Gamma^*$ is the extension added to the result when reading symbol σ after the input w, and $\Pr(E) = \{x : (\exists y \in \Sigma^* : xy \in E)\}$ is the set of all prefixes of strings in E.

But, as has been mentioned above, a SF may in some cases have more than a single LF associated to it and will be assumed that it never has more than p LFs. For example, the Spanish SF *recuerdo* may be a masculine singular noun or the first person singular of the present tense of verb *recordar*; therefore, analysis should produce two LFs. If one wants to process *recuerdo* sequentially (letter by letter) so that the longest common prefix (LCP) of all possible LFs is always produced, one finds that after reading rec, reading more letters does not increase the LCP rec since this is indeed the LCP of both LFs; it is only after detecting the end of the input that one can actually complete rec with either uerdo<n><m><sg> or ordar<v><pri><1><sg>. This behavior cannot be modelled with sequential transductions; therefore we introduce the concept of *earliest p-subsequential* transduction.

A transduction $\tau : E \to 2^{\Gamma^*}$ assigning to each string in $E \subseteq \Sigma^*$ a set of strings in Γ^*

is the *earliest p-subsequential* transduction[2] if there exists a sequential transduction η so that

$$\tau(w) = \eta(w)S(w)\ \forall w \in E,\ |S(w)| \leq p \quad (2)$$

where $S(w) \in 2^{\Gamma^*}$ is a set of at most p tails (suffixes), with

$$\begin{array}{rcl} \mu_0 & = & \mathrm{LCP}(\tau(E)); \\ \zeta(w,\sigma) & = & [\mathrm{LCP}(\tau(ww^{-1}E))]^{-1} \\ & & [\mathrm{LCP}(\tau((w\sigma)(w\sigma)^{-1}E))]; \text{ and} \\ S(w) & = & [\mathrm{LCP}(\tau(ww^{-1}E))]^{-1}\tau(w), \end{array}$$

where $ww^{-1}E = \{x \in E\ :\ w \in \mathrm{Pr}(x)\}$ and $(w\sigma)(w\sigma)^{-1}E = \{x \in E\ :\ w\sigma \in \mathrm{Pr}(x)\}$. In this way, each time that a symbol σ is read, function $\zeta(w,\sigma)$ appends the longest possible suffix to $\eta(w)$ to form $\eta(w\sigma)$, the current output prefix; finally, $\tau(w)$ is computed by concatenating the result of the sequential transduction η with a set of at most p suffixes $S(w)$. In particular, sequential tranductions are a special case of p subsequential transductions: those with $p = 1$ and $S(w) = \{\epsilon\}\ \forall w \in E$. This kind of processing allows to organize a NLP application as a pipeline, with each module receiving as early as possible the output of previous modules.

The requirement that transductions be deterministic, however, is not without problems; the only way that an input word can have more than a possible output associated to it is by producing first the LCP of all output, only producing different output suffixes of each output after seeing the whole input.

3.2 Earliest deterministic finite-state p-subsequential transducers

The usual implementation[3] earliest p-subsequential transductions for a finite UMD U (tranduction τ) is through *earliest deterministic finite-state p-subsequential transducers*, defined as follows: Given a p-subsequential transduction $\tau : E \rightarrow 2^{\Gamma^*}$, with E a finite set, the corresponding earliest deterministic (finite-state) p-subsequential transducer (EDpSST) is $T = (Q, \Sigma, \Gamma, \delta, \lambda, q_I, \psi)$, where Q is the set $Pr(E) \cup \{\perp\}$ of all input prefixes plus an absorption state; Σ is the input alphabet; Γ is the output alphabet; $\delta : Q \times \Sigma \rightarrow Q$ is the transition function

$$\delta(x,\sigma) = \begin{cases} x\sigma & \text{if } x, x\sigma \in \mathrm{Pr}(E) \\ \perp & \text{otherwise} \end{cases}; \quad (3)$$

