POLONICA XXXIV
PL ISSN 0137-9712

AGNIESZKA MYKOWIECKA
PIOTR RYCHLIK
JAKUB WASZCZUK

Definicja struktury oraz narzędzia wspomagające budowę słownika polszczyzny niewspółczesnej

1. Wstęp

W niniejszym artykule opisano prace nad elektronicznym słownikiem dawnej polszczyzny budowanym na podstawie istniejącego słownika tradycyjnego. Przedstawiono przyjęte zasady konwersji, strukturę opracowanego słownika, trudności, na jakie napotkano przy konwersji słownika na w pełni funkcjonalną wersję elektroniczną, stworzone narzędzia programistyczne oraz przetestowane metody automatycznego uzupełniania listy form. Opisywane prace wykonywane są w ramach projektu SYNAT, którego celem jest budowa narzędzi pozwalających na lepszy dostęp do polskich danych w sieci Internet. Jedno z podzadań tego projektu poświęcone jest zwiększeniu dostępności dla przeciętnego czytelnika tekstów dawnych znajdujących się między innymi w zasobach bibliotek cyfrowych. Teksty te zawierają wiele słów, które nie są już obecne we współczesnej polszczyźnie, a te słowa, które nadal są w użyciu, są często inaczej zapisane lub mają inne formy fleksyjne. Sprawia to, że przeciętnemu czytelnikowi do prawidłowego zrozumienia tekstu dawnego potrzebny jest słownik.

Z oryginalnych słowników dawnych najbardziej znany jest „Słownik języka polskiego" M. Samuela Bogumiła Lindego, 2. wydanie, Lwów 1854–1861 (OCR i wyszukiwarka elektroniczna udostępniona przez prof. Janusza Bienia http://poliqarp.wbl.klf.uw.edu.pl/ slownik-lindego/). Istniejące współczesne wydawnictwa książkowe to m.in.:

— S. Reczek, „Podręczny słownik dawnej polszczyzny", Wrocław 1968;

— „Słownik języka Jana Chryzostoma Paska", red. H. Koneczna i W. Doroszewski, t. 1–2, Wrocław 1965–1973;

— „Słownik staropolskich nazw osobowych", red. W. Taszycki, t. 1–6, Wrocław 1965–1983; t. 7: Suplement, red. M. Malec, Wrocław 1984–1987;

— „Słownik staropolski", red. S. Urbańczyk i in., t. I–XI, Kraków 1953–2003;

— „Mały słownik zaginionej polszczyzny", red. naukowy F. Wysocka, Kraków 2003;

— „Słownik polszczyzny XVI wieku", red. M.R. Mayenowa i in., t. I–XXX, Wrocław–Warszawa 1966–2002 (praca kontynuowana, http://www.spxvi.edu.pl/spxvi/);

— „Słownik polszczyzny Jana Kochanowskiego", red. M. Kucała, t. 1–4, Kraków 1995–2008;

— „Słownik języka polskiego XVII i 1. połowy XVIII wieku", Polska Akademia Nauk, Instytut Języka Polskiego, 1996– (praca nieukończona, http://sxvii.pl);

— „Słownik wyrazów zapomnianych", K. Holl, A. Żółtak, PWN, 2001;

Ze względu na okres, w którym powstawały, większość wymienionych słowników istnieje wyłącznie w formie drukowanej, a tylko niektóre są dostępne (przynajmniej częściowo) w formie elektronicznej. Przykładowo, istnieje tekstowa wersja elektroniczna wybranych tomów „Słownika polszczyzny XVI wieku", skany znajdują się w Kujawsko-Pomorskiej Bibliotece Cyfrowej udostępniającej ten zasób na licencji Creative Commons, dostępne są też na stronie http://poliqarp.wbl.klf.uw.edu.pl/slownik-polszczyzny-xvi-wieku/. Jedynie „Słownik języka polskiego XVII i 1. połowy XVIII wieku" od początku tworzony był w postaci elektronicznej.

Niestety dostępność słowników polszczyzny dawnej dla przeciętnego czytelnika jest bardzo ograniczona. Tradycyjne słowniki papierowe, nawet jeśli są udostępnione w wersji elektronicznej, to korzystanie z nich jest często dość utrudnione. Potencjalny użytkownik musi najpierw odszukać odpowiedni z nich, by potem moc znaleźć w nim, w mniej lub bardziej wygodny sposób, poszukiwane słowo. Z punktu widzenia mniej wyrobionego czytelnika bardzo wygodną funkcjonalnością byłoby przeszukiwanie wszystkich połączonych słowników, oczywiście z możliwością uzyskania informacji o źródle konkretnego wpisu. Niestety skomplikowane problemy związane z prawami autorskimi i wydawniczymi powodują, że wykorzystanie istniejących źródeł zarówno drukowanych, jak i elektronicznych i połączenie ich w jeden zasób jest trudne. Z tego względu zdecydowano się rozpocząć prace od przygotowania infrastruktury informatycznej pozwalającej na obsługę takiego słownika oraz na eksperyment pilotażowy polegający na przekształceniu jednego z tradycyjnych słowników papierowych na słownik elektroniczny o opracowanej strukturze. Jako podstawowe źródło informacji do budowanego słownika postanowiono wykorzystać „Podręczny słownik dawnej polszczyzny" Stefana Reczka (Reczek 1968). Słownik ten jest niezbyt obszerny i zawiera hasła uznane za autora za najczęściej spotykane słowa dawne, które nie są już używane w języku współczesnym lub zmieniły znaczenie. Dla tego wybranego przykładowego zasobu opracowany został zestaw programów pozwalających na transformację słownika przeznaczonego do czytania w wersji papierowej na zasób leksykalny dostępny do wyszukiwania w wersji elektronicznej.

