Vom Bild zum Text – praktische OCR für die DH

—

19.05.2021, 15–17 Uhr: Postcorrection, Hackathon

Postcorrection

OCR-Nachkorrektur (allgemein)

Robert Sachunsky

Was ist OCR-Nachkorrektur?

- **interaktiv**: durch (sachkundige) Leser, mit GUI-Unterstützung (Textbild, Korrekturvorschläge, Konkordanz)

- **automatisch**: durch Algorithmus/Modell, komplementär (zusätzliches Vorwissen), aber rein textuell-statistisch

- **nachträglich**: OCR als 1. Näherung für einzelne Textzeilen, NK genauer für ganze Seiten/Dokumente

- **integriert**: OCR für Zeichenhypothesen und -Konfidenz, NK für Textergebnis (gemeinsamer Suchraum)

- **transduktiv**: Konsistenz-Maximierung über Seite/Dokument

- **induktiv-deduktiv**: Anwendung eines vorab erlernten allgemeinen Modells

Wie funktioniert OCR-Nachkorrektur?

Art des *Vorwissens*:

1. Fehlermodell (Wahrscheinlichkeit von Zeichenverwechslungen in konkreter Eingabe oder in OCR-Modell allgemein)
2. Wörterbuch (Lexik, Orthographie)
3. Sprachmodell (Wahrscheinlichkeit von Zeichen-/Wortfolgen)

Art der *Modellierung*:

- Zeichenebene vs. Wortebene
- Symbole / Automaten (z.B. FST) vs. Vektoren / Neuronale Netze (z.B. RNN)

Wann ist OCR-Nachkorrektur sinnvoll?

Kernproblem: sprachliche Varianz sehr groß!

➔ NK kann OCR verbessern (Rekonstruktion),
aber auch *verschlechtern* (Halluzination)
➔ oft beides zugleich (veränderte Fehlercharakteristik)

Training eher in OCR oder in NK investieren?

- GT als Textbild (mit Koordinaten) vs. rein textuell (Textkorpus)
- Modelle für spez. Material/Schrift vs. spez. Textsorte

Aufgabenteilung im Gesamtsystem:

1. zwischen visueller und textueller Information
2. zwischen lokalem und globalem Kontext

Worauf ist außerdem zu achten?

Evaluierung:

- wie OCR (CER/WER… und Konfusion auf GT, Perplexität bzgl. Modell)
- zusätzlich: Korrektheit der NK-Entscheidung (Precision/Recall/F1/MCC…)

Workflow-Abhängigkeiten:

- kompatible OCR-Engine (Konfidenz, Alternativen, Multi-OCR)
- passendes OCR-Modell (Zeichensatz, Fehlercharakteristik)
- Modell-Selektion/Parameter mittels Metadaten (Sprache, Domäne, Zeit)

OCR Nachkorrektur (Apoco)

Florian Fink

Überblick

- entwickelt am Centrum für Informations- und Sprachverarbeitung (CIS) der LMU München im Rahmen von OCR-D
- Projektbeteiligte: Klaus U. Schulz, Tobias Englmeier, Florian Fink
- automatische Nachkorrektur (apoco)
- interaktive automatische Nachkorrektur (pocoweb)
- Nachkorrektur basiert auf transduktivem Fehlermodell (*Profiler*) und induktiven Sprachmodell (*Entscheider*)

Profiler

- berechnet ein dokumentenabhängiges Sprachprofil:
  - vermutete historische Schreibvarianten: *mod → hist (t → th, ei → ey, ...)*
  - vermutete OCR-Fehler
  - zugrundeliegende moderne Wörter
- berechnet für jeden Token *w(ocr)* eine Menge möglicher Interpretationen *w(mod,cand) –α→ w(hist,cand) –β→ w(ocr)*:
  - α- (hist. Schreibvar.) und β- (OCR-Fehler) Kanäle können leer sein
  - Interpretationen haben eine Gewichtung
  - jedes *w(ocr)* hat eine sortierte Menge aus Interpretationen *w(hist,cand)*
- Profiler-Kandidaten dienen der Feature-Extraktion bei der Klassifikation

Profiler-Ressourcen

- modernes Vollformenlexikon
- Liste historischer Schreibvarianten
- (optional) weitere moderne bzw. historische Lexika
- (optional) historische Ground-Truth zur Abschätzung der Anfangswahrscheinlichkeiten historischer Schreibvarianten
- Profiler-Ressourcen für Deutsch, Latein und Griechisch auf Github

Vollformenlexikon und Schreibvarianten

modernes Vollformenlexikon:

✂ ----------

alter

alten

altem

altes

altäre

altären

altamerikaner

altangesehene

✂ ----------

historische Schreibvarianten (Format: mod:hist):

