Introduction to Text Mining

Part III: Text Mining using Rules

Henning Wachsmuth

https://cs.upb.de/css

Text Mining using Rules: Learning Objectives

Concepts

- Different types of “hand-crafted” rules for text mining
- The use of lexicons in text mining
- Benefits and limitations of hand-crafted rules

Text analysis techniques

- Text segmentation using hand-crafted decision trees
- Information extraction from text using lexicons
- Rewriting of text spans using finite-state transducers

Covered text analyses

- Tokenization
- Sentence splitting
- Attribute extraction
- Morphological analysis
- Stemming

Outline of the Course

What Is Text Mining using Rules?

Text Mining using Rules

Text mining (recap)

- Automatic discovery of information from natural language text.
- Uses several text analyses to identify and structure information.

Hand-crafted rules

- In text mining, a hand-crafted rule is a definition of how to analyze text, which has been manually defined by a human.
- Analyses include 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 text analyses.

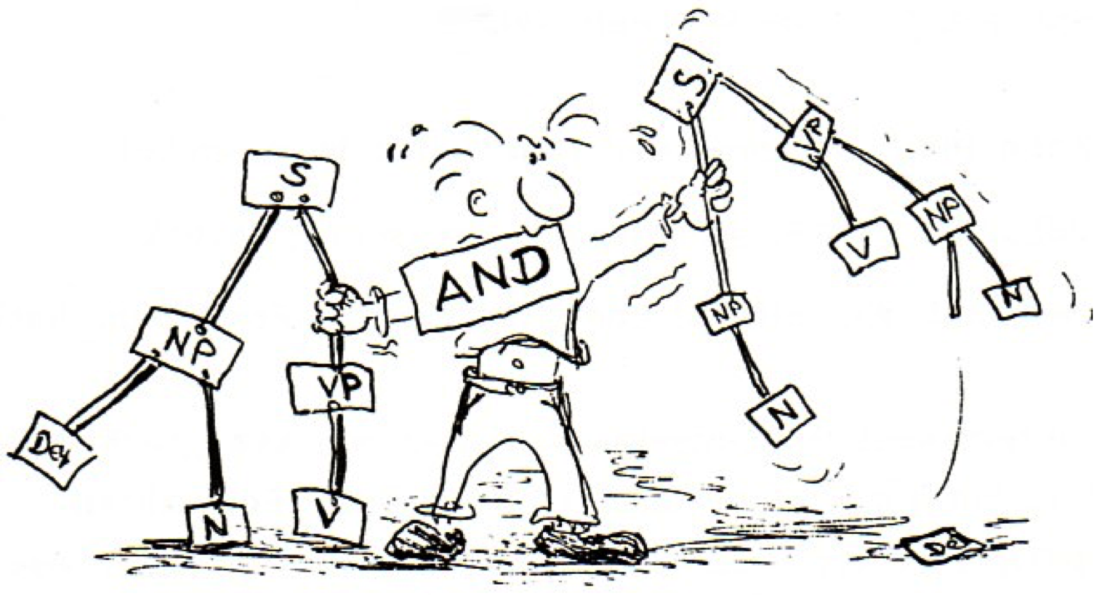
Rule-based text mining methods

- Text analyses that are done based on hand-crafted rules only.
- Aka: Knowledge-based inference or the knowledge-based approach.

Text Mining using Rules
Human Expert Knowledge

Observation

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



Encoding of knowledge

- Decision rules, lexicons, rewrite rules, string patterns, grammars, ...

Text Mining using Rules

Selected Types of Rules and Knowledge

Decision rules

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

Lexicons

Simple word lists
antagonist
anthology
antithesis
...

Lexicon with frequencies	
the	12 345 678
mining	1989
paderborn	42
...	...

Lexicon with confidences	
price	0.59
location	0.95
service	0.61
...	...

Rewrite rules

`(*vowel*) y → i` (if a span contains vowel and ends with 'y', replace 'y' with 'i')

Regular expressions

`[^a-zA-Z][tT]he[^a-zA-Z]` (matches instances of "the")

Text Mining using Rules

Types of Rule-based Methods

Covered in this part of the course

- Decision trees. Application of a hand-crafted series of decision rules to input text spans, to infer information from them.
- Lexicon matching. Matching of terms from a given lexicon with input text spans, to find information in them.
- Finite-state transducers. Matching of string patterns with input text spans, to rewrite the spans into output text spans.

Later in this course

- Regular grammars. Matching of string patterns in form of regular expressions with input text spans, to find information in them.
- Other grammars. Checking whether a given grammar generates a text span, to derive the structure of the text span.

Text Mining 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.
- Usually done with machine learning.

  Machine learning will also be dealt with later in the course.
- Aka: Statistical inference or the data-driven approach.

Rule-based vs. machine learning methods

- For most text analyses, the best results are nowadays achieved with machine learning.
- Particularly in industry, rule-based methods are still common, because they may be well-controllable and explainable.
- All rule-based methods have a statistical counterpart in some way.

Hand-crafted Decision Trees

Decision Trees

What is a decision tree?

- A decision tree is (the 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*.

