Janusz S. Bień

Optyczne rozpoznawanie znaków w słowniku Lindego Koncepcja, narzędzia i realizacja (propozycja do dyskusji)

24.03.2014, …, 4.01.2015,26.05.2018

1. Program minimum

W związku z nieotrzymaniem grantu zadania na najbliższą przyszłość muszą być mocno ograniczone.

Może wykorzystać jakoś *Bogactwa mowy polskiej* (`http://www.osinski.ibi.uw.edu.pl`)?

Proponowałem poprzednio (8.04.2014):

1. Ulepszenie nowych skanów — usunięcie zbędnych marginesów, wyprostowanie, może coś jeszcze.
2. Konwersja na DjVu za pomocą didjvu. Warto najpierw wygenerować same maski i je obejrzeć w celu ustalenia, czy jest sens edytować je ręcznie.
3. OCR za pomocą ocrodjvu z wykorzystaniem parametrów Tesseracta właściwych dla danego fragmentu.
4. Zaadoptowanie konspektu i sekcji ze starych skanów.
5. Stworzenie dotychczasowymi metodami i udostępnienie korpusu bazującego na nowych wynikach OCR.

Proponowałem również

6. […] proponuję najpierw zrealizować te zadania tylko dla pierwszego tomu.
7. Równolegle można pracować nad udostępnieniem słownika w EPrints (`eprints.wbl.klf.uw.edu.pl`) w sposób nadający się do harwestowania.
8. Można rozważyć wykonanie eksperymentu z wykorzystaniem danych treningowych PCSS (`http://dl.psnc.pl/download/tesseract_traineddata.zip`).

Przedstawiony program minimum wymaga jednak dalszej minimalizacji oraz zmiany kolejności zadań.

Punkt 1 w zasadzie został wykonany, punkt 8 odsuwa się w nieokreśloną przyszłość. Pozostałe punkty są do realizacji w uproszczonej postaci.

2. Planowany wynik ewentualnego grantu lub projektu

Długoterminowa koncepcja od 8.04.2014 nie uległa zasadniczym zmianom, ale obecnie rozszerzyłbym ją o dwa elementy:

1. wskanowanie oryginałów obu wydań, por. np. `http://www.panscan.pl/#onas` (ScanRobot),

2. rozważenie alternatywnych form udostępnianie tekstowej postaci słownika, por. GoldenDict (http://goldendict.org/) i czeski system http://sourceforge.net/projects/dict-system/ prezentowany w kilku publikacjach.

Podstawowym celem jest nadal otrzymanie słownika w formacie DjVu z wysokiej jakości tekstem ukrytym, wtórnym w stosunku do bogatszego w informacje tekstu w hOCR. Choć nadal najważniejsza jest poprawność tekstu polskiego, obecnie wydaje się, że droga to tego będzie prowadzić przez rozpoznanie innych języków występujących w słowniku, przede wszystkim niemieckiego.

Dokument powinien mieć też konspekt (ang. *outline*), powinien mieć też odnotacje zawierające np. rozwinięcia skrótów.

Słownik w tym formacie powinien być udostępniony w Bibliotece Dygitalizacyjnej KLF.

Dodatkowo słownik powinien być udostępniony za pomocą Poliqarp dla DjVu, a utworzony na jego podstawie korpus powinien posiadać odpowiednie sekcje. Wskazane byłoby otagowanie skrótów ich rozwinięciami. Pożądane byłoby zachowanie informacji o fontach, jeśli będziemy nią dysponować.

Do słownika powinien zostać stworzony indeks haseł obsługiwany przez djview4poliqarp bazujący na wstępnej wersji indeksu dla tzw. starych skanów (https://bitbucket.org/jsbien/ilindecsv/).

Do rozważenia jest sporządzenie również innego rodzaju indeksów, np. obcojęzycznych odpowiedników haseł — zwiększyłoby to zainteresowanie projektem za granicą.