Przedstawiony proces rozpoczyna się od opisanej w następnym rozdziale zamiany tekstu tradycyjnego słownika na formę elektroniczną. W rozdziale 3. opisana jest opracowana struktura słownika skonstruowana w oparciu o metaformat LMF (Lexical

Markup Framework, http://www.lexicalmarkupframework.org, Francopoulo i in. 2009, Francopoulo 2013). W kolejnym rozdziale przedstawione są szczegóły konwersji zawartości wybranego słownika na ten format uznawany obecnie za standard reprezentacji różnego typu słowników. Rozdział 6. poświęcony jest binarnej reprezentacji słownika historycznego, opracowanej pod kątem jego wykorzystania przez inne programy analizujące teksty dawne, na przykład program oznaczający formy w tekście informacjami słownikowymi. W rozdziale tym opisana jest również biblioteka automatów umożliwiająca efektywne przechowywanie słowników morfologicznych (i nie tylko). W dalszej części opisane zostały dwie metody automatycznego uzupełniania słownika dawnej polszczyzny. Celem pierwszej jest dodanie informacji o częściach mowy do poszczególnych haseł słownika, natomiast drugiej — uzupełnienie słownika o formy występujące w tekstach historycznych. Na koniec przedstawiono prace nad klastrowaniem zbioru słów na leksemy. Proces klastrowania słów może posłużyć np. do automatycznego skonstruowania słownika na podstawie nieprzetworzonego wcześniej zbioru tekstów, dlatego jest on szczególnie interesujący w kontekście języków o małej liczbie zasobów.

2. Przetwarzanie wstępne

Pierwszym krokiem przy tworzeniu elektronicznej wersji słownika było zeskanowanie jego wersji książkowej. Otrzymane w ten sposób obrazy przetworzono standardowym programem do rozpoznawania znaków OCR (Optical Character Recognition), w wyniku czego utworzony został plik w formacie Microsoft Word. Zastosowanie standardowego oprogramowania OCR było możliwe, ponieważ w książkowej wersji słownika użyto współczesnego kroju czcionki. Następnym krokiem było wyodrębnienie z tak powstałego pliku wszystkich haseł słownika wraz z opisami i zapisanie ich w postaci tekstowej. Identyfikacja haseł w wynikach OCR nie była łatwa, czego można było się spodziewać analizując inne doświadczenia z oprogramowaniem OCR (np. Reynaert 2008). W wersji książkowej hasła oraz ich opisy umieszczone są w jednym paragrafie, przy czym definiowane hasło, jak również podhasła wyróżnione są pogrubioną czcionką. Niestety, nie najlepsza jakość skanów sprawiła, że program OCR często nie rozpoznawał prawidłowo pogrubienia czcionki. Spowodowało to w niektórych przypadkach sklejenie kilku haseł w jedno lub podzielenie tekstu dotyczącego jednego hasła. Pomocną informacją, która w wielu przypadkach pozwoliła na prawidłową identyfikację haseł, była informacja dotycząca głębokości wcięcia wiersza tekstu. Każde rozpoznane hasło wraz z podhasłami, ich definicjami i przykładami użycia zostało zapisane w jednej linii tekstu w pliku tekstowym. Dodatkowo hasła zostały odseparowane od siebie pustymi liniami.

Tak utworzony plik wymagał ręcznego poprawienia z uwagi na wspomnianą już wyżej złą jakość skanów. Ponadto sam program OCR, przystosowany do współczesnego języka polskiego, zastępował niekiedy nierozpoznane sekwencje znaków innymi, występującymi w jego słowniku. Najczęstszymi poprawkami, które musiały być wy-

konane ręcznie, było wstawianie lub usuwanie znaków diakrytycznych, wstawianie znaków „«” i „»”, odpowiednio na początku i końcu definicji (zgodnie z konwencją przyjętą w słowniku) oraz usuwanie znaków nieistniejących w tekście, a powstałych w wyniku interpretacji przez program OCR różnego rodzaju zabrudzeń i skaz papieru jako ciągu znaków.

Wynikowy plik zawierał 18 820 haseł.

3. Struktura słownika

Kolejnym krokiem prowadzącym do powstania elektronicznego słownika dawnej polszczyzny było opracowanie jego reprezentacji przy użyciu metaformatu LMF (Lexical Markup Framework), obecnie obowiązującego standardu opisującego metody projektowania i konstruowania słowników elektronicznych. LMF został opracowany na podstawie licznych, istniejących już słowników i stanowi próbę zbudowania metamodelu pozwalającego na usystematyzowany opis struktury elektronicznych słowników różnego rodzaju, np. słowników morfologicznych, semantycznych, jedno- i dwujęzycznych. Zasadniczym celem wykorzystywania technik opracowanych w ramach LMF ma być uproszczenie procesów projektowania, tworzenia oraz rozszerzania zasobów leksykalnych. Pierwsze podjęte decyzje opisane zostały w: Mykowiecka i in. 2011.

W kontekście metaformatu LMF zaprojektowany słownik najbliższy jest klasie słowników o nazwie „Machine-readable dictionary” (MRD). Słowniki MRD mogą być zarówno jednojęzyczne, jak i wielojęzyczne. W przypadku języka historycznego mamy do czynienia z połączeniem słownika jednojęzycznego z dwujęzycznym, gdyż dla niektórych wyrazów lub wyrażeń podane są odpowiedniki w polskim języku współczesnym.

