Textanalyse und Korpuswerkzeuge

Katja Pöllmann

Gliederung

1. Überblick zur Sprachverarbeitung
2. Korpuswerkzeuge
3. Wörter zählen
4. Wörter, Wortarten und Morphologie
5. Literaturangabe

1. Überblick zur Sprachverarbeitung

Computerlinguistik:

- Schnittstelle zwischen **Sprachwissenschaft** und **Informatik**
- CL untersucht, wie natürliche Sprache mit Hilfe des Computers algorithmisch verarbeitet werden kann.

Überblick zur Sprachverarbeitung

Ziele:

- aus linguistischer Sicht:

  Unterstützung der **sprachwissenschaftlichen Forschung** durch den Einsatz von Computern

  → z. B. durch die automatische **Analyse** großer **Korpora**, um sprachliche Phänomene zu untersuchen oder die Gültigkeit von Theorien zu prüfen

- aus industrieller Sicht:

  Entwicklung sprachverarbeitender Systeme, z. B. für

  - maschinelle Übersetzung
  - automatisches Textverständnis
  - natürlichsprachige Interaktion von Mensch und Maschine

Überblick zur Sprachverarbeitung

kommerzielle **Anwendungen:**

- Rechtschreib- und Grammatikprüfung: MS Word
- Indizierung von Texten und Informationsabfrage im Internet: Google, Yahoo
- Maschinelle Übersetzung: SYSTRAN (englisch-russisch)
- Diktat von Briefen oder Berichten: IBM ViaVoice

  → beruhen auf Spracherkennung, transkribieren Diktate automatisch in geschriebenen Text
- Sprachsteuerung von Haushaltsgeräten, um Bedienung zu erleichtern: MS Persona

Überblick zur Sprachverarbeitung

Sprachverarbeitung betrifft sämtliche Teilgebiete der **Linguistik:**

- **Phonetik:** Erzeugung und Wahrnehmung von Lauten (= Phoneme)
- **Morphologie:** Struktur der "Wörter" (= Morpheme)
- **Syntax:** Wortstellung im Satz, Funktionen von Wörtern oder Satzteilen
- **Semantik:** Bedeutung von Wörtern und Sätzen (allgemein)
- **Pragmatik:** kontextuelle Interpretation von Wörtern und Sätzen (spezifisch)
- **Dialog:** linguistische Interaktionen zum Informations-austausch

Überblick zur Sprachverarbeitung

Probleme der Verarbeitung natürlicher Sprache:

- selten 100%ige Genauigkeit
- 2 allgegenwärtige Hindernisse (z.B. bei Spracherkennung, Wortartenbestimmung, Satzanalyse):
  - Mehrdeutigkeit
  - Fehlen eines perfekten Modells
- Lexikalische Mehrdeutigkeit

1. **note** (*noun*) short piece of writing;
2. **note** (*noun*) a single sound at a particular level;
3. **note** (*noun*) a piece of paper money;
4. **note** (*verb*) to take notice of;
5. **note** (*noun*) of note: of importance.

Überblick zur Sprachverarbeitung

Modelle und ihre **Implementierung**

- Gute Modelle sind schwierig zu konstruieren.
- Sprache ist eng mit menschlichem Denken und Verstehen verbunden

  → Studien zu menschlichem Geist mit einbeziehen

  → komplex
- Viele potentielle Theorien erfordern massive Rechenleistung

  → Prozessoren und Speicher, die in der Lage sind, die Implementierung von komplexen Modellen mit umfangreichen Wörterbüchern, Korpora und Parsern zu unterstützen, gibt es erst seit kurzem

Überblick zur Sprachverarbeitung

- Verbesserung von Modellen in den letzten 10 Jahren
- liefern brauchbare Ergebnisse, auch wenn sie selten (nie?) perfekt sind
- Die meisten Tools verwenden eine begrenzte Anzahl an **Techniken**, wie
  - endliche Zustandsautomaten
  - reguläre Ausdrücke
  - umschreibende Regeln (rewriting rules)
  - Logik
  - Statistiken und
  - maschinelles Lernen

2. Korpuswerkzeuge

Korpus (Pl. Korpora)

- Sammlung von Texten oder Sprachaufnahmen

Anwendungen:

- **Lexikographie:** Erstellung von Wörterbüchern → Bedeutung von Wörtern anhand echter Belege, repräsentativer Beispiele veranschaulichen
- **Konkordanz:** Textauszug; eine bestimmte Anzahl von Zeichen/Wörtern, die dem Suchbegriff vorangehen und folgen

Korpuswerkzeuge

Language	Concordances
English	s beginning of miracles did Je
	n they saw the miracles which
	n can do these miracles that t
	ain the second miracle that Je
	e they saw his miracles which
French	le premier des miracles que fi
	i dirent: Quel miracle nous mo
	om, voyant les miracles qu'il
	peut faire ces miracles que tu
	s ne voyez des miracles et des

Korpuswerkzeuge

- **Kollokation:** charakteristische Wendungen, Beschreibung häufiger Wortpaare

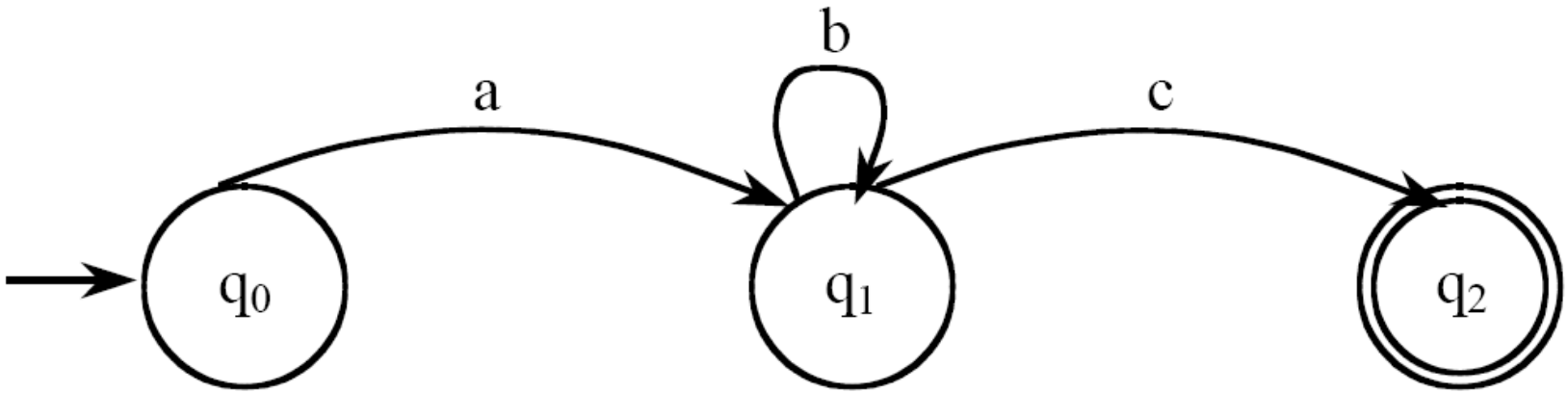
	English	French	German
You say	*Strong tea* *Powerful computer*	*Thé fort* *Ordinateur puissant*	*Kräftiger Tee* *Starker Computer*
You don't say	*Strong computer* *Powerful tea*	*Thé puissant* *Ordinateur fort*	*Starker Tee* *Kräftiger Computer*

- Korpora als Wissensquelle für Linguisten (Überprüfung von Theorien anhand reeller Daten)
- Korpora dienen auch zur Entwicklung von Wortart-Taggern oder Parsern

Korpuswerkzeuge

Endliche Zustandsautomaten

- anpassungsfähiges Werkzeug, um Texte zu durchsuchen und zu verarbeiten
- ein FSA (finite-state automata) akzeptiert (→ erkennen) und erzeugt strings (Zeichenketten → Wörter)



akzeptiert strings: ac, abc, abbbbbc, etc.