Poniższe założenia nadal wydają się słuszne, ale mało realne:

> Jeśli Poliqarp 2 zostanie udostępniony dostatecznie wcześnie, warto eksperymentalnie udostępnić część lub całość słownika również za pomocą tego oprogramowania.
>
> Głównym programistycznym wynikiem projektu powinna być istotnie nowa wersja djview4poliqarp. Inne narzędzia stworzone na potrzeby projektu też powinne być udostępnione na licencji GNU GPL.

3. Formaty danych

Podstawowym formatem dla wyników OCR jest hOCR produkowany przez tesseract i ocrodjvu.

Pomocniczo można korzystać z posiadanego przez Katedrę programu FineReader, wtedy dodatkowym formatem pośrednim byłby PDF i PDF/A. Więcej informacji zawiera PDF/A, ale program do jego obsługi (https://bitbucket.org/tomek87/pdfautils) dawno nie był przez nikogo używany i może wymagać zmian. Teoretycznie jest możliwość korzystania również z innych formatów produkowanych przez FineReader, ale w obecnych warunkach ydaje się to nieopłacalne i nierealne.

4. Megastruktura słownika

Ze względu na odmienne metadane wskazane jest przechowywanie poszczególnych tomów jako odrębnych dokumentów w Bibliotece Dygitalizacyjnej KLF. Pożądane jest udostępnienie całego słownika jako publikacji zbiorowej, ale Federacja Bibliotek Cyfrowych i Europeana i tak obsługują tylko poszczególne tomy, więc sprawa jest niskopriorytetowa.

Megastruktura ma również znaczenie praktyczne dla organizacji prac — poszczególne części niehasłowe muszą być traktowane indywidualnie.

5. Przygotowanie skanów do OCR

Pracujemy na tzw. nowych skanach słownika (wykonanych jakiś czas temu na „kaudzie” za pomocą scanhelper).

5.1. Kadrowanie

Przygotowanie skanów składa się z dwóch etapów. Pierwszy to zlikwidowanie zbędnych marginesów i ewentualne prostowanie (likwidowanie przekosu — jest nieznaczny, ale widoczny na niektórych stronach). Praktycznie jedynym stosownym narzędziem jest Scan Tailor — zadanie to już zrealizowała Joanna Bilińska z pewną pomocą Mateusza Sarneckiego.

Teoretycznie możliwe było również użycie unpaper z interfejsem Chimosza (`https://bitbucket.org/jsbien/unpaper-gui-fork`), ale byłoby to mocno ryzykowne ze względu na brak jakiegokolwiek wsparcia ze strony autorów.

Byłoby najlepiej, gdyby rozmiar w pikselach wszystkich stron był identyczny — czy udało się to osiągnąć?

5.2. Binaryzacja

Etap drugi to binaryzacja, która może być zrobiona na wiele sposobów dając różne rezultaty i w konsekwencji wpływając na wyniki OCR.

1. Scan Tailor; metoda ta nie nadaje się do bezpośredniego użycia, wyniki programy stanowiłyby tylko maskę dla programu didjvu — wydaje się, że taka komplikacja nie jest warta ewentualnych korzysci.
2. didjvu Jakuba Wilka wykorzystujący m.in. bibliotekę Gamera — zostały wykonane testy na 50 stronach słownika, a ich wyniki udostępnione pod adresem `http://teksty.klf.uw.edu.pl/6/`.
   — abutaleb — lepsza od wyników FineReadera,
   — bernsten — wyniki nieakceptowalne wzrokowo, czcionki — zwłaszcza duże — stawały się obwiedniowe.
   — djvu (domyślna metoda) — dla stron tytułówych maska zdecydowanie gorsza od abutaleb, na stronach hasłowych szare marginesy — niejasny wpływ na OCR.
   — niblack — maska bardzo czysta, duże litery stron tytuowych szarawe, ale binaryzacja tekstu wydaje się lepsza od abutaleb, por. *sedavit* na ostatniej stronie.
   — otsu — maska bardziej kontrastowa niż niblack, dukt grubszy, co czasami powoduje „zalanie”, a czasami zachowuje ciągłość znaków, które niblack rozdziela (np. ogonki).
   — sauvola — wydaje się minimalnie lepsze od niblack, zwłaszcza w przypadku ogonków; maska wakatów czystsza od otsu.
   — shading-subtraction — wszystkie litery obwiedniowe, co jest nieakceptowalne.
   — tsai — zalania podobne do niblack i otsu.
   — white-rohrer — maska z szarym tłem, wydaje się niakceptowalna.
3. konwersja na DjVu za pomocą programu FineReader — bardzo niska kompresja, maska z lukami już na pierwszej stronie, ostatnia strona „zapylona”, zdecydowanie ustępuje abutaleb.
4. Konwersja na DjVu za pomocą programu Document Express Desktop — prawdopodobnie najlepsza jakość, ale program powolny i niestabilny.

