Basic Text Processing

Tokenizing

Sentence Splitting

Word Normalisation

First steps of the processing

- For this lecture, we start with a simple plain text input, and:

  1. Detect „meaningful“ tokens
  2. Detect „words“
  3. Detect „stems“ for words
  4. Detect sentences

Quelle: Java ist auch eine Insel

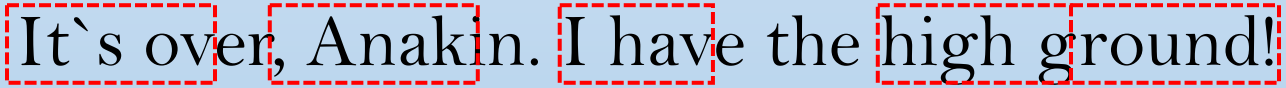
- Using:
  - Unsupervised heuristics (Regular Expressions, Porter-Stemmer algorithm, „Punkt“ tokenizer)
  - Simple techniques with Machine Learning

Task description

- Input is plain text (as a string)

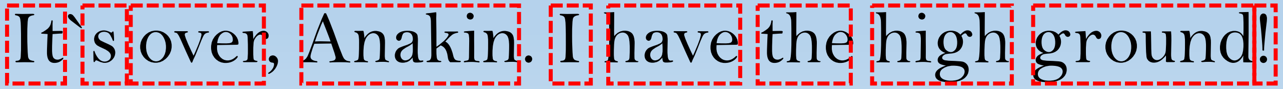
It`s over, Anakin. I have the high ground!

- Output is an arbitrary non-overlapping segmentation of the text. We call each segment a **token**



It`s over, Anakin. I have the high ground!

- But we are trying to produce tokens that make sense



It`s over, Anakin. I have the high ground!

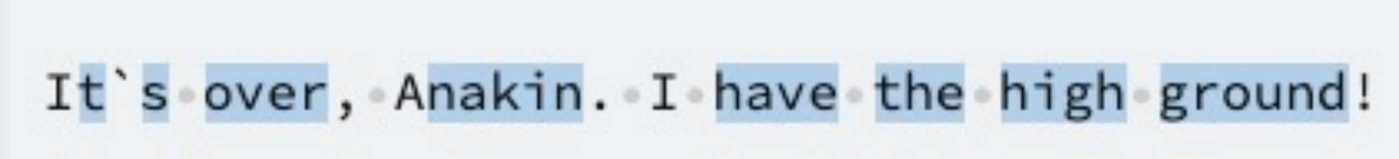
Rule based recognition – Regular Expressions

- We can write a grammar based on regular expressions
- Here, a token is a sequence of characters that matches a pattern

Pattern	Matches	First matching token
`[A-Z]`	An upper case letter	`Drenched Blossoms`
`[a-z]`	A lower case letter	`my beans were impatient`
`[0-9]`	A single digit	`Chapter 1: Down the Rabbit Hole`

Rule based recognition – Regular Expressions

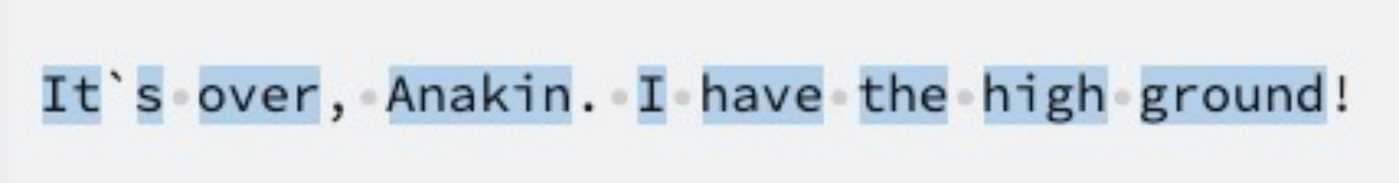
- Example: [a-z]+



It`s over, Anakin. I have the high ground!

➔ does not find capitalized words or punctuation

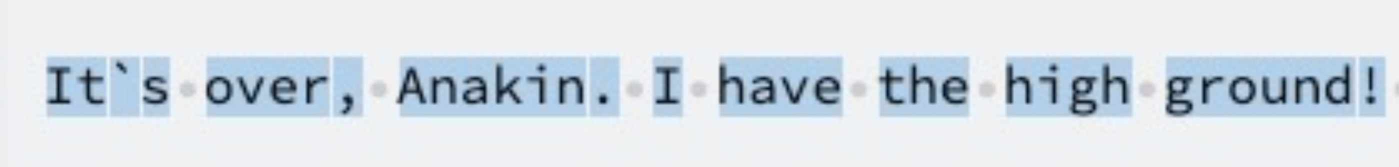
- Example : ([A-Z])?[a-z]*



It`s over, Anakin. I have the high ground!