✂ ----------

lich$:sam$

lich$:ſam$

le$:ele$

s:ſ

ss:ſſ

r:□

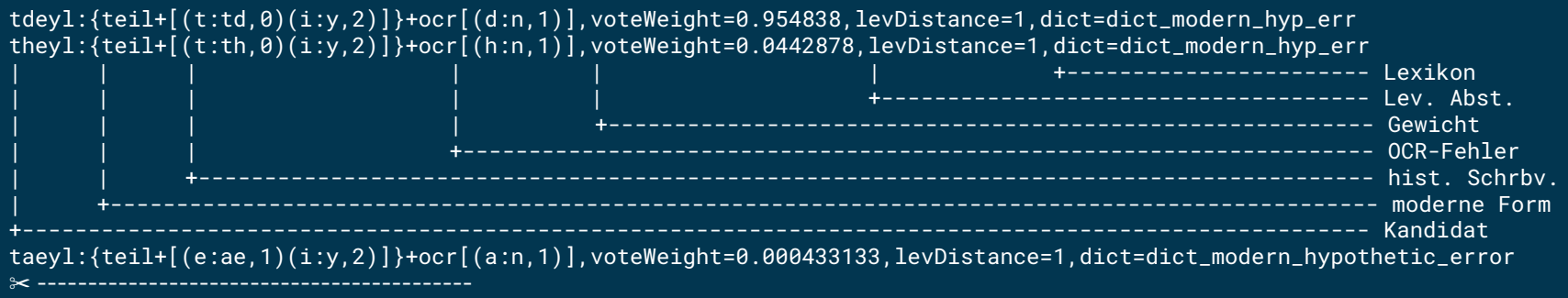
z:ʒ

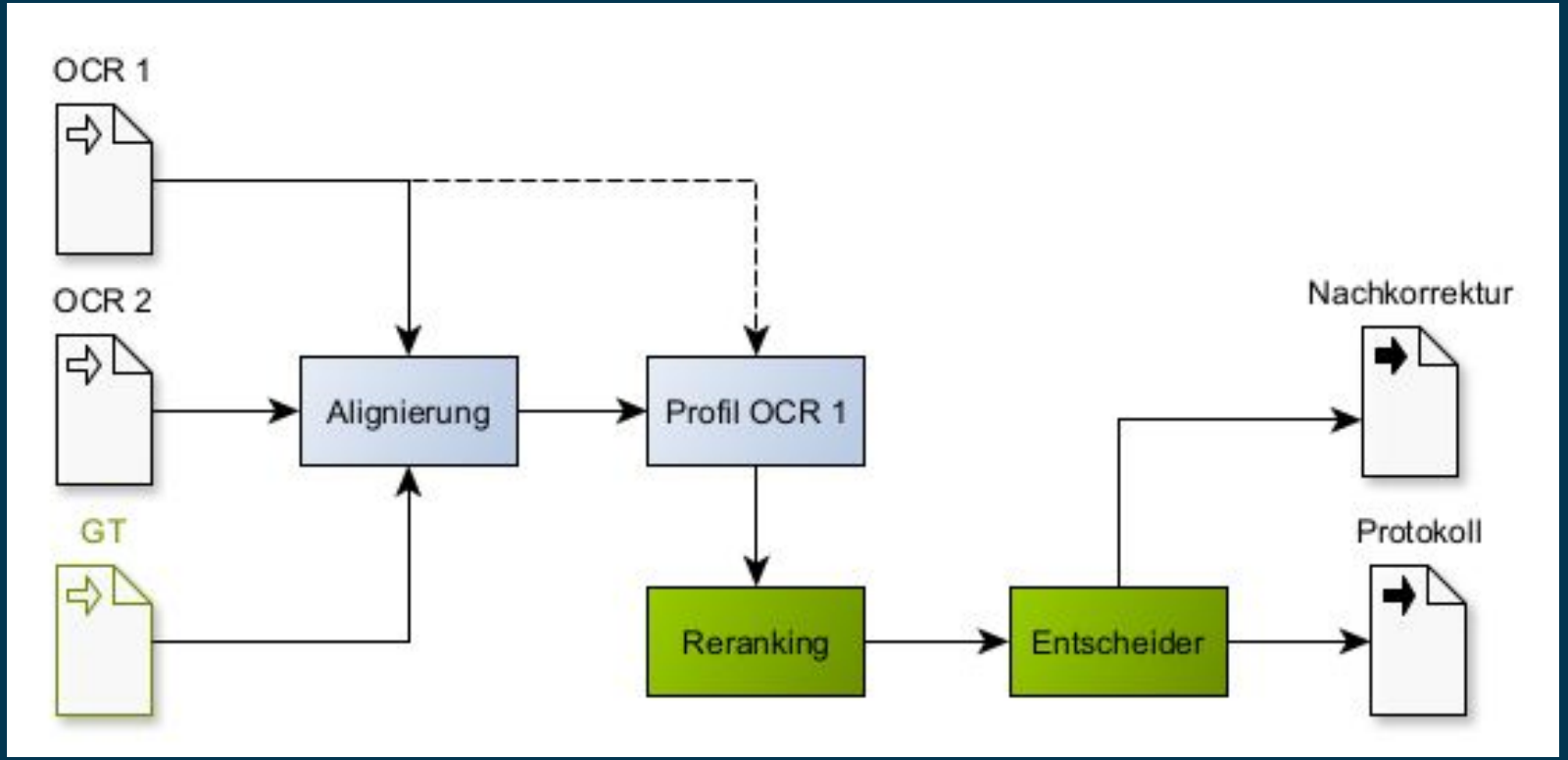
ä:e

✂ ----------

Profiler-Interpretationen

Interpretation für w(ocr) = tneil





Überblick automatisches Nachkorrektursystem

Automatische Nachkorrektur

- Eingabe:
  - eine primäre OCR
  - weitere Hilfs-OCR
  - (optional) Ground-Truth
- Aligner aligniert die OCRs (und die Ground-Truth) wortweise
- Erstellung eines Profils der primären OCR
- Reranking sortiert die Interpretationen der Token um
- Entscheider entscheidet ob ein Token mit dem bestgerankten Kandidat korrigiert werden soll oder nicht
- Entscheider und Reranker verwenden logistische Regression zur Klassifizierung
- Schreiben der Korrekturen und der Korrekturentscheidungen

Nachkorrektur mit OCR-D

- ocrd workspace init ...
- ocrd workspace add ...
- Bildvorverarbeitung, Segmentierung
- OCR-Erstellung
  - ocrd-tesserocr-recognize -O OCR1 ...
  - ocrd-tesserocr-recognizse -O OCR2 ...
- ocrd-cis-align -I OCR1,OCR2 -O ALGN -p config.json
- ocrd-cis-post-correct -I ALGN -O COR -p config.json

Auswertung der Nachkorrektur (Überblick)