Najlepszą metodą wydaje się więc sauvola, ale niblack, otsu i tsai są bardzo zbliżone.

5.3. Retusz

Jednym z elementów przygotowywania skanów może być „czyszczenie” (ang. *despeckling*). Może być wykonywane jednak na różne sposoby również na dalszych etapach, np. tworzenia masek (por. pkt 12). Roboczo zakładamy, że nie ma to jest aż tak istotnego znaczenia dla jakości OCR, i rezygnujemy z tej operacji.

Na którymś etapie chyba warto wyretuszować pieczątki na stronach, np. w tomie 5 na stronach 764 i 765.

6. Ocena wyników OCR

Kwestia ta wymaga ponownego przemyślenia. Niektóre prezentowane niżej propozycje nie wydają się obecnie celowe.

Do porównywania wyników np. różnych metod binaryzacji potrzebne jest wykorzystanie odpowiednich narzędzi. Być może zadowalające okaże się ocrevalUAtion (https://github.com/impactcentre/ocrevalUAtion). Jego wyniki mogą być dodatkowo weryfikowane za pomocą hocr-tools (https://bitbucket.org/jsbien/hocr-tools-tesseract-fix-fork).

Podstawowym zadaniem tych narzędzi jest porównywanie wyników OCR z tekstem wzorcowym (ang. *Ground-Truth*). Wydaje się wskazane ręczne stworzenie takiego tekstu. Prawdopodobnie najwygodniej to zrobić bazując na fragmentach pierwszego wydania dostępnego w WikiŹródłach. (http://pl.wikisource.org/wiki/Indeks:Samuel_Bogumi%C5%82_Linde_-_S%C5%82ownik_j%C4%99zyka_polskiego).

Uwspółcześniania pisowni można dokonać automatycznie za pomocą narzędzia https://bitbucket.org/jsbien/pol.

Można też rozważać automatyczne porównywanie wyników OCR wykonanych na różne sposoby i wybieranie wersji najlepszej.

Pierwszy krok w tym kierunku został już wykonany przez opracowanie skryptu hocr-merge dostępnego w repozytorium https://bitbucket.org/jwilk/marasca-wbl w gałęzi wbl. Program ten obsługuje tylko wyniki programu Tesseract wykonywanego z różnymi parametrami, ale jest raczej prymitywny i być może w tej formie nie nadaje się jeszcze do użytku.

Nie jest jasne, czy byłoby możliwe i celowe stworzenie programu, który w ten sposób porównywałby wyniki programów FineReader 11 (Desktop) i Tesseract. Potencjalnym problemem, występującym również w przypadku dwóch przebiegów Tesseracta, jest odmienna segmentacja na słowa — być może porównania należałoby dokonywać na poziomie znaków, co może być kłopotliwe.

Wydaje się, że podobne automatyczne wybieranie najlepszego wyniku było stosowane w projekcie http://heml.mta.ca/lace — nie ma to dla nas praktycznego znaczenia, ale zasługuje na wzmiankę przy publicznej prezentacji projektu.

7. Rozpoznanie struktury strony

Ten punkt pozostaje nadal aktualny.

Ten etap ma kluczowe znaczenie dla dalszych prac, ale jego realizacja może być kłopotliwa.