$\lambda : Q \times \Sigma \rightarrow \Gamma^*$ is the output function

$$\begin{array}{rl} \lambda(x,\sigma) = & [\mathrm{LCP}(\tau(xx^{-1}E))]^{-1} \\ & [\mathrm{LCP}(\tau((x\sigma)(x\sigma)^{-1}E))], \end{array} \quad (4)$$

for $x, x\sigma \in \mathrm{Pr}(E)$, and undefined otherwise; [4] $q_I = \{\epsilon\}$ is the initial state, and $\psi : Q \rightarrow 2^{\Gamma^*}$ is a function assigning to each state a set of tails to be appended to the output at the end of the input:

$$\psi(w) = \begin{cases} [\mathrm{LCP}(\tau(xx^{-1}E))]^{-1}\tau(w) & \text{if } w \in E \\ \emptyset & \text{otherwise} \end{cases}. \quad (5)$$

This construction is basically a trie for the strings in E endowed with output functions λ and ψ such that output is produced as early as possible (output information may be easily added along a post-order traversal of the trie). The resulting (acyclic) transducer may be easily minimized into an equivalent transducer producing the same output for all prefixes in $\mathrm{Pr}(E)$ and appending the same tails to all strings in E while using the minimal number of states. The usual technique (Mohri, 1997) iteratively refines a partition of Q into equivalence classes by testing nonequivalence in each iteration: states q and r go to different classes if (a) $\psi(q) \neq \psi(r)$, (b) if for some σ $\delta(q,\sigma)$ does not belong to the same class as $\delta(r,\sigma)$, or (c) if for some σ $\lambda(q,\sigma) \neq \lambda(r,\sigma)$.

4 Some problems associated to deterministic transducers

EDpSSTs have some inconvenient features:

- As was described in section 3, if a transduction is only validated at the end of the input, output has to be delayed. Section 6 gives data about the typical length of tails in real dictionaries.
- If a new entry is added to the MD, the new transducer has to be (almost) completely rebuilt: the LCP computations and the ensuing state equivalences may be no longer correct for most states.

[2] Mohri (1997) defines p-subsequential transductions similarly; Oncina, García, and Vidal (1993) call their 1-subsequential transductions *onward* instead of *earliest*.

[3] See, e.g. Oncina, García, and Vidal (1993), who use, however, a different notation.

[4] Note that if $\mathrm{LCP}[\tau(E)] \neq \epsilon$, all of the outputs $\lambda(\epsilon,\sigma)$ have $\mathrm{LCP}[\tau(E)]$ prepended.

As will be seen, these problems may be avoided by allowing the transducer to maintain several transduction hypotheses alive during the process, some of which could be discarded after reading some more input (i.e., a nondeterministic transducer). This kind of processing does not require a realignment of inputs and outputs but may instead preserve the alignments in an AMD.

Fig. 1 shows a nondeterministic finite-state transducer which however is deterministic with respect to input–output pairs whereas fig. 2 shows the equivalent earliest deterministic p-subsequential transducer in which state chains have been collapsed for clarity. Both transducers have been produced from the MD in table 1.

As may be seen, the FST which is deterministic with respect to input–output pairs (1) may advance along more than one transduction; a transduction may be discarded later. For example, when processing the input *tienta*, after reading `t-`, the transducer maintains a single transduction alive, but after reading `ti-`, it forms two possible outputs, `ti-` and `te-`; further along, after having read `tient-`, two possible outputs are still alive, `tient-` and `tent-`. It is after reading the character `a` that the output `tient-` is discarded and the output `tentar<vblex><pri><3><sg>` is assembled. The corresponding earliest deterministic subsequential transducer (fig. 2) can only build the output string `t-` until it sees the last letter (`a`).

