Vortrag zur Bachelorarbeit: Präzise Erkennung von Sonderzeichen aus layout-basierten Textdokumenten am Beispiel der häufigsten europäischen Sonderzeichen

Pascal Muckenhirn

Albert-Ludwigs-Universität Freiburg

30.10.2019

Gliederung

1. Einführung ins Thema und die grundlegende Problemstellung

2. Der Ansatz zur Lösung des Problems
   - Baseline-Implementierung
   - Machine-Learning-Ansatz

3. Evaluation

Gliederung

Layout-basierte Textdokumente

PDF

- Plattformunabhängiges Dateiformat
- enthalten nur Informationen über Zeichen (nicht Wörter!)
  - Schriftinformationen
  - Zeichenbreiten
  - Verwendete Encodings
  - ...
- Ziel: Gleiches Aussehen der Datei (auf jeder Plattform)

Text-Extraktion aus PDFs

Text-Extraktion aus PDFs

- komplexer Prozess
- bestehend aus mehreren aufeinander aufbauenden Schritten
- In dieser Arbeit: Nur ein Teilschritt
  $\rightarrow$ **Übersetzung von Ligaturen und diakritischen Zeichen**

Für diese Arbeit

- Nutzung der Position der Zeichen (Bounding Box)
- Nutzung der grafischen Informationen der Sonderzeichen
- Nutzung der "normalen" Zeichen (nicht Sonderzeichen)

1.Herausforderung: Ligaturen

Beispiel

1.Herausforderung: Ligaturen

Beispiel

Mögliche Ergebnisse einer Text-Extraktion:

- "e?cient"
- "ecient"
- "e\uFB03cient"

1.Herausforderung: Ligaturen

Beispiel

Mögliche Ergebnisse einer Text-Extraktion:

- "e?cient"
- "ecient"
- "e\uFB03cient"

→ **werden nicht gefunden, wenn nach "efficient" gesucht wird**

1.Herausforderung: Ligaturen

Beispiel

Mögliche Ergebnisse einer Text-Extraktion:

- "e?cient"
- "ecient"
- "e\uFB03cient"

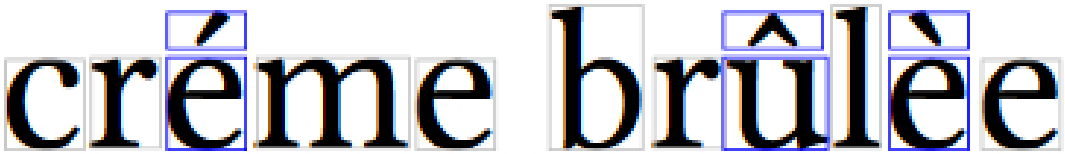
→ **werden nicht gefunden, wenn nach "efficient" gesucht wird**

Gründe für Schreibweise

- stylistisch
- historisch

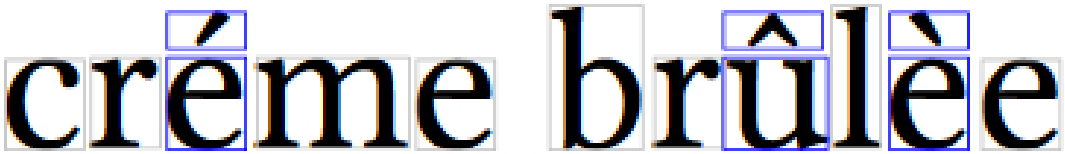
2.Herausforderung: Diakritische Zeichen

Beispiel



2.Herausforderung: Diakritische Zeichen

Beispiel

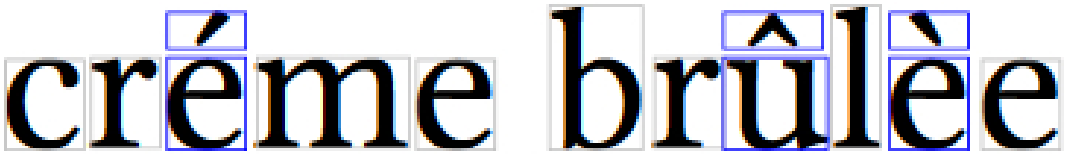


Mögliche Ergebnisse einer Text-Extraktion:

- "cr´eme br^ul`ee"
- "cre´me bru^le`e"

