ZD AUTOMOTIVE
Focus on Technology Shaping the Future

Konzeption und Implementierung eines auf maschinellem Lernen basierenden Algorithmus zur Verbesserung der Effizienz der Testautomatisierung für automotive Infotainmentsysteme

Abschlusspräsentation zur Bachelorarbeit
von Chuxuan Li

Technische Hochschule Ingolstadt

21. Februar 2018

Erstprüfer: Prof. Dr. –Ing. Daniel Großmann, THI
Zweitprüfer: Prof. Dr. –Ing. Markus Bregulla, THI
Betreuer: Dipl.-Ing. Qiang Zhou, ZD Automotive GmbH

Gliederung

1. Motivation
2. Einführung in die Themengebiete
3. Konzeption des Algorithmus
4. Implementierung
5. Ergebnisse und Auswertung
6. Zusammenfassung und Ausblick

1. Motivation

Hintergrund

- Infotainmentsystem: wichtige Schnittstelle zwischen Fahrzeug und Außenwelt
- Absicherungsaufwand wegen ansteigender Komplexität und Interaktion von Infotainment-Funktionen
- Überprüfung der Textanzeigen: eine der Hauptaufgaben der HMI-Absicherung (Human-Machine-Interface)
  - Erhöhung der Testeffizienz manuell → automatisch

Quelle:[1]

1. Motivation

Ziel des Algorithmus

- Bildverarbeitung
- Textbereiche lokalisieren
- Texterkennung

- Texte
- Koordinaten-informationen
- Fehlererkennung

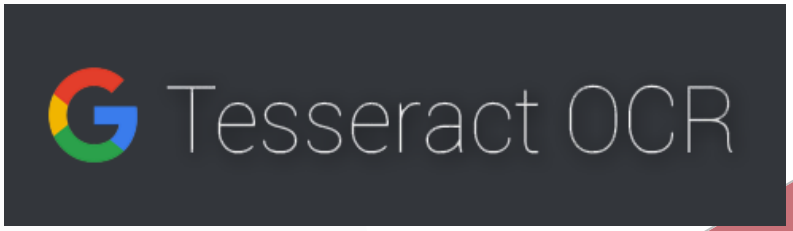
2. Einführung in die Themengebiete

OpenCV (Open Source Computer Vision Library)

- Freie Programmbibliothek für Bildverarbeitung und maschinelles Sehen

Google Tesseract OCR

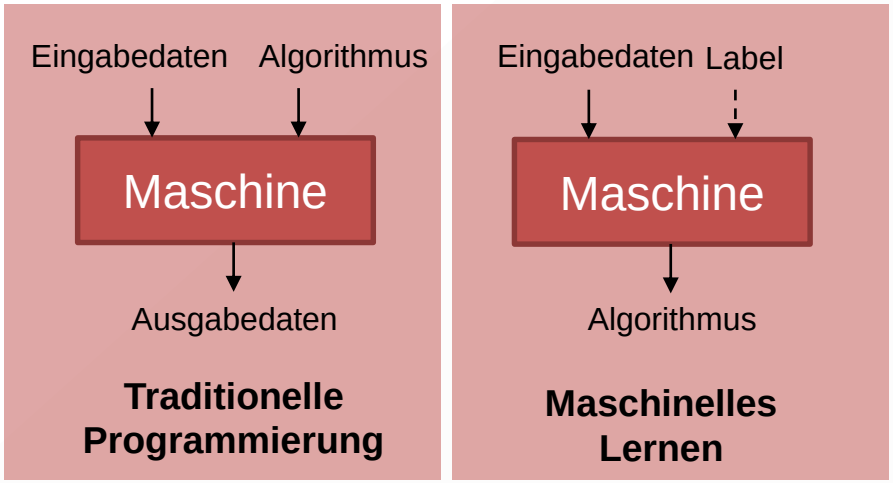
- Open-Source-OCR-Engine zur Texterkennung
- Langsam bei der Erkennung chinesischer Schriftzeichen
- Lernfähig für neues Erkennungsmodul



2. Einführung in die Themengebiete

Maschinelles Lernen

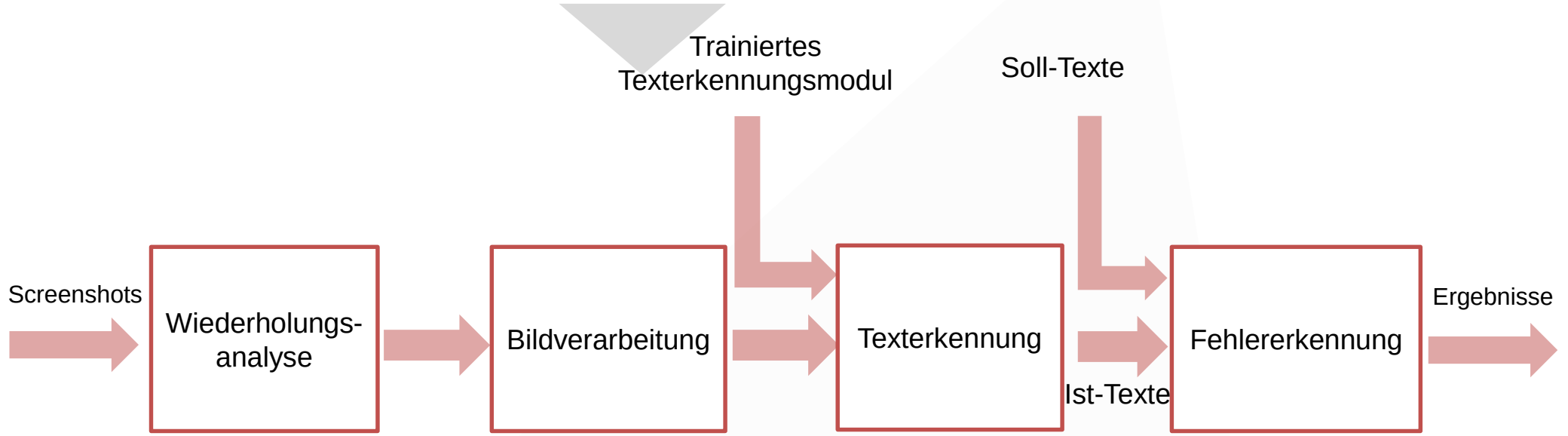
- **Begriff**: eine rechnerische Methode, die es einem Rechner ermöglichen soll, menschliches Lernverhalten nachzubilden[2].



