Introduction to Computational Linguistics

PD Dr. Frank Richter

fr@sfs.uni-tuebingen.de.

Seminar für Sprachwissenschaft

Eberhard-Karls-Universität Tübingen

Germany

Replacement Operators

- Unconditional obligatory replacement:

  A $\rightarrow$ B $=_{def}$ [[$\sim$\$[A - [ ]] [A .x. B]]* $\sim$\$[A - [ ]]]

- Unconditional optional replacement:

  A ($\rightarrow$) B $=_{def}$ [[$\sim$\$[A - [ ]] [A .x. A | A .x. B]]*
  $\sim$\$[A - [ ]]]

- Contextual obligatory replacement:

  A $\rightarrow$ B $\|$ L _ R

  meaning: “Replace A by B in the context L _ R.”

Non-determinism of *replace*

Example:	ab $\rightarrow$ ba | x
meaning:	"replace ab by ba or x non-deterministically"
Sample input:	abcdbaba
Outputs:	bacdbbaa,bacdbxa, xcdbbaa,xcdbxa

Non-determinism of *replace* (2)

Example: [a b | b | b a | a b a] $\rightarrow$ x

meaning: "replace ab or b or ba or aba by x"

Sample input: a ba aba a b a a b a

Outputs: x a axa a x x

Longest match, left-to-right replace

- For many applications, it is useful to define another version of replacement that in all such cases yields a unique outcome.
- The longest-match, left-to-right replace operator, @->, defined in Karttunen (1996), imposes a unique factorization on every input.
- The replacement sites are selected from left to right, not allowing any overlaps.
- If there are alternate candidate strings starting at the same location, only the longest one is replaced.

A Grammar for Date Expressions

1To9	=	[1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9]
0To9	=	[%0 | 1To9]
SP	=	[", "]
Day	=	[Monday | ... | Saturday | Sunday]
Month	=	[January | ... | November | December]
Date	=	[1To9 | [1 | 2] 0To9 | 3 [%0 | 1]]
Year	=	1To9 (0To9 (0To9 (0To9)))
DateExp	=	Day | (Day SP) Month " " Date (SP Year)

Marking Date Expressions

- A parser for date expressions can be compiled from the following simple regular expression:
  DateExp @-> %[... %]
- The above expression can be compiled into a finite-state transducer.
- @-> is replace operator which scans the input from left to right and follows a longest-match.
- Due to the longest match constraint, the transducer brackets only the maximal date expressions.
- The dots mean: identity with the upper string. The whole expression means: replace DateExp by DateExp surrounded by brackets.

Overgeneration Problem

- The grammar for date expressions accepts illegal dates.
- For example: it admits dates like “February 30, 2006”
- More generally:
  - If a grammar admits strings that should not be accepted by the grammar, the grammar is said to *overgenerate*.
  - If a grammar does not admit strings that should be accepted by the grammar, the grammar is said to *undergenerate*.

Tokenizing Date Expressions

Example:

Today is [Wednesday, August 28, 1996] because yesterday was [Tuesday] and it was [August 27] so tomorrow must be [Thursday, August 29] and not [August 30, 1996] as it says on the program.

Incremental Tokenization

input layer	one, two, and so on.
single word layer	one || , || two || , || and || so || on || . ||
multi-word layer	one || , || two || , || and so on || . ||

Advantages of Incremental Tokenization

- With finite-state transducers incremental tokenization is implemented by the composition operator for transducers.
- Separation of grammar specification and program code: Each analysis level is specified in a well-defined language of regular expressions.
- Transducers for each layer can be stated independently of each other
- Regular expressions can be compiled automatically into (composed) finite state transducers.

A Quick Guide to Morphology (1)

- Morphology studies the internal structure of words.
- The building blocks are called morphemes. One distinguishes between free and bound morphemes.
  - Free morphemes are those which can stand alone as words.
  - Bound morphemes are those that always have to attach to other morphemes.

A Simple Morphological Typology

- Isolating languages: no bound morphemes

A Simple Morphological Typology

- Isolating languages: no bound morphemes
- Agglutinative languages: all bound forms are affixes

A Simple Morphological Typology

- Isolating languages: no bound morphemes
- Agglutinative languages: all bound forms are affixes
- Inflectional languages: distinct features merged into single bound form; same underlying feature expressed differently, depending on paradigm

A Simple Morphological Typology

- Isolating languages: no bound morphemes
- Agglutinative languages: all bound forms are affixes
- Inflectional languages: distinct features merged into single bound form; same underlying feature expressed differently, depending on paradigm
- Polysynthetic languages: more structural information expressed morphologically

A Quick Guide to Morphology (2)

Linguists commonly distinguish three types of morphological processes:

- Inflectional morphology: refers to the class of bound morphemes that do not change word class.
- Derivational morphology: refers to the class of bound morphemes that do change word class.
- Compounding: a morphologically complex word can be constructed out of two or more free morphemes.

Inflectional Morphemes

- Bound morphemes which do not change part of speech, e.g. *big* and *bigger* are both adjectives.
- Typically indicate syntactic or semantic relations between different words in a sentence, e.g. the English present tense morpheme *-s* in *waits* shows agreement with the subject of the verb.
- Typically occur with all members of some large class of morphemes, e.g. the pural morpheme *-s* occurs with most nouns.
- Typically occur at the margins of words as affixes (prefix, suffix, circumfix)

Derivational Morphemes

- Bound morphemes which change part of speech, e.g. *-ment* forms nouns, such as *judgment*, from verbs such as *judge*.
- Typically indicate semantic relations within the word, e.g. the morpheme *-ful* in *painful* has no particular connection with any other morpheme beyond the word *painful*.
- Typically occur with only some members of a class of morphemes, e.g. the suffix *-hood* occurs with just a few nouns such as *brother*, *neighbor*, and *knight*, but not with many others, e.g. *friend*, *daughter*, *candle*, etc.
- Typically occur before inflectional suffixes, e.g. in *interpretierbare (Antwort)* the derivational suffix *bar* before the inflectional suffix *-e*.