Dla tzw. starych skanów zadawalające wyniki uzyskał Olejniczak (http://bc.klf.uw.edu.pl/223/, s. 51 i następne), który działał na wynikach programu FineReader 11; opracowane przez niego narzędzia dostępne są w prywatnym repozytorium https://bitbucket.org/tomek87/skrypty oraz w publicznym repozytorium https://bitbucket.org/tomek87/pdfautils; w repozytorium https://bitbucket.org/wybczu/pdfautils znajdują się nieprzetestowane poprawki. Dokumentacja jest szczątkowa, ale są dostępne przykładowe wyniki.

Do wyjaśnienia są następujące kwestie:

1. Czy narzędzia Olejniczaka w ogóle jeszcze działają w obecnym środowisku.
2. Czy dają się dostatecznie łatwo użyć i dostosować do obecnych celów (zmiany wymaga co najmniej format wyników).
3. Czy dają się przystosować do pracy — pośrednio lub bezpośrednio — na wynikach Tesseracta.

wykorzystując tylko ogólne idee Olejniczaka.

Ze względu na wagę tego etapu jego wyniki powinne być starannie zweryfikowane, zwłaszcza dla stron nietypowych (przede wszystkim dla stron z „tabelami rdzeni"). W pewnym stopniu może to być wykonane automatycznie i być może nawet jest już to robione przez narzędzia Olejniczaka. Nie należy jednak wykluczać weryfikacji ręcznej. Program do tego celu został nawet stworzony przez Olejniczaka, ale okazał się

niewygodny (nie pamiętam teraz jego lokalizacji, gdzieś mam zrzuty ekranu — nawet jeśli nie ma on praktycznego znaczenia, zasługuje na wzmiankę w opisie projektu).

Z informatycznego punktu widzenia stworzenie takiego narzędzia nie wydaje się trudne, ale jego specyfikacja powinna być dobrze przemyślana. Otwarte pytanie, czy powinien to być odrębny program, czy też jedna z funkcji programu o ogólniejszym przeznaczeniu. W tym drugim przypadku może to być program djview4poliqarp (lub djview4) uzupełniony o funkcję pokazywania struktury strony na ekranie i przewijanie po kolumnach zamiast po stronach (por. https://sourceforge.net/p/djvu/discussion/103286/thread/77392e32/?limit=25#ca7d/ac15).

Być może warto na wszelki wypadek zrealizować postulat https://bitbucket.org/mrudolf/djview-poliqarp/issue/28/editing-sidecar-hocr-file-s (dostęp ograniczony):

> The most needed function is adjusting (or creating) the borders of ocr_careas (content areas, in particular columns) which in the general case can be isothetic polygons. Examples:
> 1. An ocr_table (highlighted), a rectangular ocr_area (left column), an isothetic ocr_area (right column), running heads and page number:
> http://eprints.wbl.klf.uw.edu.pl/44/1/iLinde1FR11.djvu?djvuopts=&page=page0184.djvu&zoom=78&showposition=0.49,0.49&highlight=2635,4186,455,555
> 2. Two isothetic ocr_areas (left and right columns), additional problem posed by the common parts - the easiest way out seem to interpret the page as overlapping columns (using ocr_linear is probably also possible):
> http://eprints.wbl.klf.uw.edu.pl/44/1/iLinde1FR11.djvu?djvuopts=&page=page0082.djvu
> 3. A slightly simpler example:
> http://eprints.wbl.klf.uw.edu.pl/44/1/iLinde1FR11.djvu?djvuopts=&page=page0081.djvu
>
> Komentarz:
>
> Termin *isothetic* jest używany w literaturze dygitalizacyjnej, a przynajmniej w dokumentacji jednego z programów używanych w projekcie IMPACT.
> Według Wikipedii: *An isothetic polygon is a polygon whose alternate sides belong to two parametric families of straight lines which are pencils of lines with centers at two points (possibly in the infinity). The most well-known example of isothetic polygons are rectilinear polygons, and the former term is commonly used as a synonym for the latter one*.
> W praktyce chodzi więc o wielobok o kątach prostych.