2.Herausforderung: Diakritische Zeichen

Beispiel



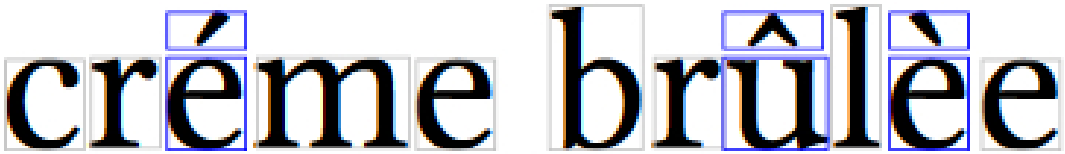
Mögliche Ergebnisse einer Text-Extraktion:

- "cr´eme br^ul`ee"
- "cre´me bru^le`e"

→ **nicht suchbar**

2.Herausforderung: Diakritische Zeichen

Beispiel



Mögliche Ergebnisse einer Text-Extraktion:

- "cr´eme br^ul`ee"
- "cre´me bru^le`e"

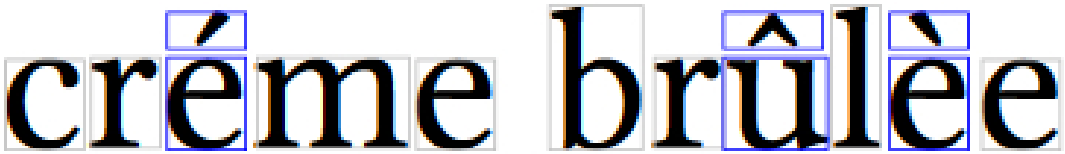
→ **nicht suchbar**

Gründe für Schreibweise

- besondere Sprechweise oder Betonung

3.Herausforderung: Reihenfolge der Zeichen

Beispiel

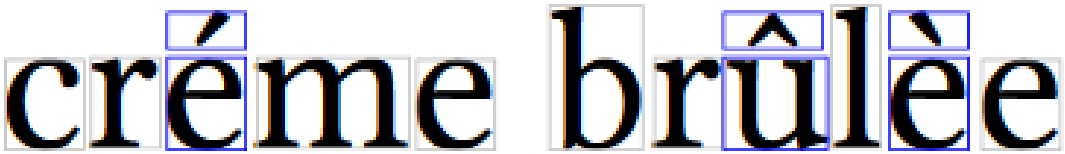


Mögliche Ergebnisse einer Text-Extraktion:

- "cr´eme br^ul`ee"
- "cre´me bru^le`e"

3.Herausforderung: Reihenfolge der Zeichen

Beispiel



Mögliche Ergebnisse einer Text-Extraktion:

- "cr´eme br^ul`ee"
- "cre´me bru^le`e"

→ **Reihenfolge nicht eindeutig**

4.Herausforderung: Gezeichnete Zeichen

æsthetics

Mögliche Speicherung der Zeichen

`null`, "s", "t", "h", "e", "t", "i", "c", "s"

Problemdefinition

Eingabe

- Wörter des PDF-Dokuments mit folgenden Eigenschaften:
  - Enthält Informationen über die enthaltenen Buchstaben
    - Text
    - Position
    - Font
    - ...
  - Kann Ligaturen und diakritische Zeichen enthalten
  - Können (nur) gezeichnet vorliegen

Ausgabe

- Durchsuchbare Wörter

Gliederung

Format der Eingabedaten

```
...
{
    "text": "entrecôte",
    "pageNum": 1,
    "positions": [66.0, 547.5, 101.8, 553.8],
    "characters": [ ...
    {
        "text": "o^",
        "image": "0ilaCrw/qAYPEEqQ.jpg",
        "parts": [{
            "text": "o",
            ...
            "pageNum: 1,
            "positions": [89.8, 547.5, 94.3, 551.6]
          }, {
            "text": "^",
            ...
            "pageNum: 1,
            "positions": [89.9, 552.2, 94.4, 553.8]
          }]
    }, ... ]
}
...
```