This suggests that transducers which are deterministic with respect to the input–output pair may be an interesting alternative: (1) they naturally integrate ambiguity in the form of nondeterminism; (2) input-output alignments used to express linguistically-motivated analyses and regularities in the MD may not be compatible with a deterministic processing of input, but may however have very compact representations; (3) they may be easily built from AMDs, because there is a direct correspondence between alignments and state paths; (4) they are more compact than their input-deterministic counterparts (see sect. 6) because they are not forced to delay their output, but instead maintain more than one output hypothesis until they are found to be invalid; and (5) they are not much slower (see section 6).

re	c	o	rd	áis
re	c	o	rd	ar<vblex><pri><2><pl>
re	c	o	rd	amos
re	c	o	rd	ar<vblex><pri><1><pl>
re	c	o	rd	amos
re	c	o	rd	ar<vblex><ifi><1><pl>
re	c	ue	rd	o
re	c	o	rd	ar<vblex><pri><1><sg>
re	c	ue	rd	o
re	c	ue	rd	o<n><m><sg>
re	c	ue	rd	a
re	c	ue	rd	ar<vblex><pri><3><sg>
	t	e	nt	áis
	t	e	nt	ar<vblex><pri><2><pl>
	t	e	nt	amos
	t	e	nt	ar<vblex><pri><1><pl>
	t	e	nt	amos
	t	e	nt	ar<vblex><ifi><1><pl>
	t	ie	nt	o
	t	e	nt	ar<vblex><pri><1><sg>
	t	ie	nt	o
	t	ie	nt	o<n><m><sg>
	t	ie	nt	a
	t	e	nt	ar<vblex><pri><3><sg>
	c	o	nt	áis
	c	o	nt	ar<vblex><pri><2><pl>
	c	o	nt	amos
	c	o	nt	ar<vblex><pri><1><pl>
	c	o	nt	amos
	c	o	nt	ar<vblex><ifi><1><pl>
	c	ue	nt	o
	c	o	nt	ar<vblex><pri><1><sg>
	c	ue	nt	o
	c	ue	nt	o<n><m><sg>
	c	ue	nt	a
	c	o	nt	ar<vblex><pri><3><sg>
	t	ie	nt	o
	t	ie	nt	o<n><m><sg>
re	t			áis
re	t			ar<vblex><pri><2><pl>
re	t			amos
re	t			ar<vblex><pri><1><pl>
re	t			amos
re	t			ar<vblex><ifi><1><pl>
re	t			o
re	t			ar<vblex><pri><1><sg>
re	t			o
re	t			o<n><m><sg>
re	t			a
re	t			ar<vblex><pri><3><sg>

Table 1: The short, partially aligned MD used to build the transducers in figs. 1 and 2.

5 Letter transducers

A convenient formalization of nondeterministic transducers is that of *letter transducers* (LTs); any finite-state transducer may be turned into an equivalent LT(Roche and