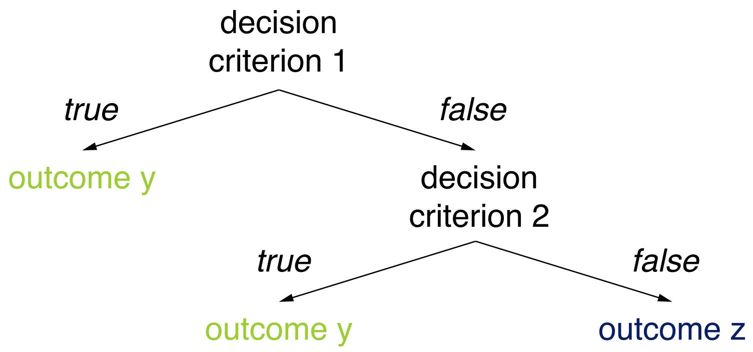
  All decision trees can be transformed into such a binary boolean form.

Decision Trees

Representations

Decision tree as a directed (tree-shaped) 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.

$$\texttt{criterion 1} \lor (\neg\texttt{criterion 1} \land \texttt{criterion 2}) \rightarrow \texttt{outcome y}$$

$$(\neg\texttt{criterion 1} \land \neg\texttt{criterion 2}) \rightarrow \texttt{outcome z}$$

Decision Trees

Hand-crafted Decision Trees

Why "hand-crafted"?

- The decision trees considered here are solely created based on human expert knowledge.
- Later, we will see decision trees that 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

Text Mining 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.

$$(\texttt{criterion 1} \wedge \ldots \wedge \texttt{criterion } n) \rightarrow \texttt{outcome y}$$

For which text analyses 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.

Text analyses covered here

- Tokenization, sentence splitting

Tokenization and Sentence Splitting

What is tokenization?

- The text 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”, “.”

What is 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.

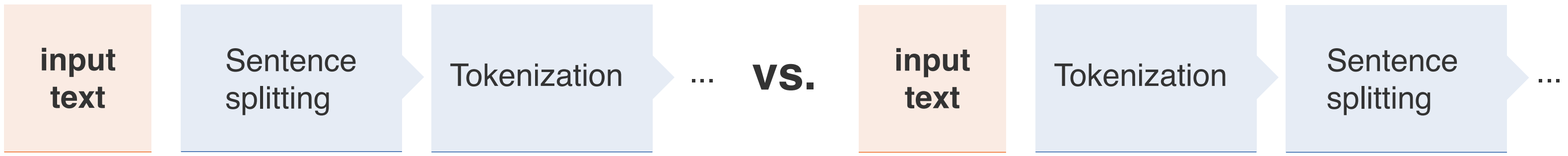
“The man sighed.”, “It’s raining cats and dogs, he felt.”

Role in text mining

- Both needed in text mining 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 to 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. reduces readability—The same holds for missing whitespaces.Really!"

  "An abbrev. reduces readability" , "The same 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

Trivial Tasks?

Controversial definitions

- A word is a unit which is bounded by spaces on both sides. (Bauer, 1988)
- Sentences end with punctuation. (Grefenstette and Tapanainen, 1994)

- Are these definitions really correct?

“Sea Containers are on the Rise

In New York Stock Exchange composite trading yesterday,
Sea Containers closed at $62.625, up 62.5 cents.”

Segmentation approaches

- Tokenization and sentence splitting may be addressed with rule-based methods as well as with machine learning.
- Own implementations not a must; algorithms exist “off-the-shelf”.
- Own implementations can be useful, in order to tune the segmentation to specific text genres.

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 (slightly modified for illustration reasons).

Sentence Splitting with a Decision Tree

Observation

- Sentence boundaries can largely be identified from the *form* of a text, i.e., without understanding the text content.
- This suggests that sentence splitting can possibly be done reasonably well with hand-crafted rules.

Sentence splitting with a decision tree

- An exemplary character-level sentence splitter is presented below.
- The approach can be understood as a binary decision tree.
- It does not require token information, 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

Pseudocode

Signature

- Input. A text given as a string.

  For simplicity already trimmed, i.e., no leading, trailing, and double whitespaces.
- Output. A list of sentences.

ruleBasedSentenceSplitting(String text)

```
List<Sentence> sentences ← ()
int start ← 0 // Character index of sentence start
int cur ← 0
while cur < text.length - 1 do
    int end ← split(text, cur) // Index after sentence end
    if end != -1 then
        sentences.add(new Sentence(start, end))
        start ← end + 1
        cur ← start
    else cur ← cur + 1
sentences.add(new Sentence(start, text.length))
return sentences
```

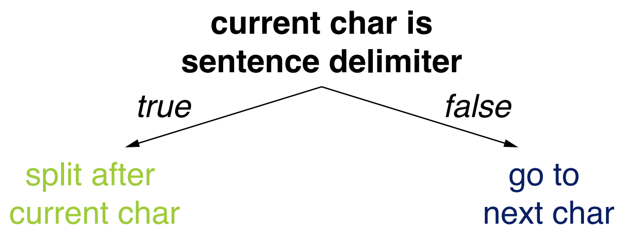
Sentence Splitting with a Decision Tree

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 Linda.” (one sentence)

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

Sentence Splitting with a Decision Tree

Example Text: Fallacious Sentence Delimiters

*“**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 (slightly modified for illustration reasons).