Verarbeitung von Ligaturen

Implementierung

- Jedes Zeichen mittels des Python-Pakets "unicodedata" abbilden

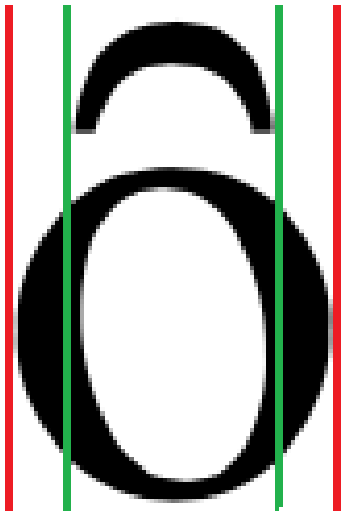
  Beispiel:"[ffi]" $\rightarrow$ "LATIN SMALL LIGATURE FFI"

  "p" $\rightarrow$ "LATIN SMALL LETTER P"

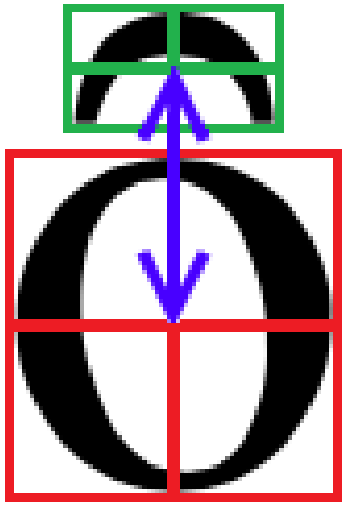
- Überprüfung auf Ligatur
- Passende(s) Rückgabezeichen konstruieren

Verarbeitung von diakritischen Zeichen

X-Überlappung



Y-Abstand



Verarbeitung von diakritischen Zeichen

Implementierung

- Für jedes Zeichen überprüfen ob es ein Diakritisches ist
  Beispiel:
  "´" $\rightarrow$ "COMBINING ACUTE ACCENT"
- Wenn ja X-Überlappung und Y-Abstand berechnen, besten Partner identifizieren und Zeichen verbinden
- Wort zurückgeben

Größtes Problem des Baseline-Ansatzes

- Gezeichnete Zeichen können nicht verarbeitet werden !

Gliederung

Idee

Aspekte zum Erreichen des Ziels

- Nutzen der Form des Zeichens
- Nutzen des Wortkontexts (umliegende Zeichen)
  Beispiel:
  e, `null`, c, i, e, n, t $\rightarrow$ efficient

Idee

Aspekte zum Erreichen des Ziels

- Nutzen der Form des Zeichens
- Nutzen des Wortkontexts (umliegende Zeichen)
  Beispiel:
  e, `null`, c, i, e, n, t $\rightarrow$ efficient

Umsetzung

- Ein Modell für Analyse der Form des Zeichens

Idee

Aspekte zum Erreichen des Ziels

- Nutzen der Form des Zeichens
- Nutzen des Wortkontexts (umliegende Zeichen)
  Beispiel:
  e, `null`, c, i, e, n, t $\rightarrow$ efficient

Umsetzung

- Ein Modell für Analyse der Form des Zeichens
  $\rightarrow$ Visuelles Modell
- Ein Modell für Analyse des Wortkontexts

Idee

Aspekte zum Erreichen des Ziels

- Nutzen der Form des Zeichens
- Nutzen des Wortkontexts (umliegende Zeichen)
  Beispiel:
  e, `null`, c, i, e, n, t $\rightarrow$ efficient

Umsetzung

- Ein Modell für Analyse der Form des Zeichens
  $\rightarrow$ Visuelles Modell
- Ein Modell für Analyse des Wortkontexts
  $\rightarrow$ Textuelles Modell
- Passende Verarbeitung zur Nutzung beider Modelle

Idee

Aspekte zum Erreichen des Ziels

- Nutzen der Form des Zeichens
- Nutzen des Wortkontexts (umliegende Zeichen)
  Beispiel:
  e, `null`, c, i, e, n, t $\rightarrow$ efficient

Umsetzung

- Ein Modell für Analyse der Form des Zeichens
  $\rightarrow$ Visuelles Modell
- Ein Modell für Analyse des Wortkontexts
  $\rightarrow$ Textuelles Modell
- Passende Verarbeitung zur Nutzung beider Modelle
  $\rightarrow$ Kombiniertes Modell