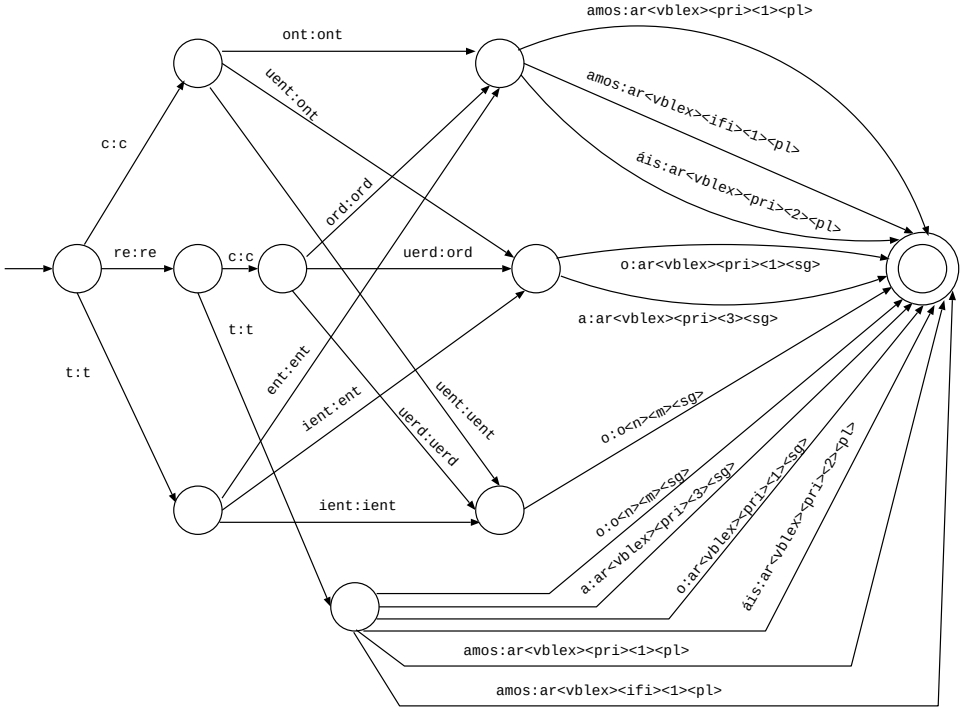

Figure 1: Nondeterministic transducer corresponding to the MD in table 1, which is deterministic with respect to input–output pairs (state chains collapsed for clarity).

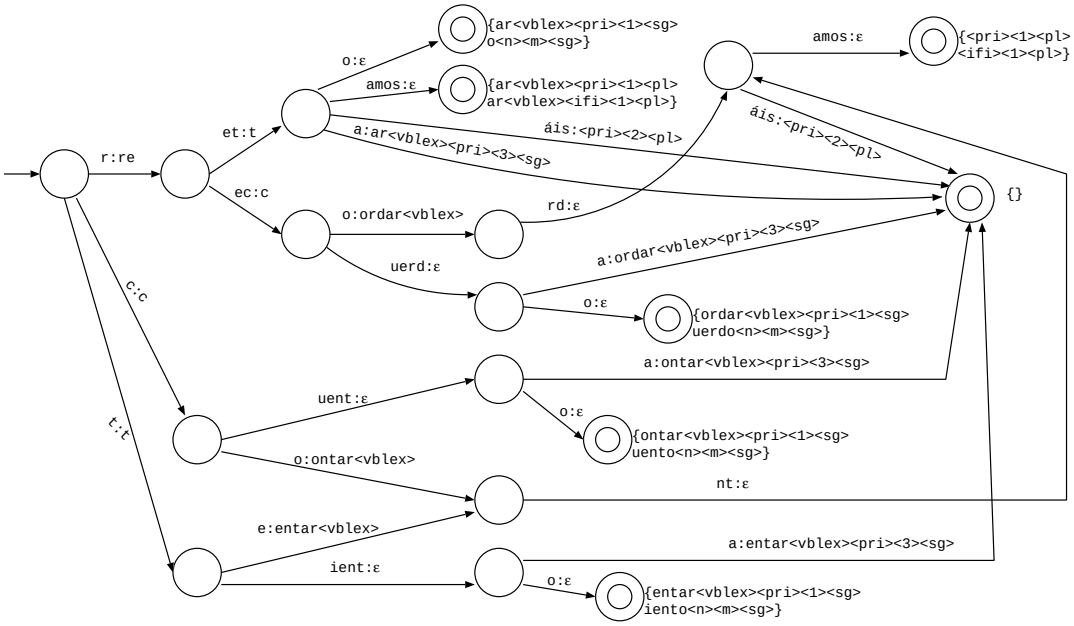

Figure 2: Deterministic p-subsequential transducer corresponding to the MD in table 1 (state chains collapsed for clarity).

Schabes, 1997). A LT is $T = (Q, L, \delta, q_I, F)$, where Q is a finite set of states; L is the set of labels defined in eq. (1), $q_I \in Q$ is the initial state, $F \subseteq Q$ is the set of acceptance states, and $\delta : Q \times L \rightarrow 2^Q$ is the transition function which assigns a finite set of destination states to each state, input symbol pair.

According to this definition, state-to-state transitions may be of three kinds: $(\sigma : \gamma)$ (a symbol is read and a symbol is written), $(\sigma : \theta)$ (a symbol is read but nothing is written), and $(\theta : \gamma)$ (a symbol is written but

nothing is read); during construction it is useful to add $(\theta : \theta)$ to L, that is, transitions that don't write or read anything (Garrido et al., 1999). A LT is *L-deterministic*, that is, deterministic with respect to alphabet L when $\delta : Q \times L \to Q$; an L-deterministic transducer need not be Σ-deterministic (i.e., deterministic when reading its input).