Korpuswerkzeuge

mathematische Definition:

- Q ist eine endliche Menge von Zuständen.
- Σ ist eine endliche Menge von Symbolen oder Zeichen: das Eingabe-Alphabet.
- q_0 ist der Startzustand, Element aus Q.
- F ist eine endliche Menge von Endzuständen, Teilmenge von Q.
- δ eine Übergangsfunktion $Q \times \Sigma \rightarrow Q$, wobei $\delta(q, i)$ den Zustand zurückgibt, zu dem sich der Automat begibt, wenn er sich in Zustand q befindet und das Eingabesymbol i verarbeitet.

Korpuswerkzeuge

Regular Expressions (regex, regulärer Ausdruck)

- sind äquivalent zu endlichen Zustandsautomaten
- setzen sich aus **literal** characters (Buchstaben, Zahlen, Interpunktionszeichen, spaces) und **metacharacters** zusammen
- sind **case-sensitive**, unterscheiden also zwischen Groß- und Kleinbuchstaben

Pattern	String
`regular`	*A section on regular expressions*
`the`	*The book of the life*

Korpuswerkzeuge

- Befehl für unseren Zustandsautomaten:

```
grep 'ab*c' myFile1 myFile2
```

Metachars	Descriptions	Examples
*	Matches any number of occurrences of the previous characters – zero or more	**ac*e** matches patterns **ae**, **ace**, **acce**, **accce**, etc. as in "The aerial acceleration alerted the ace pilot"
?	Matches at most one occurrence of the previous characters – zero or one	**ac?e** matches **ae** and **ace** as in "The aerial acceleration alerted the ace pilot"
+	Matches one or more occurrences of the previous characters	**ac+e** matches **ace**, **acce**, **accce**, etc. as in as in "The aerial acceleration alerted the ace pilot"
{n}	Matches exactly n occurrences of the previous characters	**ac{2}e** matches **acce** "The aerial acceleration alerted the ace pilot"

Korpuswerkzeuge

`{n,}`	Matches n or more occurrences of the previous characters	`ac{2,}e` matches `acce`, `accce`, etc.
`{n,m}`	Matches from n to m occurrences of the previous characters	`ac{2,4}e` matches `acce`, `accce`, and `acccce`.
`.`	Matches one occurrence of any characters of the alphabet except the new line character	`a.e` matches `aae`, `aAe`, `abe`, `aBe`, `a1e`, etc. as in "The aerial acceleration alerted the ace pilot"
`.*`	Matches any string of characters and until it encounters a new line character	

- Suche nach metacharacter (z.B. Interpunktionszeichen): \ davor (metacharacter → literal character)

Korpuswerkzeuge

Klassen von Zeichen

`[abc]`	einmaliges Vorkommen von entweder **a**, **b** oder **c**
`[ABCDEFGHIJKLMNOPQRSTUVWXYZ]`	ein Großbuchstabe, der keinen Akzent trägt
`[0123456789]+\.[0123456789]+`	Dezimalzahl
`[Cc]omputer [Ss]cience`	*Computer Science, Computer science, computer Science, computer science*
`[^…]`	Komplement einer Klasse; irgendein Zeichen, das NICHT in der Liste enthalten ist
`[^ABCD]*`	string, der nicht **A, B, C** oder **D** enthält

Korpuswerkzeuge

Vereinigung und Boolesche Operatoren

- Kombination von regulären Ausdrücken und Operatoren
- Prioritätenordnung der 3 Hauptoperatoren:
  - **Abschluss** (Kleene star, *) und andere Wiederholungsoperatoren (höchste Priorität)
  - **Verknüpfung** (implizit in strings wie *computer*), Zeilen- und Wortgrenzen
  - **Vereinigung** (a|b → entweder a oder b) (niedrigste Priorität)

- a|bc → a oder bc (Verknüpfung geht vor)
- (a|b)c → ac oder bc
- abc* → ab, abc, abccccc (Wh vor Verknüpfung)
- (abc)* → abc, abcabc, usw

Korpuswerkzeuge

Programmieren mit regulären Ausdrücken

- für etwas kompliziertere Textverarbeitung, wie Wörter ersetzen oder zählen, braucht man eine vollwertige Programmiersprache, z.B. Perl

Match

```
while ($line = <>) {
    if ($line =~ m/ab*c/) {
        print $line;
    }
}
```

- **`$line = <>`** weist der Variable **`$line`** die aktuelle Eingabezeile zu
- =~ Operator weist Perl an, die **`$line`** Variable zu durchsuchen, gibt **`true`** oder **`false`** zurück
- **`m/…/`** : match operator; grenzt den regulären Ausdruck ab

Korpuswerkzeuge

Substitute

```
while ($line = <>) {
    if ($line =~ m/ab*c/) {
        print "Old: ", $line;
        $line =~ s/ab*c/ABC/g;
        print "New: ", $line;
    }
}
```

Translate

```
tr/ABC/abc/
$line =~ tr/A-Z/a-z/;
```

- substitution: **`s/regex/replacement/`**
- translation: **`tr/search_list/replacement_list/`**
- **`tr/ABC/abc`** **`AbCdEfGhIjK`** → **`abcdEfGhIjK`**

Korpuswerkzeuge

Modifikatoren der translate-Anweisung:

- **`d` löscht** alle Zeichen der Suchliste, die in der Ersetzungsliste nicht vorkommen
- **`c`** wandelt Zeichen um, die zum **Komplement** der Suchliste gehören
- **`s` reduziert** (squeeze, squash) Zeichenfolgen, die zu identischen Zeichen umgewandelt wurden, zu einem einzigen Zeichen

```
$line =~ tr/AEIOUaeiou//d;
$line =~ tr/AEIOUaeiou/$/cs;
```

Korpuswerkzeuge

Concatenate

```
while ($line = <>) {
    $text .= $line;
}
print $text;
```

References

```
while ($line = <>) {
    while ($line =~ m/\$ *([0-9]+)\.?([0-9]*)/g) {
        print "Dollars: ", $1, " Cents: ", $2, "\n";
    }
}
```

- Perl erzeugt einen Puffer, wenn es auf eine linke Klammer trifft
- Rückverweis erfolt außerhalb der match expression mit **$<zahl>**

Arrays

```
@array = (1, 2, 3); #Array containing 1, 2, and 3
print $array[1]; #Prints 2
```

Korpuswerkzeuge

Konkordanzen in Perl

```
while ($line = <FILE>) {
    $text .= $line;
}
$string =~ s/ /\\s/g; # spaces match tabs and new lines
$text =~ s/\n/ /g; # new lines are replaced by spaces
while ($text =~ m/(.{0,$width}$string.{0,$width})/g ) {
# matches the pattern with 0..width to the right and left
    print "$1\n"; #$1 contains the match
}
```