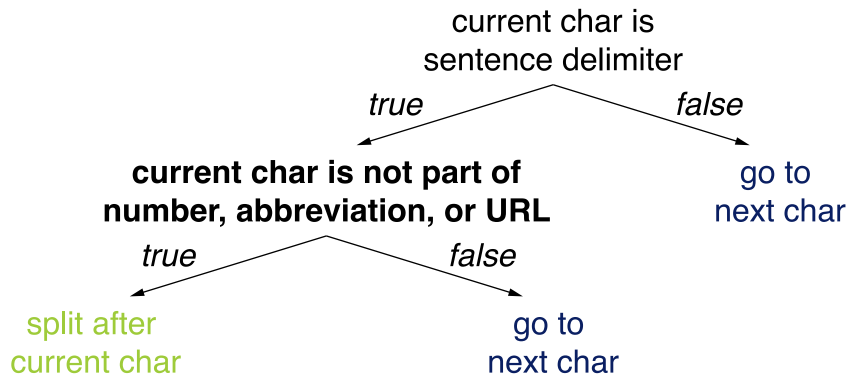
Sentence Splitting with a Decision Tree

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
```


current char is sentence delimiter
true
false
current char is not part of number, abbreviation, or URL
go to next char
true
false
split after current char
go to next char

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 Text: Other Sentence Endings

*“**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.’ [...]”

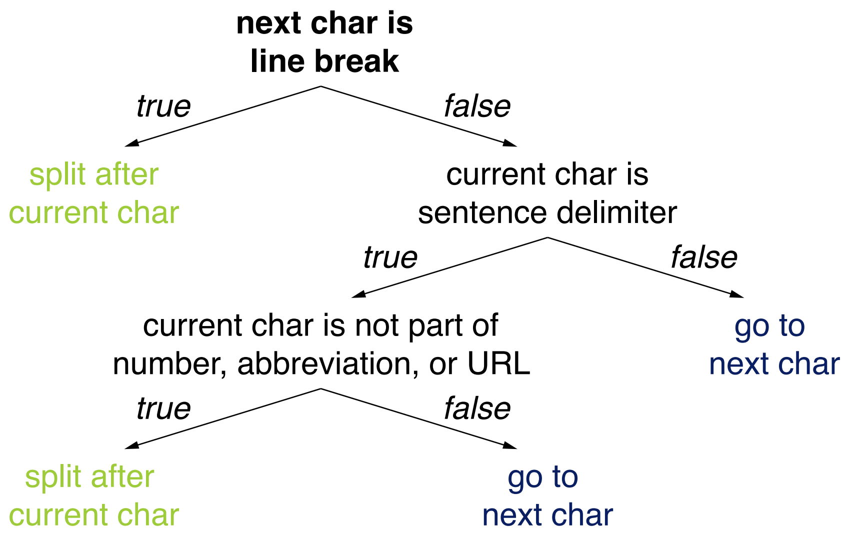
Excerpt from www.bloomberg.com/news/articles/2018-07-31/apple-forecast-tops-analysts-estimates-on-new-iphones-services (slightly modified for illustration reasons).

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 Text: Embedded Sentences

*“**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 (slightly modified for illustration reasons).

Sentence Splitting with a Decision Tree

Embedded Sentences

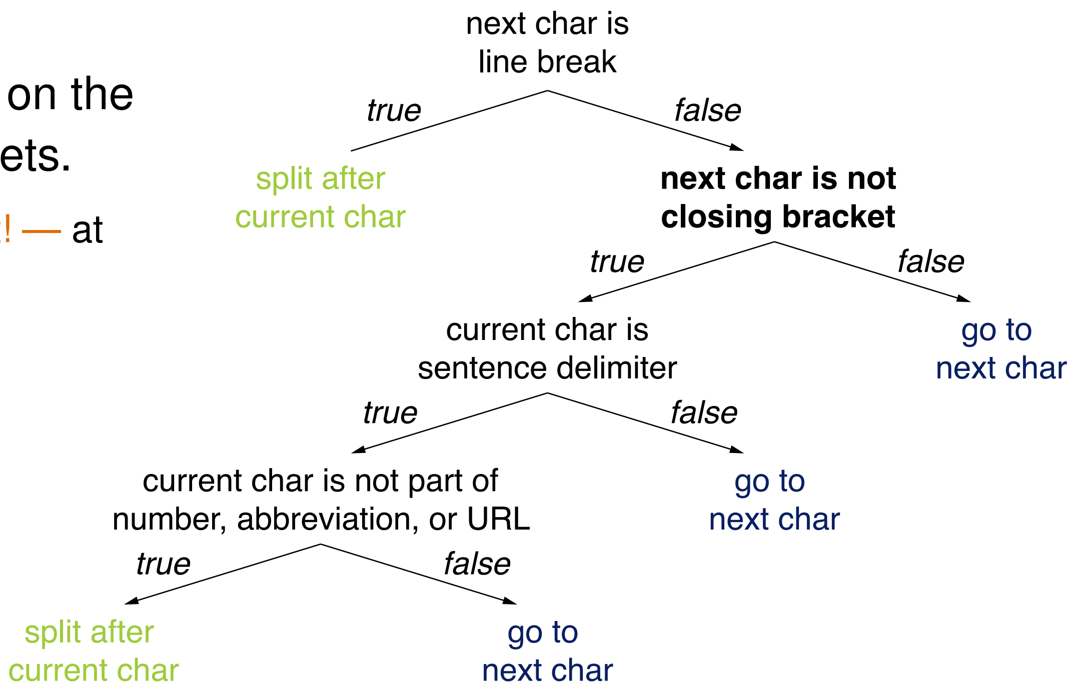
Brackets

- 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 — I love it! — at me again."



Sentence Splitting with a Decision Tree

Example Text: Quotes

*“**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 (slightly modified for illustration reasons).

Sentence Splitting with a Decision Tree

Quotes

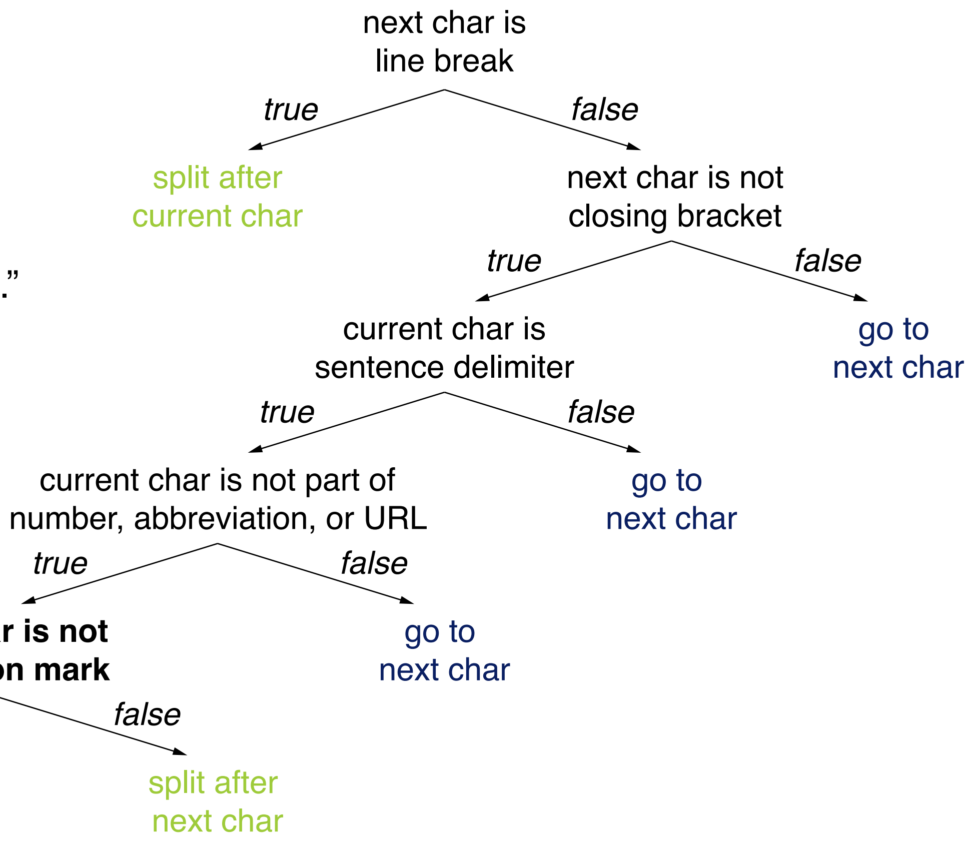
Quotation marks

- Quotation marks may shift the end of a sentence.

Challenges

- Quotations may embed sentences into others.

"'What's wrong?', Max 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 an upper-case 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

Observation

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

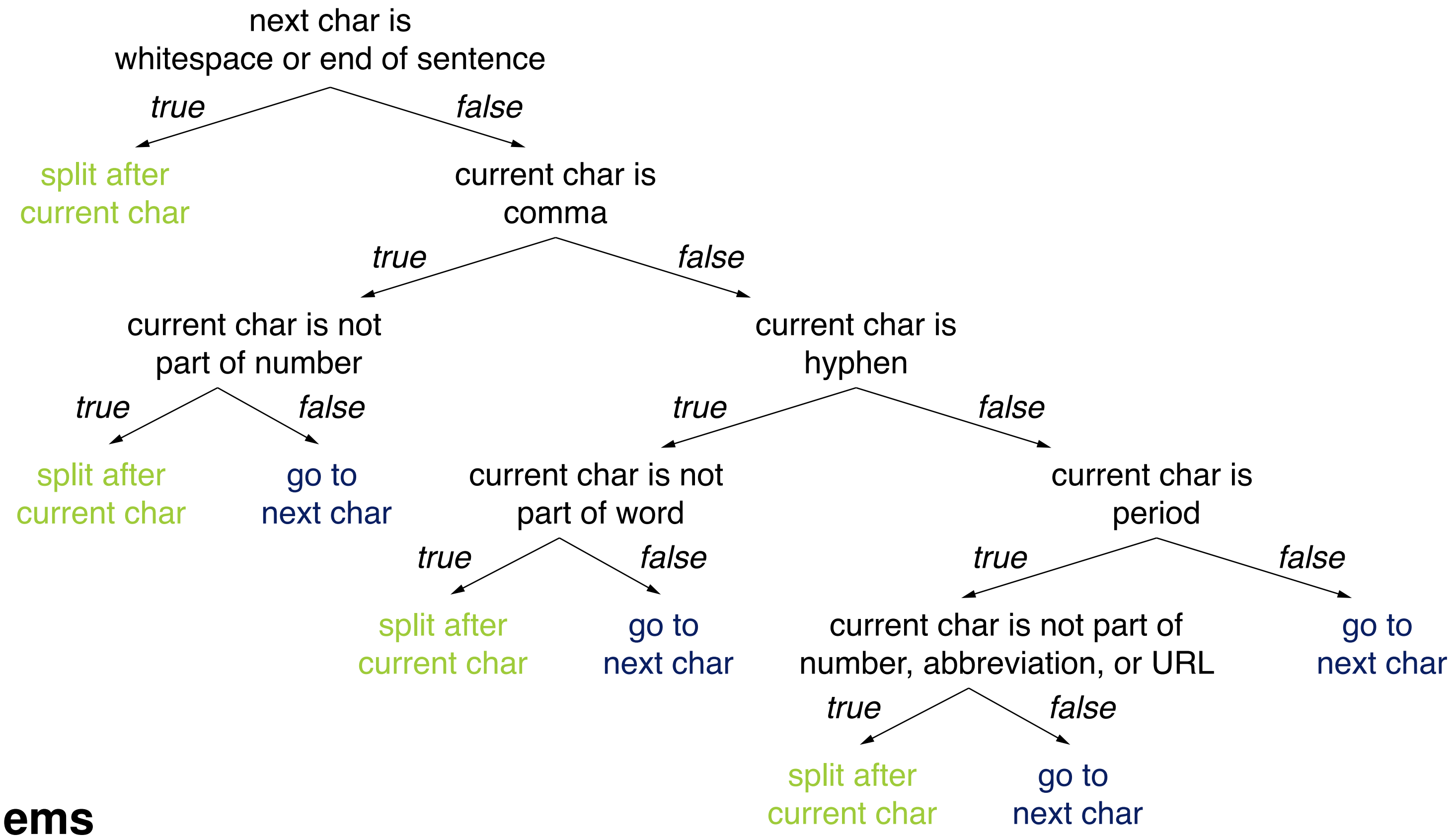
Common decision rules to find token boundaries

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

... and so on

Tokenization with a Decision Tree

Exemplary Decision Tree



Problems

- Unclear how good that works.
- Hard to tell what is missing and what effects it would have.

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”

Recovering lost token boundaries

“Thecatinthehat”	→	“the cat in the hat”
“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 different decision rules are hard to foresee.
- Adding new decision criteria may change a tree drastically.

Hand-Crafted Decision Trees

Conclusion

Benefits of hand-crafted decision trees

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

Limitations of hand-crafted decision trees

- Setting them up manually is practically infeasible for complex analyses.