Rozpoznanie struktury strony pozwala stworzyć indeks haseł głównych oraz wydzielić żywą paginę, kustosze i śródtytuły. W dalszych etapach należy wykorzystać redundancję zawartą w tych elementach, a indeks skonfrontować z indeksem stworzonym na podstawie danych przygotowanych za pomocą narzędzi Olejniczaka.

8. Analiza i korekta struktury stron

Niezbędne wydaje się wsadowe uzyskanie podstawowych statystyk na temat struktury stron.

Prawdopodobnie w tym samy przebiegu można przygotować dane do rekonstrukcji wyrazów niespójnych.

Można rozważyć tymczasową konwersję struktury strony na adnotacje i edytować je na jeden z kilku dostępnych sposobów:

— djvusmooth
— Emacs (http://elpa.gnu.org/packages/djvu.html)
— WebDjVuTextEd (http://sourceforge.net/projects/webdjvutexted/)

9. Rozpoznanie kursywy i korekta wyników

9.1. Rozpoznanie kursywy

Kluczowe znaczenie dla projektu ma poprawne rozpoznanie wszystkich skrótów — rozpoznanie kursywy wydawało się dobrą drogą do tego, ale nie jest to obecnie oczywiste.

Z dotychczasowych eksperymentów wynika, że FineReader rozpoznaje kursywę dość dobrze, a Tesseract praktycznie w ogóle.

Rozwiązaniem najbardziej eleganckim byłoby wytrenowanie Tesseracta do rozpoznawania kursywy jako dodatkowego „języka". Zadanie nie wymaga wiedzy informatycznej, ale niezbędne jest zrozumienie dość skąpej dokumentacji z pomocą m.in. archiwum listy dyskusyjnej użytkowników programu i przeprowadzenie pewnej liczby eksperymentów. Jest dużo różnych narzędzi do trenowania, niektóre z nich dostępne są również dla MS Windows — np. autor programu VietOCR (`http://vietocr.sourceforge.net/`) bardzo go zachwala, może rzeczywiście jest dobry. Warunkiem sukcesu jest biegłość w obsłudze komputera i podejście do zadania „z sercem".

Selekcja kursywy do trenowania i testowania Tesseracta teoretycznie może się odbywać na kilka sposobów:

1. djview4poliqarp Chimosza (`https://bitbucket.org/chimi/djview-fork-klf-uw`, `https://bitbucket.org/chimi/djvulibre-fork-klf-uw`), dostępny na demonstracyjnej maszynie wirtualnej z KMD, ale niestabilny,
2. djview4shapes Rudolfa — przestał działać pod nowszymi wersjami Linuksa, działa na demonstracyjnej maszynie wirtualnej (`hg-robocze/Linde4tesseract/`),
3. programy Sikory (`/mnt/MyBookT2/z_MyBookT1/MyBookT1/new-backup/galicja/hdd5-sarge-home/jsbien/Neof/`, kłopotliwe w instalacji, zaporowo wolne w działaniu)
4. korpus *font-sensitive* — jedna wersja niedostępna z powodu awarii serwera, druga w formie maszyny wirtualnej z usterkami, zasadniczym problem jest jednak redukcja rozdzielczości,
5. inne narzędzia, np. Gamera, PaRADIIT.

Preferowany jest program Rudolfa, zwłaszcza jeśli błędy zostaną usunięte szybko i będą dodane nowe funkcje. Jednak już w obecnej wersji na maszynie wirtualnej jest on używalny, należy jednak chyba działać na przetworzonej wersji skanów ze względu na minimalne — ale może istotne dla jakości OCR — „przekosy".

Tak czy inaczej warto w ten czy inny sposób wykorzystać FineReadera, jeśli będzie to możliwe technicznie,

Dla słów pisanych kursywą należy utworzyć listę frekwencyjną i ją ręcznie zweryfikować, oddzielając skróty od np. słów łacińskich. Istniejący wykaz skrótów można wykorzystać jako słownik do OCR.