- ocrd-cis-apoco print protocol -l COR ... | ocrd-cis-apoco print stats

```
Name                                = COR
Char error rate (before/after)      = 0.06530402077549924/0.0598638846601594
Char errors (before/after)          = 14585/13370
Total chars                         = 223340
Improvement (percent)               = 4.255832905737829
Error rate (before/after)           = 0.18601051545378605/0.15136848312122256
Accuracy (before/after)             = 0.8139894845462139/0.8486315168787775
Total errors (before/after)         = 7571/6161
Correct (before/after)              = 33131/34541
Total tokens                        = 40702
Successful corrections              = 1493
Missed opportunities                = 290
Infelicitous corrections            = 83
False friends                       = 469
Short errors                        = 1617
Merges                              = 133
Splits                              = 418
```

Auswertung der Nachkorrektur (Korrekturen)

- ocrd-cis-apoco print protocol -I COR | ocrd-cis-apoco print type | grep -i succ

```
id=COR_179392_line_1_1_word0002 skipped=false short=false lex=false cor=true
conf=0.7918525154224578 rank=1 ocr=velgien sug=belgien gt=belgien type=SuccessfulCorrection
id=COR_179392_line_1_8_word0005 skipped=false short=false lex=false cor=true
conf=0.8202208112107483 rank=1 ocr=uleberfahrt sug=ueberfahrt gt=ueberfahrt
type=SuccessfulCorrection
id=COR_179392_line_1_13_word0007 skipped=false short=false lex=false cor=true
conf=0.5294665408056373 rank=1 ocr=msterdam sug=amsterdam gt=amsterdam type=SuccessfulCorrection
id=COR_179392_line_1_14_word0004 skipped=false short=false lex=false cor=true
conf=0.7931126963024077 rank=1 ocr=ungebindert sug=ungehindert gt=ungehindert
type=SuccessfulCorrection
id=COR_179392_line_1_18_word0008 skipped=false short=false lex=false cor=true
conf=0.7861797669809238 rank=1 ocr=eontrolirt sug=controlirt gt=controlirt
type=SuccessfulCorrection
id=COR_179392_line_1_19_word0000 skipped=false short=false lex=false cor=true
conf=0.785752726120388 rank=1 ocr=sede sug=jede gt=jede type=SuccessfulCorrection
id=COR_179392_line_1_21_word0003 skipped=false short=false lex=false cor=true
conf=0.8198736884573661 rank=1 ocr=seisesten sug=leisesten gt=leisesten
type=SuccessfulCorrection
✂----------------------------------------
```

Auswertung der Nachkorrektur (Verschlimmb.)

- ocrd-cis-apoco print protocol -I COR | ocrd-cis-apoco print type | grep -i infel

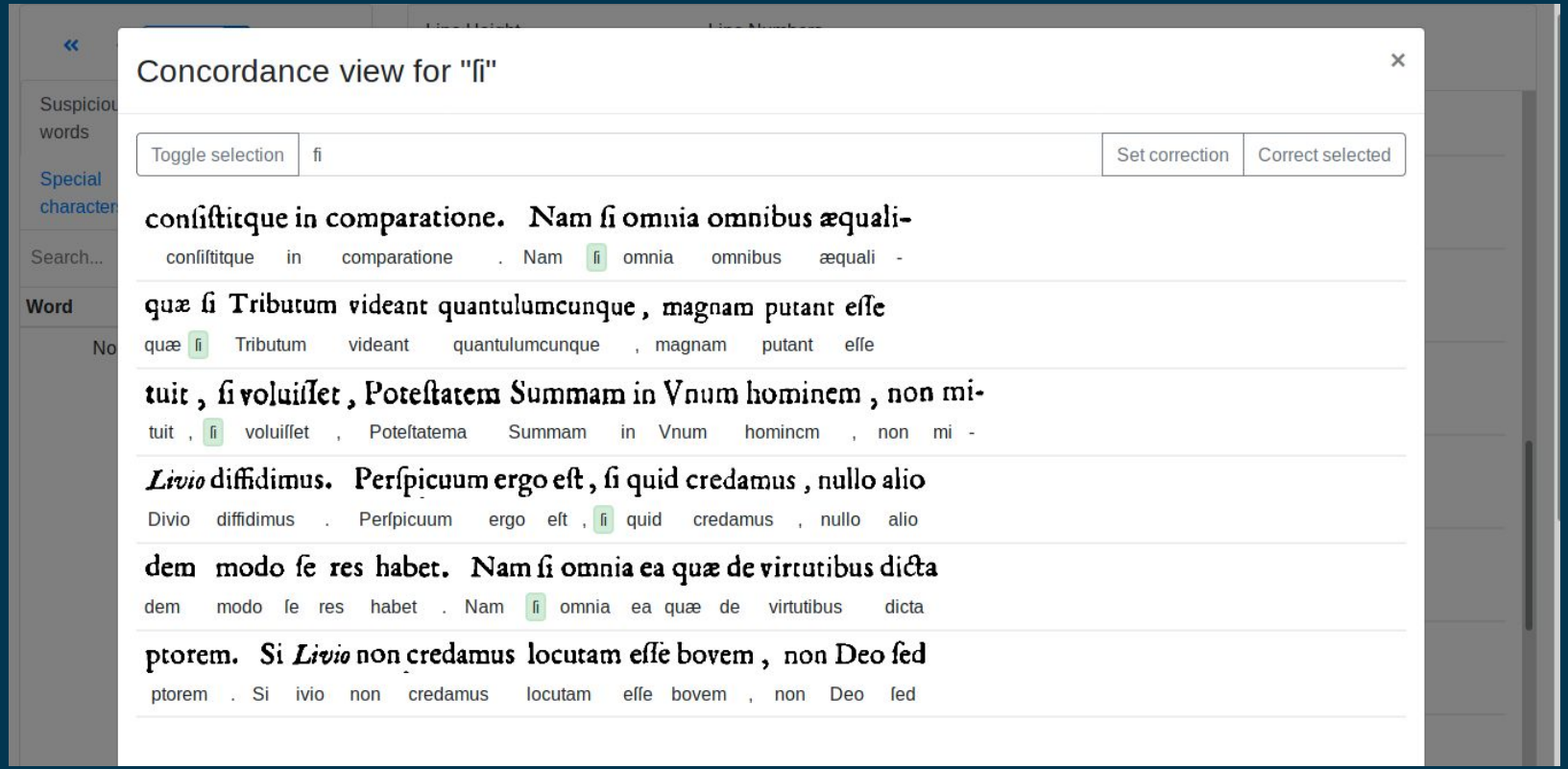
```
id=COR_179392_line_1_4_word0000 skipped=false short=false lex=false cor=true
conf=0.5117900839309029 rank=0 ocr=tiviren sug=tipiren gt=tiviren
type=InfelicitousCorrectionMissingCandidate
id=COR_179393_line_1_8_word0004 skipped=false short=false lex=false cor=true
conf=0.5238973778176376 rank=0 ocr=niederland sug=niederlande gt=niederland
type=InfelicitousCorrectionMissingCandidate
id=COR_179398_line_1_15_word0002 skipped=false short=false lex=false cor=true
conf=0.5096285956816532 rank=0 ocr=eilften sug=eiltten gt=eilften
type=InfelicitousCorrectionMissingCandidate
id=COR_179400_line_1_19_word0009 skipped=false short=false lex=false cor=true
conf=0.5117712645756004 rank=0 ocr=ostende sug=āstende gt=ostende
type=InfelicitousCorrectionMissingCandidate
id=COR_179402_line_1_23_word0003 skipped=false short=false lex=false cor=true
conf=0.5035075949269625 rank=0 ocr=ostende sug=astende gt=ostende
type=InfelicitousCorrectionMissingCandidate
id=COR_179403_line_1_3_word0011 skipped=false short=false lex=false cor=true
conf=0.7848720573059577 rank=0 ocr=kara sug=dara gt=kara
type=InfelicitousCorrectionMissingCandidate
✂----------------------------------------
```