- Several issues exist, for which the solution is unclear in general.

Implications

- Hand-crafted decision trees are only useful for simple analyses.
  Those with few decision criteria.
- For more complex analyses, machine learning is usually preferred.
- Still, hand-crafted decision rules may be used at a high level.

Lexicon-based Term Matching

Lexicons

What is a lexicon?

- A lexicon is a repository of terms (in terms of words or phrases) that represents a language, a vocabulary, or similar.

Observations

- Lexicons often store additional information along with a term.
- Lexicons are often (though not always) arranged alphabetically.

Lexicons

Selected Types of Lexicons

Just words

- Term list. The simplest form of a lexicon is just an explicit list of terms.
- Language lexicon. Words along with their stems, affixes, and inflections.
- Vocabulary. A list of terms that is known or used in a particular context.

Words and their definitions

- Dictionary. A list of terms along with their definition.
- Glossary. A vocabulary with definitions.
- Thesaurus. A dictionary of synonyms.

Words with structured information

- Gazetteers. Entity names (e.g., locations) along with meta-information.
- Frequency list. Terms with their frequency in some text collection.
- Confidence lexicons. Terms along with the confidence (or probability) that they represent a specific concept.

Lexicons
Examples

Term list

“a”, “AA”, “AAA”, “Aachen”, “aardvark”, “aardwolf”, “aba”, “abaca”, “aback”, ...

Vocabulary

Formal words		
admittedly	furthermore	meanwhile
consequently	hence	merely
conversely	incidentally	moreover
considerably	indeed	nevertheless
essentially	likewise	...

Informal words		
bastard	cuz	iffy
booze	damn	kinda
bummer	dope	puke
cop	dude	sorta
crap	hell	...

Frequency list

Word	Count	Word	Count
the	23243	a	12780
i	22225	you	12163
and	18618	my	10839
to	16339	in	10005
of	15687	...	...

Lexicons

Lexicon-Based Term Matching

Use of lexicons in text mining

- A given lexicon can be used to find all term occurrences in a text.
- The existence of a given term in a lexicon can be checked.
- The density or distribution of a vocabulary in a text can be measured.

Selected text analyses that may be based on lexicons

- Identification of terms, e.g., acronyms (see above)
- Attribute extraction, e.g., product aspects (see below)
- Morphological analysis of words (see further below)
- Sentiment analysis of texts (later in this course)