Korpuswerkzeuge

Language	Concordances
English	s beginning of miracles did Je
	n they saw the miracles which
	n can do these miracles that t
	ain the second miracle that Je
	e they saw his miracles which
French	le premier des miracles que fi
	i dirent: Quel miracle nous mo
	om, voyant les miracles qu'il
	peut faire ces miracles que tu
	s ne voyez des miracles et des

Korpuswerkzeuge

Ungenaues string matching

- um eine Menge von ähnlichen strings zu erzeugen, wendet man eine Reihe von Operationen an, die einen **source string s** in einen **target string t** verwandeln
- Operationen:
  - kopieren (erzeugt gleiche strings)
  - ersetzen
  - einfügen (im target string)
  - löschen (im target string, d.h. Zeichen des source strings wird nicht kopiert)
  - vertauschen (kopiert 2 Zeichen des source strings und vertauscht sie im target string)

Korpuswerkzeuge

Typo	Correction	Source	Target	Position	Operation
acress	actress		t	2	deletion
acress	cress	a		0	insertion
acress	caress	ac	ca	0	transposition
acress	access	r	c	2	substitution
acress	across	e	o	3	substitution
acress	acres	s		4	insertion
acress	acres	s		5	insertion

- falsch geschriebenes Wort: *acress*
- typo = tyopgraphical error
- source: correction
- target: typo

3. Wörter zählen

- Wörter werden in einem spezifischen Kontext verwendet
- **Sprachmodell:** statistische Schätzung einer Wortfolge
- wurde ursprünglich für die Spracherkennung entwickelt
- Ein Sprachmodell ermöglicht es, das nächste Wort vorherzusagen, wenn eine Folge vorangehender Wörter gegeben ist:
  - *the writer of books, novels, poetry,* etc.
  - und nicht *the writer of hooks, nobles, poultry, …*

Wörter zählen

- **Token:**
  - eigentlich: Textelement → Wörter, Interpunktionszeichen, Zahlen, Abkürzungen, oder ähnliche string-Typen
  - bei uns: eine Folge von alphanumerischen Zeichen
- **Satz**: Folge von Token, die durch . : ; ! ? beendet wird
- **Tokenisierung:**
  - zerlegt einen Zeichenstrom in Token und Sätze
  - notwendiger Schritt für die morphologische und syntaktische Analyse
  - kann auch Formatierungsanweisungen wie XML-tags entfernen
- `[l, i, s, t, ' ', o, f, ' ', c, h, a, r, a, c, t, e, r, s]` →
  `[list, of, characters]`

Wörter zählen

Tokenisierung in Perl

```
$text = <>;
while ($line = <>) {
   $text .= $line;
}
$text =~ tr/a-zåàâäæçéèêëîïôößùûüÿA-
ZÅÀÂÄÆÇÉÈÊËÎÏÔÖÙÛÜ'-,.?!:;/\n/cs;
$text =~ s/([,.?!:;])/\n$1\n/g;
$text =~ s/\n+/\n/g;
print $text;
```

Wörter zählen

- **(Wort-)Typ:** verschiedenen Wörter eines Textes
- **Token:** alle Wörter oder Symbole

Sätze aus George Orwell's *Nineteen Eighty-Four:*

- *War is peace*
- *Freedom is slavery*
- *Ignorance is strength*

→ 9 Token und 7 Typen

- **Unigrams** sind einzelne Wörter
- **Bigrams:** Folge von zwei Wörtern
- **Trigrams:** Folgen von drei Wörtern

Wörter zählen

Word	Rank	More likely alternatives
We	9	*The This One Two A Three Please In*
need	7	*are will the also do*
to	1	
resolve	85	*have know do*
all	9	*the this these problems*
of	2	*the*
the	1	
important	657	*document question first*
issues	14	*thing point to*

- im untersuchten Korpus ist *We* das 9.-wahrscheinlichste Wort, mit dem ein Satz beginnt
- *need* ist das 7.-wahrscheinlichste Wort, das auf *We* folgt

Wörter zählen

- Unigram-zähl-Algorithmus in Perl
  - Tokenisierung des Textfiles, ein Wort pro Zeile
  - Wörter mit der Hash-Tabelle zählen
  - Wörter nach Alphabet und numerischem Ranking ordnen
- **split-Funktion:** jedes Wort des Textes wird einem Array zugewiesen

```
@words = split(/\n/, $text);
```

- **Hash-Tabelle** oder inhaltsorientiertes Array: Indizierung durch strings, nicht durch Zahlen

Wörter zählen

```
$wordcount{"a"} = 21;
$wordcount{"And"} = 10;
$wordcount{"the"} = 18;
```

→ Hash-Tabelle **`$wordcount`** wird erzeugt,
mit drei Indizes, **keys** genannt: a, And, the,
und den Werten 21, 10 und 18

`%wordcount:` man bezieht sich auf den ganzen Array

Wörter zählen

```
$text = <>;
while ($line = <>) {
 $text .= $line;
}
$text =~ tr/a-zåàâäæçéèêëîïôößùûüÿA-
ZÅÀÂÄÆÇÉÈÊËÎÏÔÖÙÛÜ'-,.?!:;/\n/cs;
$text =~ s/([,.?!:;])/\n$1\n/g;
$text =~ s/\n+/\n/g;
@words = split(/\n/, $text);
for ($i = 0; $i <= $#words; $i++) {
 if (!exists($frequency{$words[$i]})) {
  $frequency{$words[$i]} = 1;
 } else {
```

Wörter zählen

```
    $frequency{$words[$i]}++;
  }
}
foreach $word (sort keys %frequency){
 print "$frequency{$word} $word\n";
}
```

Wörter zählen

- Es ist unwahrscheinlich immer die genaue Wortfolge, die man sucht, in einem Korpus zu finden, selbst wenn die Korpora Billionen von Wörtern umfassen
- Vereinfachung:

$$
\begin{aligned}
P(S) &= P(w_1 \ldots w_n) \\
&= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2)P(w_n \mid w_1 \ldots w_{n-1}) \\
&= \prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1})
\end{aligned}
$$

→ Produkt von bedingten Wahrscheinlichkeiten

S = *It was a bright cold day in April*

$$
\begin{aligned}
P(S) = {} & P(It) \times P(was \mid It) \times P(a \mid It, was) \times P(bright \mid It, was, a) \times \\
& \ldots \times P(April \mid It, was, a, bright, \ldots, in)
\end{aligned}
$$

Wörter zählen

- Problem: Kein Korpus ist groß genug, um 4-gram, 5-gram oder gar 8-gram-Statistiken zu liefern
- Deshalb: Begrenzung der n-gram Länge auf 2 oder 3
- Näherungen:
  - Bigrams: $P(w_i \mid w_1, w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$
  - Trigrams: $P(w_i \mid w_1, w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-2}, w_{i-1})$
- Mit einem Trigram-Modell:

$$P(S) \approx P(It) \times P(was \mid It) \times P(a \mid It, was) \times P(bright \mid was, a) \times \ldots \times P(April \mid day, in)$$

Wörter zählen

Bigrams:

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)}{\sum_w C(w_{i-1}, w)} = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

