UNIVERSITÄT LEIPZIG

OCR-D: Nachkorrektur mit neuronalen Netzen und endlichen Transduktoren

Robert Sachunsky, Gerhard Heyer

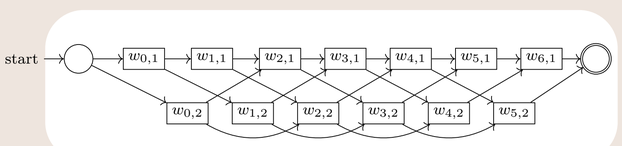
ASV Automatische Sprachverarbeitung

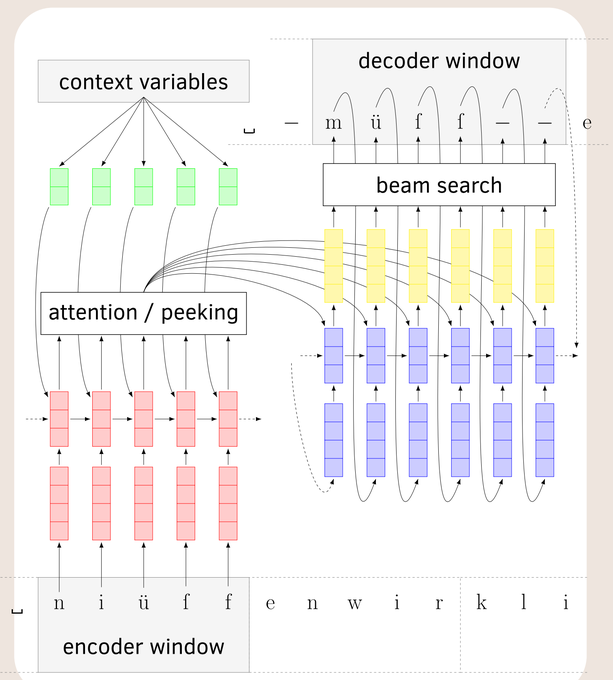
git.informatik.uni-leipzig.de/ocr-d

github.com/ASVLeipzig
github.com/bertsky

Aufgabe

- stoch. Sprachmodellierung gegen OCR-Fehler (Text)
- datengetrieben/Vorwissen
- Experimente mit ANN- und FST-Ansätzen, Kombination
- GT-Verarbeitungskette
- neue Tools, erste Modelle





RNN-Sprachmodell `ocrd_keraslm`

- auf Zeichen, mit Unterspezifikation und Kontext-Konditionierung
- Stacked/Bi-LSTM mit A*-Beamsearch und Pruning
- Zustandstransfer implizit/stateful oder explizit/inkrementell
- effiziente Graph-Dekodierung (bester Hypothesen-Pfad)
- API+CLI (standalone/OCR-D): Suche/Evaluierung/Training
- Modelle trainiert auf DTA

WFST-Nachkorrektur `cor-asv-fst`

- auf Wort-Ebene, mit Sliding-Window für Resegmentierung
- effiziente Komposition von Fehlermodell und Wortmodell
- Fenster-Rekombination/Dekodierung mit RNN-Sprachmodell
- CLI (standalone/OCR-D): Korrektur/Training

RNN-Nachkorrektur `cor-asv-ann`

- auf Zeichen-Ebene, NMT-ähnliche Encoder-Decoder-Architektur mit lokal-monotoner Soft-Attention und LM-Transfer
- Stacked/Bi-LSTM mit A*-Beamsearch und *Rückweisung*
- CLI (standalone/OCR-D): Korrektur/Evaluierung/Training
- Modelle trainiert auf GT4HistOCR/OCR-D-GT

Weiterentwicklung

- enge Integration mit OCR
- Multi-OCR-Alignment
- Training auf großen Daten
- (erneute) Evaluierung
- Performanz-Optimierung
- Adaption/Transduktion

RNN-Sprachmodell

- (Kurz-)Textklassifikation
- Sprachklassifikation

WFST-Nachkorrektur

- Morphologie, OOV
- Interpunktionsmodell

RNN-Nachkorrektur

- *exposure bias, label bias*
- Kontext-Konditionierung
- *stateful decoding*

Sonstiges

- Workflow-Experimente
- Kopplung OCR/Klassifikation mit OLR, Reading-Order
- OLR-Evaluierung

Bild-API `spec, core`

- PAGE-XML: Annotation vs. *abgeleitete* Bilddaten auf allen Hierarchie-Ebenen
- Koordinaten-*Konsistenz* absolut/relativ (Aufpunkt, Winkel, Drehpunkt)
- Umrechnung, Polygone, Maskierung/Transparenz, Attribute
- Konzept+Implementierung

Ocropus `ocrd_cis`

- OCR-D-Wrapper (Binarisierung, Despeckling, Deskewing, Dewarping, Zeilensegmentierung)
- Entflechtung CLIs und ocrolib, PIL statt Dateien, Parameter
- Zeilensegmentierung überholt (DPI/Skalen-Abhängigkeit, Robustheit, Polygone, Seite vs. Region)
- Deskewing mit Konfidenz und rein deskriptiv
- neu: Resegmentierung (Polygonalisierung)
- neu: Clipping (Maskierung von Überlappungen)

`workflow-configuration`

- `make` als Workflow-Engine für inkrementelle und GPU-/CPU-parallele Berechnung
- Makefiles für (einfache, persistente) Konfiguration
- Wiederholbarkeit, Referenzierbarkeit, Installation
- erprobte Konfigurationen
- Messungen auf GT

Binarisierung `ocrd_olena`

- Olena (Build, Repo, Upstream-Patches, CI, Docker)
- OCR-D-Wrapper (Bashlib, Konvertierung, Parameter, PAGE-XML)

Tesseract `ocrd_tesserocr, tesserocr, tesseract`

- OCR-D-Wrapper (Deskewing/OSD, Segmentierung, Erkennung)
- möglichst optimale Ausnutzung der Tesseract-API
- Prototyping für OCR-D-API
- Upstream-Patches (Wörterbücher, Whitelisting, Polygone)
- Alternativen-Ausgabe (in Kollaboration)
- Graph-Ausgabe (als Prototyp)

Deployment `ocrd_all`

- Aggregation aller kompatiblen Module (Subrepos)
- zur *Installation* und Aktualisierung aus Quellen oder per Docker
- Kapselung aller System-/Python-Abhängigkeiten und Besonderheiten
- De-/Aktivierung von Submodulen, Anpassung
- Dokumentation

Bildwandlung `ocrd_im6`

- OCR-D-Wrapper (Bashlib, Parameter, PAGE-XML)

Segmentierung `ocrd_segment, core`

- Validierung und heuristische Reparatur von Koordinaten
- Extraktion abgeleiteter Bilddaten und Metadaten