- **Grundansätze**:
  - Überwachtes Lernen (Klassifikation, Regression)
  - Unüberwachtes Lernen (Clustering)
  - Verstärkendes Lernen

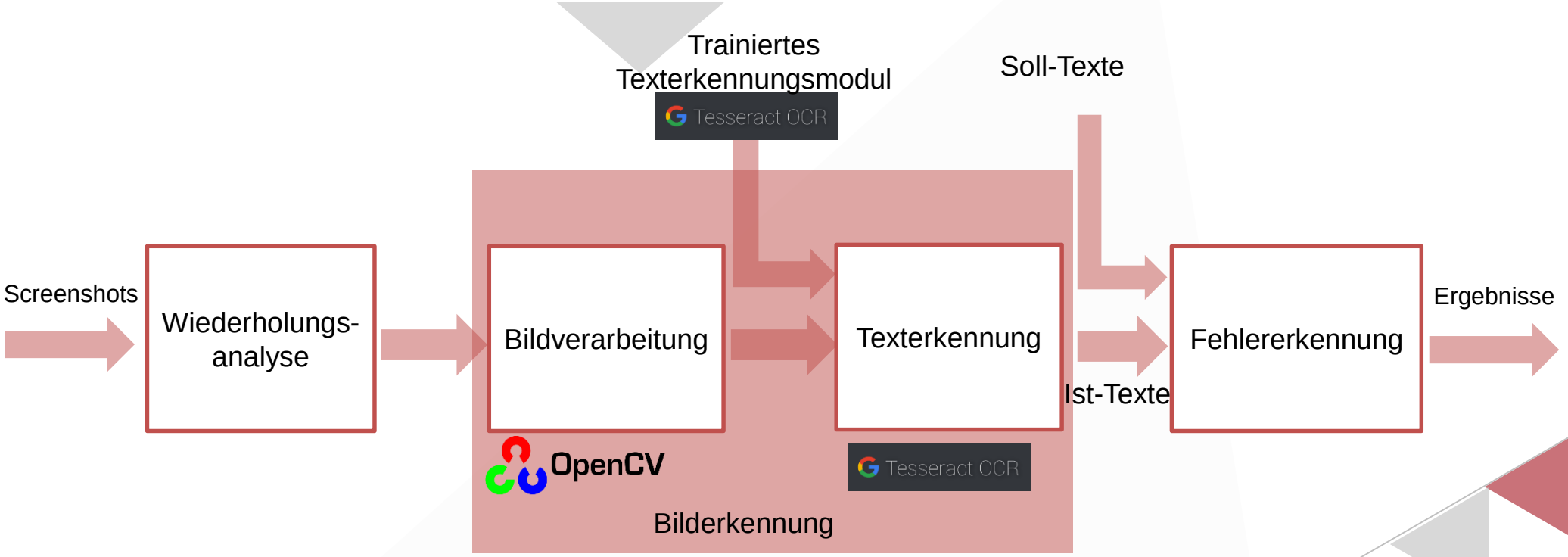
3. Konzept des Algorithmus

Darstellung des Grundkonzepts

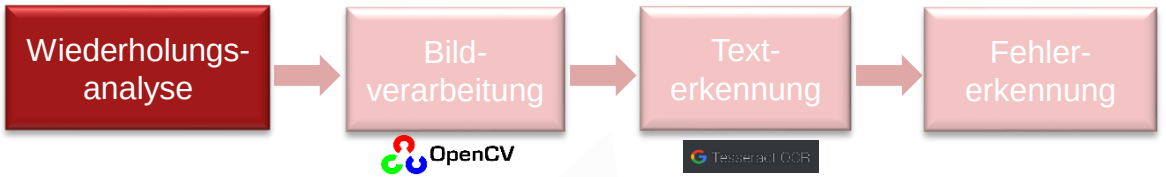


3. Konzept des Algorithmus

Darstellung des Grundkonzepts



4. Implementierung



Wiederholungsanalyse der Screenshots

- Automatischer Screenshots
  - Bei jeder Statusveränderung der HMI-Anzeige (durch Klick / Zug)

- Strukturelle Ähnlichkeit (SSIM) > 0,99
  → die wiederholenden Screenshots zu filtern

4. Implementierung

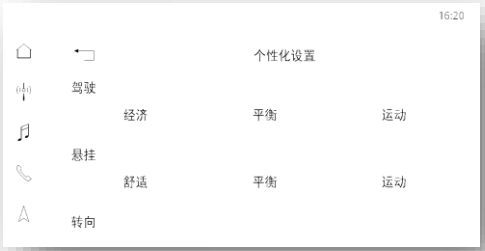
Wiederholungs-analyse → Bild-verarbeitung (OpenCV) → Text-erkennung (Tesseract OCR) → Fehler-erkennung

Verarbeitung der Screenshots

1) Bildaufnahme

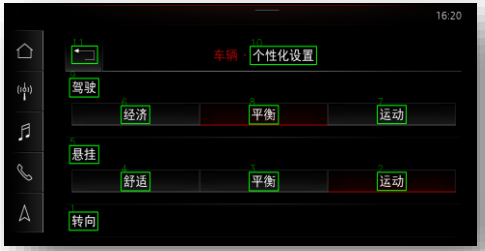
2) Grauwerttransformation

3) invertiertes Graustufenbild

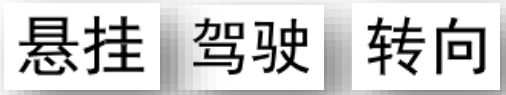


4) Binarisierung

5) Mathematische Morphologie



6) Konturverfolgung

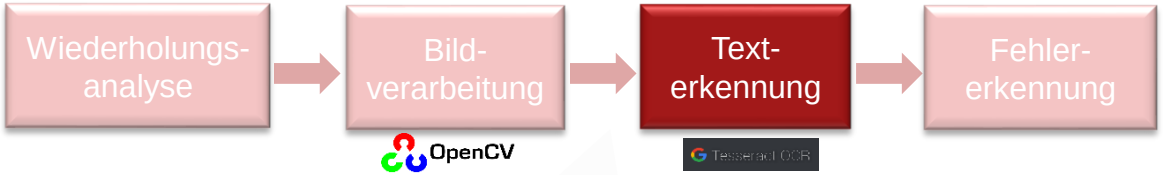


7) Extrahieren der Konturen

Texterkennung

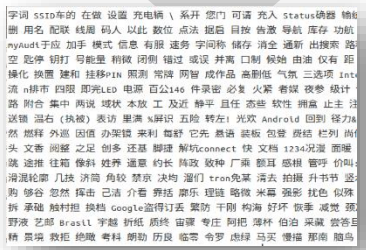
Erkennungsmodul benötigt

4. Implementierung



Training des OCR Erkennungsmoduls

Solltexte
(XML-Datei)