Trigrams:

$$P_{MLE}(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

4. Wörter, Wortarten und Morphologie

- Einteilung der Wörter in Wortarten (engl. *parts of speech*, POS)
- Substantiv, Adjektiv, Verb, Pronomen, Artikel (→ flektierbar)
- Adverb, Präposition, Konjunktion, Partikel
- Definition der Wortarten ist manchmal willkürlich
- Wortarten können in zwei Klassen unterteilt werden: **offen** und **geschlossen**

Wörter, Wortarten und Morphologie

- **Geschlossene** Klassen sind relativ stabil über die Zeit hinweg, haben eine funktionale Rolle (grammatische Morpheme)
- Bsp: Artikel, Determinanten, Pronomen, Präpositionen, Konjunktionen, Modal- und Hilfsverben

Part of speech	English	French	German
Determiners	*the, several, my*	*le, plusieurs, mon*	*der, mehrere, mein*
Pronouns	*he, she, it*	*il, elle, lui*	*er, sie, ihm*
Prepositions	*to, of*	*vers, de*	*nach, von*
Conjunctions	*and, or*	*et, ou*	*und, oder*
Auxiliaries & Modals	*be, have, will, would*	*être, avoir, pouvoir*	*sein, haben, können, werden*

Wörter, Wortarten und Morphologie

- **Offene** Klassen bilden das Gros eines Vokabulars
- Sie entstehen und verschwinden im Laufe der Zeit
- Hauptkategorien: Substantiv, Adjektiv, Verb, Adverb

Part of speech	English	French	German
Nouns	*name, Frank*	*nom, François*	*Name, Franz*
Adjectives	*Big, good*	*grand, bon*	*groß, gut*
Verbs	*to swim*	*nager*	*schwimmen*
Adverbs	*rather, very, only*	*plutôt, très, uniquement*	*fast, nur, sehr, endlich*

Wörter, Wortarten und Morphologie

Hauptkategorie	**Untergliederung nach Merkmalen**
Substantiv, Adjektiv, Pronomen, Artikel	Kasus, Numerus, Genus
Verb	Person, Numerus, Modus, Tempus, Genus verbi
Adjektiv, Adverb	Komparationsstufe

Wörter, Wortarten und Morphologie

- *Bilen framför justitieministern svängde fram och tillbaka över vägen så att hon blev rädd.*
- 'The car in front of the Justice Minister swung back and forth and she was frightened.'

```
<taglemmas>
  <taglemma id="1" tag="nn.utr.sin.def.nom"
  lemma="bil"/>
  <taglemma id="2" tag="pp"
  lemma="framför"/>
  <taglemma id="3" tag="nn.utr.sin.def.nom"
  lemma="justitieminister"/>
  <taglemma id="4" tag="vb.prt.akt"
  lemma="svänga"/>
  <taglemma id="5" tag="ab" lemma="fram"/>
```

Wörter, Wortarten und Morphologie

```
  <taglemma id="6" tag="kn" lemma="och"/>
  <taglemma id="7" tag="ab" lemma="tillbaka"/>
  <taglemma id="8" tag="pp" lemma="över"/>
  <taglemma id="9" tag="nn.utr.sin.def.nom"
  lemma="väg"/>
  <taglemma id="10" tag="ab" lemma="så"/>
  <taglemma id="11" tag="sn" lemma="att"/>
  <taglemma id="12" tag="pn.utr.sin.def.sub"
  lemma="hon"/>
  <taglemma id="13" tag="vb.prt.akt.kop"
  lemma="bli"/>
  <taglemma id="14" tag="jj.pos.utr.sin.ind.nom"
  lemma="rädd"/>
  <taglemma id="15" tag="mad" lemma="."/>
</taglemmas>
```

- *Bilen framför justitieministern svängde fram och tillbaka över vägen så att hon blev rädd.*

Wörter, Wortarten und Morphologie

Code	(schwed.) Kategorie
nn	Substantiv
utr	uter (allg. Genus)
sin	Singular
def	definit (Bestimmt-heitsgrad)
nom	Nominativ
pp	Präposition
vb	Verb
prt	Präteritum
akt	aktiv (Genus verbi)

ab	Abverb
kn	Konjunktion
sn	Subjunktion
pn	Pronomen
sub	Subjektform (bei Pronomen)
kop	Kopulaverb (?)
jj	Adjektiv
pos	Positiv (Steigerungs-grad)
ind	indefinit (Bestimmt-heitsgrad)

Wörter, Wortarten und Morphologie

Lexikon

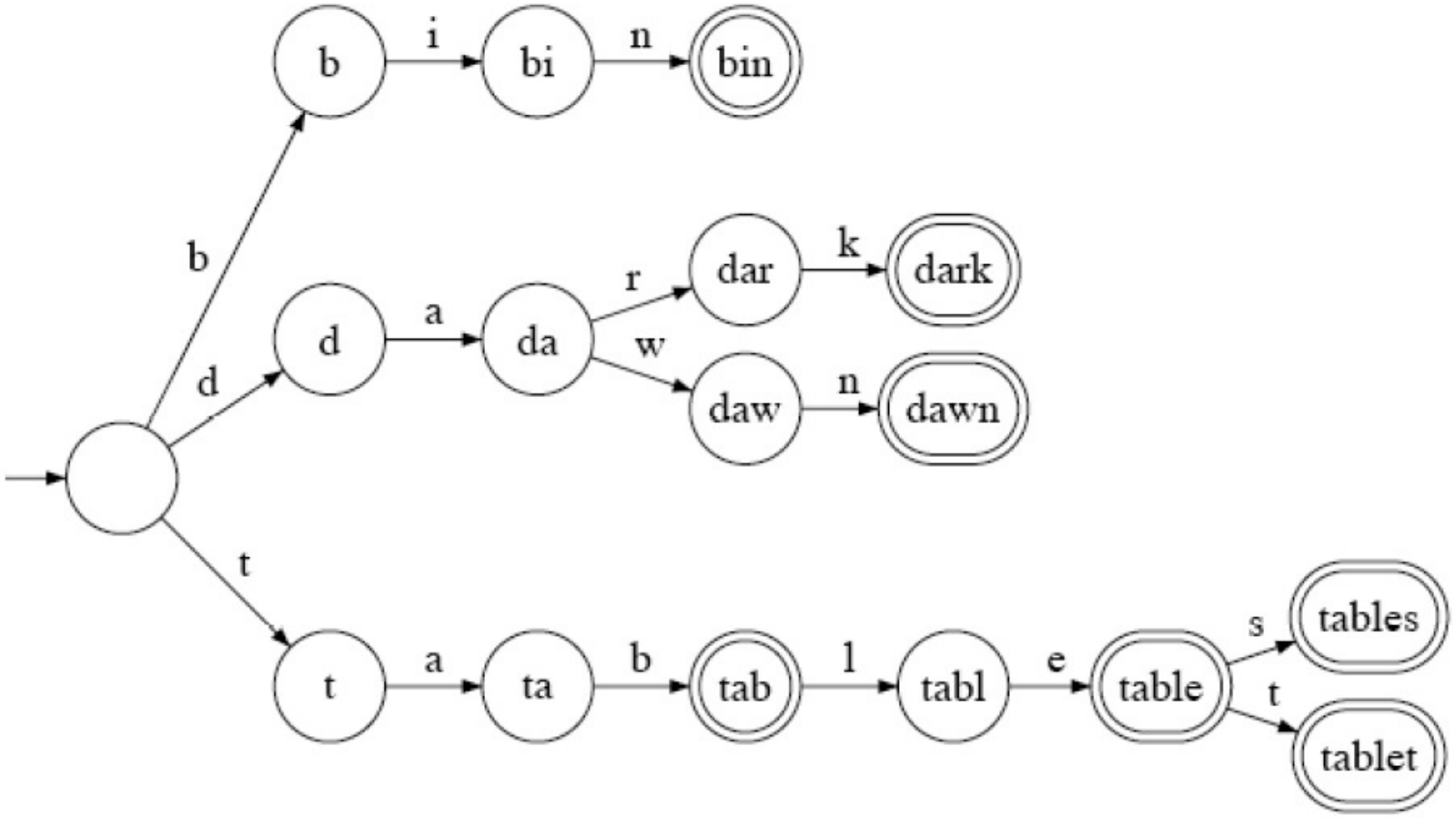
- Lexikoneinträge: **Lemma** (Lexikographie), **Lexem** (Morphologie; kleinste bedeutungstragende Einheiten einer Sprache)
- **Lexikon**: erster Baustein der meisten Sprachverarbeitungsprogramme
- Weiter Bereich: einfachen Wortliste bis hin zu sorgfältig mit Anmerkung versehenen Wörtern (Aussprache, Morphologie, syntaktischer und semantischer Markierung)
- syntaktische und semantische Mehrdeutigkeit vieler Wörter → mehrere Wörterbucheinträge

