Janusz S. Bień

Wstępna koncepcja wielopoziomowej dygitalizacji obiektowej (object multilayer digitization) na przykładzie słownika Lindego

29 stycznia 2015
(uaktualnienie adresów 26.05.2018)

1. Wstęp

Punktem wyjścia do dygitalizacji obiektowej jest szczegółowa **analiza typograficzna słownika**.

Jak można zauważyć np. na egzemplarzu z Biblioteki Łańcuckiej udostępnianym przez Google Books, w oryginale wszystkie tomy mają przedtytuł (*Schmutztitel*, `http://de.wikipedia.org/wiki/Schmutztitel`; *bastard title*, `http://en.wikipedia.org/wiki/Half_title`) treści SŁOWNIK LINDEGO. Jest on pomijany w przedrukach, co zaburza paginację — zakładamy uzupełnienie skanów o tę brakującą kartę.

W przedruku GUTENBERG-PRINT tom VI został opublikowany jako dwa woluminy z dorobionymi pseudo-oryginalnymi tytulariami. Zgodnie z oryginałem tom ten należy traktowac jako całość i uzupełnić o oryginalne karty tytułowe.

Tradycyjne wyniki dygitalizacji mają postać tekstu uzupełnionego o odpowiednie adnotacje (HTML, TEI itp.). Wynikiem dygitalizacji obiektowej jest zasób elektroniczny w formie bazy danych opisujący własności wyróżnionych w tekście obiektów. Zasób taki może mieć różnorodne zastosowanie, niekoniecznie przewidziane przez jego twórców; oczywistym i podstawowym zastosowaniem jest automatyczne wygenerowanie dygitalizacji tekstowej w zadanym formacie. Jest wskazane nadanie zasobowi International Standard Language Resource Number (`http://www.islrn.org/faq/`, usługa jest bezpłatna).

Zakładamy wykorzystanie relacyjnej bazy danych — wykorzystamy doświadczenia zdobyte przy bazodanowej dygitalizacji indeksu *a tergo* (Bień, 2014).

Omawiane dalej poziomy to pewne wygodne sposoby prezentacji jej zawartości, które mogą ulegać zmianom. Na granicy poziomów występują często wieloznaczności, co traktujemy jako rzecz normalną, wymagajacą traktowania w analogiczny sposób, jak wieloznaczności morfologiczne i składniowe znane od lat lingwistyce informatycznej.

Zakładamy intensywne wykorzystanie systemu zarządzania wersjami (mercurial) oraz możliwości logowania transakcji w bazie danych do pełnego dokumentowania procesu tworzenia i uaktualniania zawartości bazy.

Na różnych etapach dygitalizacji niezbędna będzie ręczna korekta danych, ale nie będzie to po prostu typowa korekta tekstu w trakcie jego lektury. Za pomocą istniejących oraz stworzonych na potrzeby projektu narzędzi weryfikowane będą indywidualnie wybrane aspekty dygitalizacji. Oczekujemy, że dzięki temu weryfikacja będzie szybsza, łatwiejsza i bardziej skuteczna.

Z technicznego punktu widzenia możliwe jest udostępnianie zasobu przez serwer baz danych (zapewne MySQL) dostępny zdalnie dla uprawnionych użytkowników.

2. Słownik Lindego

Uważam za celowe wskanowanie oryginałów obu wydań słowników (najlepiej egzemplarzy z BUW) w sposób gwarantujący najwyższą jakość, tzn. za pomocą ScanRobot dostępnego na zasadach komercyjnych w Pracowni Skanowania przy Muzeum i Instytucie Zoologii PAN (`http://panscan.pl`).

Aktualnie wykorzystywany reprint sprawia wrażenie opracowywanego komputerowo, co mogło wprowadzić jakieś zniekształcenia (przykładem sygnalizowane wcześniej pseudo-oryginalne strony tytułowe tomu szóstego).

Forma bazy danych pozwala w szczególności na równoległą analizę różnych aspektów słownika, np. wykaz cytatów z poszczególnych źródeł (por. `http://ebuw.uw.edu.pl/publication/339848`) czy indeksy słów dla poszczególnych języków — por. (Ptaszyk, 2007, s. 57).