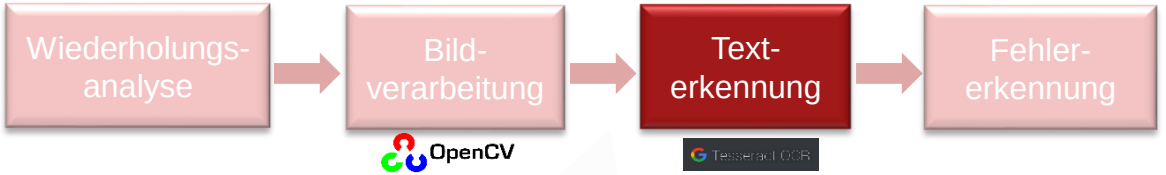
Trainingstexte

Schriftartdatei
(truetype/opentype)

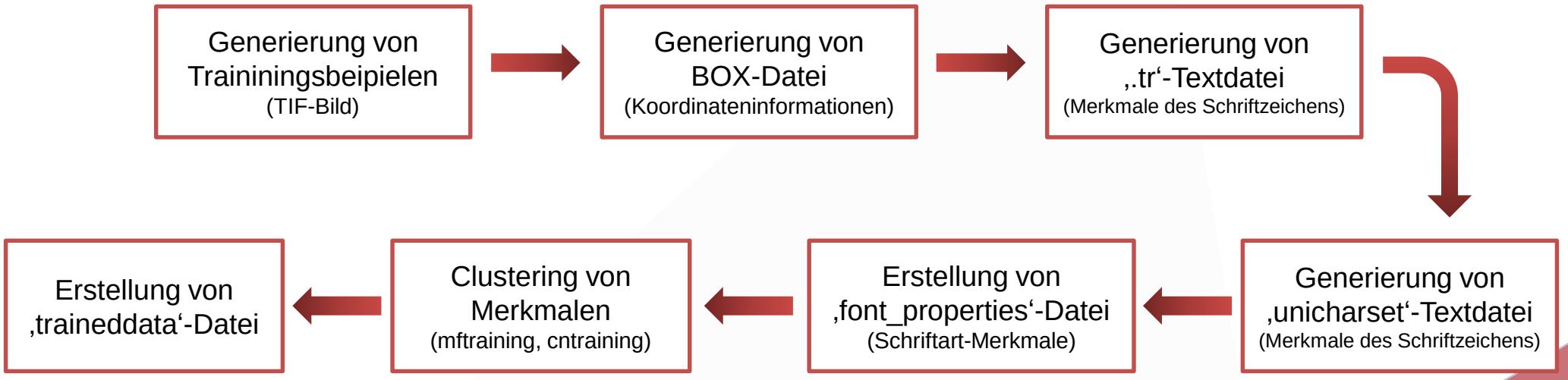
```
FAIL!
APPLY_BOXES: boxfile line 4756/围 ((532,575),(573,620)): FAILURE! Couldn't find a matching blob
FAIL!
APPLY_BOXES: boxfile line 4919/围 ((1156,443),(1196,488)): FAILURE! Couldn't find a matching blob
FAIL!
APPLY_BOXES: boxfile line 5036/提 ((3426,352),(3473,400)): FAILURE! Couldn't find a matching blob
FAIL!
APPLY_BOXES: boxfile line 5076/因 ((1879,311),(1920,355)): FAILURE! Couldn't find a matching blob
FAIL!
APPLY_BOXES: boxfile line 5107/围 ((3404,291),(3444,336)): FAILURE! Couldn't find a matching blob
APPLY_BOXES:
   Boxes read from boxfile:    5320
   Boxes failed resegmentation:      24
   Found 5296 good blobs.
Generated training data for 488 words
Page 3
row xheight=37, but median xheight = 25.5077
row xheight=30, but median xheight = 25.5077
row xheight=30, but median xheight = 25.5077
row xheight=31, but median xheight = 25.5077
row xheight=37, but median xheight = 25.5077
```

Beim Training wird die Engine über die Befehlszeilenschnittstelle im Trainingsmodus ausgeführt.

4. Implementierung

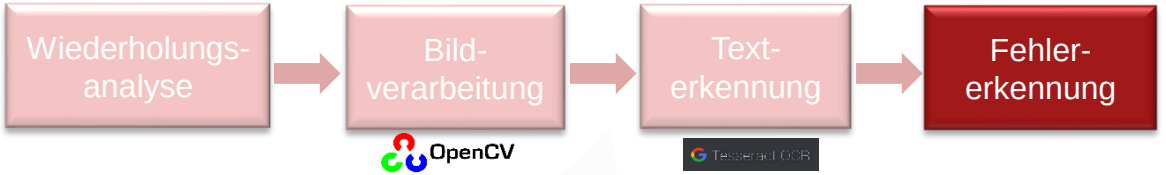


Training über die Befehlszeilenschnittstelle



→Texterkennung: Anruf von ‚traineddata'

4. Implementierung



Fehlererkennung in zwei Richtungen

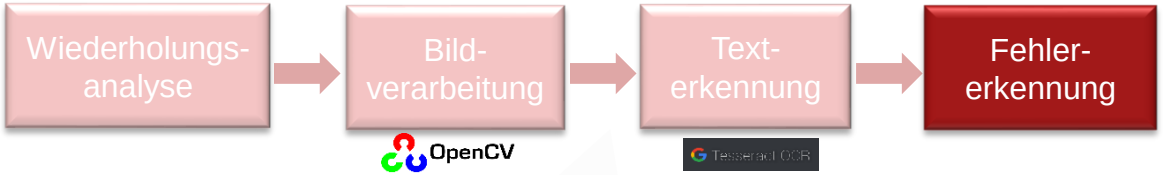
- Ist-Texte als Referenz, basierend darauf werden Soll-Texte überprüft
  → Die Fehler werden als Absicherungsergebnisse betrachtet