Konstruowany przez nas słownik jest plikiem XML (Extended Markup Language) o następującej strukturze:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<LexicalResource dtdVersion=”16”>
  <GlobalInformation>
    <feat att=”languageCoding” val=”ISO 639-6” />
  </GlobalInformation>
  <Lexicon>
    <feat att=”language” val=”polh” />
    <LexicalEntry id=”lex.1”>
    </LexicalEntry>...
    <LexicalEntry id=”lex.n”>
    </LexicalEntry>
  </Lexicon>
</LexicalResource>
```

Główną częścią tej struktury jest leksykon (<Lexicon>) stanowiący zbiór haseł leksykalnych (<LexicalEntry>), z których każde hasło reprezentuje jeden leksem

języka historycznego (polh). Każdy element <LexicalEntry> posiada atrybut o nazwie id określający unikalny identyfikator hasła w leksykonie. Każde hasło reprezentowane przez element <LexicalEntry> opatrzone jest komentarzem zawierającym numer linii w wejściowym pliku tekstowym, w której dane hasło zostało umieszczone. Dodatkowo w komentarzu może się znajdować słowo VERIFY, ale tylko w tych przypadkach, w których konieczna jest weryfikacja poprawności translacji.

Z każdym hasłem związana jest jedna forma podstawowa (<Lemma>) oraz zbiór form wyrazowych (<WordForm>). Z każdą formą związana jest jej reprezentacja w języku pisanym (<FormRepresentation>). W szczególności, jeśli dane słowo posiada kilka wariantów ortograficznych, będą one przedstawione w osobnych instancjach klasy <FormRepresentation>. Rozwiązanie to umożliwia także zapis różnych reprezentacji jednej formy przy użyciu kilku rodzajów pisma, np. formy zapisanej pismem gotyckim oraz formy transliterowanej.

Leksem może posiadać wiele znaczeń. Każde z nich jest reprezentowane przez klasę <Sense>. Znaczenie leksemu może być określone za pomocą definicji (<Definition>) (wraz z uzupełnieniem w polu <Statement>) lub poprzez podanie przykładowych kontekstów użycia słowa (klasa <Context>). W każdym z tych przypadków do opisu znaczenia leksemu służy klasa <TextRepresentation>. W przypadku haseł wielowyrazowych listę wyrazów tworzących takie hasło umieszcza się w strukturze <ListOfComponents> wchodzącej w skład elementu <LexicalEntry>. Częstym przypadkiem jest odwołanie w opisie jednego hasła do opisu innego. Odwołanie takie reprezentowane jest przez element <RelatedForm>.

O ile ekwiwalenty i definicje zapisane są w języku współczesnym, przykłady użycia są zwykle podane przy użyciu języka historycznego.

Cechą charakterystyczną słownika historycznego jest waga informacji na temat pochodzenia danego słowa (w którym tekście źródłowym słowo wystąpiło). Informacja ta pozwala np. wnioskować o tym, w jakim okresie historycznym dane słowo było używane w języku pisanym. Informacja o pochodzeniu słowa jest reprezentowana przez atrybut sourceID klasy <Representation>. Dzięki temu informacja o tekście źródłowym jest zachowana zarówno dla każdej formy <Form> hasła słownikowego, jak również dla każdego kontekstu <Context>, w którego ramach hasło występuje.

Struktura słownika dawnej polszczyzny obejmuje również kilka klas umożliwiających dołączenie informacji dotyczącej składni (<SyntacticBehaviour>, <SubcategorizationFrame> i <SyntacticArgument>) oraz semantyki (<SenseRelation>). Ewentualne dodatkowe informacje o charakterze składniowym lub semantycznym będzie można później dołączyć do słownika poprzez dodanie do modelu słownika odpowiednich klas powyższych rozszerzeń, co dzięki przyjęciu metaformatu LMF nie powinno prowadzić do kolizji z umieszczonymi już w słowniku danymi. Obecnie jedynymi informacjami składniowymi są nieliczne przykłady rekcji zapisywane jako elementy klasy <SyntacticBahaviour>.

Przykładowe hasła słownika podane będą w następnym rozdziale poświęconym opracowanym zasadom konwersji.

4. Konwersja wersji tekstowej do formatu LMF

Konwersja wersji tekstowej na wersję w formacie LMF przeprowadzana jest w kilku etapach. W pierwszym etapie następuje klasyfikacja haseł do odrębnych klas. Opisy haseł należących do jednej klasy mają podobną strukturę. W drugim kroku następuje właściwa konwersja zgodnie z określonymi dla danej klasy haseł regułami. Po przetworzeniu tekstu całego słownika następuje faza trzecia, w której ustalane są identyfikatory tych haseł, do których istnieje odwołanie w tekście innych haseł. Usuwane są wtedy redundantne wpisy, jak również tworzona jest lista haseł niezidentyfikowanych, do których jest odwołanie w tekście, a które nie znalazły się w słowniku.

Kolejną fazą jest automatyczne uzupełnienie każdego wpisu w słowniku o brakujące informacje morfologiczne. Wersja książkowa na ogół nie posiada takich informacji. Nie jest określona część mowy opisywanego hasła, a odmienione formy są przedstawione tylko w sposób skrótowy, poprzez podanie samych końcówek i to tylko w tych przypadkach, w których odmiana jest nietypowa lub różni się od tej spotykanej we współczesnej polszczyźnie.

Należy zaznaczyć, że w książkowej wersji słownika autorowi nie do końca udało się utrzymać jednolity sposób opisu wszystkich haseł, brakuje też sprecyzowania *explicite* przyjętej konwencji zapisu informacji. W wielu przypadkach z pozoru podobnie wyglądające zapisy mają różne znaczenia, co nieuchronnie prowadzi do błędnej translacji.

4.1. Format pliku wejściowego

Około 82% wszystkich haseł w przetwarzanym słowniku ma następujący wzorzec:

```
{słowo} «definicja»: cytowania.
```

lub

```
{słowo} 1. «definicja-1»: cytowania-1;
        2. «definicja-2»: cytowania-2; ...
        n. «definicja-n»: cytowania-n.
```

Zaraz po definiowanym słowie ujętym w nawiasy klamrowe następuje jeden lub więcej ponumerowanych bloków. Każdy z tych bloków zaczyna się definicją otoczoną nawiasami kątowymi „«” i „»”, po której występuje dwukropek, a następnie przykłady użycia definiowanego wyrazu rozdzielone średnikami. Prawie każdy przykład zawiera na samym końcu informację na temat źródła pochodzenia. Ilustruje to poniższy przykład.

(1)
```
{abdankować} 1. «odprawić, rozpuścić (o wojsku, służbie)»:
    Wojsk nie chciał z siebie abdankować L XVII;
    2. «rezygnować»: Chciwość hamować i rzeczom cudzym
    abdankować L XVII.
```

Pozostałych 18% haseł nie można dopasować do powyższego wzorca. Ogółem wyróżniono kilkanaście wzorców haseł. Dla każdego z nich opracowana została osobna metoda translacji na format LMF.

4.2. Reguły translacji

Przykładowe hasło (1) należące do najliczniejszej klasy haseł w słowniku ma następującą reprezentację w języku LMF.

```
<LexicalEntry id=”lex.1”>
  <!– INDEX 1 –>
  <Lemma>
    <FormRepresentation>
      <feat att=”writtenForm” val=”abdankować” />
      <feat att=”language” val=”polh” />
      <feat att=”sourceID” val=”srpsdp:L XVII” />
    </FormRepresentation>
  </Lemma>
  <Sense>
    <Definition>
      <TextRepresentation>
        <feat att=”writtenForm”
            val=”odprawić, rozpuścić (o wojsku, służbie)” />
        <feat att=”language” val=”pol” />
        <feat att=”sourceID” val=”srpsdp” />
      </TextRepresentation>
    </Definition>
    <Context>
      <TextRepresentation>
        <feat att=”writtenForm“
            val=”Wojsk nie chciał z siebie abdankować” />
        <feat att=”language” val=”polh” />
        <feat att=”sourceID” val=”srpsdp:L XVII” />
      </TextRepresentation>
    </Context>
  </Sense>
  <Sense>
    <Definition>
      <TextRepresentation>
        <feat att=”writtenForm” val=”rezygnować” />
        <feat att=”language” val=”pol” />
        <feat att=”sourceID” val=”srpsdp” />
      </TextRepresentation>
```