9.2. Korekta wyników

Pierwotna koncepcja była następująca:

> Otwartą kwestią jest metoda nanoszenia poprawek. Ewidentne systematyczne błędy OCR można korygować globalnym podstawieniem na tekście ukrytym. W przypadkach wątpliwych można chyba wykorzystać nieznacznie zmodyfikowaną funkcję tworzenia indeksów w programie djview4poliqarp. Indeks zawierający poprawki musiałby być potem przetworzony na odpowiednie podstawienia w tekście ukrytym — dla informatyka zadanie to powinno być bardzo łatwe.

Obecnie uważam za optymalną metodę użycie zmodyfikowanego programu djview4poliqarp — specyfikacja modyfikacji w przygotowaniu.

Można też rozważać użycie PoCoTo`https://github.com/thorstenv/PoCoTo`.

10. Rozpoznawanie wersalików i korekta wyników

Wersalikami pisane są hasła — ze względu na udostępnienie indeksu haseł (`https://bitbucket.org/jsbien/ilindecsv`) zadanie to ma obecnie wysoki priorytet. Wymaga ono jednak albo uruchomienia narzędzi Olejniczaka albo stworzenie nowych o podobnej funkcji.

Należy w szczególności uwzględnić przenoszenie wyrazów hasłowych do nowego wiersza, nowej kolumny lub nowej strony.

11. Wstępne rozwarstwienie językowe haseł

Wydzielenie tego etapu i nadanie mu wysokiego priorytetu do zmiana w stosunku do poprzedniej koncepcji.

Chodzi o wydzielenie — dzięki informacji uzyskanej za pomocą programu tesseract — tekstu niemieckiego pisanego fakturą, który nie jest w żadny inny sposób wydzielony formalnie.

Dla tak wydzielonego tekstu należy sporządzić listę frekwencyjną i zweryfikować ją za pomocą dostępnych programów (można konsultować się w ENeL i IMPACT), co pozwoli wyeliminować błędne rozpoznanie innych krojów pism jako fraktury.

Jak się wydaje, fraktura jest błędnie traktowana jako łacinka w przypadku krótkich słów niezawierających charakterystycznych dla fraktury liter. Być może słowa te dadzą się łatwo zauważyć na liście frekwencyjnej. Wskazane będzie poprawienie w jakiś sposób tych błędów.

12. Podstawowe rozwarstwienie językowe haseł

Ten etap nie tylko jest również kluczowy dla projektu, ale charakteryzuje się oryginalnością podejścia, co należałoby wykorzystać do przygotowania publikacji dla renomowanego czasopisma.

Dysponując wiarygodnym OCR skrótów językowych, oraz wiarygodnym podziałem strony na łamy, dzięki pracy doktorskiej J. Bilińskiej można rozpoznać w hasłach fragmenty obcojęzyczne. Powstaje oczywiście pytanie, co robić z rozpoznanymi fragmentami.

Dokument w formacie DjVu składa się z czterech warstw, oprócz tekstu ukrytego mamy tło, zadruk i maskę. Binarna maska jest niewidoczna dla użytkownika, ale to ona właśnie stanowi dane wejściowe dla ocrodjvu, a w konsekwencji dla Tesseracta. Zmieniając domyślną maskę na taką, która zawiera tylko tekst w wybranym języku, wyznaczamy tzw. ROI (ang. *region of interest*) dla OCR, przy czym użytkownik nadal widzi w całości oryginalny tekst (ale można za pomocą specjalnych adnotacji podświetlić lub w inny sposób wyróżnić ROI).

Dysponując tak rozwarstwionym słownikiem możemy wykonywać OCR dla każdego języka osobno.

Ta prosta koncepcyjnie operacja w praktyce będzie skomplikowana i żmudna ze względu na konieczność operowania dużą liczbą plików.

Pierwotne maski można tworzyć na kilka sposób, najprostszy — choć niekoniecznie najlepszy — to użycie odpowiedniej funkcji programu didjvu. Do tworzenia masek „jednojęzykowych” potrzebny jest specjalny program — zadanie dla informatyka, ale dość łatwe. Złożenie nowych masek z obrazem graficznym można wykonać również za pomocą didjvu.

Etap rozwarstwiania jest dobrym momentem na stworzenie listy wyrazów przenoszonych do nowego wiersza do wykorzystania w dalszych etapach.