Interaktive Nachkorrektur

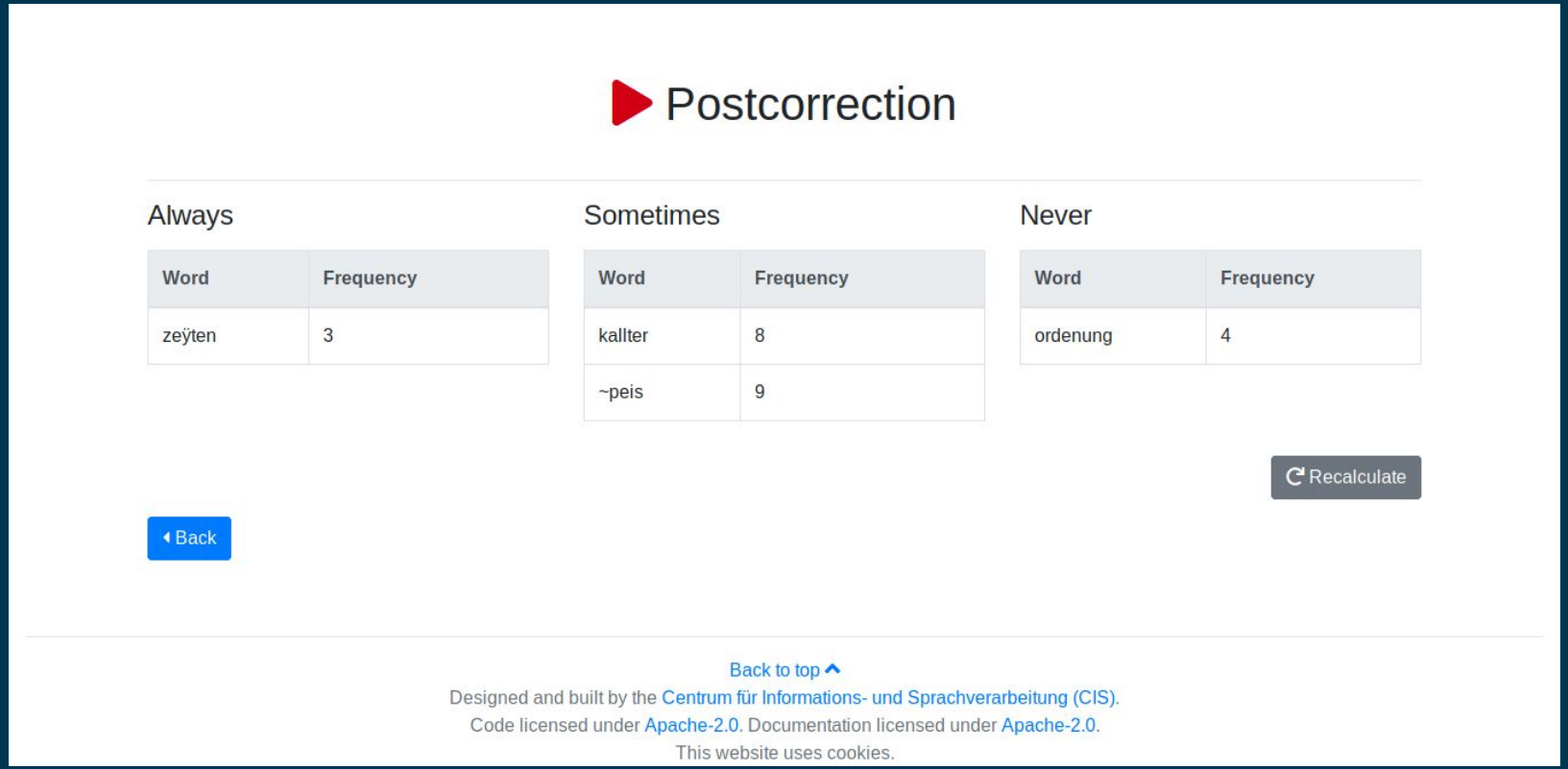
- interaktive Nachkorrektur
- Unterstützung der manuellen Nachkorrektur:
  - Stapelverarbeitung zur Korrektur von Wortfolgen
  - Auflistung und Verbesserung vermuteter OCR-Fehler
  - Auflistung von Korrekturvorschlägen
  - Schnittstelle zur automatischen Nachkorrektur
- unterstützt paralleles Arbeiten auf Dokumenten