```
    </Definition>
    <Context>
      <TextRepresentation>
        <feat att="writtenForm"
          val="Chciwość hamować i rzeczom cudzym abdankować" />
        <feat att="language" val="polh" />
        <feat att="sourceID" val="srpsdp:L XVII" />
      </TextRepresentation>
    </Context>
  </Sense>
</LexicalEntry>
```

W zapisie tym widać sposób reprezentowania dwóch znaczeń hasła. Każde z nich ma podaną definicję będącą tekstem w języku współczesnym oraz przykład użycia zacytowany za autorem słownika (identyfikator źródła poprzedzony skrótem `srpspd`).

W przypadku wielu haseł w wersji tekstowej słownika nagłówek ujęty w nawiasy klamrowe nie jest pojedynczym wyrazem.

(2) `{zableszczyć oczy} ...`
(3) `{liść, wolności} ...`
(4) `{zakrzywić: zakrzywić palec} ...`
(5) `{zalec; zalec w grobie} ...`
(6) `{ekskuzować się} ...`
(7) `{chycić, ~ się} ...`
(8) `{gibać (się)} ...`
(9) `{obiecadło, obiecado, abiecadło} ...`
(10) `{is(t)ność} ...`
(11) `{chlebojejca, -e} ...`
(12) `{fach, -u, facha, -y} ...`
(13) `{zakon, -a || -u} ...`
(14) `{iść, ~ w co} ...`
(15) `{przychodzić o co} ...`
(16) `{rozumieć (za co)} ...`

Przykłady (2)–(5) opisują hasła wielowyrazowe. Odnośniki do poszczególnych słów hasła wielowyrazowego są zapisane w postaci listy `<ListOfComponents>`. Szczególnymi przypadkami haseł wielowyrazowych słownika są czasowniki zwrotne (6)–(8), dla których nie jest tworzona struktura `<ListOfComponents>`. W przypadku (6) partykuła „się” traktowana jest jako nieodłączna składowa leksemu. W przypadkach (7) i (8) hasło przekształcane jest na dwie struktury `<LexicalEntry>` — z partykułą i bez partykuły „się”.

Nagłówek hasła może zawierać warianty pisowni, co ilustrują przykłady (9), (10) i (12). Poszczególne warianty umieszczane są w osobnych podstrukturach `<FormRepresentation>` struktury `<Lemma>`.

Na ogół hasła nie zawierają żadnej informacji morfologicznej. Wyjątkowo podawane są końcówki dopełniacza liczby pojedynczej lub mianownika liczby mnogiej (11)–(13). W takich przypadkach program translujący dołącza strukturę <WordForm> opisującą formę odmienioną i zaznacza, że tak powstała struktura wymaga weryfikacji.

W nagłówkach haseł mogą znajdować się przykłady rekcji (14)–(16). Rekcja opisywana jest strukturą <SyntacticBehaviour>.

Dodatkowe informacje zapisywane są niekiedy zaraz za nagłówkiem, przed definicją. Mogą to być informacje różnego rodzaju, np. dotyczące rekcji (17) lub stylu (18).

(17) {trzymać} (o kim, czym) «cenić, sądzić, mniemać»: ...
(18) {bigos} (mtf) «zamieszanie»: ...

Aż 5% haseł ma w swoim opisie odwołanie do innych haseł w słowniku (19).

(19) {birzwnowanie} zob. bierzwnowanie

Oprócz haseł prostych, z pojedynczym nagłówkiem, w słowniku obecne są także hasła złożone, które składają się z kilku bloków o tej samej strukturze co hasła proste.

(20) {plac} «miejsce»: ...; {~ dać} «wstąpić»: ...;
{~ otrzymać} «zwyciężyć, otrzymać pierwszeństwo»: ...
(21) {borgować} 1. «kredytować»: ...;
2. {~ komu} «przepuszczać, pobłażać komu»: ...
(22) {chlebojedźca} «familiant, domownik, podopieczny»: ...;
{chlebojedziec}: ...

W przykładzie (20) występują dwa podhasła opisujące związki frazeologiczne ze słowem z pierwszego nagłówka. Związki te umieszczone są w oddzielnych strukturach <LexicalEntry>. W przykładzie (21) mamy do czynienia z przykładem rekcji. Podhasło może również definiować alternatywną pisownię hasła głównego (22).

Hasła, których nie można było dopasować do żadnego z wyróżnionych wzorców, zostały oznaczone jako nieprzetworzone i wymagały ręcznego wstawienia do słownika.

5. Automatyczne uzupełnianie słownika

Słownik Reczka, jak każdy tradycyjny słownik papierowy, dla większości słów zawiera tylko ich formy podstawowe. Jednak w przypadku odmienionej formy nieznanego słowa dla wielu czytelników starych tekstów poprawne odgadnięcie formy podstawowej może nie być łatwe. Przydatność słownika do automatycznej analizy tekstu przez programy komputerowe w jeszcze większym stopniu zależy od pełności zawartych w nim informacji. Z tego względu chcieliśmy, by budowany przez nas słownik zawierał formy odmienione słów. Niestety automatyczna odmiana form dawnych nie byłaby możliwa, gdyż brak jest wiarygodnych reguł pozwalających na uzyskanie takich form. Najlepszym potwierdzeniem, że jakaś forma fleksyjna była faktycznie używana, jest znalezienie odpowiedniego przykładu w tekście pochodzącym z danego okresu czasu.

Zaproponowanym przez nas rozwiązaniem problemu pozyskiwania do słownika form fleksyjnych jest wyszukiwanie w tekstach form podobnych (ortograficznie) do danego leksemu i proponowanie ich wraz z zawierającym je zdaniem jako potencjalnych form fleksyjnych do późniejszej weryfikacji przez lingwistę.

Informacją słownikową, która jest bardzo istotna przy automatycznej analizie tekstów, jest kategoria gramatyczna, do której należy dane słowo. Uzupełnienie zawartości słownika o te dane okazało się możliwe dzięki informacjom uzyskanym z analizy współczesnych odpowiedników słów dawnych przy wykorzystaniu współczesnego analizatora morfologicznego.