Ze względu na dużą liczbę plików i powtarzalność operacji wydzielania języków wskazane jest stworzenie jakiś przemyślanych narzędzi wspomagających — zadanie dla informatyka, przy dobrej specyfikacji raczej łatwe.

Całą operację dobrze jest wcześniej przetestować na dwujęzycznych tekstach wstępnych, por pkt. 16.

13. Wyodrębnienie i korekta polszczyzny w hasłach

Zmiana koncepcji, por. pkt. 11!. Poniższe informacje są jednak nadal aktualne.

Na tym etapie wykorzystujemy fakt, że Tesseract całkiem dobrze odróżnia font gotycki od innych krojów pisma. Wykorzystując wyniki programu rozwarstwiamy tekst na tekst gotycki i pozostałość, prawdopodobnie w całości polskojęzyczną.

Szczegóły są do ustalenia po wykonaniu eksperymentów i ich ocenie. Nie jest jasne, czy lepiej na początku podawać parametr deu-frak+pol (niemiecki gotykiem i polski — kolejność wydaje się istotna), czy pol+de-frak, czy też wykonać osobne przebiegi dla deu-frak oraz pol i następnie skomasować je za pomocą hocr-merge.

Tekst polskojęzyczny weryfikujemy najpierw na podstawie listy frekwencyjnej, potem — w razie potrzeby — kontekstowo.

Gdyby liczba błędów była bardzo duża, można rozważyć wytrenowanie Tesseracta specjalnie do polskojęzycznego tekstu słownika, nie jestem jednak w stanie ocenić stopnia trudności i opłacalności tego zadania.

Wydaje się, że podobnie jak w przypadku korekty kursywy, poprawki można nanosić za pomocą djview4poliqarp.

Teoretycznie możliwe, ale chyba nieopłacalne, jest użycie djvusmooth (tylko dla Linuksa) lub jakiegoś narzędzia pomocniczego Tesseracta.

Niektóre narzędzia do korekty zrealizowane w ramach projektu IMPACT są sensowne, ale zniechęcają zamkniętymi licencjami, brakiem dokumentacji i bazowaniem na formatach innych niż hOCR, mogą być jednak wykorzystane jako inspiracja. Jedno z takich narzędzi jest dostępne tutaj: `https://github.com/impactcentre/PoCoTo`.

14. Korekta niemieckiego w hasłach

Zmiana koncepcji, por. pkt. 11!.

15. OCR i korekta innych języków w hasłach

Obecnie uważam, że priorytetem powinno być wydzielenie niełacińskich alfabetów.

Warto wykonać OCR dla wszystkich języków, dla których to będzie łatwe z technicznego punktu widzenia. Korekta wyników byłaby wykonywana tylko w miarę możliwości — tj. kompetencji zespołu i możliwości czasowych.

Wcześniejsze stanowisko

> Wydaje się, że na tym etapie warto wykorzystać FineReadera do OCR języków nie obsługiwanych przez Tesseracta.

prawdopodobnie wymaga modyfikacji.

Jeśli nanoszenie korekty za pomocą djview4poliqap okaże się dostatecznie wygodne, ten etap może być wykonywany z pomocą wolontariuszy i/lub studentów.

16. Niehasłowe fragmenty słownika

Jak było wspomniane, muszą one być traktowane indywidualnie. Nie jest jasne, czy opłaca się korzystać z fragmentów pierwszego wydania dostępnych w WikiŹródłach — chyba nie.

Nietypowy podział na kolumny może być uwzględniony przez ręczną edycję masek.

Można się obawiać, że wielojęzyczne *Prawidła etymologii* będę zawierać wiele błędów OCR, a metoda rozwarstwiania na języki w tym wypadku jest nieopłacalna — pozostaje chyba tylko ręczna edycja tekstów. Wybór narzędzi jest taki sam, jak w przypadku 9.2.

17. Komasowanie wyników

Uzyskanie zbiorczego tekstu ukrytego na podstawie zweryfikowanych tekstów wielojęzycznych wymaga specjalnego programu — dość łatwe zadanie dla informatyka.

