Computational Linguistics

MORPHOLOGY – TRANSDUCERS

Martin Rajman

Martin.Rajman@epfl.ch

and

Jean-Cédric Chappelier

Jean-Cedric.Chappelier@epfl.ch

Artificial Intelligence Laboratory

Objectives of this lecture

➥ Present morphology, important part of NLP

➥ Introduce transducers, tools for computational morphology

Contents

Morphology

Study of the internal structure and the variability of the words in a language:

- verbs conjugation
- plurals
- nominalization (enjoy $\longrightarrow$ enjoyment)

➜ inflectional morphology: preserves the grammatical category

give given gave gives ...

➜ derivational morphology: change in category

process processing processable processor processabilty

Morphology (2)

Interest: use *a priori* knowledge about word structure to decompose it into morphemes and produce additional syntactic and semantic information (on the current word)

processable $\longrightarrow$	process-	-able	☞ 2 morphemes
meaning:	process	possible	
role:	root	suffix	
semantic information:	main	less	

The importance and complexity of morphology vary from language to language

Some information represented at the morphological level in English may be represented differently in other languages (and vice-versa). The paradigmatic/syntagmatic repartition changes from one language to another

Example in Chinese: ate $\longrightarrow$ expressed as ”eat yesterday”

Stems – Affixes

Words are decomposed into morphemes: roots (or stems) and affixes.

There are several kinds of **affixes**:

1. prefixes: **in**- -credible
2. suffixes: incred- -**ible**
3. infixes:

   Example in Tagalog (Philippines):

   hingi (to borrow) ⟶ h**um**ingi (agent of the action)

   In slang English! ⟶ "fucking" in the middle of a word Man-fucking-hattan
4. circumfixes:

   Example in German:

   sagen (to say) ⟶ **ge**sag**t** (said)

Stems – Affixes (2)

several affixes may be combined:

examples in Turkish where you can have up to 10 (!) affixes.

uygarlaştıramadıklarimizdanmışsınızcasına

uygar	laş	tır	ama	dık	lar	imiz	dan	mış	sınız	casına
civilized	+Bec	+Caus	+NegAble	+PPart	+Pl	+P1Pl	+Abl	+Past	+2Pl	+AsIf

as if you are among those whom we could not cause to become civilized

When only prefixes and suffixes are involved: **concatenative** morphology

Some languages are not concatenative:

- infixes
- pattern-based morphology

Example of semitic languages

Pattern-based morphology

In Hebrew, the verb morphology is based on the association of

- a root, often made of 3 consonents, which indicates the main meaning,
- and a vocalic structure (insertion ov vowels) that refines the meaning.

Example: LMD (learn or teach)

LAMAD $\longrightarrow$ he was learning

LUMAD $\longrightarrow$ he was taught

Computational Morphology

Let us consider flexional morphology, for instance for verbs and nouns

Noun flexions: plural

General rule: +s

but several exceptions (e.g. foxes, mice)

Verb flexions: conjugations

- tense, mode
- regular/irregular

☞ How to handle flexions (comptutationaly)?

Computational Morphology

Example:surface form: is

canonical representation at the lexicon level (formalization): be+3+s+Ind+Pres

The objective of computational morphology tools is precisely to go from one to the other:

- Analysis: Find the canonical representation corresponding to the surface form
- Generation: Produce the surface form described by the canonical representation

Challenge: have a "good" implementation of these two transformations

Tools: associations of strings $\longrightarrow$ transducers

String associations

$$(X_1, X'_1)$$
$$\vdots$$
$$(X_n, X'_n)$$

(eaten, eat)

(processed, process)

$\vdots$

(thought, think)

Easy situation: $\forall i, \quad |X_i| = |X'_i|$ Example: (abc, ABC)

$\Rightarrow$ represented as a sequence of **character** transductions

(abc, ABC) = (a,A)(b,B)(c,C)

☞strings on a new alphabet: strings of character couples

Not so easy: If $\exists i, \quad |X_i| \neq |X'_i| \Rightarrow$ requires the introduction of empty string ε

Example: (ab, ABC) $\simeq$ (εab, ABC) = (ε,A)(a,B)(b,C)

Dealing with ε

Where to put the ε?