W niniejszym rozdziale opisujemy opracowane w trakcie trwania projektu metody automatycznego uzupełniania słownika o części mowy oraz formy podobne. Przypisywanie części mowy odbywało się w oparciu o znaczniki morfoskładniowe współczesnych odpowiedników przyporządkowanych poszczególnym hasłom słownika. Proces ten opisany jest w podrozdziale 5.1, a także w pracy: Mykowiecka i in. 2012. Dla celów uzupełniania słownika formami podobnymi do tych już w słowniku występujących opracowana została niezależna biblioteka wyszukiwania przybliżonego. Sposób wykorzystania tej biblioteki w procesie uzupełnienia słownika historycznego formami występującymi w analizowanym zbiorze tekstów historycznych opisany jest w podrozdziale 5.2.

5.1. Uzupełnianie informacji o części mowy

W pierwotnej wersji słownika nie było informacji o częściach mowy poszczególnych leksemów, ale za to dla niemal każdego leksemu określona była lista słów-odpowiedników w języku współczesnym. Przyjęta została (potwierdzona później empirycznie) hipoteza, że istnieje silna zależność pomiędzy częściami mowy leksemów słownika historycznego a częściami mowy przypisanych im ekwiwalentów. Na tej podstawie opracowany został program, który wykorzystywał odpowiedniki do określania zbiorów potencjalnych części mowy poszczególnych leksemów, z których wybierana była następnie najczęstsza propozycja i przypisywana odpowiednim hasłom słownika historycznego. Takie postępowanie, w którym wybierana jest najczęstsza odpowiedź spośród różnych wyników uzyskiwanych przez zastosowanie różnych algorytmów bądź pochodzących z analizy różnych danych, jest obecnie dość popularne i nazywane wyborem przez głosowanie.

Dla większości haseł słownika dawnej polszczyzny zbiór odpowiedników współczesnych był jednoelementowy. Nie było to jednak regułą, w związku z czym opracowana metoda musiała działać również wtedy, gdy zbiór odpowiedników zawierał więcej niż jedną formę, a w niektórych przypadkach również frazy. W związku z tym przyjęliśmy następujący dwuetapowy proces przypisywania części mowy:

- dla każdego elementu zbioru odpowiedników określane są przypisane mu przez współczesny analizator morfologiczny (lub za pośrednictwem reguł dla fraz, o czym niżej) części mowy,

- wystąpienia poszczególnych części mowy zliczane są względem całego zbioru współczesnych odpowiedników i wybierane są te, które wystąpiły największą liczbę razy.

Każda część mowy określona w pierwszym kroku powyższego procesu otrzymuje jeden punkt (głos), który jest następnie brany pod uwagę podczas zliczania w kroku drugim, za wyjątkiem części mowy przypisanych odpowiednikom niestanowiącym form podstawowych — taki przypadek sugeruje, że podany odpowiednik odnosi się do innego leksemu języka współczesnego, dlatego odpowiadająca mu część mowy otrzymuje jedynie pół głosu.

Poniżej pokazane są przykłady haseł ze słownika dawnej polszczyzny i wyniki użycia w ich kontekście opisanej metody głosowania. W pierwszych dwóch przykładach w wyniku analizy odpowiedników otrzymujemy trzy wartości części mowy (POS), z których dwie są równe, w związku z czym stanowią wynik głosowania. W trzecim przykładzie oba tagi POS występują po jednym razie i oba zostają przypisane hasłu „fuzą".

Czwarty przykład obrazuje częsty przypadek niejednoznaczności pomiędzy rzeczownikiem a rzeczownikiem odczasownikowym, przypadek który jest trudny do rozwiązania również dla ludzi.

- abecedariusz «początkujący [pact, subst], nowicjusz [subst]» → [subst]
- dojutraszek «kunktator [subst], maruda [subst], jednodniowy [adj]» → [subst]
- fuzą «szybko [adv], migiem [subst]» → [adv, subst]
- dufanie «ufność [ger, subst]» → [ger, subst]

Dodatkowe informacje na temat części mowy można uzyskać z analizy treści hasła słownika historycznego. W sytuacjach, gdy znaczenie słowa uległo zmianie, lecz jego forma pozostała niezmieniona (o czym świadczy sam fakt rozpoznania hasła przez współczesny analizator), wynik analizy morfologicznej hasła dodawany jest do wyników analizy zbioru odpowiedników i dopiero na tak uzyskanym ważonym wielozbiorze części mowy odbywa się głosowanie. Taka strategia jest szczególnie skuteczna dla haseł występujących jako elementy innych haseł wielowyrazowych i, w związku z tym, nieposiadających w słowniku Reczka swoich własnych odpowiedników współczesnych.

Dla celów określenia części mowy ekwiwalentów wykorzystany został program Morfeusz (Woliński 2006, http://sgjp.pl/morfeusz). W związku z tym przygotowana została również biblioteka umożliwiająca korzystanie z Morfeusza z poziomu programów napisanych w języku Haskell (http://hackage.haskell.org/ package/morfeusz).

Odpowiedniki wielowyrazowe parsowane były w oparciu o następującą sekwencję reguł:

- ~ [pos=inf] [orth=„się"] ⇒ inf
- ~ [pos=pact] [orth=„się"] ⇒ subst
- ~ [pos=pact] [orth!=„się"] ⇒ pact
- ~ [orth="o"] [pos=adj] ⇒ pact
- [pos=inf] ⇒ inf
- [pos=subst, case=nom] ⇒ subst
- [pos=ger, case=nom] ⇒ ger

Składnia reguł pozwala określić, że dana forma ma znajdować się na początku frazy (znak ~) lub że może być położona w dowolnym miejscu (domyślna interpretacja). Po lewej stronie reguły (na lewo od ⇒) znajduje się sekwencja warunków, z których każdy dotyczy pojedynczego słowa frazy. Warunek elementarny może dotyczyć formy danego słowa (orth) lub jego części mowy (pos). Po prawej stronie poszczególnych reguł podana jest część mowy, która zostanie uwzględniona w ramach głosowania na zasadach opisanych powyżej.

Zilustrujmy wyniki zastosowania podanych wyżej reguł względem dwóch haseł, dla których podany jest odpowiednik wielowyrazowy. W pierwszym przypadku piąta reguła (rozpoznająca bezokolicznikową formę czasownika) zostaje dopasowana, natomiast w drugim znaleziona jest mianownikowa forma rzeczownika.