Zawartość bazy danych może być prezentowana użytkownikowi na różne sposoby, np.

— warstwa tekstu ukrytego w dokumentach DjVu
— korpus DjVu
— korpus w formacie Poliqarp 2
— HTML z zachowaniem układu strony oryginału lub z jego modyfikacją

W trakcie przygotowywania takiej formy prezentacyjnej wskazane jest — jeśli format wyjściowy na to pozwala — scalenie erraty z tekstem głównym, wprowadzenia powiązań między oboma wydaniami słownika itp.

Na potrzeby interfejsu użytkownika może być potrzebne stworzenie odpowiedniego fontu.

3. Baza informacji typograficznych

Baza ta zawiera informacje wytworzone przez program rozpoznawania znaków. Podstawowe dane pobieramy z programu tesseract w formie plików hOCR generowanego przez ocrodjvu. Podobne dane można otrzymać z programu tesseract z łatą umożliwiającą wypisywanie wyniku w formacie TSV (por. `(http://tesseract-hocrtsv.herokuapp.com/`, `http://teksty.klf.uw.edu.pl/12/1/alice_1.png.hocr.tsv` `https://bitbucket.org/jsbien/linde-info/`); warto w miarę możliwości zachować zgodnośc z formatem TSV.

Zakładamy segmentację do poziomu pojedynczych znaków. Jest ona dostępna w plikach hOCR generowanych przez ocrodjvu, ale w niestandardowej formie zaprojektowanej *ad hoc* Przez Jakuba Wilka — znajduje się ona w komentarzu na końcu pliku (por. np. `http://teksty.klf.uw.edu.pl/20/3/hOCR1.zip`). W bardziej czytelnej postaci jest dostępna w plikach djvused generowanych przez ocrodjvu (por. np. `http://teksty.klf.uw.edu.pl/20/4/djvused1.zip`) — jeśli dobrze pamiętam, ocrodjvu dwukrotnie uruchamia tesseracta z różnymi parametrami i w ten sposób komasuje wyniki. Być może warto w jakiś sposób wykorzystać faktt dostępnosci danych w formacie djvused i kodu, który go generuje.

Dane mogą one być uzupełniane i konfrontowane z danymi z innych podobnych programów (przede wszystkim FineReader).

Punktem wyjścia jest zestaw własności przedstawiony niżej odnoszący się do pojedynczej strony.

Niektóre wartości odnoszą się tylko do niektórych typów elementów; dla pozostałych otrzymują one wartości puste lub ustalone umownie — np. własności fontu mogą być przypisane jednostkom większym niż znak, jeśli wszystkie elementy podrzędne mają takie same ich wartości.

Szczególny charakter ma pole `text`. W sposób pierwotny jego wartość jest wypełniana przez dane programu OCR, ale różne programy — a nawet różne przebiegi tego

samego programu — mogą dostarczać tych wartości dla elementów różnych poziomów. Jeśli dysponujemy tą informacją na poziomie znaków, to dodatkowo rekonstruujemy ją dla słów i ewentualnie wierszy.

1. version_id (umowny odsyłacz do metadanych danej wersji, informujących o pliku wejściowym, użytym programie do OCR itp.; choć będziemy dążyć do tego, żeby istniała tylko jedna wersja bazy, do tej utworzenia potrzebne będą wersje alternatywne).
2. page_id (umowny identyfikator strony — patrz niżej)
3. level (zagnieżdżenie elementu)
4. page_id (kolejny numer strony czyli elementu `div` z atrybutem `ocr_page`; informacja praktycznie chyba nieprzydatna, ale zachowana na wszelki wypadek);
5. block_num (kolejny numer bloku czyli elementu `div` z atrybutem `ocr_carea` na danej stronie — jak w TSV, w hOCR numery są globalne! 0 dla level = 1);
6. par_num (kolejny numer akapitu czyli elementu `p` z atrybutem `ocr_par` w danym bloku — jak w TSV, w hOCR numery są globalne! 0 dla level > 3);
7. line_num (kolejny numer wiersza czyli elementu `span` z atrybutem `ocr_line` w danym akapicie — jak w TSV, w hOCR numery są globalne! 0 dla level > 4);
8. word_num (kolejny numer słowa czyli elementu `span` z atrybutem `ocrx_word` w danym wierszu — jak w TSV, w hOCR numery są globalne! 0 dla level > 5);
9. char_num (kolejny numer znaku w słowie — jeśli segmentacja na znaki nie jest dostępna w hOCR, to jest utworzona na podstawie `ocrx_word`)
10. left (współrzędna gabarytu elementu, pobierana z sekcji bbox atrybutu `title` danego elementu; 0 tylko dla elementów, dla których hOCR nie zawiera ich gabarytów, czyli w praktyce znaki rozpoznane przez FineReader)
11. top (współrzędna gabarytu elementu, pobierana z sekcji bbox atrybutu `title` danego elementu; 0 tylko jak wyżej)
12. width (rozmiar gabarytu elementu, pobierany z sekcji bbox atrybutu `title` danego elementu; 0 tylko jak wyżej)
13. height (rozmiar gabarytu elementu, pobierany z sekcji bbox atrybutu `title` danego elementu; 0 tylko jak wyżej)
14. baseline (pole tekstowe, zawierające zawartość sekcji baseline atrybutu `title` elementu `span` klasy `ocr_line`). Według specyfikacji
    > baseline pn pn-1 … p0 - a polynomial describing the baseline of a line of text