Generierung der Trainingsdaten

Visuelles Modell

```
[13]    ['.../ligatures-diacritics/../00pBRon8.jpg']
```

Textuelles Modell

```
[14]    [13, 43, 16, 6] [[198.4, 203.6, 208.6, 212.8],
                         [203.6, 208.3, 212.8, 217.0],
                         [84.7, 84.6, 84.6, 84.6],
                         [91.0, 88.7, 88.7, 88.7]]
```

Format der Eingabedaten

```
...
{
    "text": "entrecôte",
    "pageNum": 1,
    "positions": [66.0, 547.5, 101.8, 553.8],
    "characters": [ ...
    {
        "text": "o^",
        "image": "0ilaCrw/qAYPEEqQ.jpg",
        "parts": [{
            "text": "o",
            ...
            "pageNum: 1,
            "positions": [89.8, 547.5, 94.3, 551.6]
        }, {
            "text": "^",
            ...
            "pageNum: 1,
            "positions": [89.9, 552.2, 94.4, 553.8]
        }]
    }, ... ]
}
...
```

Visuelles Modell

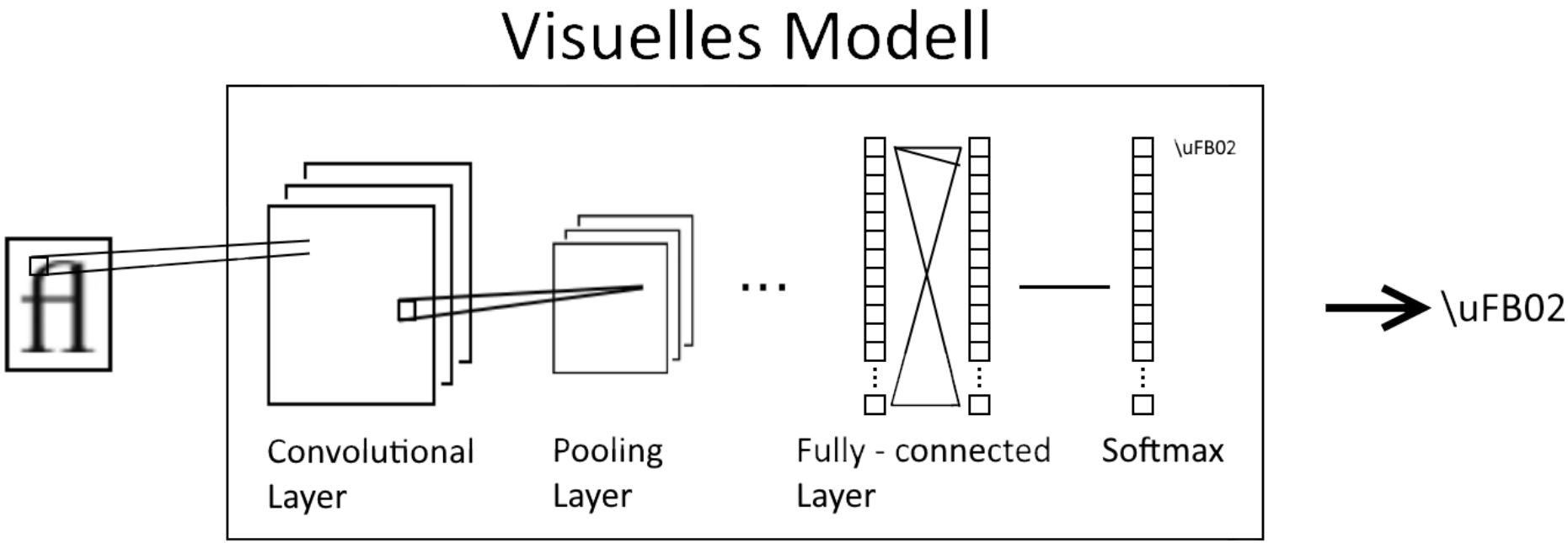
Visuelles Modell

- Analyse der Form des Zeichens
- "Convolutional Network"

Besonderheiten

- Ausschließliches Vorhersagen der Sonderzeichen

Visuelles Modell



Textuelles Modell

Textuelles Modell

- Analyse des Kontextes des Zeichens
  Beispiel:
  e, `null`, c, i, e, n, t $\rightarrow$ efficient
- mittels Bidirektionaler LSTM-Schichten