Example:(ab,ABC) $\simeq$ (εab, ABC)

but also (ab,ABC) $\simeq$ (aεb, ABC)

or (ab,ABC) $\simeq$ (abε, ABC)

General case:

$$\binom{n}{m} \qquad (\text{with } m < n)$$

Hard problem in general $\longrightarrow$ need for a convention

Transducer (definition)

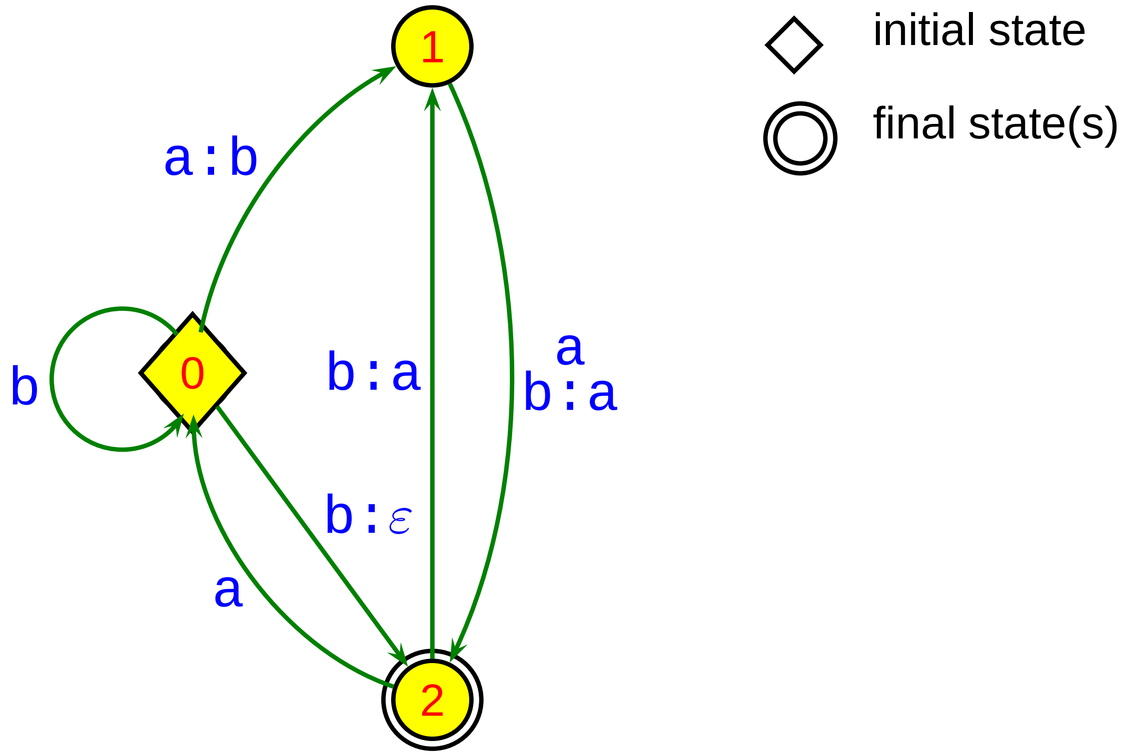
Let Σ_1 and Σ_2 be two enumerable sets (alphabets), and

$$\Sigma = \Big((\Sigma_1 \cup \{\varepsilon\}) \times (\Sigma_2 \cup \{\varepsilon\}) \Big) \setminus \{(\varepsilon, \varepsilon)\}$$

A transducer is a DFSA on Σ

Σ_1 : ”left” language	Σ_2 : ”right” language
: upper language	: lower language
: input language	: output language

Example



Some transductions: (bb,b) [0,0,2] (ababb,baab) [0,1,2,0,0,2]

Different usages of a transducer

❶ association checking $(\text{abba}, \text{baaa}) \in \Sigma^*$?

❷ Generation: $\text{string}_1 \longrightarrow \text{string}_2$ $\text{bbab} \longrightarrow$?