- Soll-Texte als Referenz, basierend darauf werden Ist-Texte überprüft
  → Die betroffenen Fehler wirken als Hinweise zur Verbesserung der Generierung von Testfällen

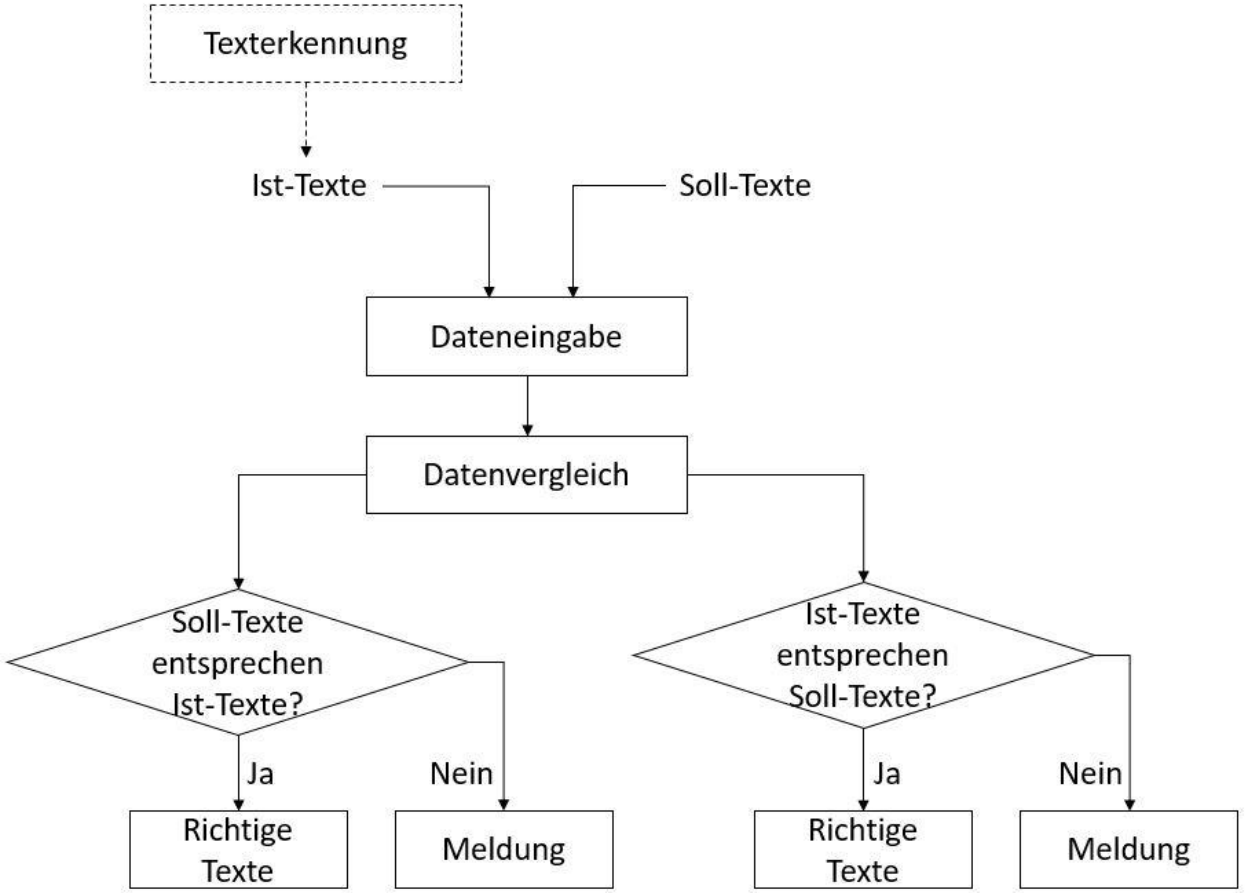
Soll-Texte	Ist-Texte
✓	✓
✓	✗
✗	✓

Mögliche Situationen beim Vergleich

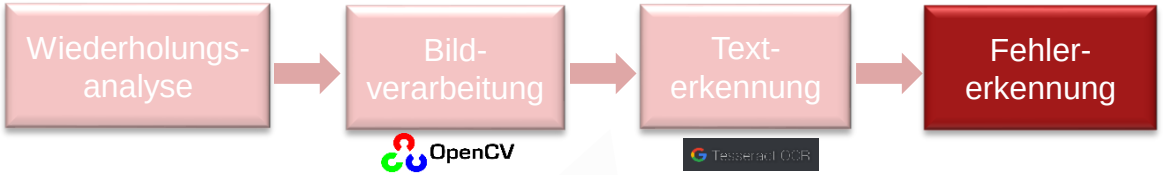
4. Implementierung



Fehlererkennung in zwei Richtungen



4. Implementierung



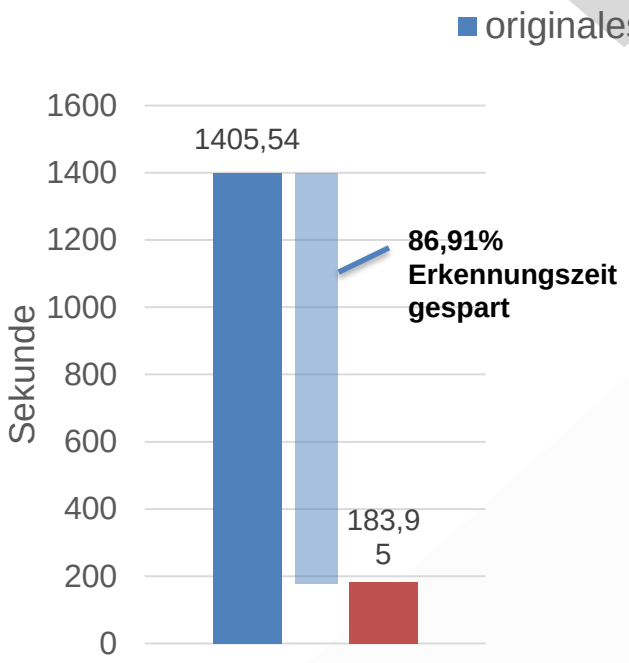
Beispielanzeige für die Fehlererkennungsergebnisse