Textuelles Modell

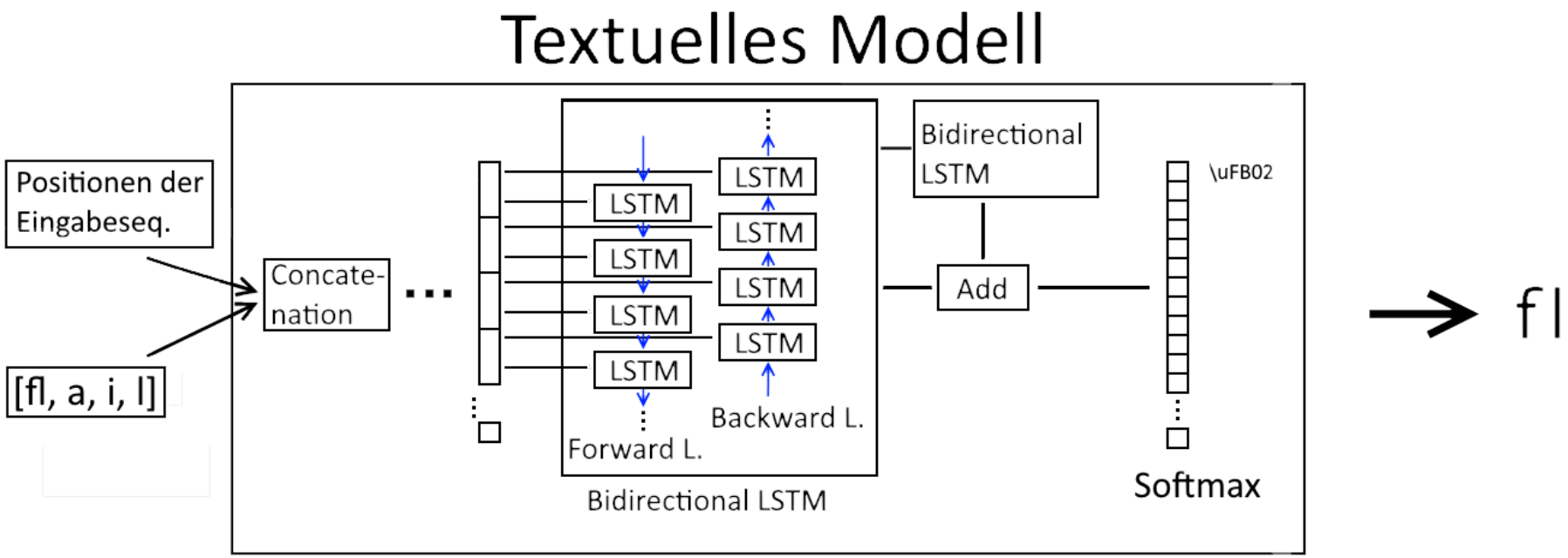
Textuelles Modell

- Analyse des Kontextes des Zeichens
  Beispiel:
  e, `null`, c, i, e, n, t $\rightarrow$ efficient
- mittels Bidirektionaler LSTM-Schichten

Besonderheiten

- Ausschließliches Vorhersagen der Sonderzeichen
- Ausgabe angepasst an "Kombiniertes Modell"
  - Beispielsausgabe: ["ffi"] anstatt ["f", "f", "i"]