Wörter, Wortarten und Morphologie

letter trees or tries ("try ees")

- nützliche Datenstrukturen, um große Lexika zu speichern und Wörter schnell zu suchen
- Idee: Wörter als Bäume von Zeichen zu speichern und gleiche Zweige zu verwenden, soweit die Buchstaben zweier Wörter übereinstimmen
- In Programmen: ineinander verschachtelte Listen, wobei jeder Zweig eine Liste ist
- **erste Element** eines Zweiges: Wurzelbuchstabe
- **Blätter** eines Baums: lexikalischen Einträge

Wörter, Wortarten und Morphologie



Wörter, Wortarten und Morphologie

- in Prolog:

```
[
 [b, [i, [n, bin]]]
 [d, [a, [r, [k, dark]],
         [w, [n, dawn]]]]
 [t, [a, [b, tab,
             [l, [e, table,
                     [s, tables],
                     [t, tablet]]]]]]
]
```

Wörter, Wortarten und Morphologie

Morphologie

- Sprache = Menge von Morphemen, die sich in lexikalische (**Lexeme**) und **grammatische** Morpheme untergliedern lässt

	Word	Morpheme decomposition
English	*disentangling*	*dis+en+**tangle**+ing*
	rewritten	*re+**write**+en*
French	*désembrouillé*	*dé+em+**brouiller**+é*
	récrite	*re+**écrire**+te*
German	*entfesselnd*	*ent+**fesseln**+end*
	geschrieben	*ge+**schreiben**+t (ie)*

Wörter, Wortarten und Morphologie

- Morphologie:
  Unterteilung in **Flexion** und **Derivation**
- **Flexion:** Änderung der Gestalt eines Wortes zum Ausdruck seiner grammatischen Funktion innerhalb eines Satzgefüges → Angleichung an syntaktische Merkmale (Numerus, Genus, Tempus, …)
- Flexion ist relativ vorhersehbar, also regelmäßig
- Man kann die jeweilige Form, die der Kontext verlangt, bilden, wenn das Lemma, die Wortart und grammatische Merkmale bekannt sind (vgl. Eintrag im Wörterbuch)

Wörter, Wortarten und Morphologie

- Flexion

	Plural of nouns	Morpheme decomposition
English	*hedgehogs*	*hedgehog+s*
	churches	*church+es*
	sheep	*sheep+Ø*
French	*hérissons*	*hérisson+s*
	chevaux	*cheval+ux*
German	*Gründe*	*Grund+(¨) e*
	Hände	*Hand+(¨) e*
	Igel	*Igel+Ø*

Wörter, Wortarten und Morphologie

- **Derivation**/Ableitung: an das Grundwort (Wurzel, Stamm) wird ein **Affix** (Präfix, Suffix) angehängt, so dass ein neues Wort entsteht → Teil der **Wortbildung**
- die meisten Affixe können nur an eine bestimmte lexikalische Kategorie (Wortart) angehängt werden
- einige Affixe verändern die Wortart nicht, andere haben dagegen einen Wortklassenwechsel zur Folge

	English	French	German
Prefixes	***fore+****see,* ***un+****pleasant*	***pré+****voir,* ***dé+****plaisant*	***vor+****sehen,* ***un+****angenehm*
Suffixes	*manage+**able**,* *rigor+**ous***	*gér+**able**,* *rigour+**eux***	***er****+reich+****bar*** *ernst+**haft***

- weitere Wortbildungsmöglichkeit: **Komposition** = Aneinanderhängen zweier Wörter

Wörter, Wortarten und Morphologie

Morphologische Verarbeitung:

- **Generierung:** Erzeugung eines Wortes aus einer Menge von Morphemen
- **Parsing**: Zerlegung eines flektierten, abgeleiteten oder zusammengesetzten Wortes in Morpheme
- **Lemmatisierung:** Überführung eines Wortes in seine Grundform, dem kanonischen Wörterbucheintrag

English		French		German	
Generation →					
dog+s	*dogs*	*chien+s*	*chiens*	*Hund+e*	*Hunde*
work+ing	*working*	*travailler+ant*	*travaillant*	*arbeiten+end*	*arbeitend*
un+do	*undo*	*dé+faire*	*défaire*		
← Parsing					

Wörter, Wortarten und Morphologie

Mehrdeutigkeit

- Lemmatisierung ist oft mehrdeutig. Ein isoliertes Wort kann zu mehreren Lesarten führen.
- Lösung: **Kontext** des Satzes; normalerweise ist nur eine Lesart syntaktisch oder semantisch möglich

Wörter, Wortarten und Morphologie

	English	French	German
Words	*Run*	*Marche*	*Lauf*
Words in context	1. *A **run** in the forest* 2. *Sportsmen **run** everyday*	1. *Une **marche** dans la forêt* 2. *Il **marche** dans la cour*	1. *Der **Lauf** der Zeit* 2. ***Lauf** schnell!*
Lemmatization	1. **run**: noun singular 2. **to run**: verb present 3rd person plural	1. **marche**: noun singular feminine 2. **marcher**: verb present 3rd person singular	1. **Der Lauf**: noun, sing, masc 2. **laufen**: verb, imperative, singular

Wörter, Wortarten und Morphologie

- speicheraufwändige Methode: alle Wörter mit ihren flektierten Formen in ein Wörterbuch zu stecken, statt einen Parser zu implementieren, der diese Aufgabe übernimmt
- Gängige morphologische Parser basieren auf dem **Zwei-Ebenen-Modell** von Kimmo Koskenniemi
- Dieses verlinkt die **Oberflächenform** eines Wortes – das Wort wie es in einem Text vorkommt – mit seiner **lexikalischen** oder zugrunde liegenden Form – seinen Morphemfolgen