    Niestety aktualnie znaczenie wartości podawanych przez tesseract nie jest jasne. Są one wypisywane przez funkcję `AddBaselineCoordsTohOCR` w `api/baseapi.cpp`; w ogólnym wypadku może wystąpić również sekcja `textangle`. Dla elementów innych niż `ocr_line` wartość może być wyliczana według jakiegoś algorytmu; 0 jeśli hOCR nie podaje tej informacji dla `ocr_line`.
15. conf (ufność rozpoznania, pobierana z sekcji x_conf atrybutu `title` danego elementu; wartość 0 jeśli hOCR nie zawiera tej informacji)
16. dir (kierunek pisma, pobierany z atrybutu `dir` danego elementu; z wyjątkiem nielicznych wstawek w alfabecie hebrajskim zawsze `ltr` czyli *left to right*). Wartość dziedziczona przez elementy podrzędne, oraz przez te elementy nadrzędne, dla których jest to jedyna wartość elementów podrzędnych; w przeciwnym razie 0. Wartość 0 również wtedy, gdy hOCR w ogóle nie zawiera tej informacji.
17. lang (język, pobierany z atrybutu `lang` danego elementu). Zasady dziedziczenia — jak wyżej.
18. font (informacja pobierana z sekcji x_font atrybutu `title` danego elementu — dostępna na razie tylko w wersji rozwojowej, ~~(por. `http://teksty.klf.uw.edu.pl/13/`~~). Zasady dziedziczenia — jak wyżej.
19. font_size (informacja o foncie, pobierana z sekcji x_fsize atrybutu `title` danego elementu — dostępna na razie tylko w wersji rozwojowej ~~(por. `http://teksty.klf.uw.edu.pl/13/`~~). Zasady dziedziczenia — jak wyżej.

20. font_style (eprezentowany przez element `strong`, element `em` lub brak tych elementów). Zasady dziedziczenia — jak wyżej.
21. text (rozpoznany tekst w UTF-8 z wykorzystaniem Private Use Area dla znaków, słów i wierszy — pobierany bezpośrednio z hOCR lub utworzony na podstawie elementów nadrzędnych lub podrzędnych).

Wydaje się wskazane kilkakrotne przetwarzanie tekstu z różnymi parametrami. Choć program tesseract może teoretycznie rozpoznawać teksty wielojęzyczne, to zwiększanie liczby rozpoznawanych języków pogarsza jakość rozpoznawania. W związku z tym planujemy zadawać programowi jednocześnie co najwyżej dwa języki do rozpoznania. W konsekwencji w bazie będziemy mieć kilka wersji danych typograficznych i do dalszego przetwarzania będziemy wybierać wyniki najbardziej prawdopodobne — wydaje się to wygodniejsze, niż wstępnie zaimplementowane przez Wilka komasowanie plików hOCR, które nie radziłoby sobie z rozbieżnosciami w gabarytach. Trochę podobną metodologię zastosował projekt LACE (`http://heml.mta.ca/lace`).