Textuelles Modell



Kombiniertes Modell

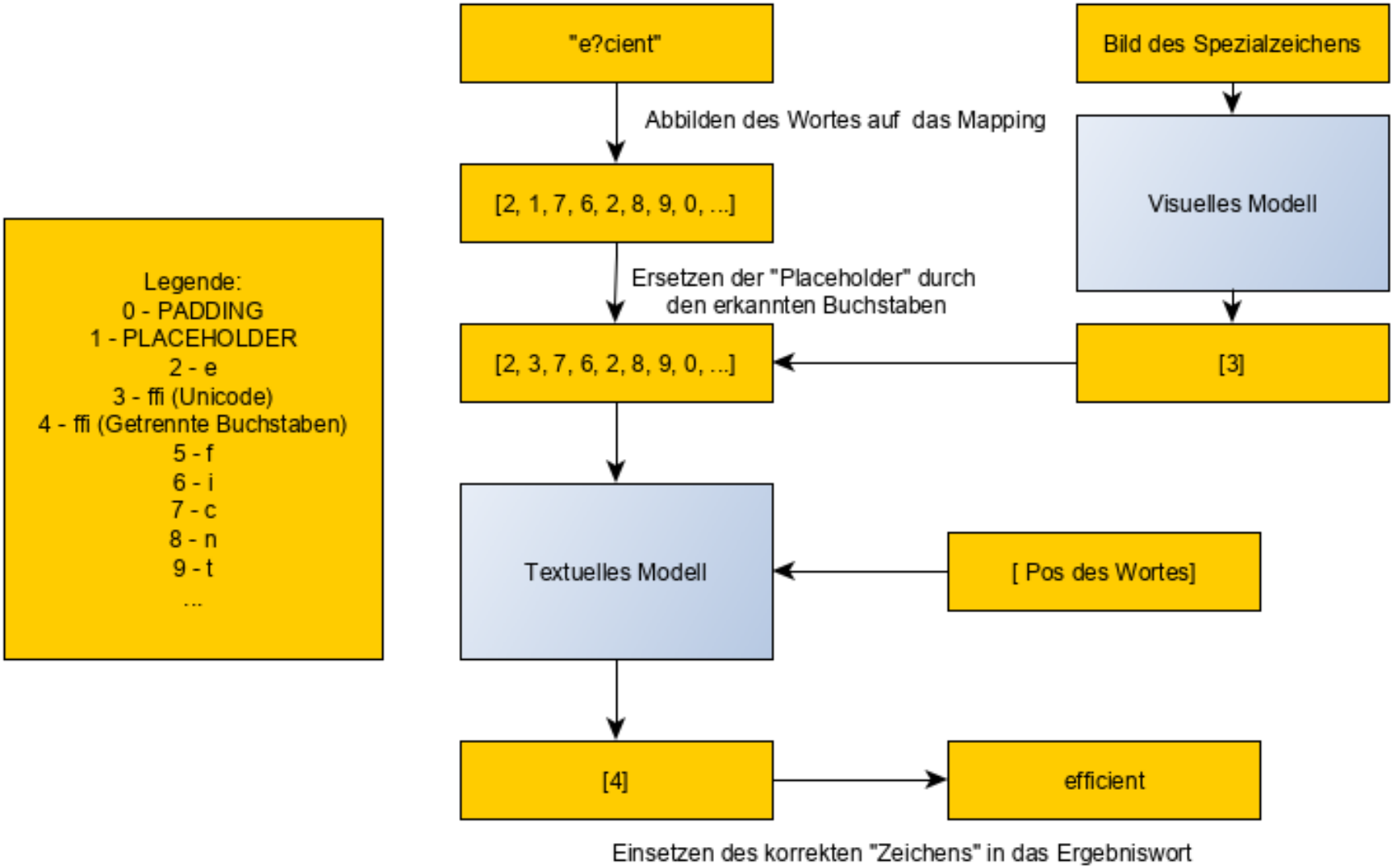
Kombiniertes Modell

- Kein Machine-Learning-Modell
- Gemeinsame Nutzung beider Machine-Learning-Modelle