➔ does not find punctuation

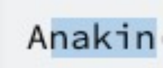
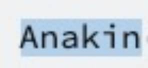
- Example : ([A-Za-z]+)|([.,;!?`])

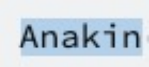


It`s over, Anakin. I have the high ground!

https://regexr.com/

Rule based recognition – Regular Expressions

- Failure categories:
- The process of creating the regular expressions brings 2 classes of failures:
  - Match on strings that should not be matched on: (False-positives or FP)
  -  Anakin instead of  Anakin
  - ➔ "nakin" = 1 False-positive

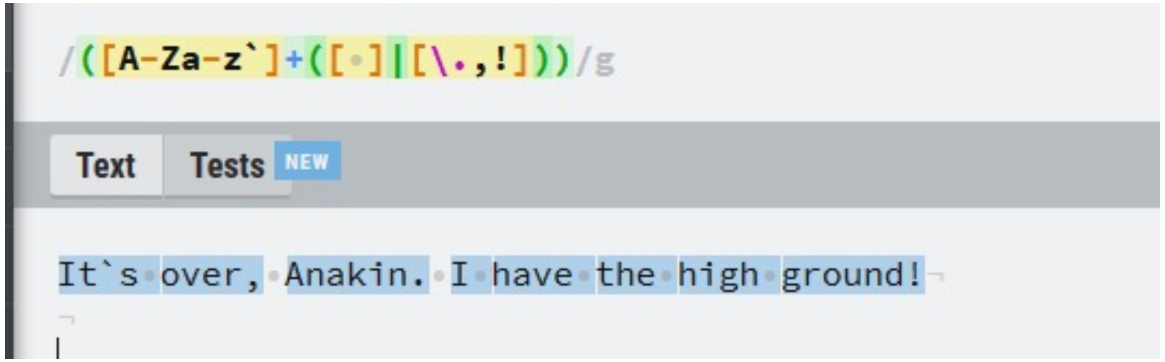
  - Do not match on strings that should be matched on: (False-negative or FN)
  -  Anakin was not found
  - ➔ Missing "Anakin" = 1 False-negative

Rule based recognition – Regular Expressions

- Write a grammar based on regular expressions
- Example:

```
Text := (rToken(rWS|rBreak))+
rToken := [A-Za-z]+
rWS = [ \t\r\n]
rBreak := [.,;:!?]
```

Results in:



```
/([A-Za-z`]+([ ]|[\.,!]))/g
```

```
It`s over, Anakin. I have the high ground!
```

https://regexr.com/

Tokenization of Natural Language

- We saw some generic ways to tokenize textual input
- Main challenges for natural language
  - What even is a useful (or meaningful) token
    - Daily dose of Ibuprofen according to ICD-10-GM-2017:I10

    ➔in general, this depends on your goals and your domain! There is no one-trick pony
  - Classical questions:
    - Abbreviations: dept. (department)
    - Phone numbers (0123-565141)
    - Law (Zur Anwendung d. § 311b Abs. 2)
    - Email, Headers, Currencies, domain specific codes, …

How to build/apply a rule-based tokenizer

- A domain expert (might be you) defines what a meaningful token is
  ➔The tokenizer you use has to be adaptable in some way

- You can make use of several implementations of „General-purpose“ approaches
  - E.g. in NLTK:
    - Whitespace-Tokenizer
    - RegExpTokenizer
    - „Punkt“-Tokenizer (we will go into more detail here later)

Tokenizing using Machine Learning

- For this lecture, we define a scenario for classification:

1. What is an instance? (It represents the structure of our problem)
2. How to describe an instance (In terms of features)
3. What kind of classifier (SVM, MaxEnt, Decision Tree, Deep Learning, …)
4. How to **train** the classifier (This, yet again, depends on the problem)
5. How to **apply** the classifier?
6. How to evaluate the classifier (What data, what kind of metrics)

Tokenizing using Machine Learning

1. What is an instance?
   - For tokenization, we only have the text at hand, so we could convert every character into an instance (this is not the only option, but an intuitive one!)