❸ Analysis: $\text{string}_2 \longrightarrow \text{string}_1$? $\longrightarrow \text{ba}$

❶: easy: ($=$ FSA: nothing special)

What about ❷ and ❸ ?

Transduction

Walk through the FSA following one or the other element of the couple (projections)

! ☞ not deterministic in general!

The fact that a transducer is a deterministic (couple-)FSA does not at all imply that the automaton resulting from one projection or the other is also deterministic!

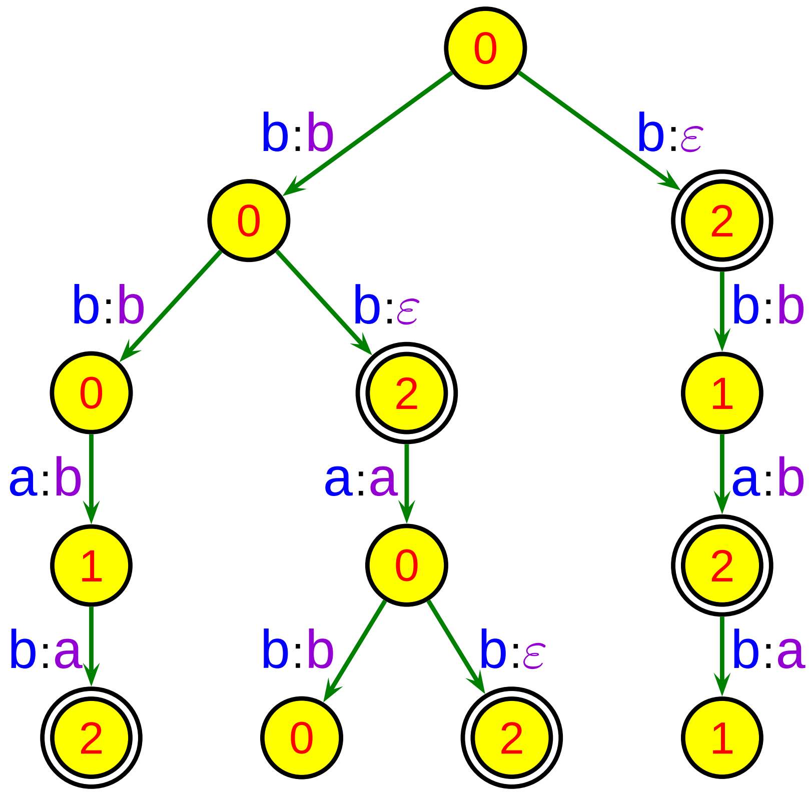
$$\left.\begin{array}{l}\text{non-deterministic evaluation}\\[1em]\text{backtracking on ”wrong” solutions}\end{array}\right\}\Rightarrow \text{The projection is not constant time (in general)}$$

When a transducer is deterministic with respect to one projection or the other, it is called a **sequential transducer**

A transducer in not sequential in general. In particular if one language or the other (upper or lower) is not finite, it is not sure that a sequential transducer can be produced.

Transduction (2)

Example: bbab $\longrightarrow$?

