Automatische OCR-Korrektur mit LLMs

Eine Evaluierung

Worum geht es …

- OCR wird heute von OCR-Engines (Tesseract, Abbyy, Kraken, Calamri, Google,…) bei guten Scans in guter Qualität geliefert.
- Trotzdem gibt es immer noch Fehler in der OCR.
- Wir haben uns gefragt, ob diese Fehler durch den Einsatz von Large-Language-Modellen (z.B. ChatGPT) behoben werden können.
- Als Beispiel-Datensatz dienten Zeitungen von 1823.

Unser Ansatz im Überblick

- Wir haben einen Schwerpunkt in Zeitungsdigitalisierung
- Somit ist der „textuelle Master“ der Artikel
  - ggf. über mehrere Spalten oder Seiten !
- Ablauf
  - Normaler Digitalisierungs-Durchlauf mit docWizz (CCS-eigene Software)
  - Inkl. manueller „perfekter“ Nachkorrektur der Artikelstruktur
  - Und, OCR-GT Erstellung zum Vergleich
  - Artikel-Texte werden dann den LLMs übergeben zum „korrigieren“
  - Vergleich des Ergebnisses der normalen Digitalisierung und nach der LLM-Korrektur mit der OCR-GT

Unser Ansatz im Detail

- Erster Schwerpunkt war „Prompt Engineering“
- Die Ergebnisse dann analysieren, um zu verstehen
  - welches Pre- und Postprocessing nötig ist
  - Welche OCR-Fehler korrigierbar sind, welche ggf. nicht, welche neue Fehler erzeugt werden

Prompt Engineering - I

Correct OCR-induced errors in the text, ensuring it flows coherently with the previous context.

Follow these guidelines:

1. Fix OCR-induced typos and errors:
   - Correct words split across line breaks
   - Fix common OCR errors (e.g., 'rn' misread as 'm')
   - Use context and common sense to correct errors
   - Only fix clear errors, don't alter the content unnecessarily
   - Do not add extra periods or any unnecessary punctuation
2. Maintain original structure:
   - Keep all headings and subheadings intact
3. Preserve original content:
   - Keep all important information from the original text
   - Do not add any new information not present in the original text
   - Remove unnecessary line breaks within sentences or paragraphs
   - Maintain paragraph breaks
   - Do not change uppercase words to lowercase
   - The text is from 1823, preserve the language of that time
4. Maintain coherence:
   - Ensure the content connects smoothly with the previous context
   - Handle text that starts or ends mid-sentence appropriately

IMPORTANT: Respond ONLY with the corrected text. Preserve all original formatting, including line breaks. Do not include any introduction, explanation, or metadata.

Prompt Engineering - II

- Nach diesem initialen Prompt wurden dann kontinuierlich als Schleife über alle Artikeltexte über alle „Textabschnitte“ Prompts erstellt mit
  - “Previous context, do not correct:”
    - Folgend einem Textabschnitt vor dem zu korrigierenden Text (beim ersten Abschnitt = “”)
  - “Text to correct:”
    - Folgend dem Textabschnitt mit dem zu korrigierenden Text
  - Zum nächsten Schleifendurchlauf wurde der “Previous context” mit dem Textabschnitt belegt, der gerade zur Korrektur vorgelegt wurde.

Preprocessing

- Als wichtige Evaluierung im Pre-Processing wurden verschiedene Varianten von “Textabschnitten” ausprobiert, in die der Artikeltext zerlegt wurde.
  - Absätze, Zeilen, Sätze, “ein paar Worte”,...
- → “Sätze” haben sich als am Robustesten herausgestellt (siehe auch später Fehlerbilder)

Erkenntnisse

- Grundsätzlich können LLMs sehr ordentlich (typische) OCR-Fehler korrigieren („rn<->m“)
- Bei unserem Vergleich zwischen ChatGPT und einem lokalen LLM (Llama-variante), war ChatGPT deutlich besser – vielleicht, weil wir einen deutlich längeren, genaueren Prompt formulieren konnten.
- LLMs sind grundsätzlich „gesprächig“, d.h. sie tendieren dazu (Halb-)Sätze zu ergänzen.
- LLMs sind auf moderne Sprache trainiert, so werden alte Wörter („to-day“) einfach in die moderne Variante gewandelt.
- Auch grammatikalisch gibt es Unterschiede. So wurde ein Adverb in der Vergangenheit im Englischen eher an leicht anderer Position verwendet → da werden dann Wortvertauschungen vorgenommen
- Interpunktionen aller Art (z.B. diverse Variante von . , „ ‘) können auch hier nicht eindeutig erkannt werden.

Postprocessing

- Ein wesentlicher Aspekt des Postprocessing war es zu identifizieren, wo das LLM neue Fehler eingebracht hat, z.B.
  - Erkennen, wenn neue (Halb-)Sätze dazu erfunden worden sind.
  - Erkennen, wenn Wort-Reihenfolgen vertauscht worden sind.
- Hilfreich sind auch (Zeit-)spezifische Wörterbücher („to-day“).
- Ignoriere Interpunktion.

Fazit

- Bewusst „weich“ formuliert:
  - LLMs können dazu beitragen, die OCR-Qualität automatisch zu verbessern, wenn man gewissen Rahmenbedingungen akzeptiert bzw. durch Pre-/Postprocessing sicherstellt.
- Für uns (CCS) hat das durchaus ein (kommerziellen) Nutzen unter bestimmten Projektbedingungen einzusetzen.

Kontakt

CCS Content Conversion Specialists – Hamburg/Germany

Stefan von der Heide (CTO)

Stefan.vonderHeide@content-conversion.com

Telefon: +49 1579 2366592

info@content-conversion.com

www.cloutodo.de

Backup

Spezielle technische Herausforderung

- Um die Fehlerrate zur OCR-GT zu berechnen, wird Levenshtein-Distance genommen.
- Dafür gibt es gute, optimierte Algorithmen, die mit „linearer-Ordnung“ zurechtkommen, um die „Distance“ als Ergebnis zu berechnen.
- Wir wollten zur Analyse jedoch die „Editor-Methode“ (insert,delete,replace) mit als Ergebnis bekommen.
- Dafür gibt es (unseres Wissens nach) keine optimierten Methoden. Somit sind wir (mindestens) bei „quadratischer Ordnung“
- Für unsere durchaus langen Artikeltexte hat das den Großteil der (Rechen-)-Zeit (auf lokaler CPU) ausgemacht.
- Als Heuristik haben wir ein „Halbierungsverfahren“ angewendet.
  - Dazu haben wir im mittleren Textabschnitt eine Überschneidung gesucht, und angenommen, dass das wirklich der identische Text ist. Und dann den erweiterten Levenshtein auf beide Hälften angewendet.
- Kennt jemand dazu eine bessere Lösung?