Specjalnego programu będzie wymagać też naniesienie adnotacji — djview4poliqarp może być rozszerzony o ich obsługę.

18. Konwersja na korpus

Istotnym elementem nowej koncepcji jest zrobienie wspólnego korpusu dla starych i nowych skanów.

18.1. Metadane

— author
— editors
— title (inny dla całości, inny dla tomów)
— edition
— place
— year
— version (1 — stare skany, 2 — nowe)

18.2. Tagi

Pole form hasłowych wykorzystujemy na rekonstrukcję wyrazów podzielonych, por. 18.4.

Poniższa propozycja jest nieznaczną modyfikacją propozycji Wilka z mejla z 29 kwietnia 2013 r. (skróty języków zgodne z BCP 47), ograniczoną do potrzeb pierwszego etapu pracy:

— `orig`: źródło tekstu, por. koncepcję pseudohomonimów (pkt. 18.5). Przykładowe propozycje wartości: `FR10pol` (stare skany), `tess3.3deu-frak+pol` (nowe skany), `indeks` (wyraz potwierdzony przez indeks *a tergo*) itp.
— `lang`: `&#xFFFD;` (REPLACEMENT CHARACTER, U+FFFD: oznaczenie nierozpoznanego języka), `de` (niemiecki); dla interpunkcji i skrótów należy przemyśleć, jaka wartość byłaby optymalna.
— `script`: `&#xFFFD;` (REPLACEMENT CHARACTER, U+FFFD: oznaczenie nierozpoznanego pisma), `latf` (łacinka fraktura).
— `series`: aktualnie tylko `&#xFFFD;`, później także `medium` i `bold`.
— `shape`: aktualnie tylko `&#xFFFD;`, później także `upright` i `italic`.
— `size`: aktualnie tylko `&#xFFFD;`.
— `wconf` (*word confidence*: `0`, …, `9` (wyliczone na podstawie wartości dostarczanych przez program tesseract, oznaczające odpowiednie wartości z przedziału 0–0,1 itd.), `&#xFFFD;` (brak informacji o ufności), ✓ (CHECK MARK, U+2713: rozpoznanie powierdzone w jakiś dodatkowy sposób). Przykład użycia: `wconf='[5-9]'`.

Konwersja na korpus będzie wymagać specjalnego programu — chyba dość łatwe zadanie dla informatyka. można się wzorować na istniejących programach.

18.3. Segmentacja na tokeny

Do rozważenia jest przypisanie każdemu tokenowi identyfikatora: numer tomu, numer strony, numer kolumny, numer wiersza, numer we wierszu. Dane te mogą zostać zapisane w formie arkusza kalkulacyjnego lub tabeli w bazie danych.

18.4. Wyrazy niespójne

Chodzi o wyrazy przenoszone do nowego wiersza, nowej kolumny lub strony.

Pole formy hasłowej pierwszego członu wyrazu niespójnego zawiera całą zrekonstruowaną formę, dla pozostałych członów pola te są puste (wydaje się to technicznie możliwe).

W Poliqarp 2 ten problem ma być rozwiązywalny elegancko, ale przystosowanie djview4poliqarp do współpracy z Poliqarp 2 może być nietrywialne.

18.5. Pseudohomonimy

Ograniczenie do tylko dwóch pól tekstowych w całostce można obejść wprowadzając całości będące z technicznego punktu widzenia homonimami. Jest to na pewno możliwe w programie Poliqarp, należy sprawdzić — chyba doświadczalnie — czy jest to obsługiwane również przez marasca. Dalej zakładamy odpowiedź pozytywną.

19. Uwagi końcowe

Priorytem jest obecnie wybór jednej lub kilku metod binaryzacji i wykonanie OCR za pomocą programu tesseract.

Można uniknąć podejmowania decyzji stosując wszystkie metody jednocześnie — czasochłonne, ale proste koncepcyjnie.

Nie jest niestety jasne, czy OCR za pomocą programu FineReader jest wart zachodu.