Interaktive Nachkorrektur (Pocoweb)

Interaktive Nachkorrektur (Konkordanz)



Interaktive Nachkorrektur (automatisch)



Aktuelle Arbeiten

- Verbesserungen und Auswertung der automatischen Nachkorrektur
- Fertigstellung der interaktiven Nachkorrektur
- Finalisierung der interaktiven automatischen Nachkorrektur
- Arbeiten zur Behandlung von Splits/Merges
- zwei allgemeine Modelle (trainiert auf 2 OCRs):
  - 19th trainiert auf GT4HistOCR
  - pre19th trainiert auf GT4HistOCR
- spezielle Modelle bringen kaum Verbesserungen
- Anpassung der Profiler-Ressourcen verbessern die Nachkorrektur

Referenzen

- Profiler Ressourcen und Anleitungen auf Github
- automatische Nachkorrektur auf Github
- interaktive Nachkorrektur auf Github
- Ulrich Reffle and Christoph Ringlstetter. Unsupervised profiling of OCRed historical documents. Pattern Recognition
- Tobias Englmeier, Florian Fink, Klaus U. Schulz: A-I-PoCoTo: Combining Automated and Interactive OCR Postcorrection. DATeCH
- Uwe Springmann, Christian Reul, Stefanie Dipper, Johannes Baiter: Ground Truth for training OCR engines on historical documents in German Fraktur and Early Modern Latin

OCR-Nachkorrektur (ASV)

Robert Sachunsky

Zwei Module, zwei Ansätze

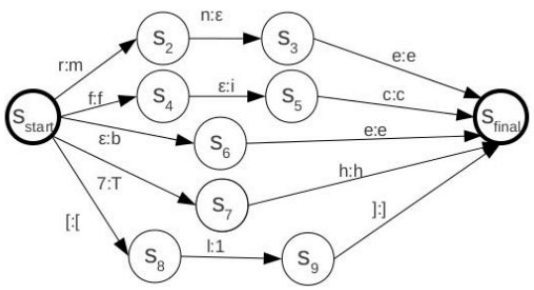
WFST-basierte Nachkorrektur cor-asv-fst:

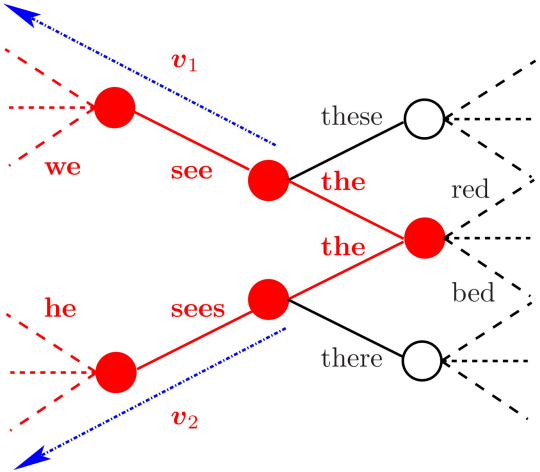
- OCR-Hypothesen als Akzeptor
- induktives Fehlermodell als Transduktor
- (historisches) Lexikon als Akzeptor
- Berechnung durch FST-Komposition
- schnelles RNN-Sprachmodell
- Kombination per (inkrementeller) Graph-Dekodierung mit Pruning
- A*-Beam-Decoder für Pfadsuche
- Sliding-Window-Verarbeitung → Wort-Resegmentierung
- dynamische Rückweisungsschwelle
- OCR-D-Schnittstelle

LSTM-basierte Nachkorrektur cor-asv-ann:

- induktives Gesamtmodell als Encoder-Decoder-RNN mit Attention (wie maschinelle Übersetzung)
  - (Encoder ≈ implizites Fehlermodell)
  - (Decoder ≈ implizites Sprachmodell)
- monoton-lokales Attention-Modell (gegen “Vergessen” und “Einrasten”)
- Lücken/Fremdzeichen: Unterspezifikation
- A*-Beam-Decoder für Pfadsuche
- durchgängig auf Zeichenebene
- Alignierung → Wort-Resegmentierung
- dynamische Rückweisungsschwelle
- OCR-D-Schnittstelle