Dane typograficzne wymagają weryfikacji lub innej modyfikacji na każdym poziomie, a wyniki weryfikacji powinne być zapisywane — być może w osobnej powiązanej bazie. Poniżej wymienione są przykładowe możliwe testy.

1. nieuzasadnione zmiany linii pisma — prawdopodobnie wynik traktowania skaz papieru jako znaków lub inny błąd segmentacji.
2. level (zagnieżdżenie elementu) — zbyt duża wartość oznacza błąd rozpoznania.
3. block_num (kolejny numer bloku) — należy oczekiwać błędów w rozpoznaniu podziału strony na łamy, potrzebne są też pewne decyzje konwencjonalne (paginę i sygnatury proponuję zaliczać do odpowiedniego łamu). Należy pamiętać o tabelach radiks i akapitach wspólnych dla obu łamów. Nie wszystkie bloki są prostokątne. Niewłaściwa liczba bloków sygnalizuje błąd rozpoznania.

Strony identyfikujemy za pomocą globalnego numeru strony w tomie, z uzwględnieniem następujacych reguł:

— w numeracji pomijamy tytularia przedruku,
— w numeracji uwzgledniamy tytularia oryginału pominięte w przedruku,
— tom szósty, podzielony na dwa wolumeny w przedruku GUTENBERG-PRINT, traktujemy jako jedną całość.

Pomocniczo będziemy korzystać z alternatywnej identyfikacja stron i ich własności, które mogą wyglądać następująco:

1. Wydanie (reprint GUTENBERG-PRINT traktujemy jako wydanie czwarte — pole może być pomijane, dopóki pracujemy tylko na jednym wydaniu).
2. Tom (1–6).
3. Część tomu (1-2 dla tomu szóstego, dla pozostałych 0 — pole istotne tylko dla wydania czwartego).
4. Numer arkusza (ma znaczenie dla analizy układu strony — wartość początkowa wpisana ręcznie, potem uzupełniana i weryfikowana automatycznie).
5. Numer strony w arkuszu (pierwsza zawiera sygnaturę główną, trzecia sygnaturę pomocniczą).
6. Numer sekcji paginacyjnej (zostaną zdefiniowane w osobnym dokumencie).
7. Typ paginacji (arabska, rzymska, ślepa, mieszana itp. — zostaną zdefiniowane w osobnym dokumencie i wprowadzone ręcznie).
8. Numer strony w sekcji paginacyjnej.
9. Typ układu strony (np. hasłowa z tabelą, wakat - nietypowe wartości wpisane ręcznie, typowe uzupełnione i weryfikowane automatycznie).

4. Poziomy unilateralne

Obiektami najniższego poziomu są obrazy czcionek na skanach. Mogą one być niejednoznaczne z różnych powodów na jednym lub wielu egzemplarzach dzieła: skazy papieru, wady druku (za mało lub za dużo farby drukarskiej), defekt czcionki. W ogólnym

wypadku nieoczywista może też być segmentacja skanów na obrazy czcionek. Poziom ten nazywamy typograficznym (z segmentacją) lub graficznym (bez segmentacji).

Kolejnym poziomem są czcionki odtworzone na podstawie ich obrazów. Wierne ich odtworzenie nie jest niezbędne dla dalszych etapów dygitalizacji, ale może być przydatne np. do badania proweniencji dzieła. Czcionki mają różnorodne własności — wielkość, krój itp; szczególna własnością jest wykaz ich użycia w tekście z dokładną lokalizacją na stronie, zarówno względną (wiersz, numer), jak i bezwzględną (współrzędne w pikselach). Poziom ten nazywamy faksymilowym.

Następnym poziomem są znaki kodowe w sensie Unicode; niektórym czcionkom niezbędne jest arbitralne przypisanie kodów z obszaru użytku prywatnego. Znaki wykorzystujemy jako wierne reprezentacje czcionek — na przykład *vv* lub *rv* w funkcji *w* jest reprezentowane jako *vv* lub *rv*. Poziom ten nazywamy znakowym.

Wszystkie wymienione poziomy są unilateralne w sensie Saloniego, czyli w żaden sposób nie nawiązują do znaczenia odpowiednich napisów.