- Analysis of style, e.g., formal vs. informal language
- Named entity recognition, e.g., location names
- Spelling correction of words

... and so on

Attribute Extraction with Lexicon-based Term Matching

What is attribute extraction?

- The text analysis that extracts certain attributes of some entity from text.
- Input. A text, usually at least split into tokens and sentences.
- Output. The list of all extracted attributes (including their text positions).

"We spent one night at that hotel. The service at the front desk was perfect and our room looked clean and cozy... but this alone never justifies the price!"

Role in text mining

- Used for tasks such as aspect-based sentiment analysis or the extraction of complex events.

Example here: Extraction of hotel aspects

- An approach that creates a lexicon of aspects covered in hotel reviews to then use it for extraction is presented below.
- The approach can easily be transferred to other terms.

Attribute Extraction with Lexicon-based Term Matching

Why is lexicon matching not trivial?

- Some terms sometimes but not always denote an aspect of an entity.

"The food in the hotel was great." vs. "We left the hotel to go for food."

Hotel aspect confidence lexicon

- A lexicon of hotel aspects where each term is assigned a value $\in \{0, 1\}$.
- The value represents the confidence that a term really is a hotel aspect.

Approach in a nutshell

1. Create confidence lexicon based on a collection of reviews.
2. Choose a threshold $\tau \in [0, 1]$.
3. Extract each term in a new review that is in the lexicon and that has a confidence value of at least τ.
4. Prefer longer terms over shorter terms.

"in-room service" vs. "service"

Attribute Extraction with Lexicon-based Term Matching

Creation of a Confidence Lexicon

How to compute confidence values?

- Assume we are given a training set of hotel reviews where all aspects $a_1, \ldots, a_k$ have been marked.
- Then the confidence value of an aspect a_i is given by the fraction of marked occurrences a_i under all occurences of a_i in the training set.

Excerpt from confidence lexicon (derived from 900 training TripAdvisor reviews)

Hotel aspect	Confidence
minibar	1.00
towels	0.97
a/c	0.92
wi-fi	0.83
front desk	0.74
shuttle	0.65
alcohol	0.50
waiter	0.40
buffet	0.21
people	0.01

Attribute Extraction with Lexicon-based Term Matching

Pseudocode

Signature

- Input. A tokenized `text`, a confidence `lexicon`, and a threshold τ.
- Output. A list of extracted aspects.