- Anzeige richtige Texte
- Meldung: von XML-Datei unerwartete erkannten Texte
- Meldung: Soll-Texte nicht erkannt

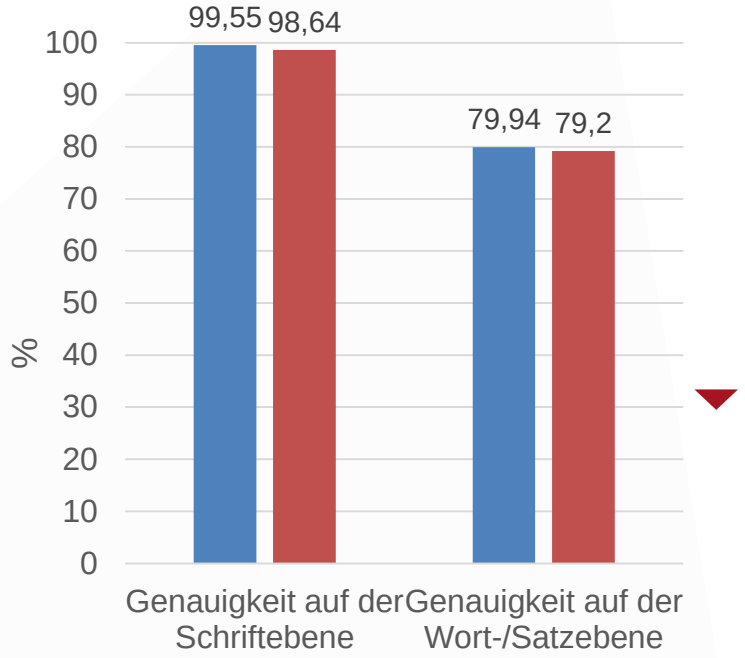
```
==========================================================
Correct:

经济  found 1 times
舒适  found 4 times
个性化设置  found 3 times
悬挂  found 2 times
转向  found 3 times
安静  found 1 times
当前  found 1 times
自动  found 1 times
运动  found 5 times
车辆  found 1 times
平衡  found 5 times
发动机声音  found 1 times
驾驶  found 1 times

==========================================================
Recognized texts from screenshots are not found in the xml file:

口工2柱
(similarity: 0)  found 1 times
带开力加宙音
(similarity: 0.166666666667)  found 1 times
2单Av纸
(similarity: 0.2)  found 1 times

==========================================================
Test cases in the xml file are not found in screenshots:

Audi 驾驶模式选择
(similarity: 0.0909090909091)
ausgewogen
(similarity: 0)
komfortabel
(similarity: 0)
ausgewogen
(similarity: 0)

processing time: 7.520642s
```

5. Ergebnisse und Auswertung

Erkennungsergebnisse für Menü ‚CAR‘



Erkennungszeit

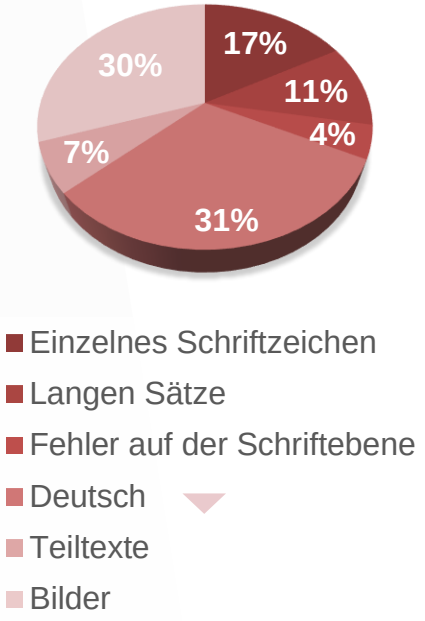


Genauigkeit

5. Ergebnisse und Auswertung

Fehleranalyse auf der Wort/Satzebene

- Falsche Erkennung von Tesseract
  - Unstabile Performance bei der Erkennung von Text aus einzelnem Schriftzeichen
  - Unstabile Performance bei der Erkennung von langen Sätze
  - Falsche Erkennung wegen Fehler auf der Schriftebene
  - Deutsch in den meisten Fälle nicht richtig erkannt
- ‚Teiltexte' im Screenshot erkannt
- Bilder im Screenshot nicht entfernt und als Texte erkannt



6. Zusammenfassung und Ausblick

Zusammenfassung

- Bildverarbeitung und Fehlererkennung erfolgreich durchgeführt
- Geschwindigkeit deutlich erhöht
- Kein großer Unterschied zwischen der Genauigkeit der originalen und trainierten Module

- Begrenzung der Open-Source-Engine Tesseract
- Intelligentere Methode zur Layout-Analyse notwendig

Weitere Arbeit

- Fehlerbeseitigung auf der Schriftebene
- Fehlererkennung für alle Screenshots

6. Zusammenfassung und Ausblick

Ausblick

- Automatische Fehlererkennung in anderen Sprachen
- Die einzelnen Funktionen des Algorithmus als Schnittstelle in andere Programmen einstecken
  - Wiederholungsanalyse
  - Bildverarbeitung
  - Texterkennung
- Textbereiche auf andere Weise zu lokalisieren + Texterkennung

Vielen Dank für Ihre Aufmerksamkeit!

Quelle

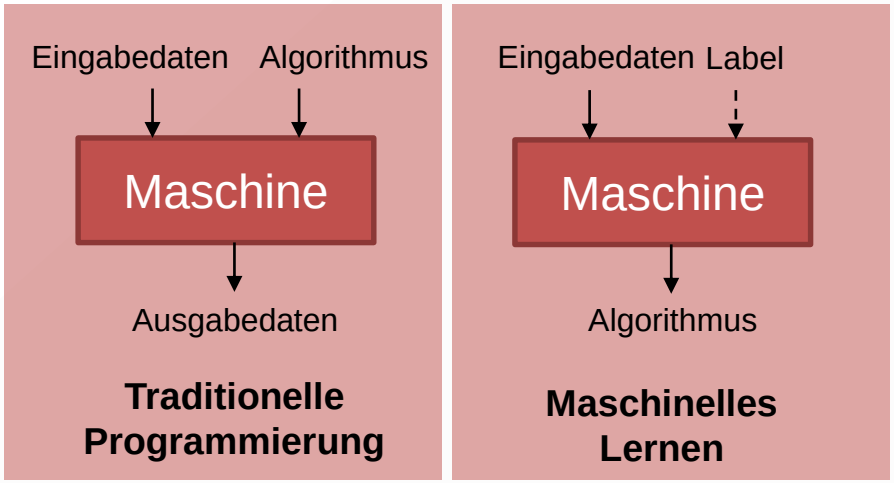
1. MMI Navigation plus, https://www.audi.de/de/brand/de/neuwagen/a7/a7sportback/ausstattung.html#.
2. Mehryar Mohri, Afshin Rostamizadeh und Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