Wörter, Wortarten und Morphologie

Generation: Lexical to surface form →		
English	*dis+en+tangle+ed*	*disentangled*
	happy+er	*happier*
	move+ed	*moved*
French	*dés+em+brouiller+é*	*désembrouillé*
	dé+chanter+erons	*déchanterons*
German	*ent+wirren+end*	*entwirrend*
	wieder+ge+schreiben+en	*wiedergeschrieben*
Parsing: ← Surface to lexical form		

Wörter, Wortarten und Morphologie

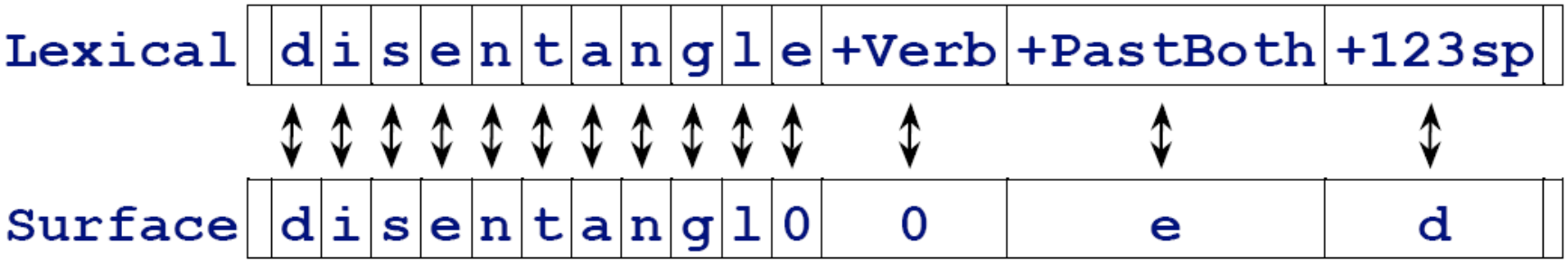
- Abgleich beider strings Buchstabe für Buchstabe,
  d.h. der erste Buchstabe der ersten Form wird auf den ersten Buchstaben der zweiten Form abgebildet usw.

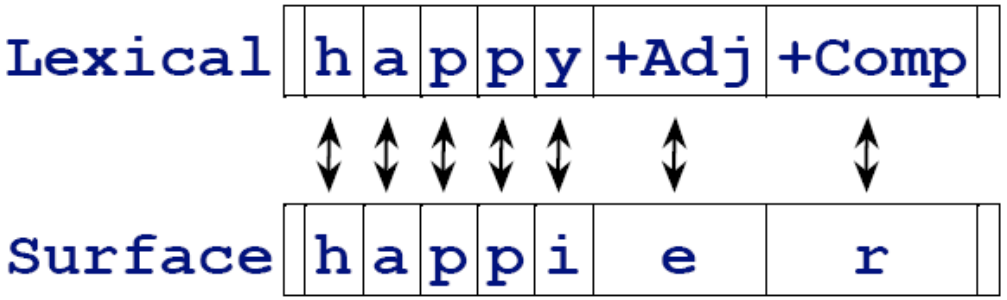
English	**dis+en+tangle+ed**	**happy+er**	**move+ed**
	dis0en0tangl00ed	**happi0er**	**mov00ed**
French	**dé+chanter+erons**	**cheval+ux**	**cheviller+é**
	dé0chant000erons	**cheva00ux**	**chevill000é**

- Output des Xerox Parsers für *disentangled* und *happier*:
  disentangle+Verb+PastBoth+123SP
  happy+Adj+Comp

Wörter, Wortarten und Morphologie

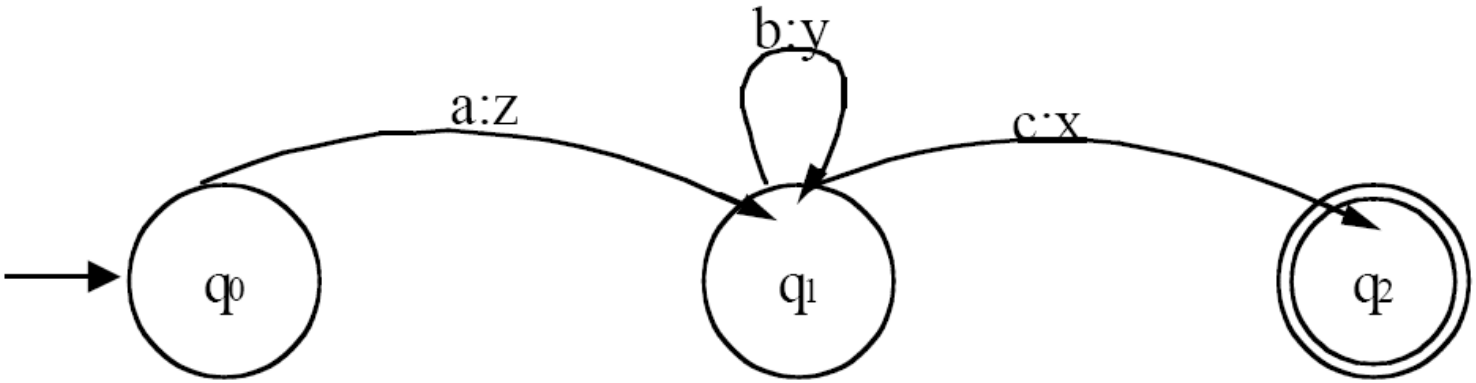
- Angesichts dieser neuen lexikalischen Formen muss der Parser die Symbole der Merkmale mit Buchstaben oder Nullen abgleichen





Wörter, Wortarten und Morphologie

- Das Zwei-Ebenen-Modell wird mit **endlichen Zustands-Umwandlern** (finite-state transducers, FST) implementiert
- Umwandler sind Automaten, die String-Paare akzeptieren, übersetzen oder generieren
- Bögen: das erste Symbol ist der Input, das zweite der Output
- der String abbbc wird umgewandelt in zyyyx