```
extractLongestAspects(String text, Map lexicon, double τ)
    List<Term> aspects ← ()
    List<Token> tokens ← text.toTokens()
    int maxTokens ← lexicon.getLongestAspect().length
    for int i ← 0 to tokens.length-1 do
        int j ← min{i+maxTokens-1, tokens.length-1}
        while j ≥ i do
            String term ← text[tokens[i].begin, tokens[j].end]
            if lexicon.contains(term) and lexicon.get(term)≥τ then
                aspects.add(new Aspect(term.begin, term.end))
                i ← j
                break // leave while loop
            j ← j - 1
    return aspects
```

Attribute Extraction with Lexicon-based Term Matching

Evaluation of the Approach

What does the threshold τ do?

- The higher τ, the more likely an extracted aspect really is the aspect, but the fewer aspects will be extracted.
- τ trades *precision* (i.e., the proportion of correctly extracted aspects) against *recall* (i.e., the proportion of found aspects).

  The harmonic mean of precision and recall is the so-called F_1-score.

Evaluation of the approach (on 600 test TripAdvisor reviews)

τ	**Precision**	**Recall**	**F_1-score**
0.1	0.739	**0.460**	0.566
0.2	0.768	**0.460**	0.575
0.3	0.785	0.457	0.578
0.4	0.794	0.456	**0.580**
0.5	0.808	0.448	0.576
0.6	0.820	0.429	0.563
0.7	0.846	0.354	0.499
0.8	0.864	0.284	0.427
0.9	**0.893**	0.144	0.265

Attribute Extraction with Lexicon-based Term Matching

Some Insights from Analyzing Hotel-related Terms *

Some the most often named aspects (in 2100 reviews on TripAdvisor)

1. Room. Mentioned in 80% of all reviews.
3. Location. Seen positive in 85% of all reviews.
8. Service. If seen negative, highest overall score in 0% of all reviews.
20. Towels. Seen negative in 67% of all reviews.
24. Parking. If seen negative, highest overall score in 12% of all reviews. But if seen positive, lowest score in 0% of all reviews.

Specific tokens (in 44,220 user comments on HRS)

- Most frequent.
  “the”, “and”, “to”, “was”, “a”, “in”, “very”, “is”
- Most clearly positive.
  “close”, “easy”, “friendly”, “modern”, “nice”
- Most clearly negative.
  “been”, “because”, “booked”, “cold”, “dirty”, “or”, “hot”, “so”, “them”

Lexicon-based Term Matching

Conclusion

Benefits of lexicon-based methods

- Lexicon-based methods are particularly reliable for unambiguous terms.

  For certain types of terms, such as location names, huge gazetteer lists exist.
- Lexicons with confidence values can be used to trade the precision against the recall of matchings.

  Such lexicons can be built from training data.

Limitations of lexicon-based methods

- Information that is not in the employed lexicons can never be found.
- Ambiguous terms require other methods for disambiguation.
- Composition of different information (as in relations) are hard to handle with lexicon-based approaches.

Implications

- Lexicons are most suitable for (more or less) closed-class terms.
- Lexicons are often useful as part of other methods.

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 is an extension of an FSA that reads one string and generates another. It represents the set of all relations between two sets of strings.

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

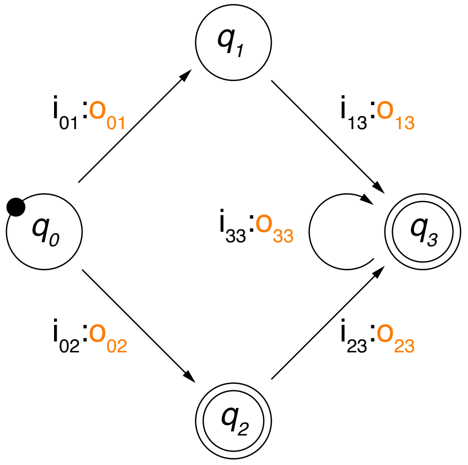
Q A finite set of $n > 0$ states, $Q = \{q_0, ..., q_n\}$.

Σ 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, $F \subseteq Q$.

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



Finite-State Transducers (FSTs)

Text Mining 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 Σ.
- Set relator. Compute relations between sets of strings I and O, such that $i \in I$ and $o \in O$.