Do opisu tych poziomów mogą być zaadoptowane narzędzia NPT (Ekstrakcja i etykietowanie kształtów znaków — shape extraction and labelling, przeglądarka kształtów znaków — a shape browser).

Warto podkreślić, że aktualnie nie jest znany pełny repertuar występujacych w słowniku znaków. Najbardziej rzuca się w oczy nietypowy znak paragrafu (por. (Bilińska, 2013, s. 98)), w cytatach obcojęzycznych występuje długie s, można również spotkać np. wariant znaku ℞ (RESPONSE, U+211F). Kilka razy występuje również ⁊(TIRONIAN SIGN ET, U+204A) w kształcie *r rotunda*, por. `https://en.wikipedia.org/wiki/Tironian_notes`, `https://bitbucket.org/jwilk/ocrodjvu/issue/15/`. Sporadycznie występuje również znak ☞ (WHITE RIGHT POINTING INDEX, U+261E).

5. Segmentacja unilateralna

Segmentacja unilateralna to segmentacja typograficzna czyli podział obrazu strony na takie obszary, jak pagina, łamy, sygnatury, przypisy, tabele oraz poszczególne wiersze tekstu. Podział ten w zasadzie daje się przeprowadzić w sposób obiektywny, choć dostępne narzędzia nie pozwalają go uzyskać w sposób w pełni automatyczny.

Do segmentacji typograficznej zaliczamy też segmentację wierszy przez odstępy, która jednak nie jest kompletnie jednoznaczna — z powodu ciasnego składu nie zawsze można odróżnić odstępy międzywyrazowe od międzyliterowych. Dlatego segmentację na tradycyjne wyrazy zaliczamy do poziomów bilateralnych (patrz niżej).

Ważnym efektem ubocznym typograficznej segmentacji unilateralnej jest opracowanie intuicyjnej konwencji lokalizacji fragmentu tekstu w słowniku. **Wiersze są najmniejszą jednostką tekstu w słowniku, które daje się jednoznacznie zidentyfikować na podstawie wyglądu strony**, dlatego należy przydzielić im identyfikatory, np. postaci numeru tomu, numeru sekcji, numeru strony, identyfikatora bloku (np. łamu) i lokalizacji wiersza w bloku (numer w bloku lub numer akapitu i numer w akapicie).

6. Poziomy bilateralne

Pierwszym poziomem bilateralnym jest poziom teksteli (por. Bień 2011, w druku). Na tym poziomie bierzemy pod uwagę funkcję czcionek, więc *vv* lub *rv* w funkcji *w* będzie reprezentowane jako *w*, a częsty w słowniku Lindego znak ź (LATIN SMALL LETTER Z, COMBINING COMMA ABOVE) będzie reprezentowany przez *ż*.

W przypadku słownika Lindego występuje systematycznie problem częstego pomijania diakrytów w wersalikach, które nie zawsze mogą być zrekonstruowane jednoznacznie — por. (Bień, 2014). Na poziomie tekstelowym powinne one być w razie potrzeby odtworzone.

Na tym poziomie nadal zachowane są ewentualne ligatury.

7. Poziomy transkrypcyjne

Poziomy transkrypcyjne opisują obiekty powstałe z segmentacji na słowa strumienia teksteli uporządkowanych w kolejności czytania; oznacza to sklejanie wyrazów przenoszonych do nowej linii, kolumny i strony z pominięciem dywiza (niewykluczone są jednak przypadki, kiedy interpretacja dywizu nie jest jednoznaczna).

Jedną z naistotniejszych własności słowa jest język.

Do poziomów transkrypcyjnych zaliczamy w szczególności normalizację i modernizację pisowni.

Bibliografia

Bień, Janusz S. (2014). "Elektroniczny indeks do słownika Lindego". URL: https://www.slideshare.net/jsbien/elektroniczny-indeks-do-sownika-lindego.

Bilińska, Joanna A. (2013). "Analiza i leksykograficzny opis struktury słownika Lindego na potrzeby dygitalizacji". Prac. dokt. Uniwersytet Warszawski. URL: https://depotuw.ceon.pl/handle/item/349.

Ptaszyk, Marian (2007). Słownik języka polskiego Samuela Bogumiła Lindego. Toruń: Wydawnictwo Uniwersytetu Mikołaja Kopernika.