Back-up

Maschinellen Lernens

Fachbegriffe des maschinellen Lernens*

- **Beispiel**: Elemente oder Instanzen von Daten, die zum Lernen oder zur Auswertung eines Algorithmus verwendet werden
- **Merkmal**: Eine oder mehrere Eigenschaften, die das Beispiel repräsentieren
- **Label**: Werte oder Kategorien, die den Beispielen zugewiesen sind
- **Trainingsbeispiel**: Beispiele zum Training eines Lernalgorithmus
- **Validierungsbeispiel**: Beispiele zum Einstellen der Parameter des Lernalgorithmus
- **Testbeispiel**: Beispiele zur Bewertung der Performance des Lernalgorithmus



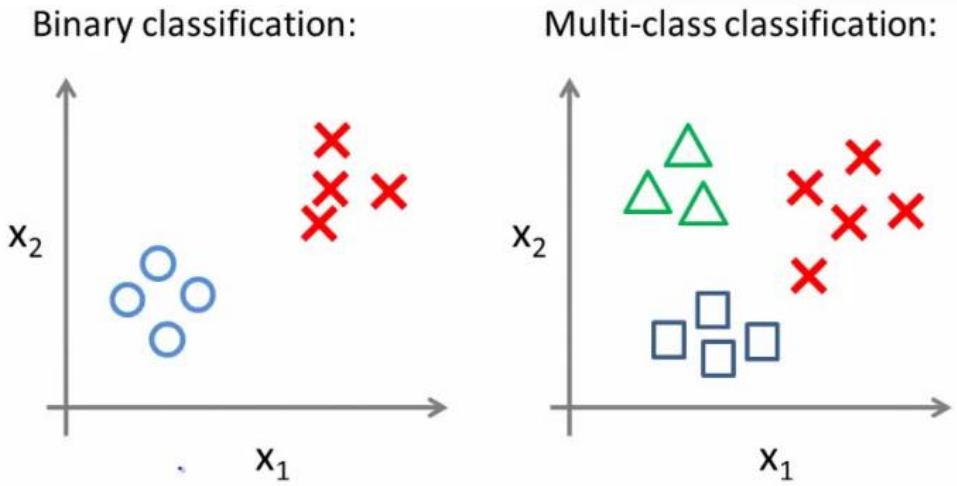
* Quelle: Mehryar Mohri, Afshin Rostamizadeh und Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

Grundansätze des maschinellen Lernens

Klassifikation

Jedem Trainingsbeispiel eine Kategorie zuzuweisen, finde für neue Beispiele die richtige Klasse.

- k-NN, Entscheidungsbäume, NN, SVM



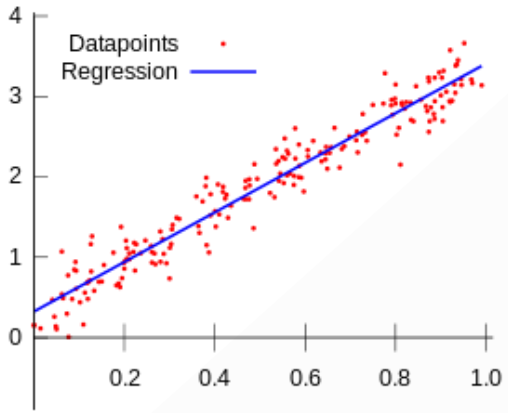
Bildquelle: http://www.geeksforgeeks.org/getting-started-with-classification/

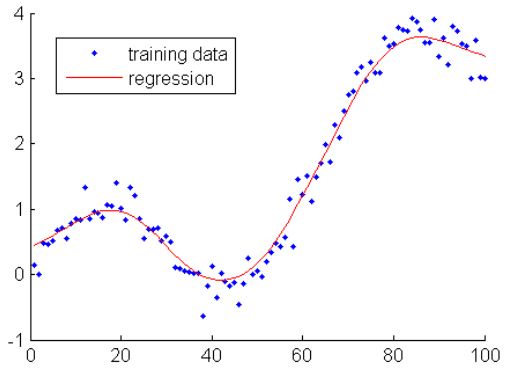
Grundansätze des maschinellen Lernens

Regression

Beziehungen zwischen einer abhängigen Variable und einer oder mehrere unabhängigen Variablen zu modellieren, finde für neue Beispiele die richtige Werte.
Ausgabe: kontinuierliche Werte

- Lineare Regression
- Nichtlineare Regression





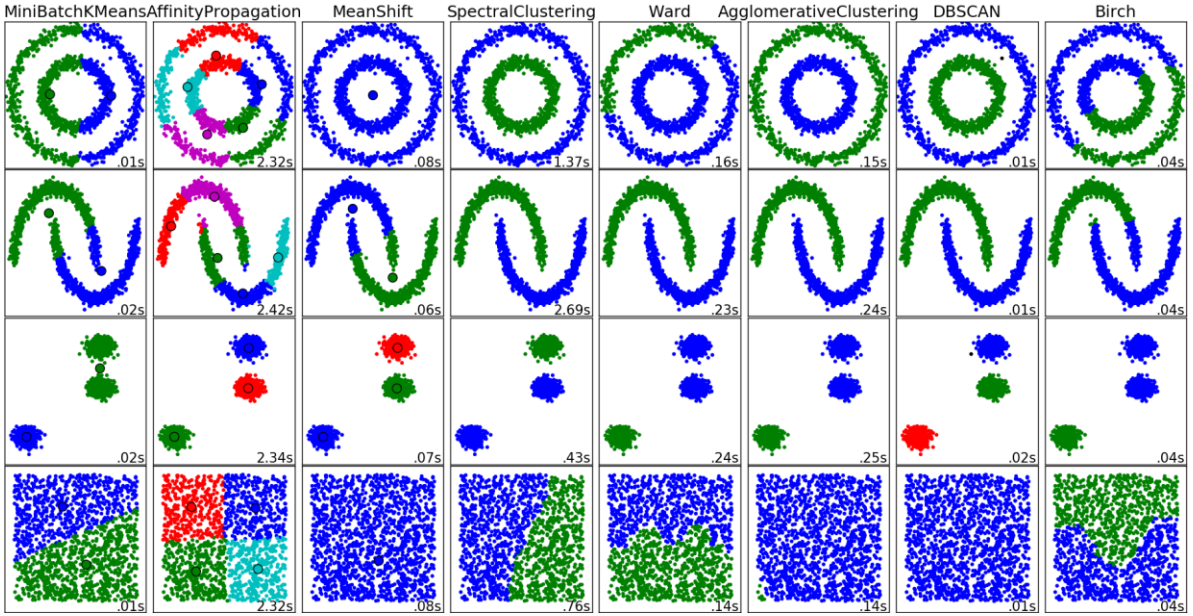
Bildquelle: https://commons.wikimedia.org/wiki/File:LinearRegression.svg