Text analyses covered here

- Morphological analysis, word normalization

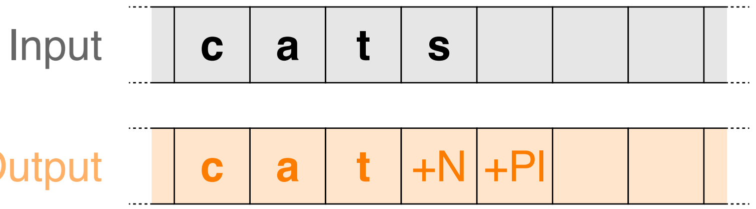
What is morphological analysis?

- The text analysis that breaks down a word into its different morphemes.
- Sometimes used in text mining 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.

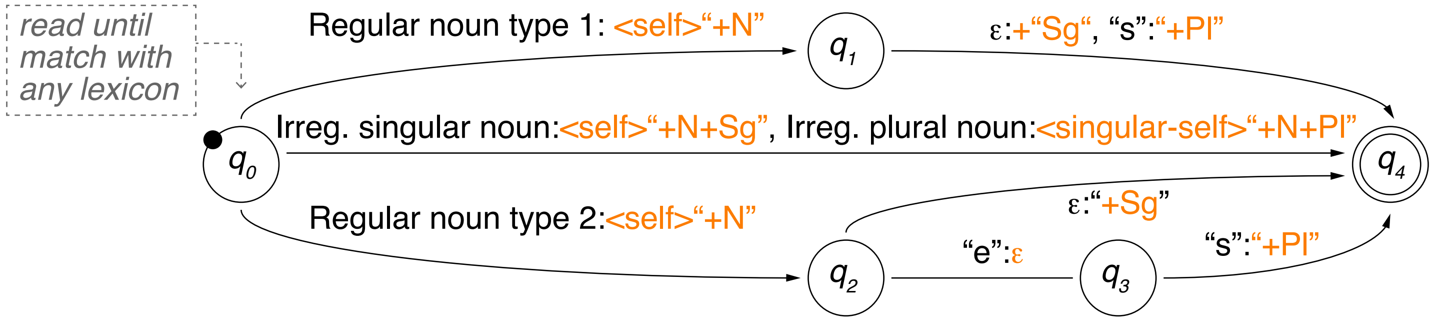


Knowledge needed for morphological analysis

- Lexicon. Stems and 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 *



read until match with any lexicon
Regular noun type 1: <self>"+N"
q1
ε:+"Sg", "s":"+Pl"
q0
Irreg. singular noun:<self>"+N+Sg", Irreg. plural noun:<singular-self>"+N+Pl"
q4
ε:"+Sg"
Regular noun type 2:<self>"+N"
"e":ε
"s":"+Pl"
q2
q3

(ε empty word, <self> output is input, <singular-self> 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

What is word normalization?

- The conversion of all tokens into a canonical form, thereby defining equivalence classes of terms.

  Technically, a token is not converted, but its canonical form is stored in addition.
- Used in text mining 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 methods: stemming and lemmatization.

What is stemming?

- The text analysis that identifies the stem of a token.

“playing” $\rightarrow$ “play” “derive” $\rightarrow$ “deriv” “am” $\rightarrow$ “am”

What is lemmatization?

- The text analysis that identifies the lemma of a token.

“playing” $\rightarrow$ “play” “derive” $\rightarrow$ “derive” “am” $\rightarrow$ “be”

Stemming with Finite-State Transducers

Stemming with affix elimination

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

  "connects", "connecting", "connection" $\rightarrow$ "connect" (correct stem)

  "automate", "automatic", "automation" $\rightarrow$ "automat" (not a real stem)

- The elimination may be based on prefix and suffix forms only.

  Both prefixes and suffices are closed classes within a language.

Porter Stemmer

- The most common stemmer for English is sketched below.
- The Porter Stemmer is based on a series of cascaded rewrite rules.
- It 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 token).
- Output. The 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 S.endsWith(S1) and holds(<premise>, S-S1) 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 defining 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: Selection of 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: Functions of 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. $\text{Y} \rightarrow \text{I}$
- 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 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

Observations

- The application of rules is trivial. The knowledge is *in* the rules.
- The rules are specific to English (adaptation to other languages exist).

Issues

- Difficult to modify: 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”

Stemming with Finite-State Transducers

Combining Rules with Lexicons *

Krovetz Stemmer

- Adds lexicons to the finite-state transducer again.
- The lexicon captures well-known cases.
- The patterns capture new words not found in the lexicon.

Approach in a nutshell

1. If input token present in lexicon, replace with stem.
2. If not present, check token for choppable inflection suffixes.
3. If chopped token present in lexicon, replace with stem.
4. If still not present, try to add different suffixes.

Properties

- Produces words, not stems (more readable, similar to lemmatization).

  Captures irregular cases such as “is”, “be”, and “was”.

- Comparable effectiveness to Porter stemmer.

  Fewer wrongly found stems, some more missed stems.