- setkować «karać co setnego» ⇒ inf
- cetno «liczba parzysta» ⇒ subst

W tabelach 1 i 2 prezentujemy wyniki opisanego wyżej procesu przypisywania części mowy. W momencie wyliczania statystyk słownik składał się z 18 820 haseł. W tabeli 1. znajdują się informacje, ile było haseł, którym w wyniku głosowania przypisana została określona liczba części mowy. W pierwszej linii zawierającej znak „–" znajduje się liczba haseł, które stanowią jedynie odnośnik do innych haseł i w związku z tym nie są tutaj rozpatrywane (np. odyć zob. odejdź). Około 5% haseł słownika Reczka stanowią odnośniki. Mniej niż 1% haseł nie otrzymało żadnej etykiety POS, do 92,5% haseł przypisana została jedna etykieta, a dla pozostałych 5,2% haseł powyższa metoda zwróciła więcej niż jedną etykietę. Tabela 2. zawiera liczby wystąpień najczęściej przypisywanych części mowy.

Implementację opisanego wyżej procesu przypisywania części mowy można znaleźć pod adresem https://github.com/kawu/hist-pl/tree/master/poser. Wszystkie hasła słownika zostały przejrzane przez lingwistę, który oceniał poprawność przypisań części mowy i korygował błędne oznaczenia.

Tabela 1. Statystyki przypisywania etykiet POS — liczba przypisanych etykiet

liczba etykiet POS	liczba haseł
–	605
0	174
1	17236
2	888
3	78
4	11

Tabela 2. Statystyki przypisywania etykiet POS – rozkład kategorii gramatycznych

kategoria gramatyczna	etykieta	liczba haseł
rzeczownik	subst	8828
czasownik	verb	4718
przymiotnik	adj	3051
przysłówek	adv	868
odsłownik	ger	522
imiesłów przymiotnikowy	ppas	229
imiesłów przysłówkowy	pact	141
partykuła	part	146
spójnik podrzędny	comp	62
spójnik współrzędny	conj	46

Spośród ogólnej liczby haseł wstępna automatyczna weryfikacja wskazała jako wątpliwe 1441 (5,7%) haseł, dla których uzyskano więcej niż jedną lub nie uzyskano żadnej propozycji części mowy lub których struktura wewnętrzna nie była zgodna z opracowanymi schematami. Dla tych najtrudniejszych do automatycznego przetworzenia haseł dokonano 548 zmian w określeniu części mowy, w tym 174 przypisania dla form, które nie miały podanej żadnej propozycji. Najczęstsze z dokonanych zmian przedstawione są w tabeli 3. W przypadku, gdy hasło miało przypisaną więcej niż jedną część mowy, na 110 przypadków tylko w 18 pośród kandydatów nie było prawidłowej odpowiedzi.

Tabela 3. Statystyki błędów przypisywania etykiet POS

etykiety		liczba zmian	przykład
poprawna	przypisane		
adj	subst	18	eburowy
adv	subst	28	gwoli
adj	–	39	dziewiątodny
adv	–	24	gdziebykoli
inf	–	23	nasprawować
subst	–	53	mordasz

adj	adj pact adj subst	63	gęgliwy jedorny
comp	comp qub	15	eżby
ger	subst ger	15	nagnajanie
subst	adj subst	26	łaziebnik

5.2. Dodawanie form podobnych

W celu uzupełniania słownika formami wyrazowymi wykorzystane zostały trzy zasoby:

- konteksty użycia podane w słowniku Reczka, a więc po konwersji dostępne w samym słowniku LMF,
- teksty zebrane w ramach projektu IMPACT (http://www.impact-project.eu/),
- zbiór tekstów zebranych w trakcie projektu przez współpracujących lingwistów.

Proces wstępnego przetwarzania powyższych zasobów uwzględniał prostą segmentację na słowa, w której ramach znaki interpunkcyjne były ignorowane. Teksty z projektu IMPACT zostały uprzednio transliterowane do pisowni współczesnej (zob. rozdział 7). Każdy segment wynikowy był traktowany w następnej fazie przetwarzania niezależnie.

Zebrane teksty posłużyły jako źródło form odmienionych, nieobecnych w budowanym słowniku. W celu zidentyfikowania haseł słownikowych odpowiadających formom występującym w danych wejściowych wykorzystaliśmy metodę bazującą na wyszukiwaniu przybliżonym, gdzie podobieństwo między dwiema formami określane jest na podstawie tzw. uogólnionej odległości edycyjnej. Dla każdej formy tekstu wejściowego określana była najbardziej podobna forma występująca już w słowniku, jak również odpowiadające jej hasło leksykalne i, o ile odległość edycyjna pomiędzy wyszukiwaną formą oraz tą znalezioną była niższa niż zadana z góry wartość progowa, znalezione hasło wzbogacane było o nową formę odmienioną.

Oczywiście proces ten może wprowadzić do słownika błędy, dlatego każda dodana w ten sposób forma oznaczana była w słowniku elektronicznym dodatkowym atrybutem określającym jej pochodzenie i świadczącym o jej niepewnym statusie, co ułatwiło dalsze prace nad poprawianiem zawartości słownika. Warto również zauważyć, że metoda ta jest tym skuteczniejsza, im więcej form odmienionych znajduje się już w słowniku, ponieważ łatwiej wtedy o znalezienie hasła odpowiadającego wyszukiwanej formie.

Odległość pomiędzy dwoma słowami definiuje się, tak jak w przypadku tradycyjnej odległości edycyjnej, jako minimalną sumę kosztów operacji potrzebnych do przekształcenia jednego słowa w drugie, przy czym uwzględniane są trzy rodzaje operacji elementarnych — zamiana, dodanie oraz usunięcie znaku. W uogólnionej odległości edycyjnej koszt tych operacji (nieujemna liczba rzeczywista) może zależeć od znaków, które biorą w niej udział, jak również od pozycji, na której dana operacja ma miejsce.

Definiując odległość edycyjną dla języka historycznego oparliśmy się na przesłance wywodzącej się z analizy języka współczesnego, zgodnie z którą formy odmienione zadanego leksemu częściej różnią się na końcowych pozycjach niż na początkowych, co uwzględniliśmy na poziomie odległości edycyjnej poprzez nadanie operacjom edycji występującym na początkowych pozycjach słowa wyższego kosztu niż analogicznym operacjom odbywającym się na jego końcowych pozycjach.