Grundansätze des maschinellen Lernens

Clustering

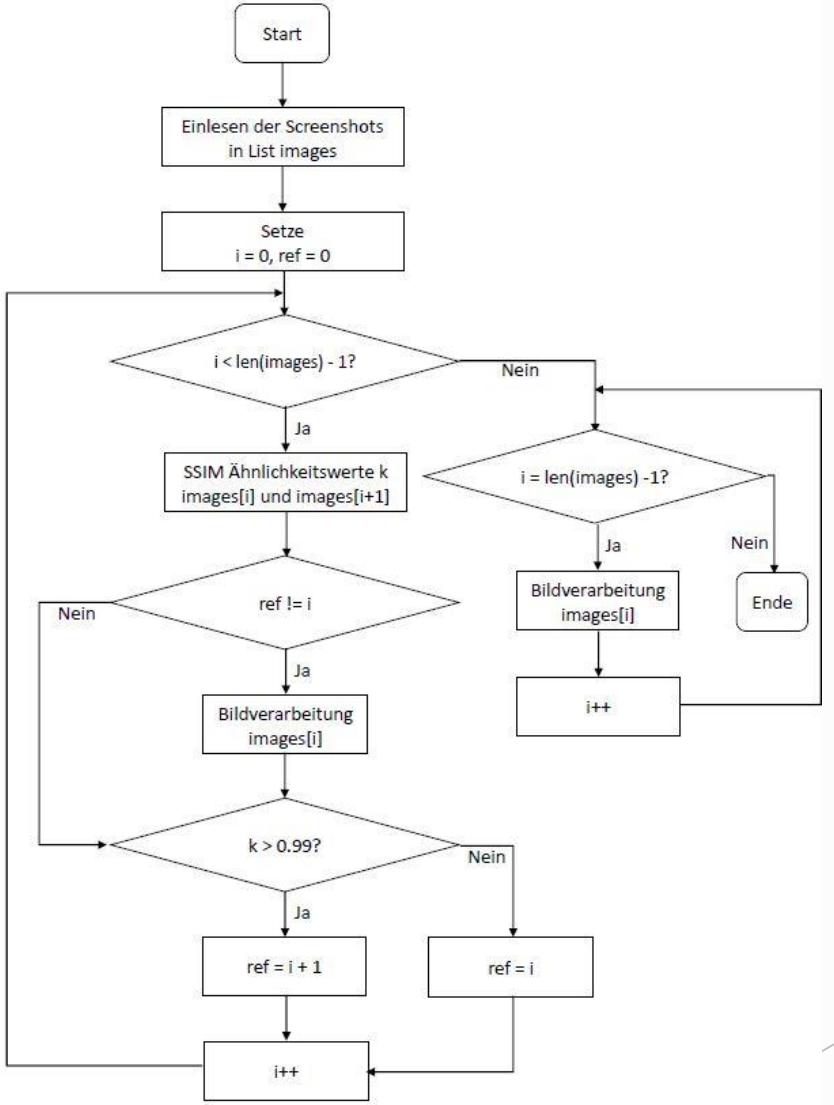
Beispieldatensätze anhand von Ähnlichkeiten in Gruppen zusammenzufassen, keine Kategorien (Labels) als Klassifikationsergebnisse vordefiniert.
Das Netz erstellt somit selbständig Klassifier.