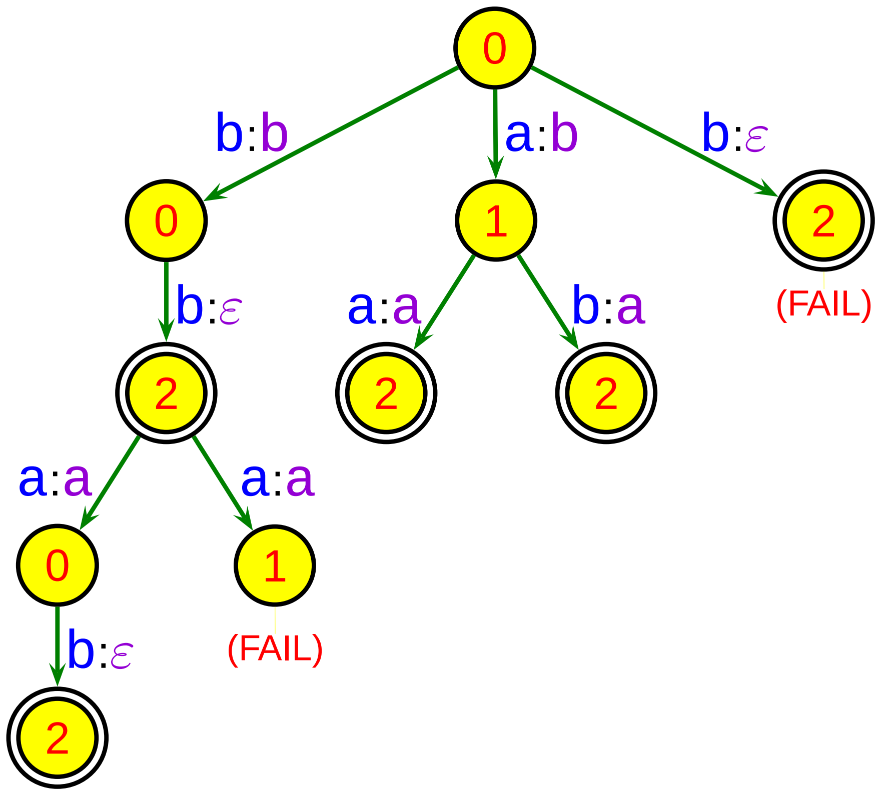


bbab $\longrightarrow$ bbba

bbab $\longrightarrow$ ba

Transduction (3)

Example: ? $\longrightarrow$ ba



aa $\longrightarrow$ ba   ab $\longrightarrow$ ba   bbab $\longrightarrow$ ba

Operations and Regular Expressions on Transducers

- All FSA regular expressions: concatenation, or, Kleene closure (`*`), ...

  Example:(concatenation) "`a:b c:a`" recognizes `ac` and produces `ba`

- cross-product of regular languages: $E_1 \otimes E_2$ recognizes $L_1 \times L_2$

  example: `a+` $\otimes$ `b+` $\longrightarrow (a^n, b^m) \qquad \forall\, n \geq 1, m \geq 1$!! this is $\neq (a \otimes b)$+

- Composition of transducers: $T = T_1 \circ T_2$

$$(X_1, X_2) \in T \iff \exists Y : (X_1, Y) \in T_1 \text{ and } (Y, X_2) \in T_2$$

- Reduction: extraction of the upper or the lower FSA

(Other) examples of applications

(morphology)

- ★ text-to-speech (grapheme to phoneme transduction)
- ★ specific lexicon representation (composition of some access and inverse fonctions)
- ★ filters (remove/add/modify marks; e.g. HTML)
- ★ text segmentation

Computational morphology using transducers

Use of composition:

- → Identification of a paradigm (T_1)
- → Implementation of this paradim (T_2)
- → Exception handling (T_3)

Example: input: chat+NP, fox+NP, ... ("+NP" means "noun plural")

T_1: ([a-z]+)(\+NP $\otimes$ \+1) paradigm identification: plural nouns (trivial here: only one paradigm (+1))

T_2: ([a-z]+)(\+1 $\otimes$ \+Xs) plural inflection of nouns (regular part)

T_3: ([a-z]+)(h\+Xs $\otimes$ hes | x\+Xs $\otimes$ xen | ... | [^hx...](\+X$\otimes\varepsilon$)s) correction of exceptions

$T_1 \circ T_2 \circ T_3$: plural for nouns

Computational morphology using transducers (2)

Detailed example on the plural of nouns:

general case: add a terminal ’s’

cat+NP $\longrightarrow$ cats, dog+NP $\longrightarrow$ dogs, ...

Exceptions (several kind):

- fly flies
- fox foxes, but ox oxen!
- ..

Method: find all the paradigms (linguists’ role) and implement a transducer for each of them

☞ add the paradigm identification in the lexical description

Keypoints

- Flexional and derivational morphologies, their roles
- Main functions of transducers: association checking, generation and analysis
- Deterministic and not deterministic nature of transduction

References

E. Roche, Y. Schabes, *Finite-state Language Processing*, pp. 14-63, 67-96, A Bradforf Book, 1997.