A string $w' \in \Gamma^*$ is taken to be an output associated to input string $w \in \Sigma^*$ if there is a path, that is, a sequence $s_0 \xrightarrow{l_1} s_1 \xrightarrow{l_2} s_2 \ldots \xrightarrow{l_n} s_n$, with $s_i \in Q, a_i \in L$ and $s_{j+1} \in \delta(s_j, l_j)$., from the initial state $s_0 = q_I$ to an acceptance state $s_n \in F$ such that the concatenation of the input (resp. output) part of the transition labels (dropping the θs) is w (resp. w'). In principle, there may be more than one path for a given input–output pair (w, w'): this should be avoided, and is partially, by determinization. But in this construction it is natural to have a finite number of valid outputs $\tau(w) = \{w'_1, w'_2, \ldots\}$ for a given input w.

Input in a LT is processed by maintaining a list of alive state–output pairs (ASOPs) which is updated after reading each input symbol. Before reading any input, this set is initialized to the null-input closure of $(q_I, (\epsilon, \epsilon))$, that is, $\mathcal{V}^{[0]} = \{(q, z) : q \in \delta^*(q_I, (\epsilon, z))\}$, where δ^* is the obvious extension of δ to input-output string pairs. After reading symbol $\sigma[t]$, a set $\mathcal{V}^{[t]}$ is built from $\mathcal{V}^{[t-1]}$ as follows:

$$\mathcal{V}^{[t]} = \{(q, z\gamma) : \quad q \in \delta^*(q', (\sigma[t], \gamma)) \\ \wedge\, (q', z) \in \mathcal{V}^{[t-1]}\}. \qquad (6)$$

After reading the whole word, $\tau(w) = \{z : (q, z) \in \mathcal{V}^{[|w|]} \wedge q \in F\}$. LTs may easily be determinized and minimized with respect to L by using customary algorithms (Hopcroft and Ullman (1979), p. 19–25, p. 68), since they are isomorphic to finite automata.

6 Results

This section describes a quantitative comparison between LTs which are deterministic with respect input–output pairs (L-deterministic LTs or L-DLTs) and earliest deterministic p-subsequential (finite-state) transducers (EDpSSTs), using realistic MDs for the morphological analysis of representative text corpora. To perform the comparison, EDpSSTs are converted into L-DLTs which are deterministic with respect to their input, except for the fact that each of the tails in non-null $\psi(q)$ sets is turned into null-input paths (which lead to input indeterminism in case that $|\psi(q)| > 1$); these last transducers are called Σ-quasideterministic LTs or Σ-QDLTs (construction detailed below).

Dict.	Corpus size	Ambig. rate
d_{500}	224 007	1.6(0.5)
d_{1000}	250 390	1.6(0.5)
d_{2000}	277 004	1.5(0.5)
d_{4000}	331 443	1.5(0.5)
d_{6000}	390 281	1.5(0.5)
d_{8000}	838 759	1.2(0.4)
d_{12000}	1 040 156	1.2(0.4)
d_{15000}	1 241 661	1.2(0.4)
d_{16500}	1 260 273	1.2(0.5)

Table 2: Average number of LFs per SF (ambiguity rate) for a series of inclusive dictionaries

The Spanish corpus used contains about two millon words, mainly from newspaper text. For each possible linguist-aligned MD, the corpus is filtered so that unknown words are removed. In the following, the names of dictionaries refer to the number of *lemmas* they contain. Each AMD contains the previous one in the list plus new pairs.

Table 2 shows the average lexical ambiguity of Spanish, computed as the number of LFs assigned to each SF, averaged on the corpus (standard deviations shown in parentheses). Ambiguity manifests itself in EDpSSTs as the cardinal of tail sets $\psi(q)$.