Wörter, Wortarten und Morphologie

- Bsp: Konjugation des frz. Verbs *chanter* 'singen' im Futur

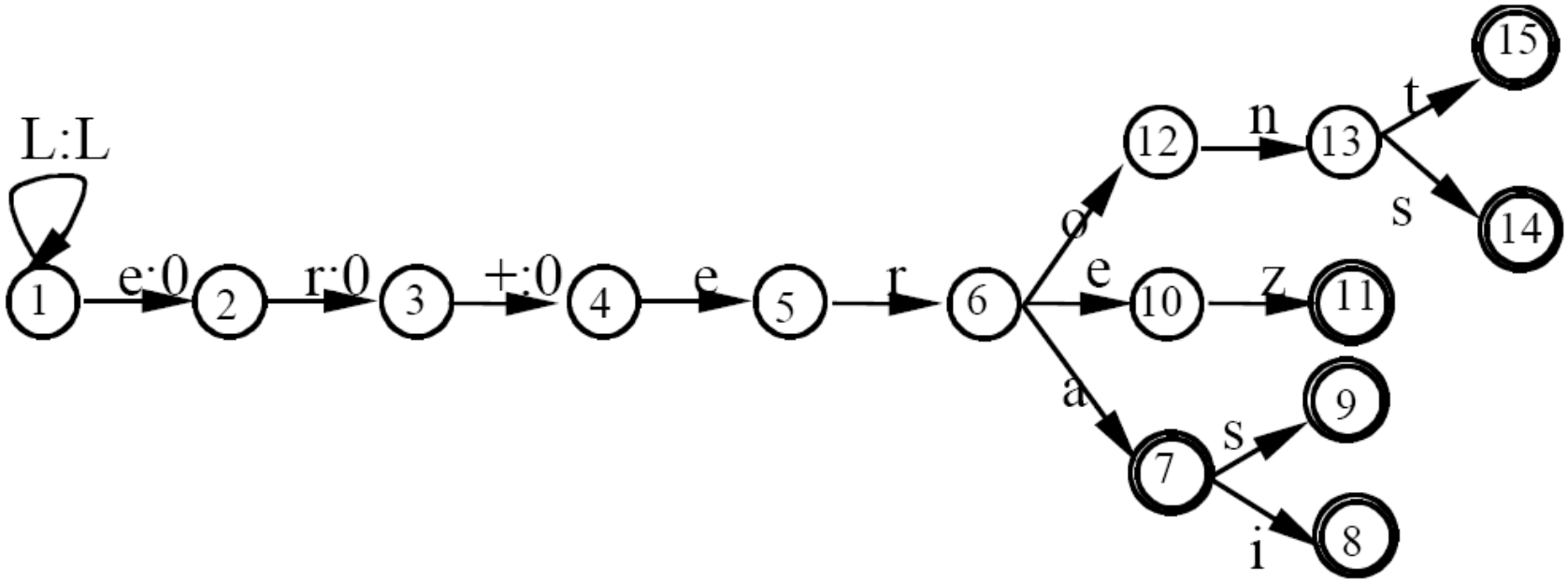
	1st	2nd	3rd
singular	*chanterai*	*chanteras*	*chantera*
plural	*chanterons*	*chanterez*	*chanteront*

- Abgleich der lexikalischen und der Oberflächenform

`chant000erai`	`chant000eras`	`chant000era`
`chanter+erai`	`chanter+eras`	`chanter+era`
`chant000erons`	`chant000erez`	`chant000eront`
`chanter+erons`	`chanter+erez`	`chanter+eront`

Wörter, Wortarten und Morphologie

- Endlichen Zustands-Umwandler für das Futur aller regelmäßigen frz. Verben der 1. Gruppe (enden auf *-er*)



Wörter, Wortarten und Morphologie

- Morphologische Regeln zur Umwandlung lexical:surface

Rules	Description
`a:b ⇒ lc __ rc`	`a` is transduced as `b` **only** when it has `lc` to the left and `rc` to the right
`a:b ⇐ lc __ rc`	`a` is **always** transduced as `b` only when it has `lc` to the left and `rc` to the right
`a:b ⇔ lc __ rc`	`a` is transduced as `b` **always and only** when it has `lc` to the left and `rc` to the right
`a:b /⇐ lc __ rc`	`a` is **never** transduced as `b` when it has `lc` to the left and `rc` to the right

Wörter, Wortarten und Morphologie

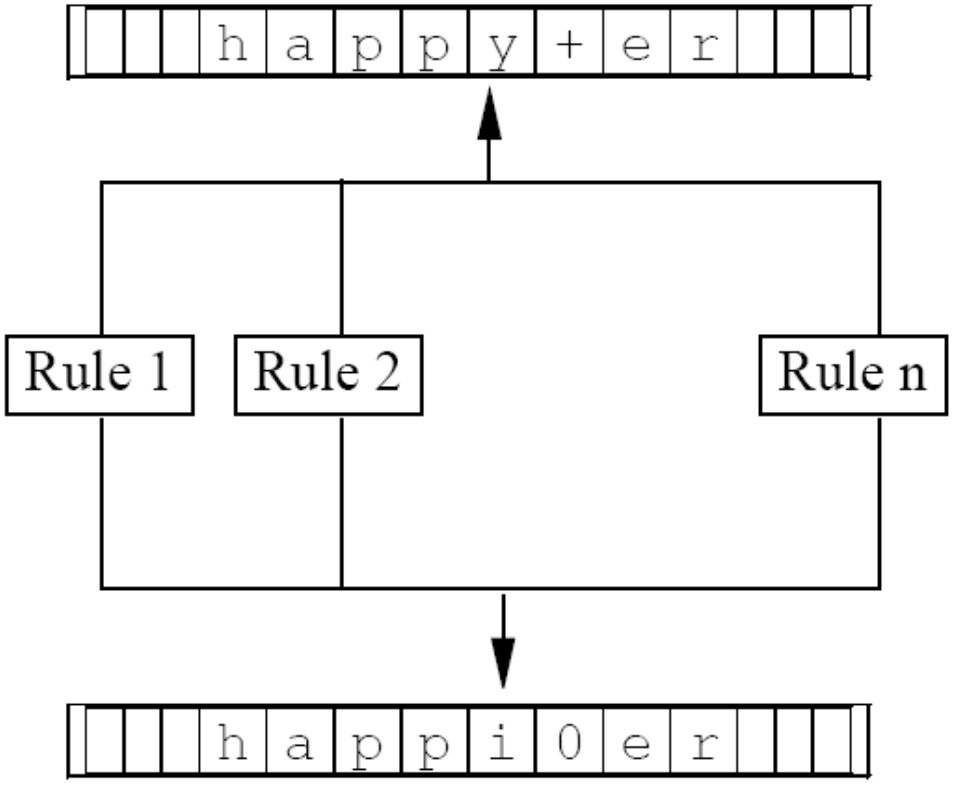
- Im Englischen kann ein lexikalischen *y* einem Oberflächen-*i* entsprechen, wie z.B. in *happier*.
- Es tritt auf, wenn dem *y* ein Konsonant vorausgeht und *-er*, *-ed* oder *-s* folgt

```
1.y:i ⇐ C:C __ +:0 e:e r:r
2.y:i ⇐ C:C __ +:e s:s
3.y:i ⇐ C:C __ +:0 e:e d:d
```

- Bsp:
  - happy+er > happier
  - party+s > parties
  - marry+ed > married

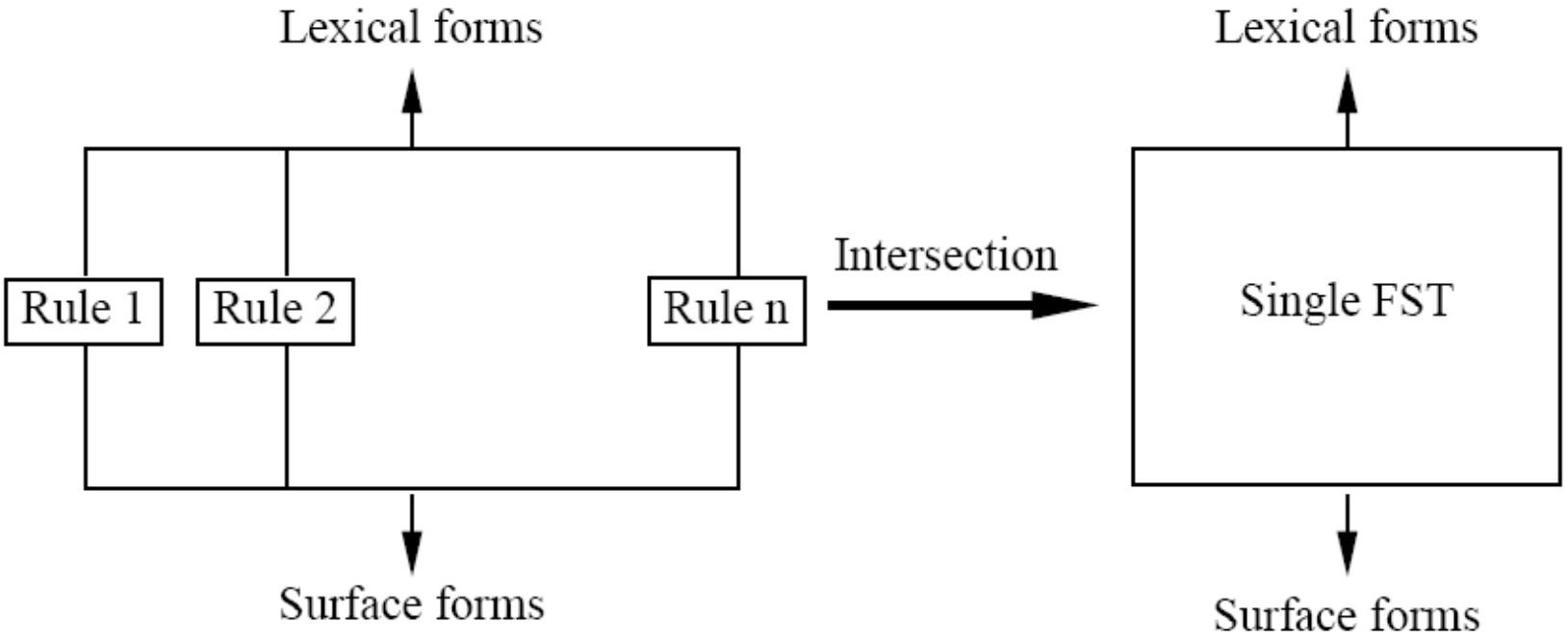
Wörter, Wortarten und Morphologie

- Sämtliche Regeln werden parallel angewandt



Wörter, Wortarten und Morphologie

- Die parallelen Umwandler werden zu einem einzigen kombiniert, indem man die Schnittmenge bildet



Wörter, Wortarten und Morphologie

- unregelmäßige französische Verben

Infinitive	courir	dormir	battre	peindre	écrire
1st pers sing	cours	**dors**	**bats**	peins	écris
2nd pers sing	cours	**dors**	**bats**	peins	écris
3rd pers sing	court	**dort**	**bat**	peint	écrit
1st pers plur	courons	dormons	battons	**peignons**	**écrivons**
2nd pers plur	courez	dormez	battez	**peignez**	**écrivez**
3rd pers plur	courent	dorment	battent	**peignent**	**écrivent**

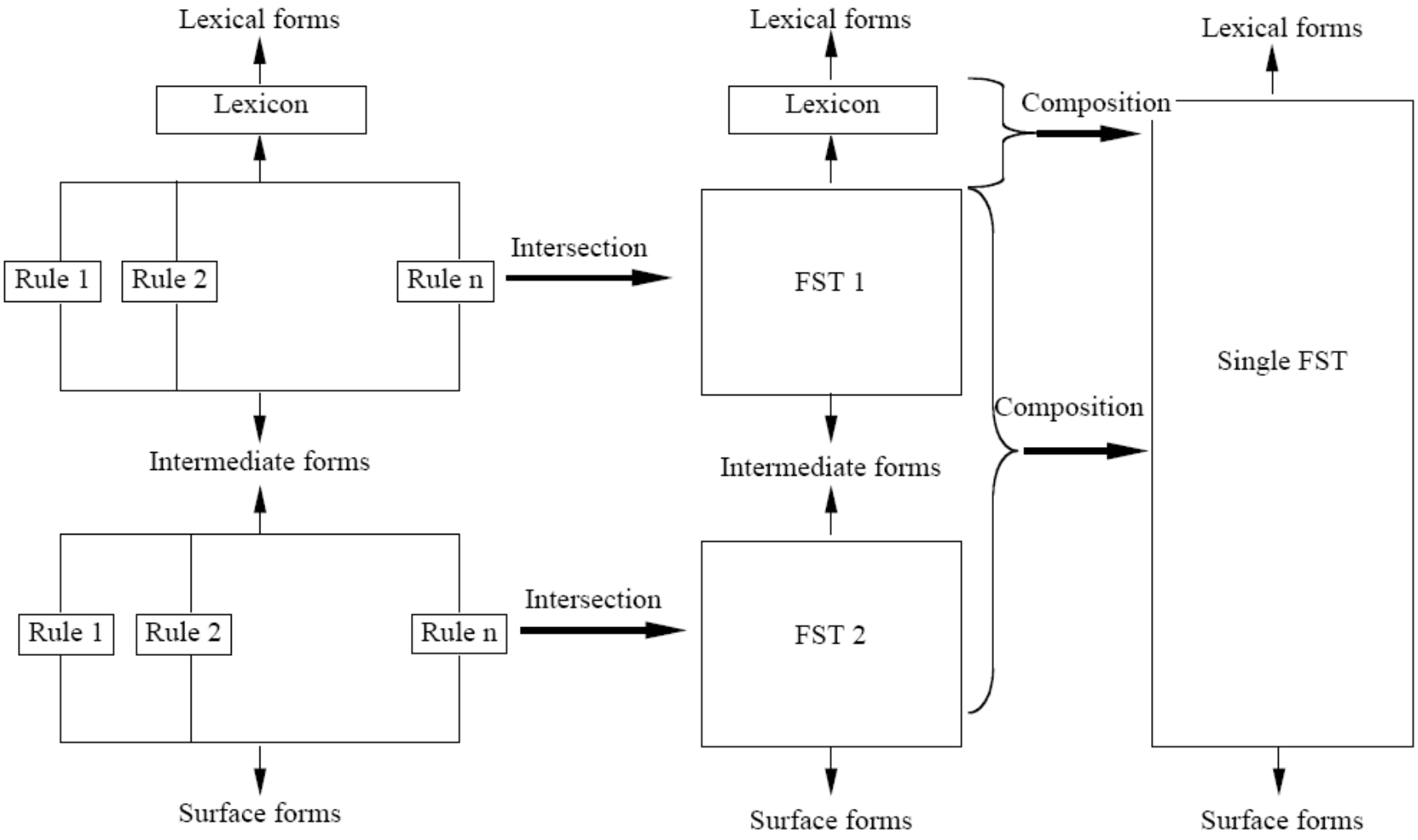
Wörter, Wortarten und Morphologie

- Zerlegung in einfache Regeln:

Lexical form: stem	dormir	+IndP +SG +P1
	↕	↕
Intermediate form: inflection	dorm	+IndP +SG +P1
	↕	↕
Intermediate form: deletion of	dorm	s
m followed by ***s***	↕	↕
Surface form	dor	s

- Komposition und Schnitt von endlichen Zustands-Wandlern, kombiniert mit einem Lexikon → einen einzigen Wandler

Wörter, Wortarten und Morphologie



Literaturangabe

- Nugues, Pierre M. (2006): *An Introduction to Language Processing with Perl and Prolog.* An Outline of Theories, Implementation, and Application with Special Consideration of English, French, and German. Berlin, Heidelberg: Springer.