- K-means, dichteverbundenes Clustern

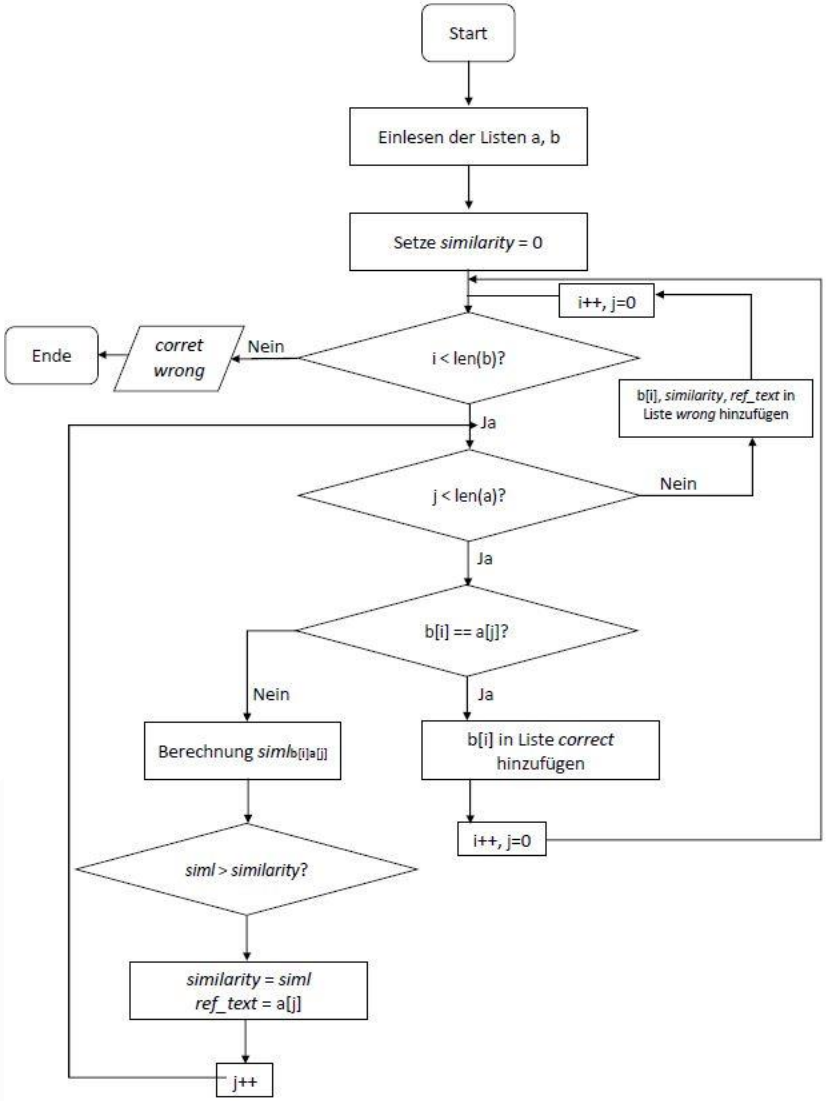


Bildquelle: http://scikit-learn.org/0.18/auto_examples/cluster/plot_cluster_comparison.html

Ablaufdiagramm für die Wiederholungsanalyse der Screenshots



Ablaufdiagramm für die Fehlererkennung (Ist-Texte als Referenz)



Binarisierung

‚.tr'-Datei

The *.tr file format

Every character in the box file has a corresponding set of entries in the .tr file (in order) like this:

```
<fontname> <character> <left> <top> <right> <bottom> <pagenum>
 4
mf <number of features>
<x> <y> <length> <dir> 0 0
...
cn 1
<ypos> <length> <x2ndmoment> <y2ndmoment>
if <number of features>
<x> <y> <dir>
...
tb 1
<bottom> <top> <width>
```

The Micro Features (`mf`) are polygon segments of the outline normalized to the 1st and 2nd moments. The `mf` line will followed by a set of lines determined by <number of features>.
x is x position [-0.5,0.5]
y is y position [-0.25,0.75]
length is the length of the polygon segment [0,1.0]
dir is the direction of the segment [0,1.0]

The Char Norm features (`cn`) are used to correct for the moment normalization to distinguish position and size (eg `c` vs `C` and `,` vs `'`)

‚unicharset‘-Datei

`character` `properties` `glyph_metrics` `script` `other_case` `direction` `mirror` `normed_form`

- `character`
  The UTF-8 encoded string to be produced for this unichar.
- `properties`
  An integer mask of character properties, one per bit. From least to most significant bit, these are: isalpha, islower, isupper, isdigit, ispunctuation.
- `glyph_metrics`
  Ten comma-separated integers representing various standards for where this glyph is to be found within a baseline-normalized coordinate system where 128 is normalized to x-height.
  - `min_bottom`, `max_bottom`
    The ranges where the bottom of the character can be found.
  - `min_top`, `max_top`
    The ranges where the top of the character may be found.
  - `min_width`, `max_width`
    Horizontal width of the character.
  - `min_bearing`, `max_bearing`
    How far from the usual start position does the leftmost part of the character begin.
  - `min_advance`, `max_advance`
    How far from the printer’s cell left do we advance to begin the next character.
- `script`
  Name of the script (Latin, Common, Greek, Cyrillic, Han, NULL).
- `other_case`
  The Unichar ID of the other case version of this character (upper or lower).
- `direction`
  The Unicode BiDi direction of this character, as defined by ICU’s enum UCharDirection. (0 = Left to Right, 1 = Right to Left, 2 = European Number…)
- `mirror`
  The Unichar ID of the BiDirectional mirror of this character. For example the mirror of open paren is close paren, but Latin Capital C has no mirror, so it remains a Latin Capital C.
- `normed_form`
  The UTF-8 representation of a "normalized form" of this unichar for the purpose of blaming a module for errors given ground truth text. For instance, a left or right single quote may normalize to an ASCII quote.

‚font_properies'-Datei

The font_properties file

Now you need to create a `font_properties` text file. The purpose of this file is to provide font style information that will appear in the output when the font is recognized.

Each line of the `font_properties` file is formatted as follows: `fontname` `italic` `bold` `fixed` `serif` `fraktur`
where `fontname` is a string naming the font (no spaces allowed!), and `italic`, `bold`, `fixed`, `serif` and `fraktur` are all simple `0` or `1` flags indicating whether the font has the named property.

Example:

```
timesitalic 1 0 0 1 0
```

The `font_properties` file will be used by the `shapeclustering` and `mftraining` commands.

When running `mftraining`, each `fontname` field in the *.tr file must match an `fontname` entry in the `font_properties` file, or `mftraining` will abort.

Note: There is a default font_properties file, that covers 3000 fonts (not necessarily accurately) located in the langdata repository.

‚font_properies'-Datei

More about the `properties` field

Here is another example. For simplicity only the first two fields in each line are shown in this example. The other fields are omitted.

```
...
; 10 ...
b 3 ...
W 5 ...
7 8 ...
= 0 ...
...
```

Char	Punctuation	Digit	Upper	Lower	Alpha	Binary num	Hex.
;	1	0	0	0	0	10000	10
b	0	0	0	1	1	00011	3
W	0	0	1	0	1	00101	5
7	0	1	0	0	0	01000	8
=	0	0	0	0	0	00000	0

In columns 2-6, `0` means 'No', `1` means 'Yes'.

Clustering

mftraining

```
mftraining -F font_properties -U unicharset -O lang.unicharset lang.fontname.exp0.tr
```

The -U file is the unicharset generated by `unicharset_extractor` above, and lang.unicharset is the output unicharset that will be given to `combine_tessdata`.

`mftraining` will output two other data files: `inttemp` (the shape prototypes) and `pffmtable` (the number of expected features for each character).

NOTE: `mftraining` will produce a `shapetable` file if you didn't run `shapeclustering`. You **must** include this `shapetable` in your traineddata file, whether or not `shapeclustering` was used.

cntraining

```
cntraining lang.fontname.exp0.tr lang.fontname.exp1.tr ...
```

This will output the `normproto` data file (the character normalization sensitivity prototypes).