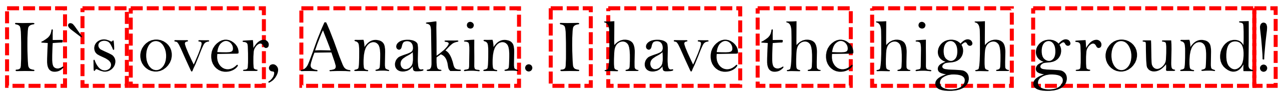
It`s␣over,␣Anakin.␣I␣have␣the␣high␣ground!

First three instances

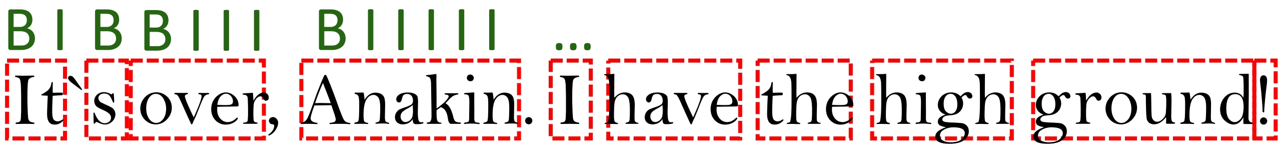
Tokenizing using Machine Learning

1. What is an instance?

- Reminder, looking for a meaningful split in tokens



- These blocks can be represented by the so-called **B**eginning/**I**nside/**O**utside notation (BIO tagging)

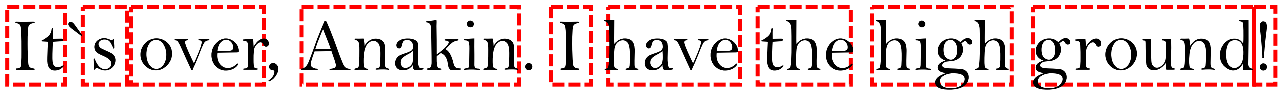

B I B B I I I B I I I I I ...
It`s over, Anakin. I have the high ground!

- All spaces get the symbol "O", so they are not part of a token! (left out for readability)

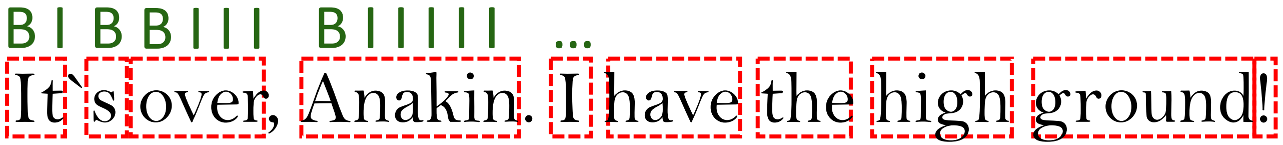
Tokenizing using Machine Learning

1. What is an instance?

- Reminder, looking for a meaningful split in tokens


It`s over, Anakin. I have the high ground!

- These blocks can be represented by the so-called **B**eginning/**I**nside/**O**utside notation (BIO tagging)


B I B B I I I B I I I I I ...
It`s over, Anakin. I have the high ground!

- An instance is a character, and for each character one of the symbols "B" or "I" or "O" is predicted

Tokenizing using Machine Learning

2. How to describe an instance (In terms of features)?
   - Features are basically the way we tell the computer what we are dealing with
   - Features might be classical hand-crafted or Embeddings generated from some other (smart) procedure

- Example: It`s␣over,␣Anakin.␣I␣have␣the␣high␣ground!