Next, the results of a comparison between the L-DLTs and the Σ-QDLTs deriving from these dictionaries has been performed. Σ-QDLTs are obtained from EDpSSTs as follows:

- Expand transitions $\lambda(q, \sigma) = \gamma_1\gamma_2 \ldots \gamma_n$ with $n > 1$, into a state path with transitions labeled $(\sigma, \gamma_1), (\theta, \gamma_2), \ldots, (\theta, \gamma_n)$.
- For each state q and for each tail $\gamma_1\gamma_2 \ldots \gamma_n \in \psi(q)$, build a state path starting at q and ending in a single acceptance state q_F with transitions labeled $(\theta, \gamma_1), (\theta, \gamma_2), \ldots (\theta, \gamma_n)$. This is the only source of input nondeterminism.
- Minimize the resulting Σ-QDLT with respect to L. This is necessary to merge equivalent states resulting from the previous expansion.

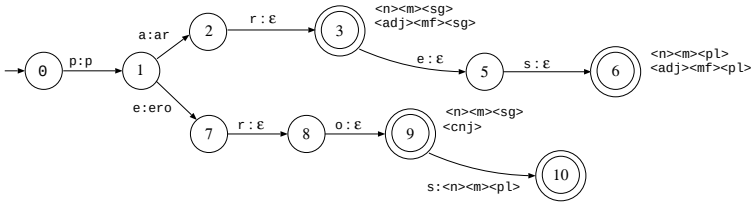

Figure 3: A minimal p-subsequential transducer.

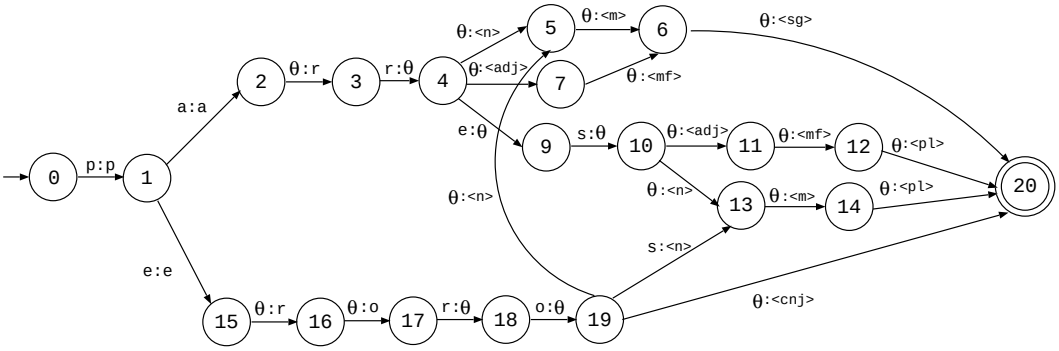

Figure 4: The minimal Σ-QDLT resulting from expanding the p-subsequential transducer in fig 3 and then minimizing it.

This transformation is performed in order to make a fair comparison between both strategies which is based on the same finite-state architecture. For example, expansion and minimization of the minimal p-subsequential transducer in fig. 3 results in the Σ-QDLT shown in fig. 4.

Table 3 shows the results obtained when turning each MD into a L-DLT and a Σ-QDLT. As may be seen in fig. 5, the size (number of states) of L-DLT is about 2.5 times smaller than that of the corresponding Σ-QDLT. Table 3 shows a comparison between the number of alive state-output pairs (ANASOPs) or nondeterminism rate for each MD, averaged on the corresponding corpus (standard deviation in parentheses). Results show that the average nondeterminism rate of L-DLT is not much larger than that of Σ-QDLT; in this last case, the only source of nondeterminism are the p-subsequential tails. Indeed, the nondeterminism rate observed is of the order of the lexical ambiguity of the texts themselves. In summary, the sacrifice in time efficiency resulting from the use of L-DLT instead of Σ-QDLT appears small in comparison to the large improvement in space efficiency. Note that these results may be interpreted as evaluating the effect of using or dropping the alignment in an AMD and suggest that alignments may lead to very compact transducers.

Dictionary	ANASOPs (Σ-QDLT)	ANASOPs (L-DLT)
d_{500}	1.3(0.6)	1.9(1.0)
d_{1000}	1.3(0.6)	1.8(0.9)
d_{2000}	1.3(0.6)	1.7(0.9)
d_{4000}	1.3(0.5)	1.6(0.9)
d_{6000}	1.3(0.5)	1.6(0.9)
d_{8000}	1.2(0.5)	1.5(0.8)
d_{12000}	1.3(0.5)	1.6(0.9)
d_{15000}	1.3(0.5)	1.5(0.9)
d_{16500}	1.3(0.5)	1.5(0.9)

Table 3: Non-determinism (expressed as the average number of alive states-output pairs or ANASOPs) as a function of the size of the MD for Σ-QDLTs and L-DLTs.