W przeprowadzonych eksperymentach koszty poszczególnych operacji edycji zdefiniowane były następująco:

- koszt dodania znaku jest równy 1,
- koszt usunięcia znaku jest równy 1,
- koszt zamiany znaku *a* na znak *b* wynosi:
  - 0, jeśli *a* i *b* są identyczne,
  - 0,5, jeśli oba znaki różnią się jedynie wielkością,
  - `subst(a,b)` w pozostałych przypadkach, przy czym to funkcja grupująca podobne do siebie znaki (np. „e" i „ę" albo „l" i „ł") i przypisująca im wartości pomiędzy 0,5 a 1.
- Na koniec koszt każdej z powyższych operacji jest przemnażany przez współczynnik zależny od pozycji, na której dana operacja ma miejsce. Współczynnik ten, mający wartości w przedziale [0,1], maleje wraz z rosnącymi wartościami pozycji operacji.

Zapisaną w sposób formalny definicję wykorzystanej odległości edycyjnej można również znaleźć pod adresem https://github.com/kawu/hist-pl/blob/master/similar/src/NLP/HistPL/ Similar/Cost.hs.

Poniższa lista obrazuje wyniki wyszukiwania przybliżonego przykładowych słów względem aktualnej wersji słownika historycznego oraz zdefiniowanej wyżej miary podobieństwa. Na prawo od → pokazane są najbardziej podobne słowa znalezione w słowniku oraz odległości pomiędzy poszczególnymi (wyszukiwanymi i znalezionymi) słowami.

- abdankowała → abdankowali, 0,333
- abdankowaliśmy → abdankowali, 0,75
- fuza → fuzą, 0,167
- setkowali → setkować, 0,6

Metody wykorzystujące tak zdefiniowaną odległość edycyjną zostały wykorzystane przy próbie automatycznego pozyskania form fleksyjnych. Analiza przykładów umieszczonych w poszczególnych hasłach pozwoliła na zidentyfikowanie 14 574 form fleksyjnych. Przykładowo, do hasła *żołdat* dodane zostały cztery formy: *żołdaci*, *żołdata*, *żołdatem*, *żołdatów*, a do hasła *błądzić* formy: *błądzicie*, *błądzą*, *błądząc*, *błędzi*, *błędzili*, *błędzić*, *błądził*. Najczęściej w przykładach rozpoznawano jednak tylko jedną formę fleksyjną hasła, na przykład dla *błąkanina* formę *błąkanin*. Prawie wszystkie zaproponowane formy były faktycznie fleksyjnymi formami odpowiedniego hasła. Jednymi z bardzo nielicznych przykładów negatywnych są forma *blachy* zaproponowana w haśle *blachowni* oraz forma *kmiecku* pochodząca z frazy *jakoby po*

kmiecku mówim zaproponowana obok poprawnej formy *kmiećskiej* w haśle *kmiećski / kmiecki.*

6. Transliteracja

Dodatkową trudnością przy rozpoznawaniu słów dawnych stanowią różnice w ich pisowni, zarówno jeśli chodzi o warianty ortograficzne, jak i typ stosowanej czcionki. Co prawda, zgodnie z przyjętymi założeniami, w słowniku historycznym mogą znajdować się hasła zapisane przy użyciu różnych rodzajów pisowni, jednak w praktyce większość z nich zapisana jest przy użyciu pisowni współczesnej. Z uwagi na fakt, że jednym z przewidywanych zastosowań opracowanego słownika może być oznaczanie tekstów dawnych informacjami morfologicznymi, zaimplementowano moduł transliteracji, który pozwala na przekształcenie napisów wykorzystujących czcionki dawne do formy współczesnej (oczywiście nie zawsze przekształcenie to jest stuprocentowo poprawne). Do zamiany pisowni tekstu przygotowany został osobny program regułowy. Dodatkowo opracowany został zestaw reguł specjalnie dostosowany do transliteracji tekstów historycznych zgromadzonych w ramach projektu IMPACT.

Przyjęliśmy założenie, że transliteracja odbywa się na poziomie słów. Przygotowanie modułu transliteracji sprowadzało się do implementacji opracowanych reguł. Przykłady reguł transliteracji zawiera tabela 4. Reguły te, stosowane bez uwzględnienia szerszego kontekstu zarówno tekstowego jak i historycznego, nie zawsze dają poprawne rezultaty. Przykładowo wynikiem zastosowania zdefiniowanego zbioru reguł do napisu *appelluiącego* jest napis *apelującego.*

Tabela 4. Przykładowe reguły transliteracji

Sekwencja rozpoznana	Sekwencja wstawiana	Przykłady	Opis słowny
cż, rż, sż	cz, rz, sz	Rżym — Rzym, máczugá — maczuga	należy opuszczać kropkę nad *ż* w dwuznakach
źi, śi, ći, dźi, ńi, żi, ċi, ṅi	zi, si, ci, dzi, ni	śiekli — siekli, śćięty — ścięty, wżiął — wziął, ṅie — nie	nie ma jednak sposobu, żeby przeprowadzić operację wstawienia diakrytów tam, gdzie ich nie ma, np. scięty — ścięty
ſM, ſ̀M	śM	ieſli — jeśli, ſláchtá — ślachta, ſmiał — śmiał, ſćiáná — ściana	M — spółgłoska miękka

Wykorzystany zestaw reguł można scharakteryzować następująco:

- Reguły są postaci $s_1 \rightarrow s_2$, gdzie s_1 oznacza ciąg znaków słowa wejściowego, który w ramach transliteracji powinien zostać zamieniony na ciąg znaków s_2 w alfabecie wyjściowym.
- Reguły mają charakter kontekstowy: można zadawać warunki na napisy bezpośrednio po lewej i po prawej stronie wejściowego ciągu s_i.

- Zestaw reguł tworzy kaskadę: na regułach określony jest porządek liniowy i jeśli względem dwóch zachodzących na siebie podsłów s_1 i s_2 słowa wejściowego można zastosować dwie reguły z zestawu, stosowana jest pierwsza z nich (mniejsza w relacji porządku).
- Reguły mogą być niedeterministyczne.

Na poziomie implementacji dodany został następujący element specyfikacji działania kaskady: reguły powinny być stosowane tak długo, jak długo istnieje przynajmniej jedna reguła postaci $s_1 \rightarrow s_2$ taka, że s_1 jest podsłowem słowa wejściowego. