

# Introduction to Natural Language Processing

## Part III: NLP using Rules

Henning Wachsmuth

<https://ai.uni-hannover.de>

# Learning Objectives

## Concepts

- Different types of “hand-crafted” rules for NLP
- The use of templates in NLP
- Benefits and limitations of hand-crafted rules

## Methods

- Text segmentation using hand-crafted decision trees
- Rewriting of text spans using finite-state transducers
- Discourse generation using predefined templates

## Cover tasks

- Tokenization
- Sentence splitting
- Morphological analysis
- Stemming
- Description generation

# Outline of the Course

I. Overview

II. Basics of Linguistics

III. NLP using Rules

- Introduction
- Hand-crafted Decision Trees
- Finite-State Transducers
- Template-based Generation

IV. NLP using Lexicons

V. Basics of Empirical Methods

VI. NLP using Regular Expressions

VII. NLP using Context-Free Grammars

VIII. NLP using Language Models

IX. Practical Issues

# Introduction

# NLP using Rules

## NLP (recap)

- Understanding and generating human-readable text (or speech) using computational methods
- Decode structured information from language, or encode it in language.

## Hand-crafted rule (in NLP)

- A definition of how to process text, which has been manually defined by a human developer.
- Processing may mean the segmentation of text, the rewriting of text, the inference of information from text, and similar.
- A rule encodes human expert knowledge of texts and/or tasks

## Rule-based NLP

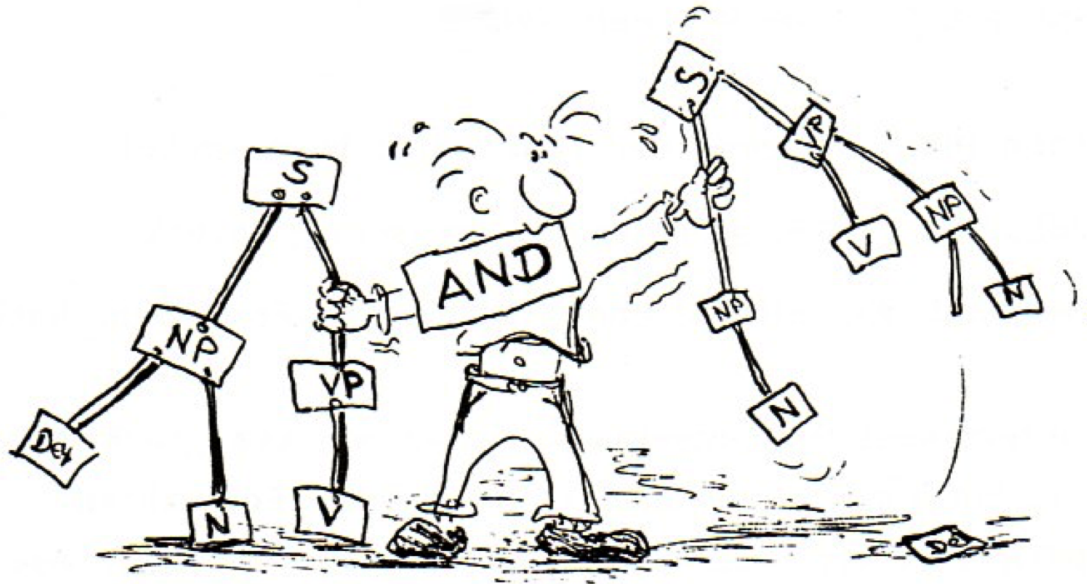
- Analysis and synthesis techniques based on hand-crafted rules
- Aka: Knowledge-based inference or the knowledge-based approach

# NLP using Rules

## Human Expert Knowledge

### Observation

- The quality of any rule-based NLP method rises and falls with the human expert knowledge encoded.



### Encoding of knowledge

- Decision/Rewrite rules, string templates/patterns, lexicons, grammars...

# NLP using Rules

## Rule-based Techniques in this Lecture Part

### Hand-crafted decision trees

- Series of decision rules that classify or segment input text spans

```
if char ∈ { '.', '?', '!'} then return true
else return false
```

### Finite-state transducers

- Series of rules that rewrite matching input spans into output spans

```
(*vowel*) y → i
```

If a span contains vowel and ends with 'y', replace 'y' with 'i'.

### Template-based generation

- Predefined string templates filled with information to create new text

```
"I am <stance> <issue>, because <reason>."
```

Example: "I am *con death penalty* because *the death penalty kills innocent people.*"

# NLP using Rules

## Rules vs. Statistics

### Alternative to hand-crafted rules?

- (Semi-) Automatic definition of implicit or explicit rules using statistics derived from a given dataset
- Done with probabilistic techniques or machine learning  
First statistical techniques later in the course
- Aka: Statistical inference or the data-driven approach

### Rule-based vs. statistical methods

- For most analysis and synthesis tasks, the best results are nowadays achieved with statistical/neural techniques.
- Particularly in industry, rule-based techniques are still used, because they are often well-controllable and explainable.
- All rule-based methods have a statistical counterpart in some way.



# Hand-crafted Decision Trees

# Decision Trees

## Decision tree

- A (representation of a) series of one or more *decision rules*, which lead to one of a set of predefined *outcomes*

## Decision rule

- A decision rule has a conditional *decision criterion* that can be tested and that leads to one of a set of alternative *options*.
- An option is an outcome or a decision tree itself.

## Binary decision tree

- A decision tree where each decision criterion has two options
- In such a tree, the rules can be modeled as if-then-else statements:

```
if decision criterion holds then option a else option b
```

- From here on, we consider only binary decision trees whose criteria can be either *true* or *false*.

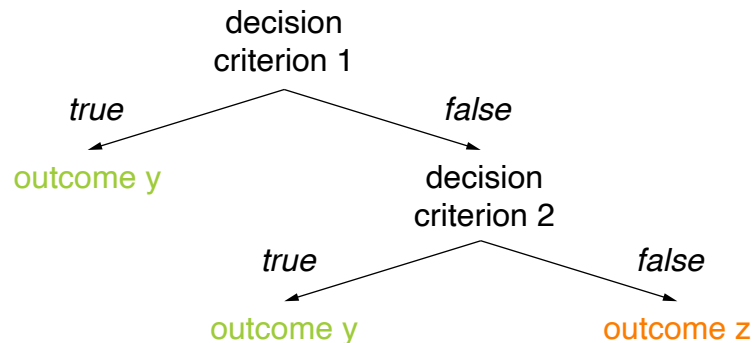
All decision trees can be transformed into such a binary boolean form.

# Decision Trees

## Representations

### Decision tree as a directed graph

- **Inner nodes.** Decision criteria, each capturing a single conditional rule  
The root is simply the first decision criterion to be considered.
- **Leaf nodes.** Potential outcomes from a given set of outcomes
- **Edges.** Options available for the decision criterion of the source node



### Decision tree as logical formulas

- A decision tree can be understood as a set of logical implications.

```
critterion 1 ∨ (¬critterion 1 ∧ critterion 2) → outcome y
(¬critterion 1 ∧ ¬critterion 2) → outcome z
```

# Decision Trees

## Hand-crafted Decision Trees

### Why “hand-crafted”?

- The decision trees considered here are solely created based on human expert knowledge.
- In machine learning, decision trees are created automatically based on statistics derived from data.

### Hand-crafted vs. statistical decision trees

- **Hand-crafted.** The set of decision criteria and their ordering of resulting decision rules are defined manually.
- **Statistical.** The best decision criteria and the best ordering (according to the data) are determined automatically.

### Notice

- Expert knowledge always governs the set of *candidate* decision criteria.

# Decision Trees

## NLP using Hand-crafted Decision Trees

### When to use?

- Decision tree structures get complicated fast.
- The number of decision criteria to consider should be small.
- The decision criteria should not be too interdependent.
- **Rule of thumb.** Few criteria with clear connections to outcomes.

```
(criterion 1  $\wedge$  ...  $\wedge$  criterion n)  $\rightarrow$  outcome y
```

### For which tasks to use?

- Theoretically, there is no real restriction.
- Practically, they are most used for shallow lexical or syntactic analyses.
- **Rule of thumb.** The surface form of a text is enough for the decisions.

### Tasks covered here

- Tokenization, sentence splitting

# Tokenization and Sentence Splitting

## Tokenization

- The task analysis that segments a span of text into its single tokens
- **Input.** Usually a plain text, possibly segmented into sentences
- **Output.** A list of tokens, not including whitespace between tokens

“The”, “man”, “sighed”, “.”, “It”, “s”, “raining”, “cats”, “and”, “dogs”, “,”, “he”, “felt”, “.”

## Sentence splitting

- The text analysis that segments a text into its single sentences
- **Input.** Usually plain text, possibly segmented into tokens
- **Output.** A list of sentences, not including space between sentences

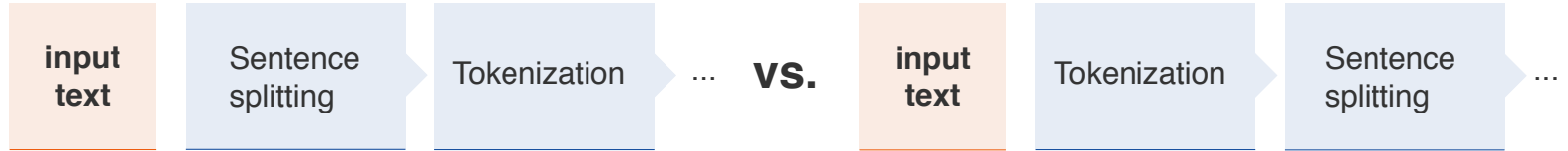
“The man sighed.”, “It’s raining cats and dogs, he felt.”

## Role in NLP

- Both are needed in NLP as preprocessing for most other analyses.
- Often, the first analyses performed on natural language text

# Tokenization and Sentence Splitting

## What First?



## Dilemma

- Knowing token boundaries helps identify sentence boundaries.

“Not all periods split sentences, e.g. those in acronyms.”

→ “Not”, “all”, “periods”, “split”, “sentences”, “,”, “e.g.”, “those”, “in”, “acronyms”, “.”

- Knowing sentence boundaries helps to identify token boundaries.

“An abbrev. harms readability—This also holds for missing whitespaces.Really!”

→ “An **abbrev.** harms readability” , “This also holds for missing **whitespaces.**”, “**Really!**”

## Schedule of the two text analyses

- The default is to tokenize first, but both schedules exist.
- An alternative is to do both text analyses jointly.

# Tokenization and Sentence Splitting

## Development of Approaches

### Need for development?

- Various rule-based and/or statistical methods for tokenization and sentence splitting are found in many code libraries.
- So, own approaches are not a must.
- They may still be useful, to tune the segmentation to certain text genres.

### Sentence splitting with a decision tree

- A character-level sentence splitter is presented below that tackles the task with a binary decision tree.
- It needs no tokens, so it can be scheduled first.

### Approach in a nutshell

1. Process an input text character by character.
2. Decide for each character whether it is the last character in a sentence.



# Sentence Splitting with a Decision Tree

## Example text

### ***“Apple Shares Jump on iPhone Sales Projection***

*Apple Inc. shares jumped 4.3 percent Wednesday after the company projected sales that suggest consumers are still snapping up the company’s high-end iPhones even as updated models are on the horizon.*

*The U.S.-based technology giant said on Tuesday it expects fiscal fourth-quarter revenue between \$60 billion and \$62 billion (Analysts were looking for \$59.4 billion, according to data compiled by Bloomberg!). The shares were trading at \$198.50 at 9:35 a.m. in New York, a record.*

*‘These results and guidance will increase investor confidence,’ Shannon Cross of Cross Research wrote in a note to investors. ‘We expect the vast majority of Apple’s product line-up to be refreshed during the next couple of quarters which should support near-term results.’ [...]*”

Excerpt from [www.bloomberg.com/news/articles/2018-07-31/apple-forecast-tops-analysts-estimates-on-new-iphones-services](http://www.bloomberg.com/news/articles/2018-07-31/apple-forecast-tops-analysts-estimates-on-new-iphones-services) (slightly modified for illustration reasons).

# Sentence Splitting with a Decision Tree

## Pseudocode

### Signature

- **Input.** A text given as a string

For simplicity, assumed to be trimmed: no leading, trailing, and double whitespaces

- **Output.** A list of sentences

### `decisionTreeBasedSentenceSplitting(String text)`

```
1.   List<Sentence> sentences ← ()
2.   int start ← 0 // Character index of sentence start
3.   int cur ← 0
4.   while cur < text.length - 1 do
5.       int end ← split(text, cur) // Index after sentence end
6.       if end != -1 then
7.           sentences.add(new Sentence(start, end))
8.           start ← end + 1
9.           cur ← start
10.      else cur ← cur + 1
11.      sentences.add(new Sentence(start, text.length))
12.      return sentences
```

# Sentence Splitting with a Decision Tree

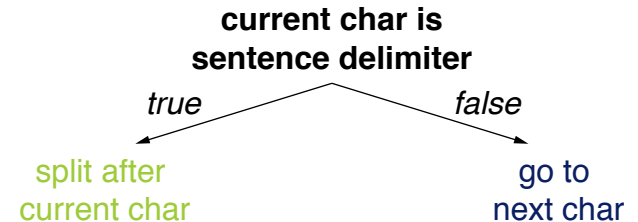
## Candidate Sentence Delimiters

### Sentence delimiters

- Most sentences in well-formed text end with a period, a question mark, or an exclamation mark.

### `split(String text, int cur)`

1. `if text[cur] ∈ { '.', '?', '!' } then`
2.     `return cur + 1`
3. `return -1`



### Challenges

- Colons are usually seen as sentence delimiters, if a full sentence is following. This requires “looking ahead”.

“They have two children: Max and Maja.” (one sentence)

“The reason is the following: Max and Maja are their children.” (two sentences)

# Sentence Splitting with a Decision Tree

## Examples for Fallacious Sentence Delimiters

### Example text

#### ***“Apple Shares Jump on iPhone Sales Projection***

*Apple Inc. shares jumped 4.3 percent Wednesday after the company projected sales that suggest consumers are still snapping up the company’s high-end iPhones even as updated models are on the horizon.*

*The U.S.-based technology giant said on Tuesday it expects fiscal fourth-quarter revenue between \$60 billion and \$62 billion (Analysts were looking for \$59.4 billion, according to data compiled by Bloomberg!). The shares were trading at \$198.50 at 9:35 a.m. in New York, a record.*

*‘These results and guidance will increase investor confidence,’ Shannon Cross of Cross Research wrote in a note to investors. ‘We expect the vast majority of Apple’s product line-up to be refreshed during the next couple of quarters which should support near-term results.’ [...]*”

Excerpt from [www.bloomberg.com/news/articles/2018-07-31/apple-forecast-tops-analysts-estimates-on-new-iphones-services](http://www.bloomberg.com/news/articles/2018-07-31/apple-forecast-tops-analysts-estimates-on-new-iphones-services) (slightly modified for illustration reasons).

# Sentence Splitting with a Decision Tree

## Fallacious Sentence Delimiters

### Common tokens containing punctuation

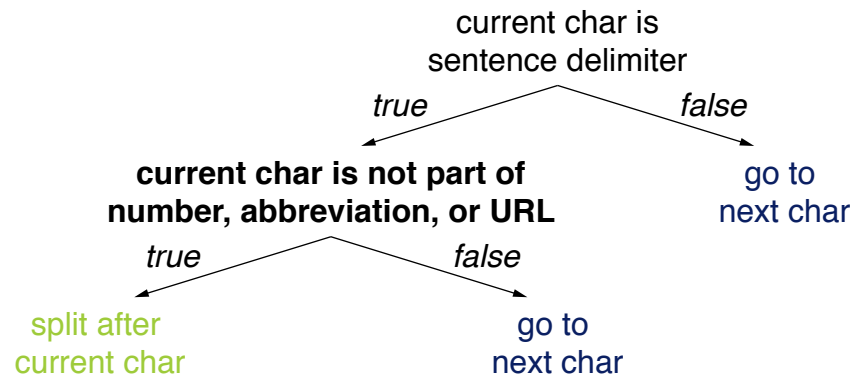
- Numbers with decimals or ordinals, such as “42.42” and “1.”
- Abbreviations, including acronyms, such as “abbrev.” and “a.m.”
- URLs, such as “https://www.args.me/?q=feminism”

```
split(String text, int cur)
```

```
// Code omitted on this and  
// on forthcoming slides
```

### Identification of such tokens

- Numbers and URLs follow clear patterns.
- Abbreviations need a lexicon.



### Challenges

- Many of these tokens may also occur at sentence endings.

# Sentence Splitting with a Decision Tree

## Example for Other Sentence Endings

### Example text

#### *“Apple Shares Jump on iPhone Sales Projection*”

*Apple Inc. shares jumped 4.3 percent Wednesday after the company projected sales that suggest consumers are still snapping up the company’s high-end iPhones even as updated models are on the horizon.*

*The U.S.-based technology giant said on Tuesday it expects fiscal fourth-quarter revenue between \$60 billion and \$62 billion (Analysts were looking for \$59.4 billion, according to data compiled by Bloomberg!). The shares were trading at \$198.50 at 9:35 a.m. in New York, a record.*

*‘These results and guidance will increase investor confidence,’ Shannon Cross of Cross Research wrote in a note to investors. ‘We expect the vast majority of Apple’s product line-up to be refreshed during the next couple of quarters which should support near-term results.’ [...]*”

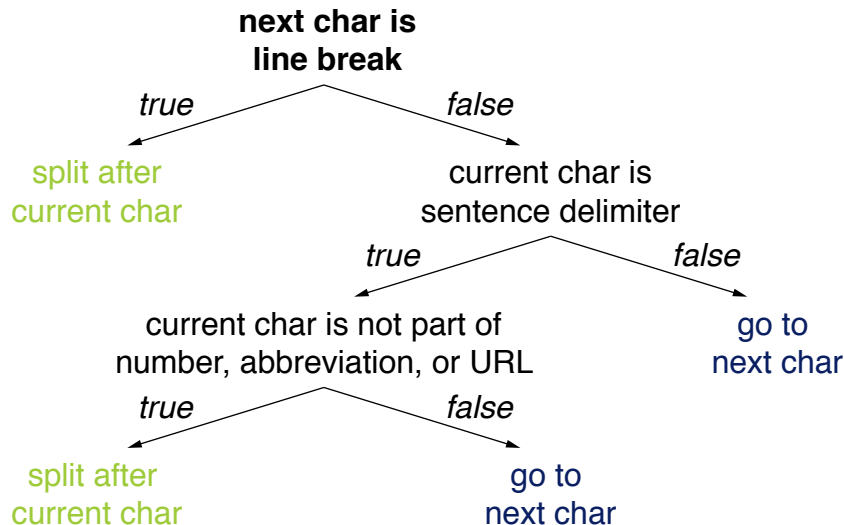
Excerpt from [www.bloomberg.com/news/articles/2018-07-31/apple-forecast-tops-analysts-estimates-on-new-iphones-services](http://www.bloomberg.com/news/articles/2018-07-31/apple-forecast-tops-analysts-estimates-on-new-iphones-services) (slightly modified for illustration reasons).

# Sentence Splitting with a Decision Tree

## Other Sentence Endings

### Line breaks

- In well-formed text, line breaks are unambiguous splitters of sentences.
- Titles often do not end with a delimiter, but are followed by line breaks.



### Challenges

- Some text formats add line breaks after every 80 characters (or similar).
- Text extracted from files such as PDFs often has additional line breaks.

# Sentence Splitting with a Decision Tree

## Example for Embedded Sentences

### Example text

#### ***“Apple Shares Jump on iPhone Sales Projection***

*Apple Inc. shares jumped 4.3 percent Wednesday after the company projected sales that suggest consumers are still snapping up the company’s high-end iPhones even as updated models are on the horizon.*

*The U.S.-based technology giant said on Tuesday it expects fiscal fourth-quarter revenue between \$60 billion and \$62 billion (Analysts were looking for \$59.4 billion, according to data compiled by Bloomberg!). The shares were trading at \$198.50 at 9:35 a.m. in New York, a record.*

*‘These results and guidance will increase investor confidence,’ Shannon Cross of Cross Research wrote in a note to investors. ‘We expect the vast majority of Apple’s product line-up to be refreshed during the next couple of quarters which should support near-term results.’ [...]*”

Excerpt from [www.bloomberg.com/news/articles/2018-07-31/apple-forecast-tops-analysts-estimates-on-new-iphones-services](http://www.bloomberg.com/news/articles/2018-07-31/apple-forecast-tops-analysts-estimates-on-new-iphones-services) (slightly modified for illustration reasons).



# Sentence Splitting with a Decision Tree

## Embedded Sentences

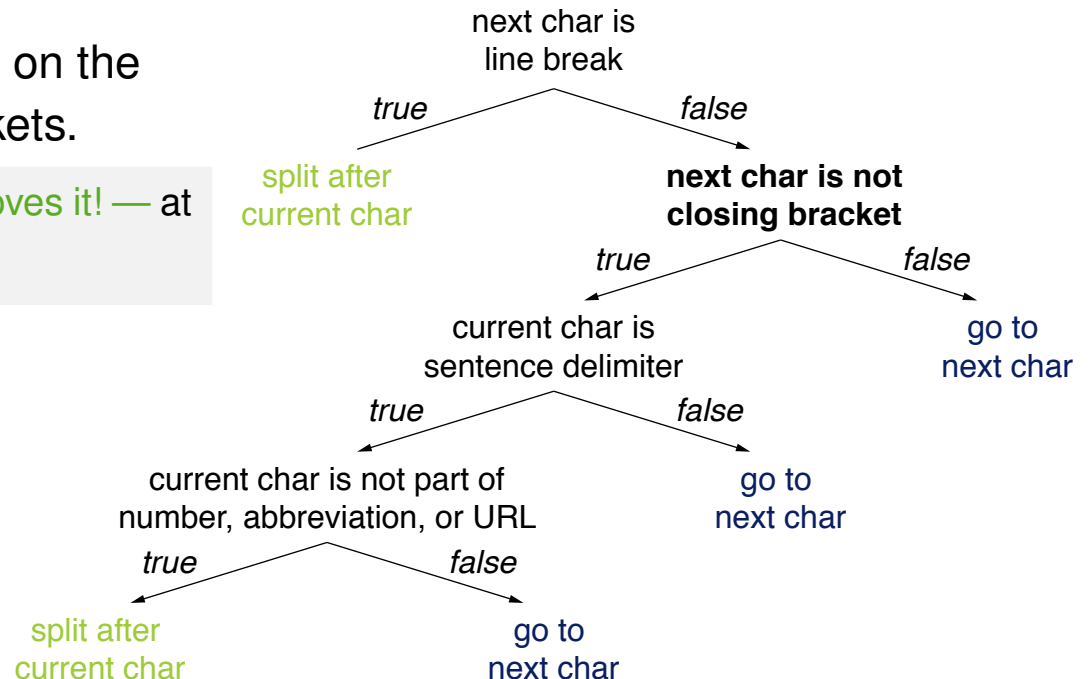
### Brackets

- Brackets, usually parentheses, may embed full sentences into others.
- Closing brackets thus “overrule” potential preceding sentence endings.

### Challenges

- Hyphens may take on the roles of such brackets.

“Max smiled — she loves it! — at her again.”



# Sentence Splitting with a Decision Tree

## Example for Quotes

### Example text

#### ***“Apple Shares Jump on iPhone Sales Projection***

*Apple Inc. shares jumped 4.3 percent Wednesday after the company projected sales that suggest consumers are still snapping up the company’s high-end iPhones even as updated models are on the horizon.*

*The U.S.-based technology giant said on Tuesday it expects fiscal fourth-quarter revenue between \$60 billion and \$62 billion (Analysts were looking for \$59.4 billion, according to data compiled by Bloomberg!). The shares were trading at \$198.50 at 9:35 a.m. in New York, a record.*

*‘These results and guidance will increase investor confidence,’ Shannon Cross of Cross Research wrote in a note to investors. ‘We expect the vast majority of Apple’s product line-up to be refreshed during the next couple of quarters which should support near-term results.’ [...]*”

Excerpt from [www.bloomberg.com/news/articles/2018-07-31/apple-forecast-tops-analysts-estimates-on-new-iphones-services](http://www.bloomberg.com/news/articles/2018-07-31/apple-forecast-tops-analysts-estimates-on-new-iphones-services) (slightly modified for illustration reasons).

# Sentence Splitting with a Decision Tree

## Quotes

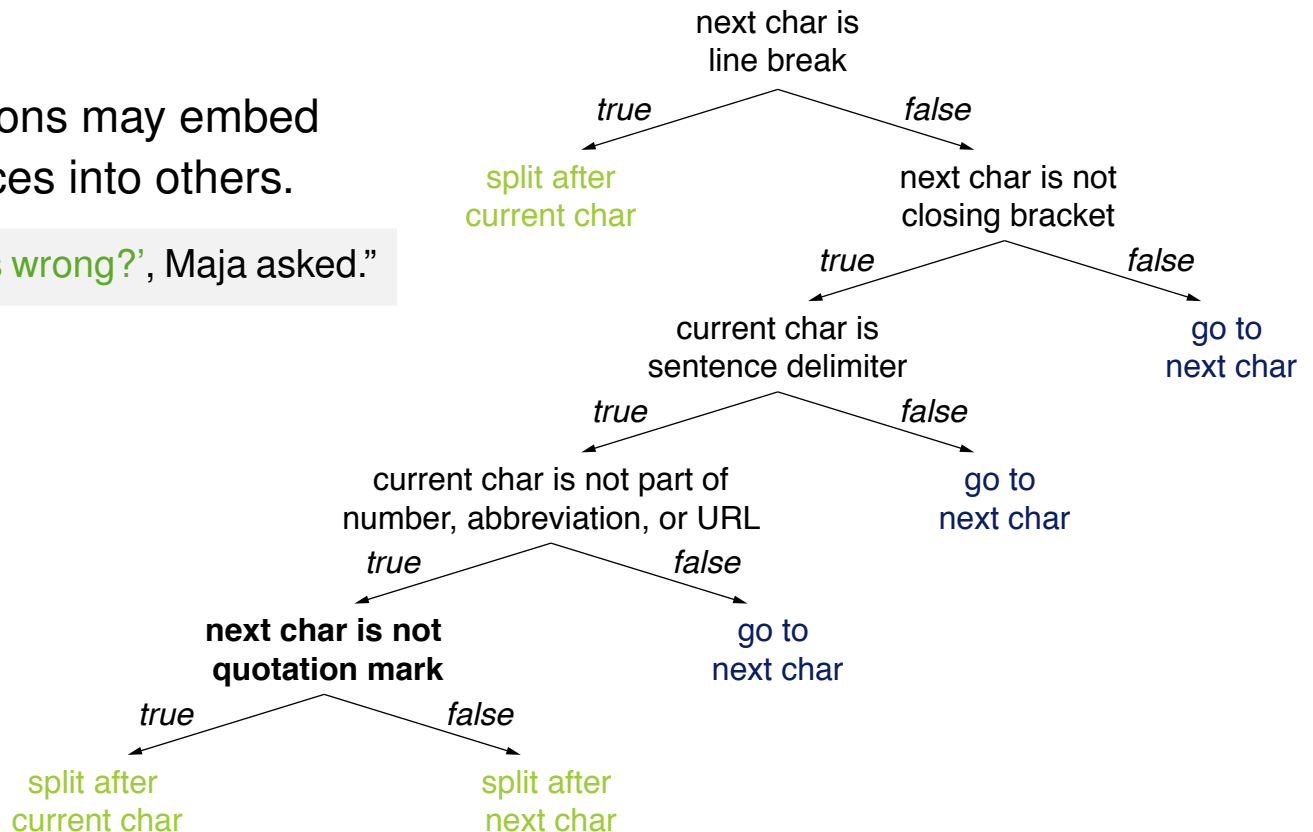
### Quotation marks

- Quotation marks may shift the end of a sentence.

### Challenges

- Quotations may embed sentences into others.

“‘What’s wrong?’, Maja asked.”



# Sentence Splitting with a Decision Tree

## Further Challenges

### Grammatical flaws

- The introduced rules to some extent assume a text to be well-formed.
- This largely holds for genres such as news articles, but less for more informal texts, such as those found on social media.

### Capitalization

- Some splitters require sentences to start with a capital letter.  
Notice that the presented approach does not consider capitalization at all.
- Inconsistent capitalization is particularly common on social media.

“i was thinking... A LOT.” (one sentence)

“i was thinking... a lot happened in this time.” (two sentences)

### And much more

- Ellipses (“...”), multiple sentence delimiters in a row (“!?!”), unknown acronyms (“Btww.”), smileys (“:-)”), ...

# Tokenization with a Decision Tree

## Tokenization with a decision tree

- Tokenization faces similar problems as sentence splitting.
- An analog decision tree could be created for this analysis.
- If the approach above is applied before, knowledge about sentence boundaries can be exploited.

## Potential decision criteria

- **End of sentence.** Always indicates a boundary
- **Whitespace.** Strongly indicates a boundary
- **Comma.** Indicates a boundary, unless part of a number
- **Hyphen.** Indicates a boundary, unless part of a word
- **Period.** Indicates a boundary, unless part of a number, abbrev., or URL
- **Letters and digits.** Usually do not indicate a boundary

# Tokenization with a Decision Tree

## Specific Tokenization Issues

### Selected controversial cases

“Finland’s capital”	→	“Finland”+“s”+“capital”	vs.	“Finland’s”+“capital”
“Hewlett-Packard”	→	“Hewlett”+“-”+“Packard”	vs.	“Hewlett-Packard”
“state-of-the-art”	→	“state”+“-”+“of”+“-”+“the”+“-”+“art”	vs.	“state-of-the-art”
“4 242”	→	“4”+“242”	vs.	“4 242”

### Lost token boundaries

“Thetabledownthere” → “the table down there” vs. “theta bled own there”

### Missing whitespaces

- **Chinese.** Multiple syntactically and semantically correct segmentations

“country-loving person” 爱国|人 vs. 爱|国人 “love country-person”

- **Hashtags.** May entail similar problems

“#nowthatcherisdead” (on Twitter after Thatcher died)

# Hand-Crafted Decision Trees

## General Issues

### Issues with decision criteria

- The connection of criteria to outcomes is often not straightforward.

```
if (#positive words > #negative words) then positive (correct?)
```

- For numeric decision criteria, thresholds may be needed.

```
if (sentEnd-sentStart < min) then go to next char (what minimum?)
```

- Often, a weighting of different decision criteria is important.
- It is unclear how to find all relevant criteria.

### Issues with decision trees

- Decision trees get complex fast, already for few decision criteria.  
Many approaches use thousands of criteria. In theory,  $2^n$  combinations of  $n$  criteria.
- The mutual effects of decision rules are hard to foresee.
- Adding new decision criteria may change a tree drastically.

# Hand-Crafted Decision Trees

## Benefits and Limitations

### Benefits

- Precise rules can be specified with human expert knowledge.
- The behavior of (small) hand-crafted decision trees is well-controllable.
- Decision trees are considered to be easily interpretable.

### Limitations

- The bigger the trees get, the harder it is to adjust them.
- Setting them up manually is practically infeasible for complex tasks.
- Including weightings is all but straightforward for decision trees.

### Implications

- They are only useful for simple tasks: those with few decision criteria.
- For more complex tasks, machine learning is usually preferred.
- Still, hand-crafted decision rules may be used at a high level.



# Finite-State Transducers

# Finite-State Transducers (FSTs)

## Recap finite-state automata (FSAs)

- An FSA is a state machine that reads a string from a regular language.
- It represents the set of all strings belonging to the language.

## Finite-state transducer (FST) (aka Mealy Machine)

- An FST extends an FSA in that reads one string and writes another.
- It represents the set of all relations between two sets of strings.

## FST as a 5-tuple $(Q, \Sigma, q_0, F, \delta)$

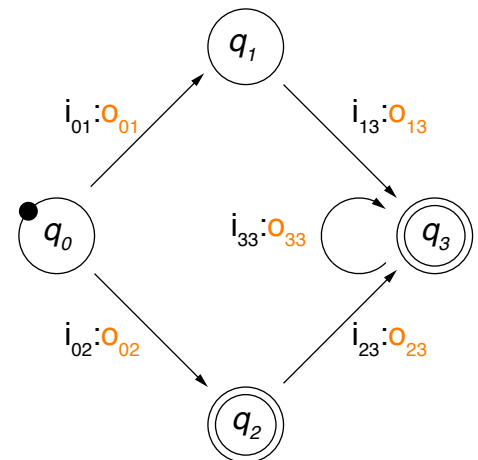
$Q$  A finite set of  $n > 0$  states,  $Q = \{q_0, \dots, q_n\}$

$\Sigma$  An alphabet of complex symbols  $i:o$ , where  $i$  is an input symbol,  $o$  an output symbol

$q_0$  A start state,  $q_0 \in Q$

$F$  A set of final states,  $F \subseteq Q$

$\delta$  A transition function between states triggered based on  $i:o$ ,  $\delta : Q \times \Sigma \rightarrow Q$



# Finite-State Transducers (FSTs)

## NLP using FSTs

### Four ways of employing an FST

- **Translator / Rewriter.** Read a string  $i$  and output another string  $o$ .
- **Recognizer.** Take a pair of strings  $i:o$  as input. Output “accept” if  $i:o \in \Sigma$ , “reject” otherwise.
- **Generator.** Output pairs of strings  $i:o$  from  $\Sigma$ .
- **Set relator.** Compute relations between sets of strings  $I$  and  $O$ , such that  $i \in I$  and  $o \in O$ .

### Tasks covered here

- Morphological analysis, word normalization (stemming)

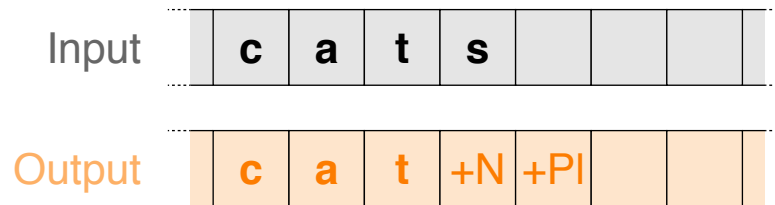
### Morphological analysis

- The text analysis that breaks down a word into its different morphemes
- Sometimes used in NLP as preprocessing for tasks that require deeper grammatical analysis

# Morphological Analysis with Finite-State Transducers

## Morphological analysis as rewriting

- **Input.** The fully inflected surface form of a word
- **Output.** The stem + the part-of-speech + the number (singular or plural).
- This can be done with an FST that reads a word and writes the output.

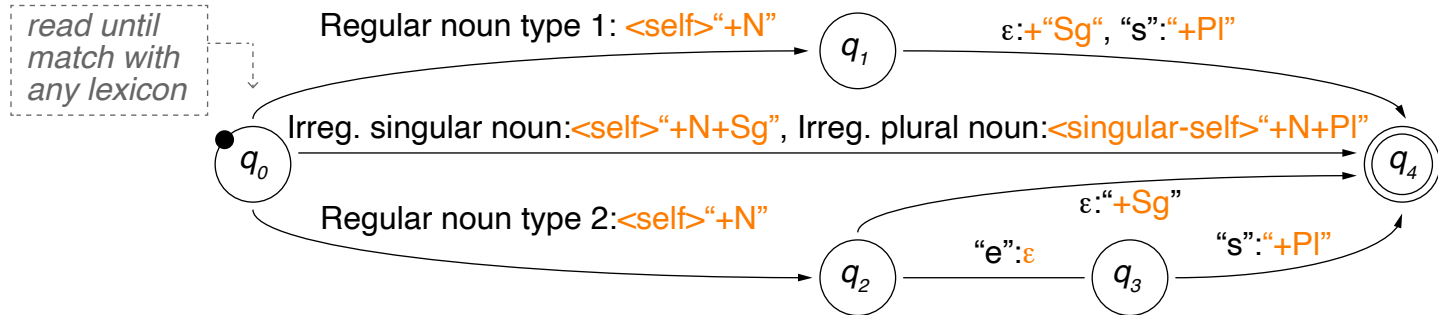


## Knowledge needed for morphological analysis

- **Lexicon.** Stems with affixes, together with morphological information
- **Morphotactics.** A model that explains which morpheme classes (e.g., plural “-s”) can follow others (e.g., noun) inside a word
- **Orthographic rules.** A model of the changes that may occur in a word, particularly when two morphemes combine

# Morphological Analysis with Finite-State Transducers

## Simple Example: English Nominal Number Inflection \*



( $\epsilon$  empty word,  $\langle \text{self} \rangle$  output is input,  $\langle \text{singular-self} \rangle$  output is singular of input)

## Lexicons

- Regular noun type 1 (plural form with “-s”). “cat”, “zero”, ...
- Regular noun type 2 (plural form with “-es”). “bus”, “hero”, ...
- Irreg. singular noun. “mouse”, “try”, ...
- Irreg. plural noun (maps to singular). “mice” → “mouse”, “tries” → “try”, ...

## Notice

- Much knowledge is captured in the lexicons; they must contain all individual regular and irregular noun stems.

# Word Normalization

## Word normalization

- The conversion of all words in a text into some defined canonical form  
Technically, a word is not converted, but its canonical form is stored in addition.
- Used in NLP to identify different forms of the same word
- Character-level and morphological methods exist, all with pro's & con's.

## Common character-level word normalizations

- **Case folding.** Converting all letters to lower-case (or upper-case, resp.).

“First” → “first”

“CamelCase” → “camelcase”

“US” → “us” (reasonable?)

- **Removal of special characters.** Keep only letters and digits.

“U.S.A.” → “USA”

“tl;dr” → “tldr”

“42.42” → “4242” (reasonable?)

- **Removal of diacritical marks.** Keep only plain letters without diacritics.

“café” → “cafe”

“Barça” → “Barca”

“Tú” → “Tu” (reasonable?)

# Word Normalization

## Morphological Normalization

### Morphological normalization

- Identification of a single canonical representative for morphologically related wordforms
- Reduces inflections (and partly also derivations) to a common base
- Two alternative techniques: stemming and lemmatization

### Stemming

- The text analysis that identifies the stem of a token

“playing” → “play”

“derive” → “deriv”

“am” → “am”

### Lemmatization

- The text analysis that identifies the lemma of a token

“playing” → “play”

“derive” → “derive”

“am” → “be”

# Stemming with Finite-State Transducers

## Stemming with affix elimination

- Stem a word with rule-based elimination of prefixes and suffixes.

“connects”, “connecting”, “connection” → “connect” (suffixes eliminated)  
“embodied”, “body”, “bodies” → “bod” (both eliminated)

- The elimination may be based on prefix and suffix forms only.  
Both prefixes and suffixes are closed classes within a language.

## Porter Stemmer

- The most common stemmer for English
- Based on a series of cascaded rewrite rules
- Can be implemented as a lexicon-free FST

## Approach in a nutshell

1. Rewrite longest possible match of a given token with a set of defined character sequence patterns.
2. Repeat Step 1 until no pattern matches the token anymore.



# Stemming with Finite-State Transducers

## Porter Stemmer: Pseudocode

### Signature

- **Input.** A string  $s$  (representing a word)
- **Output.** The identified stem of  $s$

### Hand-crafted pattern matching rules

- Nine ordered rule sets, each with 3–20 rules `<premise> S1 → S2:`

```
if S ends with S1 and the part before S1 fulfills <premise>
then replace S1 by S2
```

```
PorterStemmer(String S) // clean-up rules left out
```

```
1. for each ruleSet do
2.     for each rule <premise> S1 → S2 ∈ ruleSet do
3.         if endsWith(S, S1) and fulfills(S-S1, <premise>) then
4.             S ← S-S1 + S2
5.             break // leave inner for loop
6. return S
```

# Stemming with Finite-State Transducers

## Porter Stemmer: Premises

### Premises

- Patterns specifying certain attributes of string sequences
- The patterns were defined hand-crafted based on expert knowledge.

### Premise patterns used by the Porter Stemmer

( $*S'$ )  $S-S1$  ends with a string  $S'$ .

( $*V*$ )  $S-S1$  contains some vowel  $V$ .

“Vowel”: All real vowels as well as ‘y’ after a consonant, as in “lovely”

( $*CC$ )  $S-S1$  ends with two identical consonants  $C$ .

“Consonant”: All real consonants, but ‘y’ only after a vowel, as in “toy”

( $*CVC'$ )  $S-S1$  ends with  $CVC'$  where  $C' \notin \{ 'W', 'X', 'Y' \}$ .

( $m > x$ ) Number  $m$  of sequences of vowels followed by consonants in  $S-S1$  is larger than some  $x$

Example: For  $m = 2$ , a sequence would be “uances”.

# Stemming with Finite-State Transducers

## Porter Stemmer: Selected Rules

Rule set	<premise>	S1	→	S2	Examples (gray: no rule match)
1		sses		ss	caresses → caress
1		ies		i	ponies → poni
1		ss		ss	caress → caress
1		s		ε	cats → cat
2a	(m>0)	eed		ee	agreed → agree, feed → feed
2a	(*v*)	ed		ε	plastered → plaster, bled → bled
2a	(*v*)	ing		ε	motoring → motor, sing → sing
3	(*v*)	y		i	happy → happi, sky → sky
4	(m>0)	ational		ate	relational → relate
4	(m>0)	biliti		ble	sensibiliti → sensible
...					
6	(m>0)	al		ε	revival → reviv
...					

Full list at <http://snowball.tartarus.org/algorithms/porter/stemmer.html> (notice: Numbering of steps differs in different sources)

# Stemming with Finite-State Transducers

## Porter Stemmer: Rule Sets

### Each rule set represents a specific function

- Set 1. Plural nouns and third person singular verbs
- Set 2a. Verbal past tense and progressive forms
- Set 2b. Clean-up: Add specific word endings
- Set 3.  $Y \rightarrow I$
- Set 4. Derivational morphology I: Multiple suffixes
- Set 5. Derivational morphology II: Remaining multiple suffixes
- Set 6. Derivational morphology III: Single suffixes
- Set 7a. Clean-up: Remove specific vowel endings
- Set 7b. Clean-up: Remove double letter endings

### Notice

- Maximum one rule per rule set is applied.

# Stemming with Finite-State Transducers

## Porter Stemmer on an Example Text

### Original text

*“A relevant document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer.”*

### Porter-stemmed text

*“A relevant document will describ market strategi carri out by U.S. compani for their agricultur chemic, report predict for market share of such chemic, or report market statist for agrochem pesticid, herbicid, fungicid, insecticid, fertil.”*

# Stemming with Finite-State Transducers

## Porter Stemmer: Analysis

### Issues

- Difficult to modify, that is, the effects of changes are hardly predictable
- Tends to overgeneralize:

“policy” → “police”

“university” → “universe”

“organization” → “organ”

- Does not capture clear generalizations:

“European” and “Europe”

“matrices” and “matrix”

“machine” and “machinery”

- Generates some stems that are difficult to interpret:

“iteration” → “iter”

“general” → “gener”

### Observations

- The application of rules is trivial. The knowledge is *in* the rules.
- The rules are specific to English (adaptations to other languages exist).
- The lack of a lexicon has limitations (alternatives with lexicons exist).

# Finite-State Transducers (FSTs)

## Benefits and Limitations

### Benefits

- As for decision trees, precise rules can be specified by human experts.
- The behavior of FSTs for simple rewriting tasks is well-controllable.
- They also tend to be computationally efficient.

### Limitations

- FSTs tend to overgeneralize or to have low coverage.
- For more complex tasks, FSTs easily get very complicated.
- They are rather restricted to tasks analyzing surface forms is enough.

### Implications

- FSTs should rather be used where approximate results are sufficient.
- Some tasks can be easier accessed with regular expressions.
- For advanced rewriting tasks, neural methods are the only option.

# Template-based Generation



# Templates

## Template

- We consider templates for natural language generation (NLG) here.
- Such templates define constraints and points of variation for any text instance to be generated.
- Most common types: *sentence templates* and *discourse templates*

## Sentence template

- Representation of a sentence as boilerplate text and parameters
  - **Parameters.** To be filled by instance-specific concepts and values
  - **Boilerplate text.** More or less unchanged in any text
- Minor adjustments may be done for grammatical reasons.

```
"I am <stance> <issue>, because <reason>."
```

## Discourse template

- Hierarchical or sequential representation of the organization of a text
- Based on discourse relations, series of sentence templates, or similar

# Templates

## Creating and Filling Templates

### Where do templates come from?

- Template are often created specifically for a given task based on domain expert knowledge.
- Alternative: Infer them semi-automatically from recurring text patterns.

```
“Smoking should be banned.”  
“The death penalty should be abolished.” → “<controversial topic>  
“Abortion should be legalized.”          should be  
                                           <accept/reject term>.”
```

### With what to fill templates?

- The parameters of templates are the input required for a template.
- This input comes from data or knowledge bases, is entered by a user, computed on-the-fly, or similar
- In addition, lexicons are often needed to map the input to terms.

```
“gender balance” pro [x] con [] → “Gender balance should be promoted.”
```

# Templates

## Using Templates

### For what to employ templates?

- Recurring texts with conventional form and structure
  - Situations where natural language is preferred over structured data
  - Precise requirements of how texts should look like
  - Writing assistance of humans in recurring tasks
- ... and similar

### Applications in practice

- **Answer** questions of predefined types, such as those in Jeopardy
  - **Formulate** learned rules, such as those of decision trees
  - **Explain** medical information, such as patient diagnoses
  - **Produce** texts of predefined forms, such as job offers
  - **Report** on recurring events, such as soccer games
  - **Describe** products and services, such as hotels
- ... among various others

# Templates

## Example: Auto-Generated Hotel Descriptions



### Stay in the heart of Hannover

You're eligible for a Genius discount at Concorde Hotel am Leineschloss! To save at this property, all you have to do is [sign in](#).

Located in the city center, Concorde am Leineschloss is directly next to the gothic Marktkirche Church and a 10-minute walk from Hannover Train Station. The hotel offers 24-hour reception services and soundproofed windows.

The modern-style rooms at Concorde Hotel am Leineschloss are decorated in soft pastel tones to enhance relaxation. They all include a flat-screen TV, a work desk and private bathrooms with hairdryer.

Many local cafés and restaurants in the Old Town (Altstadt) provide hearty local food and beers.

Located 131 m from the Hannover Markthalle/Landtag Metro Station that provides direct access to Hannover's main sights. Langenhagen International Airport is a 15-minute drive away.

This is our guests' favorite part of Hannover, according to independent reviews.

Solo travelers in particular like the location – they rated it **9.5** for a one-person stay.

**Concorde Hotel am Leineschloss has been welcoming Booking.com guests since Apr 3, 2006**

#### Property Highlights

##### Perfect for a 1-night stay!

📍 Located in the heart of Hannover, this hotel has an excellent location score of 9.7

Popular with solo travelers

##### Breakfast Info

Buffet

🅅 Parking available at the hotel

##### Loyal Customers

✔ There are more repeat guests here than most other properties.

Reserve

# Template-based Generation

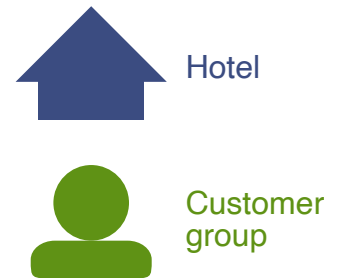
## Template-based generation

- Automatic or semi-automatic synthesis of texts based on sentence and discourse templates
- **Input.** Goal of what to generate, information represented in some way
- **Output.** A natural language text, conveying the information

## Case study

- Below, we exemplify the generation for the description of a given hotel for a given customer group
- For simplicity, we focus on a hotel's customer ratings

Notice: The given examples may not fully match practice.



## Data-to-text

- Template-based generation is a data-to-text problem, i.e., structured data is to be encoded in unstructured text
- The data may be given, or is selected as part of the generation process
- Template-based generation follows the *standard NLG process*.

# Template-based Generation

## Natural Language Generation Process

### Standard NLG process (Reiter and Dale, 1997)

1. Content determination. What to say
2. Discourse planning. When to say what
3. Sentence aggregation. What to say together
4. Lexicalization. How to say what to say
5. Referring expression generation. Decide how to refer to it
6. Linguistic realization. How to say all together

### NLG architectures

- The process shows what is to be done *conceptually* in any NLG task.
- Multiple (or all) process steps may be tackled at the same time.
- Traditionally, the process is often realized in three components:

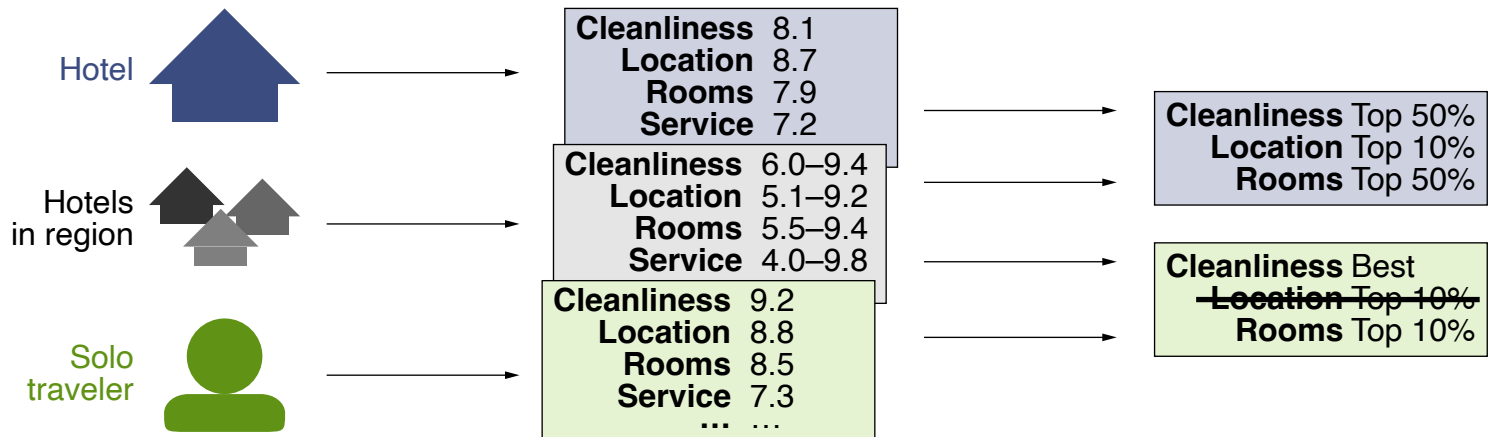


# Template-based Generation

## Content Dermination

### Content determination

- **Task.** Decide what information should be communicated in a text
- **Process.** Retrieve and filter information from some knowledge base
- **Result.** Entities, attributes, values, and relations



### Example: Hotel description generation

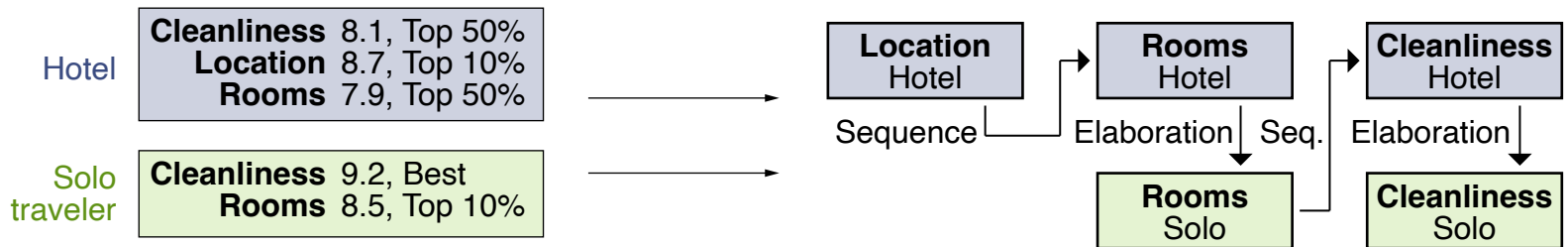
- Retrieve customer ratings of given hotel, and of other hotels in region
- Filter top  $k = 3$  above-average ratings of given hotel
- Filter respective better ratings in given customer group (solo travelers)

# Template-based Generation

## Discourse Planning

### Discourse planning

- **Task.** Organize the whole text in a coherent way
- **Process.** Order and structure information using discourse knowledge
- **Result.** A sequence or tree structure of discourse relations



### Example: Hotel description generation

- Order filtered ratings by importance according to domain knowledge
- Put hotel rating first, then group-specific rating
- Structure ratings with discourse relations (here, sequence/elaboration)

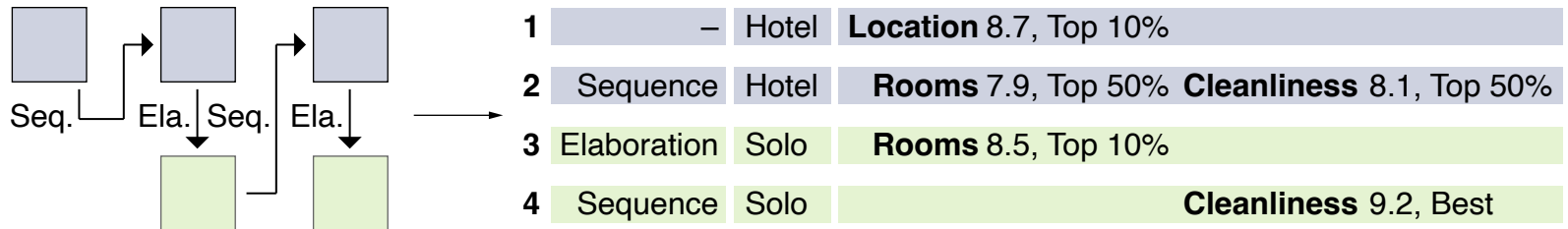


# Template-based Generation

## Sentence Aggregation

### Sentence aggregation

- **Task.** Organize individual information in a fluent and readable way
- **Process.** Aggregate the information to be communicated into sentences
- **Result.** A structured representation of each sentence



### Example: Hotel description generation

- Always aggregate entity and attribute into same sentence
- Aggregate those hotel ratings into one sentence whose rating matches
- Always have group-specific ratings in individual sentences

# Template-based Generation

## Lexicalization

### Lexicalization

- **Task.** Encode the information to be conveyed in natural language
- **Process.** Select words and phrases to express the information
- **Result.** A first representation in natural language

1	■ ■ ■	→	1	–	“our guests”	“ <b>location</b> ”	“one of the best”
2	■ ■ ■		2	“moreover”	“our guests”	“ <b>rooms and cleanliness</b> ”	“overproportionally good”
3	■ ■ ■		3	–	“solo travelers”	“ <b>rooms</b> ”	“among the most preferred”
4	■ ■ ■		4	“notably”	“solo travelers”	“ <b>cleanliness</b> ”	“the very best”, “9.2”

### Example: Hotel description generation

- Always use standard terms to refer to entities
- Select overselling descriptions for top ratings and vague descriptions for above-average ratings
- Use lexical variation in case of repeating descriptions

# Template-based Generation

## Referring Expression Generation

### Referring expression generation

- **Task.** Replace identifiers of information in a natural, yet clear way
- **Process.** Select adequate coreferences where connections are clear
- **Result.** A refined natural language representation

1	■ ■ ■ ■	→	1	–	“our guests”	“ <b>location</b> ”	“one of the best”
2	■ ■ ■ ■		2	“moreover”	“they”	“ <b>rooms and cleanliness</b> ”	“overproportionally good”
3	■ ■ ■ ■		3	–	“solo travelers”	“ <b>rooms</b> ”	“among the most preferred”
4	■ ■ ■ ■		4	“notably”	“they”	“ <b>cleanliness</b> ”	“the very best”, “9.2”

### Example: Hotel description generation

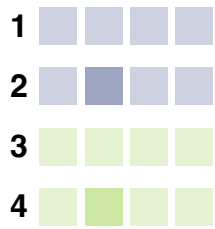
- Use standard entity and attribute terms at first occurrence
- Use personal pronoun if entity or attribute occurs two times in a row
- Always use the original value descriptions

# Template-based Generation

## Linguistic Realization

### Linguistic realization

- **Task.** Generate a morphologically and syntactically correct text
- **Process.** Fill templates and adjust text according to rules of grammar
- **Result.** The final output text



**Our** guests find the **location** of the hotel to be **one of the best** in this region, according to independent reviews.

**Moreover, they** see **rooms and cleanliness** as **overproportionally good**.

The **rooms** are **among the most preferred** by **solo travelers** in particular.

**Notably, they** consider the **cleanliness the very best** – they rated it **9.2**.

### Example: Hotel description generation

- Fill information into sentence templates
- Arrange sentences according to discourse templates
- Capitalize sentence beginnings, adjust singular/plural, and similar

# Templates

## Benefits and Limitations

### Benefits

- Very sophisticated language patterns can be specified.
- As for the techniques above, the behavior is well-controllable.
- Templates enable near-perfect effectiveness in focused tasks.

### Limitations

- They are usually domain-specific and presuppose what can be said.
- They allow for low linguistic variation only, limiting applicability.
- They require much manual labor, limiting scalability.

### Implications

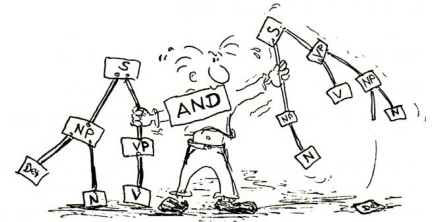
- Templates may be useful where precise recurring phrasing is needed.
- They barely suffice for more free natural communication.
- Templates are also used as prototypes in recent neural methods.

# Conclusion

# Conclusion

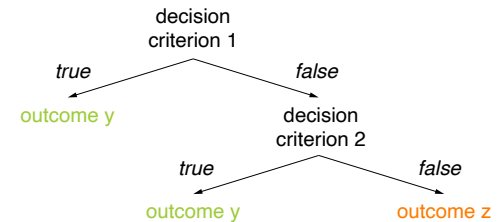
## NLP using rules

- Text analysis is based on manually defined rules.
- The rules encode human expert knowledge.
- The rules may be based on lexicons of terms.



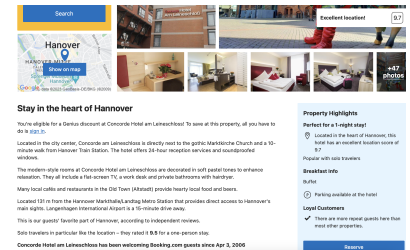
## Types of rule-based NLP

- Hand-crafted decision trees for analyzing text
- Finite-state transducers for rewriting text
- Template-based generation for creating text



## Benefits and limitations

- Behavior well-controllable for simple tasks
- Otherwise, rules easily explode or are unknown
- Not state of the art, but still used much in practice



# References

## Some content and examples taken from

- Ehud Reiter and Robert Dale (1997). Building Applied Natural Language Generation Systems. *Natural Language Engineering* 3(1):57–87. Cambridge University Press.
- Daniel Jurafsky and Christopher D. Manning (2016). *Natural Language Processing*. Lecture slides from the Stanford Coursera course.  
<https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html>.
- Daniel Jurafsky and James H. Martin (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, 2nd edition.
- Christopher D. Manning and Hinrich Schütze (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.
- Henning Wachsmuth (2015): *Text Analysis Pipelines — Towards Ad-hoc Large-scale Text Mining*. LNCS 9383, Springer.