Stemming with Finite-State Transducers

Krovetz 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.”

Krovetz-stemmed text

“A relevant document will describe marketing strategy carry out by U.S. company for their agriculture chemical, report prediction for market share of such chemical, or report market statistic for agrochemic pesticide, herbicide, fungicide, insecticide, fertilizer.”

Finite-State Transducers (FSTs)

Conclusion

Benefits

- Similar to decision trees, precise rules can be specified with human expert knowledge.
- Behavior of FSTs for focused rewriting tasks is well-controllable.

Limitations

- FSTs are meant for applications where an output text is to be created based on an input text.
- FSTs tend to overgeneralize or to have low coverage.
- For more complex tasks, FSTs get very complicated (as decision trees).

Implications

- FSTs should rather be used where approximate results are sufficient.
- Some tasks can be easier accessed with regular expressions.

Conclusion

Pros and Cons of Text Mining using Rules

Pros

- Rules can often be derived from world knowledge and human intuition.
- Human experts can define very precise rules for many tasks.
- No or few training data of the given task is needed.
- Behavior can be controlled well — as long as the tasks remain simple.
- Behavior can mostly be easily explained.

Cons

- Hand-crafted rules are hard to handle for more complex tasks.
- Not practical where a weighting of several text features is needed.
- For some tasks, it is just unclear how to specify rules manually.

  A typical example is authorship attribution.

Alternatives

- Grammar-based approaches, such as regular expressions.
- Machine learning methods that learn statistical weightings of features.

  Features can represent rules, regular expressions, or something similar.

General Observations about Text Mining

Correctness vs. effectiveness

- Text mining algorithms are rarely correct, i.e., their output contains errors from time to time.
- Rather, they have a certain effectiveness in terms of precision, recall, ...

Types of errors

- There are two general kinds of errors, often with a trade-off.
- False positives. Wrong information that was inferred from a text.
- False negatives. Correct information that was not inferred from a text.

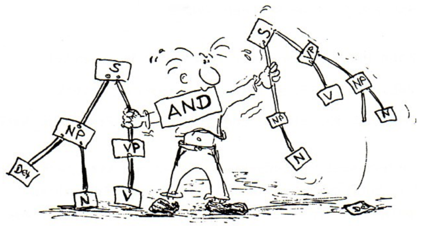
Need for data

- Training data is needed to develop certain text mining methods.
- Test data is needed to evaluate the effectiveness of methods.
- The available data is a (if not *the*) decisive factor in text mining.

Summary

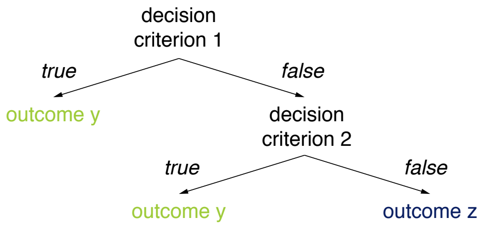
Text mining using rules

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



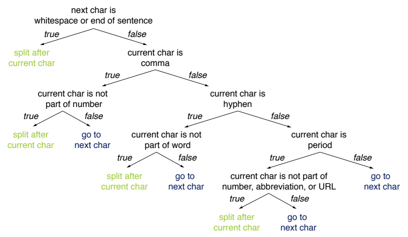
Types of rule-based text mining

- Decision trees with series of conditional rules.
- Lexicon-based matching of specific terms.
- Finite-state transducers for rewriting text.



Benefits and limations

- Behavior can be controlled well for simple tasks.
- Often, too many rules needed or rules unknown.
- State-of-the-art methods are often *not* rule-based.



References

Some content and examples taken from

- Daniel Jurafsky and Christopher D. Manning (2016). Natural Language Processing. Lecture slides from the Stanford Coursera course. https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html.
- Matthias Hagen (2018). Natural Language Processing. Slides from the lecture at Martin-Luther-Universität Halle-Wittenberg. https://studip.uni-halle.de/dispatch.php/course/details/index/8b17eba74d69784964cdefc154bb8b95.
- 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.