FST-RNN-Graphdecoder mit Sliding-Window



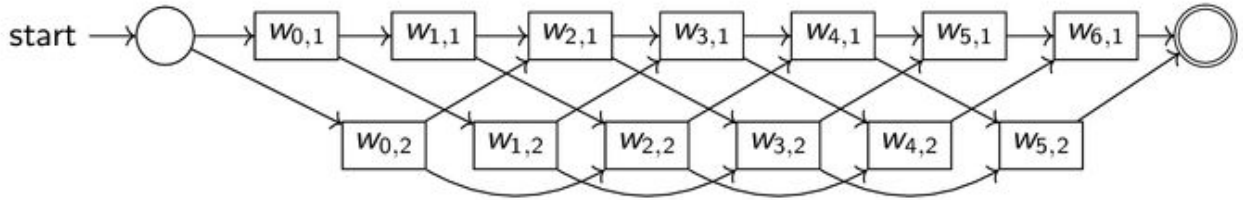


Philoſophen von Jndien durch Griechenland bis

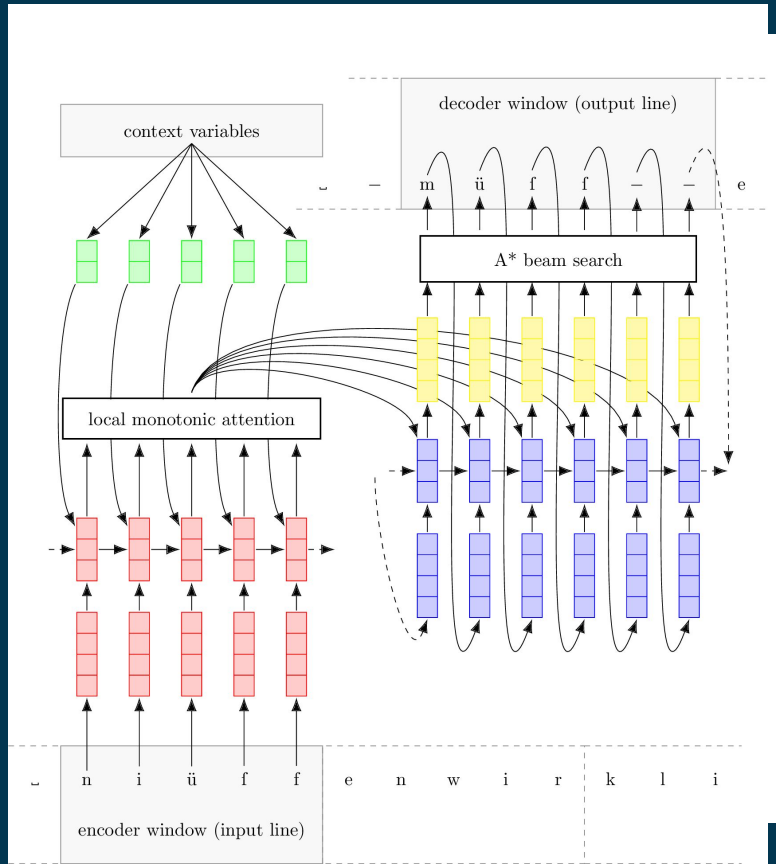
Philoſophenvon␣Fndicn␣dur<h␣Gricche␣nland␣bls ✘

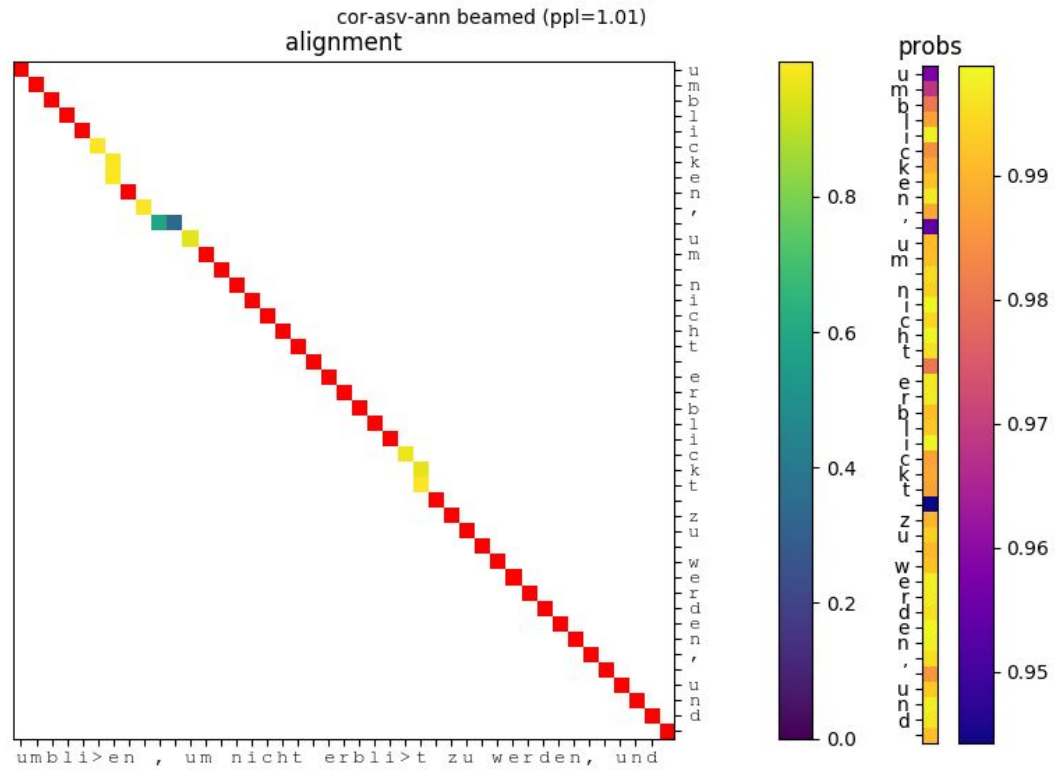
Philoſophen␣von␣Jndien␣
Jndien␣durch␣
durch␣Grieche␣
Griechenland␣
land␣bis

Philoſophen␣von␣Jndien␣durch␣Griechenland␣bis ✓



Encoder-Decoder-Modell mit Attention-Alignierung





Modelle

- in beiden Modulen:
  - weitere CLI-Tools für Datenhaltung und Training
  - Subrepos für vortrainierte Modelle (cor-asv-ann-models, cor-asv-fst-models)
  - Probleme während Projektlaufzeit:
    1. kein repräsentativer großer GT → GT4HistOCR-Subkorpus für Fraktur 19. Jh.
    2. keine adäquaten OCR-Modelle → Tesseract `Fraktur|frk|deu-frak`, Ocropus
    3. keine OCR-Engine mit Hypothesenausgabe → "blind" (außer `deu-frak`)
  - vortrainierte Modelle: nur für diese Konfigurationen, nur in-domain brauchbar!