To consider the amount of output delayed to the end of input, note that, for MD d_{16500}, the average number of characters written per character read by a Σ-QDLT is 0.7 (standard deviation 1.2) and that the average number of characters delayed to the end of input is 2.2 (standard deviation 2.4).

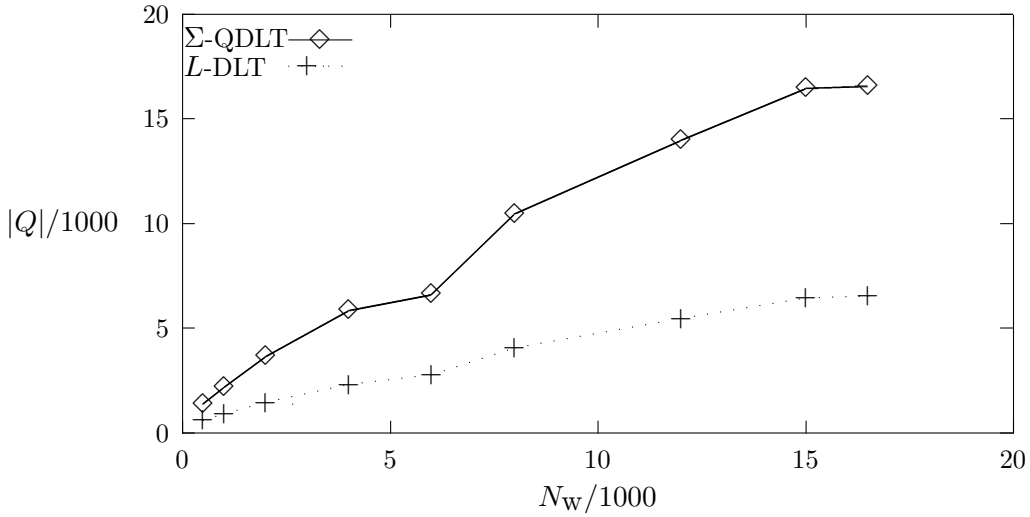

Figure 5: Number of states $|Q|$ in L-DLT and Σ-QDLT as a function of the number of words N_w in the MD.

7 Concluding remarks

We show how nondeterministic letter transducers may be a viable alternative to p-subsequential transducers (Mohri, 1997) when it comes to implement morphological analysers and generators for a natural language application: they are very compact while showing a low rate of indeterminism (i.e., keep in average less than two alive states during processing). Nondeterministic letter transducers are currently in use in the MT system interNOSTRUM (Canals-Marote et al., 2001).

References

Canals-Marote, R., A. Esteve-Guillen, A. Garrido-Alenda, M. Guardiola-Savall, A. Iturraspe-Bellver, S. Monserrat-Buendia, S. Ortiz-Rojas, H. Pastor-Pina, P.M. Perez-Antón, and M.L. Forcada. 2001. El sistema de traducción automática castellano-catalán interNOSTRUM. *Procesamiento del Lenguaje Natural*, 27:151–156. XVII Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural, Jaén, Spain, 12-14.09.2001.

Garrido, A., A. Iturraspe, S. Montserrat, H. Pastor, and M. L. Forcada. 1999. A compiler for morphological analysers and generators based on finite-state transducers. *Procesamiento del Lenguaje Natural*, (25):93–98.

Hopcroft, J. E. and J. D. Ullman. 1979. *Introduction to automata theory, languages, and computation.* Addison-Wesley, Reading, MA.

Mohri, Mehryar. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Oncina, J., P. García, and E. Vidal. 1993. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458.

Roche, E. and Y. Schabes. 1997. Introduction. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing.* MIT Press, Cambridge, Mass., pages 1–65.

Wagner, Robert A. and Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173.