Compounding

- A compound is a word formed by the combination of two independent words.
- The parts of the compound can be free morphemes, derived words, or other compounds in nearly any combination:
  - *girlfriend* (two independent morphemes),
  - *looking glass* (derived word + free morpheme),
  - *life insurance salesman* (compound + free morpheme).

Morphology: The Naive Solution

The simplest, but for most cases naive solution:

- Compile a full-form lexicon which lists all possible word forms together with their morphological analyses.
- If a given word has only one morphological analysis, the full-form lexicon stores exactly one reading.
- If a given word has more than one morphological analysis, the full-form lexicon stores all possible readings separately.

Morphological Analysis: Lemmatization

- Lemmatization refers to the process of relating individual word forms to their citation form (lemma) by means of morphological analysis.
- Lemmatization provides a means to distinguish between the total number of word tokens and distinct lemmata that occur in a corpus.
- Lemmatization is indispensible for highly inflectional languages which have a large number of distinct word forms for a given lemma.

Examples from English (1)

Input: *spies*

Analysis:

spies spy+Noun+Pl

spies spy+Verb+Pres+3sg

Input: *travelling*

Analysis:

travelling travel+Verb+Prog

travelling travelling+Adj

travelling travelling+Noun+Sg

Examples from English (2)

Input: *foxes*

Analysis:

foxes fox+Noun+Pl

foxes fox+Verb+Pres+3s

Input: *moved*

Analysis:

moved move+Verb+PastBoth+123SP

moved moved+Adj

Examples from German (1)

Input: *Staubecken*

Analysis:

1. Stau+Noun+Common+Masc+Sg#
   Becken+Noun+Common+Neut+Sg+NomAccDat
2. Stau+Noun+Common+Masc+Sg#
   Becken+Noun+Common+Neut+Pl+NomAccDatGen
3. Staub+Noun+Common+Masc+Sg#
   Ecke+Noun+Common+Fem+Pl+NomAccDatGen

Examples from German (2)

```
<form>hat</form> <ENGLISH>has</ENGLISH>
<lemma wkl=VER typ=AUX pers=3 num=SIN modtemp=PRÄ>haben</lemma>
<lemma wkl=VER pers=3 num=SIN modtemp=PRÄ konj=NON>haben</lemma>

<form>man</form> <ENGLISH>one</ENGLISH>
<lemma wkl=PRO typ=IND kas=NOM num=SIN gen=ALG stellung=STV>man</lemma>

<form>mir</form> <ENGLISH>me</ENGLISH>
<lemma wkl=PRO typ=REF kas=DAT num=SIN gen=ALG pers=1>sich</lemma>
<lemma wkl=PRO typ=PER kas=DAT num=SIN gen=ALG pers=1>ich</lemma>

<form>gesagt</form> <ENGLISH>told</ENGLISH>
<lemma wkl=VER form=PA2 konj=SFT>sagen</lemma>
<lemma wkl=PA2 gebrauch=PRD komp=GRU>gesagt</lemma>

<form>,</form>
<lemma wkl=SZK>,</lemma>

<form>ja</form> <ENGLISH>right</ENGLISH>
<lemma wkl=ADV typ=MOD>ja</lemma>
```

Stemmers

- Stemmers are the simplest type of morphological analyzer.
- One of the main advantages of stemmers is that they do not require a lexicon.
- The function of a stemmer is to remove the most common morphological and inflexional endings from words.
- Its main use is as part of a term normalisation process that is usually done when setting up Information Retrieval systems.

Finite-State Morphology

- Basic Idea: Encode morphological analysis and generation as composition of finite-state transducers.
- Resources needed:
  - Morpho-syntactic lexicon that specifies which combinations of free and bound morphemes are grammatical
  - Context-sensitive replacement rules for spelling alternations

2-level Rules: Restriction Operators

Two-level morphology employs a set of particular restriction operators:

=>	the correspondence only occurs in the environment
<=	the correspondence always occurs in the environment
<=>	the correspondence always and only occurs in the environment
/<=	the correspondence never occurs in the environment

2-level Rules: Restriction Operators

Two-level morphology employs a set of particular restriction operators:

=>	the correspondence only occurs in the environment
<=	the correspondence always occurs in the environment
<=>	the correspondence always and only occurs in the environment
/<=	the correspondence never occurs in the environment

Idea: Rules with restriction operators function as constraints on the mapping between lexcial and surface form of morphs.

Toy Rules for English (1)

i:y-spelling

```
die+ing tie+ing
dy00ing ty00ing
```

Rule: i:y <= _ e:? +:0 i

Elision

```
agree+ed dye+ed hoe+ed hoe+ing
agre00ed dy00ed ho00ed hoe0ing
```

Rule: e:0 <= C { V, y } _ +:? e:e

with V = { a e i o u } and
C = { b c d f g h j k l m n p q r s t v w x y z sh ch }

Toy Rules for English (2)

Epenthesis

```
fox+s kiss+s church+s spy+s
foxes kisses churches spies
```

Rule: +:e <=> { C_{sib}, y:i, o:o } _ s

with C_{sib} = { s x z sh ch }