Feature-Index	Feature Description	Value
0	Current Character	o
1	Previous Character	␣
2	Next Character	v

Tokenizing using Machine Learning

- In general, we could use any feature we want (depending on the classifier of course,…)
- Potential features for tokenization
  1. Characters in neighbourhood (at positions: -1,1,-2,2, …)
  2. Strings in an interval around the character (interval [-2,2],[-2,1],[-2,0], …)
  3. Dictionaries with existing words! (And whether some substring is part of it)
  4. The current character is a „.“ and is part of an abbreviation dictionary
  5. …

Tokenizing using Machine Learning

3. What kind of classifier (SVM, MaxEnt, Decision Tree, Deep Learning, …)

4. How to **train** the classifier?
   - Convert each instance into machine readable representation (our feature space)
   - Enrich each instance with a label
     - Example: **B**/**I**/**O** tagging for each character
   - Train your classifier using an appropriate training algorithm:
     - Gradient Descent
     - L-BFGS
     - Additive Training
     - Improved Iterative Scaling
     - …

Tokenizing using Machine Learning

5. How to apply the classifier?

- In our case, it is simple:
- Use the trained classifier to predict a {B,I,O} label for each instance (character)
  Details might yet again vary for each classifier

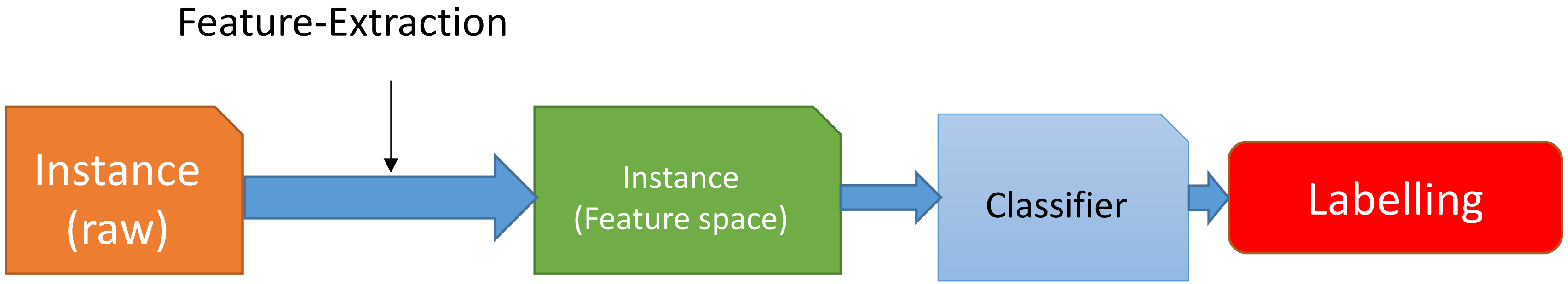
Tokenizing using Machine Learning

6. How to evaluate the classifier?

- We compare the predictions of our classifier with the manually assigned labels from our test data
- Several options:
  - Label-accuracy $\mathrm{acc} = \frac{|correct|}{|goldlabels|} = \frac{|goldlabel=classifierlabel|}{|goldlabels|}$
  - Label-F1 (based on FP and FN)
  - Entity-F1
  - Weighted Entity-F1
  - Micro/Makro F1
  - …
- Will be addressed in more detail in the chapter "Evaluation"