Modell	CER (OCR)	CER (cor-asv-ann)	CER (cor-asv-fst/Lexikonorakel)
dta19.Fraktur (Tesseract4)	4,9 %	2,3 %	2,8 %
dta19.deu-frak (Tesseract3)	8,4 % (mit NFKC: 5,5 %)	3,9 % (mit NFKC: 3,8 %)	4,7 %
dta19.fraktur (Ocropus1)	7,8 %	4,2 %	3,8 %
dta19.fraktur-jze (Ocropus1)	5,7 %	4,3 %	4,3 %

Beispiel: s-ſ-Rekonstruktion per Nachkorrektur

1. Volltext mit “ſ” nehmen, automatisch auf “s” reduzieren
2. Paar (vorher/nachher) als Rekonstruktion trainieren → NK-Modell
3. OCR ohne “ſ” (oder modernen Text) nehmen, Modell anwenden:

```
ocrd-cor-asv-ann-process -I TEXT -O CORR \
    -P model_file s2s.gt4histocr.s-ſ.*.h5 -P rejection_threshold 0.9
```

4. Evaluierung: Schritte 1 und 3 auf anderen Daten (als “GT”), dann:

```
ocrd-cor-asv-ann-evaluate -I ORIG,TEXT,CORR -O EVAL \
    -P confusion 20 -P metric Levenshtein
-   234 lines 0.023±0.031 CER overall / ORIG vs TEXT
-   234 lines 0.001±0.005 CER overall / ORIG vs CORR
- most frequent confusion / ORIG vs TEXT: ([(152, ('s', 'ſ'))], 6151)
- most frequent confusion / ORIG vs CORR: ([(5, ('ſ', 's'))], 6151)
(davon: 3 falsch in ORIG, 2 falsch in CORR: “Auſſprechung”, “deſwegen”)
```

OCR Nachkorrektur (Qurator)

Robin Schäfer

Motivation

Möglichkeiten zur Verbesserung der OCR Qualität

1. Verbesserung der OCR Pipeline
2. Nachkorrektur von fehlerhaften OCR Daten

Hier wird Ansatz 2 verfolgt. Die Nachkorrektur erfolgt automatisiert.

Herausforderung: Fehler sollen korrigiert werden ohne korrekte OCR Daten fälschlicherweise zu verändern (“Verschlimmbesserung”)

Technischer Ansatz

Die Nachkorrektur wird als Übersetzungsproblem definiert; dieses versuchen wir mit Machine Learning, genauer Deep Learning, zu lösen.

Unkorrigierte/Fehlerhafte OCR Daten werden in korrigierte OCR Daten “übersetzt” (analog zur Sprachübersetzung, z.B. Deutsch zu Englisch).

Während des Trainings des Übersetzungsmodells werden die OCR Daten mit korrekten Ground Truth Daten verglichen.

Es werden keine Wörterbücher etc. verwendet.

Daten

Für das Trainieren haben wir OCR von 63 Werken aus dem Deutschen Textarchiv erstellt, die aus dem 17. - 19. Jahrhundert stammen. (CER: 1,1 %)

Mit besserer CER wird das Korrekturproblem anspruchsvoller; eine CER von 1,1 % ist bereits relativ niedrig.

OCR Daten und GT wurden auf Zeilenebene aligniert (mit dem von Mike Gerber implementierten dinglehopper Tool).

Nicht-deutschsprachige Zeilen wurden entfernt, um ein besseres Trainingsergebnis zu erhalten.

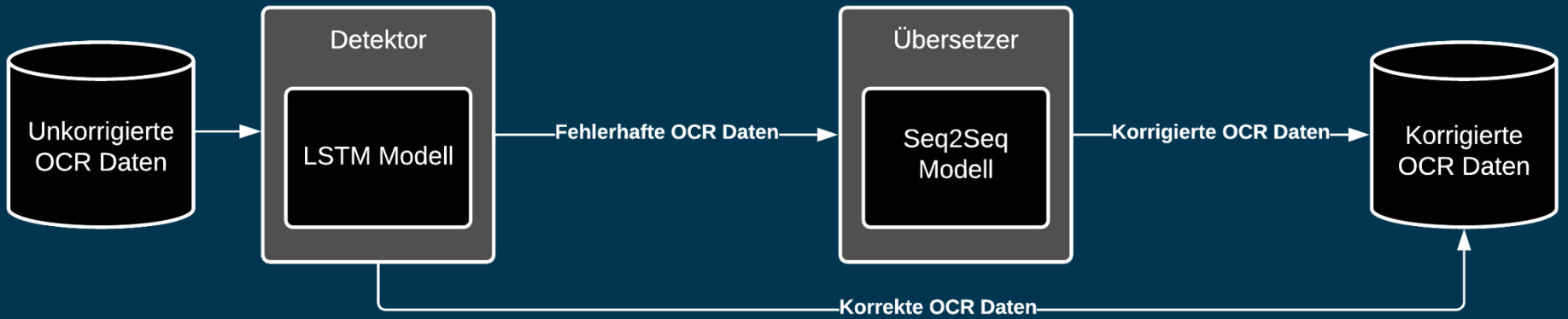
Komponenten der Nachkorrektur-Pipeline

Zur Erinnerung: der Anteil der Verschlimmbesserungen soll möglichst niedrig bleiben.

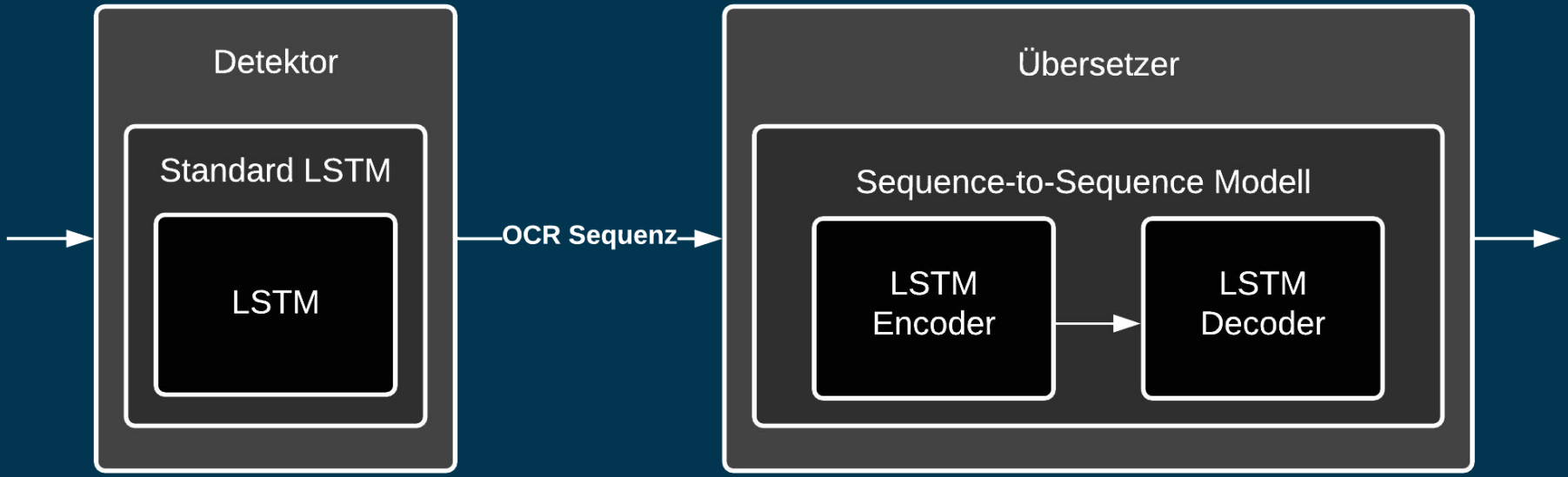
Wir erreichen dies über zwei Modellkomponenten:

- Detektor: Identifizierung von fehlerhafter OCR
- Übersetzer: Korrektur von fehlerhafter OCR

2-Schritt-Pipeline



2-Schritt-Pipeline (etwas mehr Details)



Parameter für Detektor Training

```
(qurator) robin@robin-ThinkPad-T570:~/Qurator/mono-repo$ train-detector --help
Usage: train-detector [OPTIONS] OCR_DIR GT_DIR TARGETS_DIR MODEL_OUT_DIR
                      TOKEN_TO_CODE_DIR

  Train detector component of OCR post-correction pipeline.

  Arguments:
  ocr-dir -- The absolute path to the OCR data
  gt-dir -- The absolute path to the GT data
  targets-dir -- The absolute path to the targets
  model-out-dir -- The absolute path for the trained models
  token-to-code-dir -- The absolute path to the token-encoding mapping

Options:
  --hidden-size INTEGER  Hidden dimension of RNN architecture. (default: 512)
  --batch-size INTEGER   The training batch size. (default: 200)
  --n-epochs INTEGER     The number of training epochs. (default: 1000)
  --lr FLOAT             The learning rate. (default: 0.0001)
  --node-type TEXT       The RNN type (LSTM/GRU). (default: lstm)
  --n-layers INTEGER     The number of RNN layers. (default: 2)
  --bidir / --no-bidir   --bidir: Train model bidirectional; --no-bidir: Train
                         model monodirectional. (default: false)

  --dropout-prob FLOAT   The dropout probability. (default: 0.2)
  --help                 Show this message and exit.
```

Parameter für Übersetzer Training

```
(qurator) robin@robin-ThinkPad-T570:~$ train-translator --help
Usage: train-translator [OPTIONS] OCR_DIR GT_DIR MODEL_OUT_DIR
                        TOKEN_TO_CODE_DIR

  Train translator component of OCR post-correction pipeline.

  Arguments:
  ocr-dir -- The absolute path to the OCR data
  gt-dir -- The absolute path to the GT data
  model-out-dir -- The absolute path for the trained models
  token-to-code-dir -- The absolute path to the token-encoding mapping

Options:
  --approach TEXT                 The OCR post-correction approach ("seq2seq" or
                                  "gan").

  --hidden-size INTEGER           Hidden dimension of RNN architecture.
                                  (default: 512)

  --batch-size INTEGER            The training batch size. (default: 200)
  --n-epochs INTEGER              The number of training epochs. (default: 1000)
  --lr FLOAT                      The learning rate. (default: 0.0001)
  --n-layers INTEGER              The number of RNN layers. (default: 2)
  --attention / --no-attention    --attention: Use attention mechanism; --no-
                                  attention: Use no attention mechanism.
                                  (default: True)

  --dropout-prob FLOAT            The dropout probability. (default: 0.2)
  --teacher-ratio FLOAT           The teacher ratio probability. (default: 0.5)
  --help                          Show this message and exit.
```

Beispiele (blau=korrekt; orange=fehlerhaft)

1.
OCR: und die rundlage seines
GT: und die Grundlage seines
Korrigiert: und die Grundlage seines

2.
OCR: werden, es aher nichtsdestoweniger
GT: werden, es aber nichtsdestoweniger
Korrigiert: werden, es aber nichtsdestoweniger

3.
OCR: ler uroper aufgestellten Stühle
GT: der Europäer aufgestellten Stühle
Korrigiert: der Euraner aufgestellten Stühle

Bisherige Ergebnisse

Wir haben die Pipeline mit und ohne Detektor Modell getestet.

Ansatz	CER (vorher)	CER (nachher)
Mit Detektor	1,1 %	0,9 %
Ohne Detektor	1,1 %	2,1 %

Ein Verbesserung von 1,1 % auf 0.9 % bedeutet eine relative Verbesserung von 18,2 %. Verschlimmbesserungen: 0.3 %

Dieses Modell ist recht konservativ, um eine möglichst geringe Rate an Verschlimmbesserungen zu erzielen.

Veröffentlichung von Code und Ergebnissen

Ansatz und Ergebnisse wurden in einem Paper veröffentlicht:

Schaefer & Neudecker 2020: A Two-Step Approach for Automatic OCR Post-Correction.
https://www.aclweb.org/anthology/2020.latechclfl-1.6/

Der Code ist auf Github zu finden:

https://github.com/qurator-spk/sbb_ocr_postcorrection

Ausblick OCR Nachkorrektur

Die OCR Nachkorrektur Pipeline soll in den OCR-D-Workflow integriert werden.

Die Nachkorrektur Pipeline kann über verschiedene Wege verbessert werden:

1. Alternativer Trainingsansatz: Generative Adversarial Networks (GAN)
2. Optimierung des Übersetzers: Alternativer Attention-Algorithmus

Hackathon

Daten?

Probleme?

Ergebnisse?

Publikumsbeteiligung!