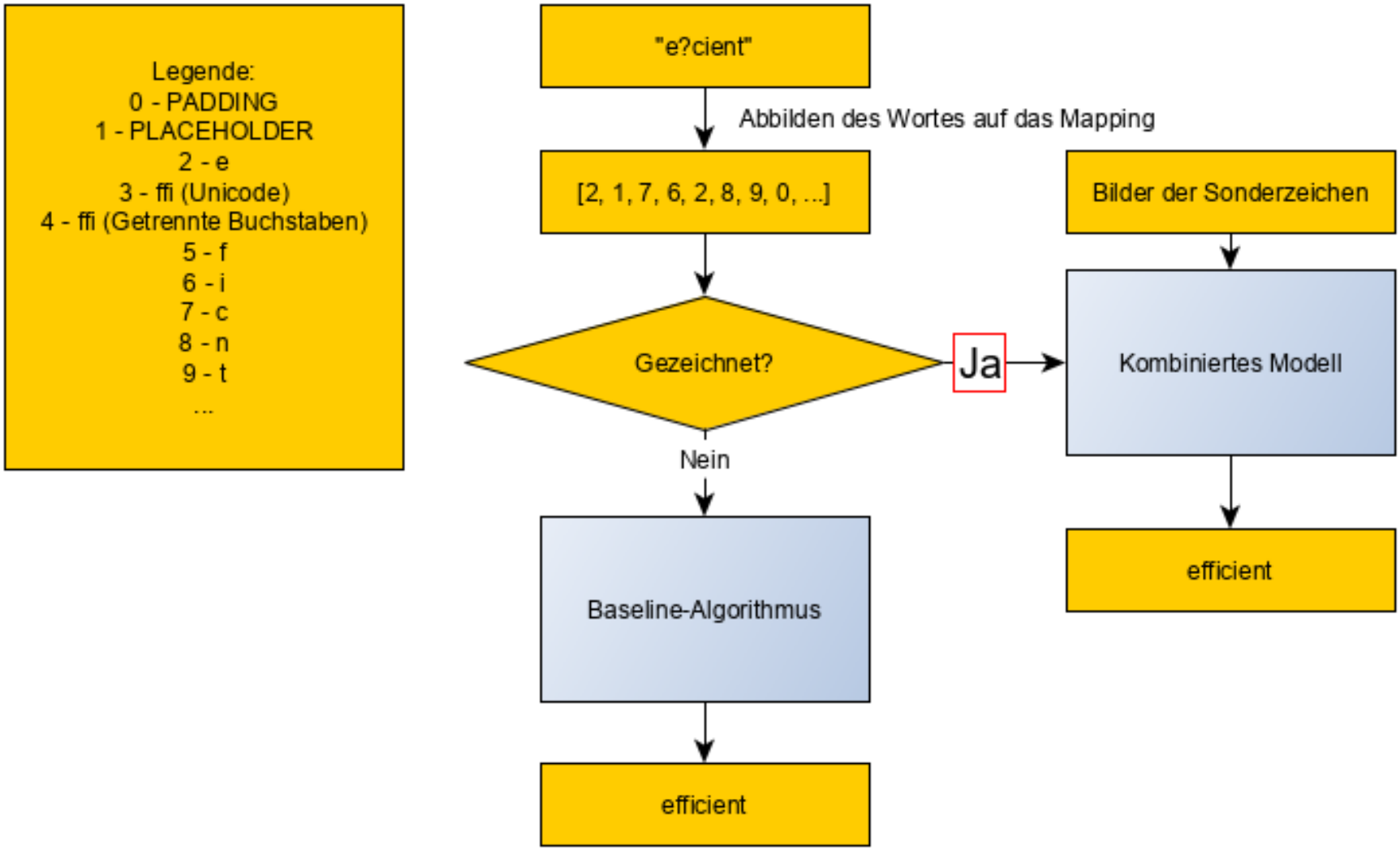
Besonderheiten

- Beim Einlesen: Alle Sonderzeichen durch "PLACEHOLDER" ersetzt
- Ausnutzen des Ausgabeformates des Textuellen Modells für passendes Einsetzen
  - Beispielsausgabe: ["ffi"] anstatt ["f", "f", "i"]