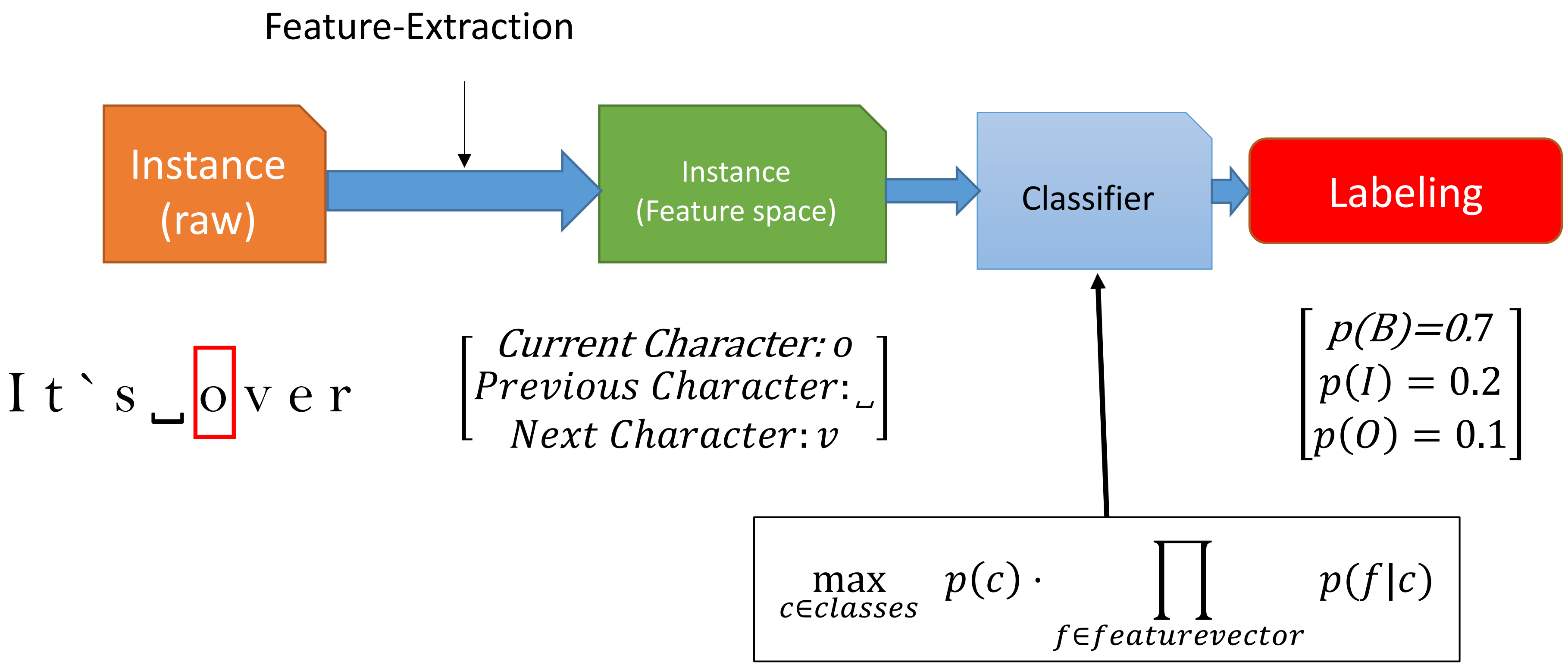
Machine Learning for Tokenization

- We are trying to learn a classifier
  - A classifier is any sort of parametric function, learning refers to determining specific values for the parameters
  - A classifier takes an instance, represented as a feature vector as its input
  - And produces a labelling (might be a vector of probabilities)



Machine Learning for Tokenization

- An example:



I t ` s ␣ o v e r

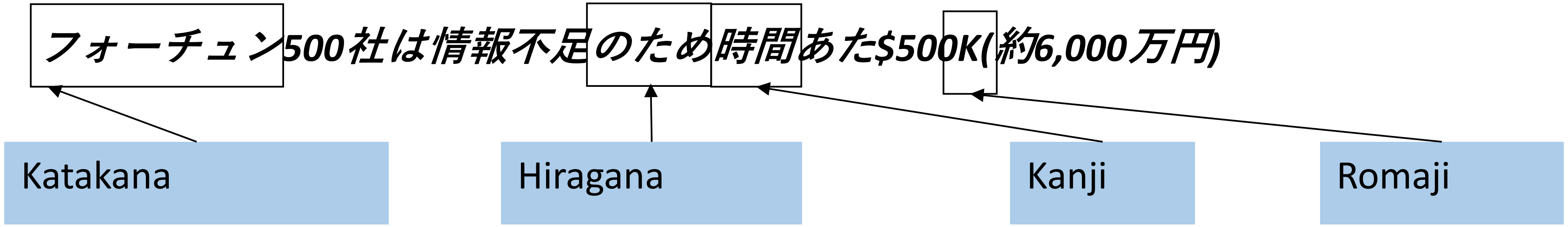
$$\begin{bmatrix} Current\ Character: o \\ Previous\ Character: ␣ \\ Next\ Character: v \end{bmatrix}$$

$$\begin{bmatrix} p(B)=0.7 \\ p(I) = 0.2 \\ p(O) = 0.1 \end{bmatrix}$$

$$\max_{c\in classes} \; p(c) \cdot \prod_{f \in featurevector} p(f|c)$$

Tokenization: Language Issues

- No spaces in Japanese and Chinese :
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida



フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

Tokenization: Language Issues

- A simple procedure for this is „Maximum Matching“

1. Given an input string and a dictionary
2. Place a pointer to the begin of the string
3. Find the longest word of the dictionary, starting at the position of the pointer
4. Move pointer forward accordingly
5. Go to 3

Max-match segmentation illustration

- Thecatinthehat

the cat in the hat

- Thetabledownthere

the table down there

theta bled own there

- Doesn’t generally work in English!

- But works astonishingly well in Chinese
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达

- Modern probabilistic segmentation algorithms even better