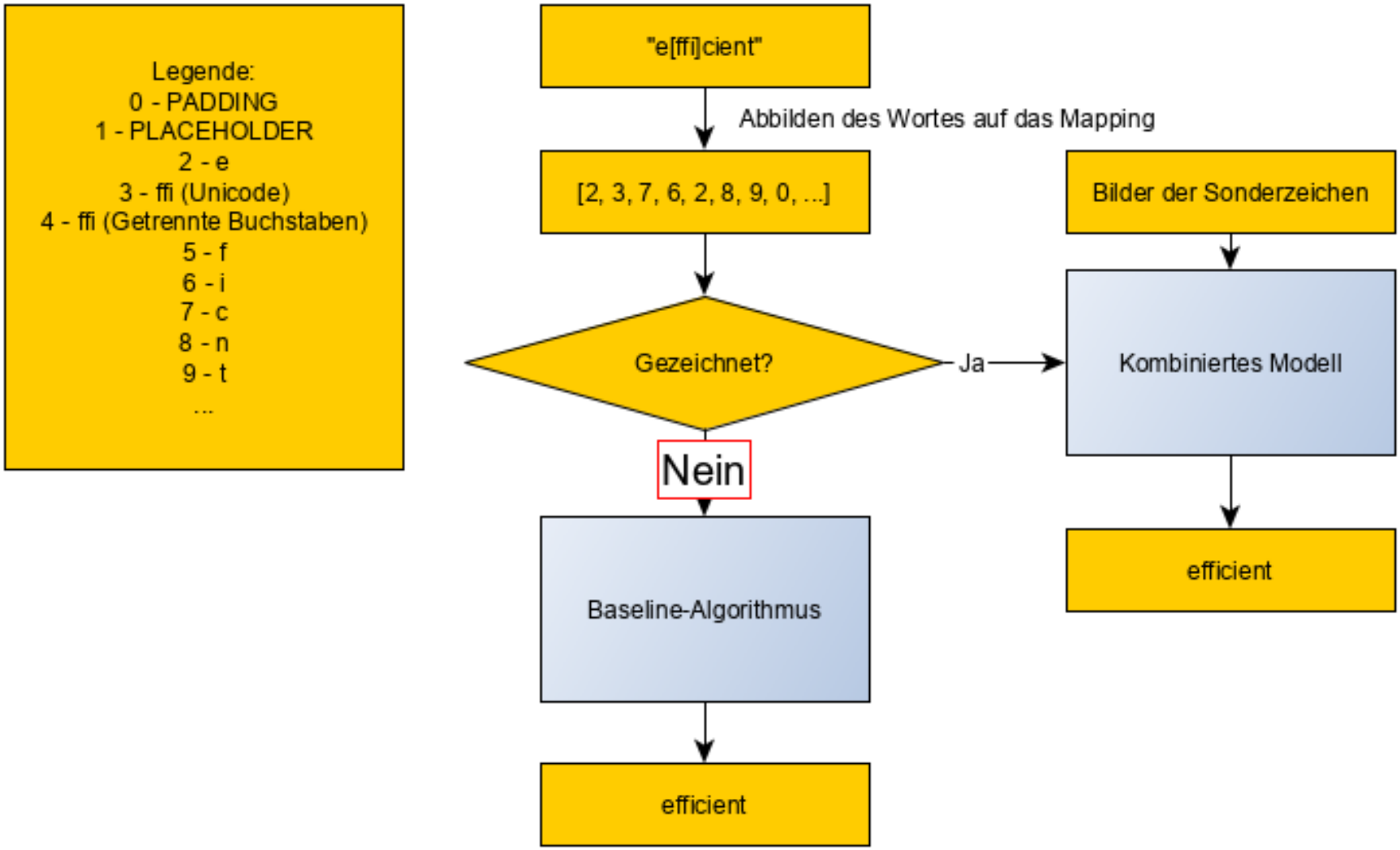
Kombiniertes Modell



Kombiniertes Modell mit Baseline-Algorithmus



Kombiniertes Modell mit Baseline-Algorithmus



Gliederung

Evaluation

Evaluationsmaß

Genauigkeit - Accuracy

Evaluation

Evaluationsmaß

Genauigkeit - Accuracy

Versionen von Evaluationsdateien

- "Normal" (1042 Dateien)
  Vollständige Infos sind vorhanden (Sonderzeichen mit Unicodes)
- "Gezeichnet" (510 Dateien)
  Sonderzeichen liegen nur gezeichnet vor (Kein Unicode hinterlegt)

Evaluation im Vergleich mit GROBID & pdftotext

Algorithmus	Ergebnis für alle PDFs	Ergebnis für "normale" PDFs	Ergebnis für "gezeichnete" PDFs	Laufzeit pro Dokument
Baseline	72,45 %	99,99 %	16,90 %	0,21 sec
pdftotext	58,91 %	87,00 %	1,52 %	0,43 sec
GROBID	70,08 %	96,00 %	17,13 %	2,76 sec
Kombiniertes Modell	97,54 %	99,51 %	93,53 %	192,39 sec
Kombiniertes Modell mit Baseline-Algorithmus	97,86 %	99,99 %	93,53 %	64,21 sec

Referenzen

Die Grafiken auf Folie 21 und 23 sind teilweise übernommen aus folgenden Quellen:

- https://www.eforce21.com/bilderkennung-mit-convolutional-neural-networks/
- https://www.sciencedirect.com/science/article/pii/S0010482518300738
- https://www.statworx.com/de/blog/bilder-lernen-mit-neuronalen-netzen/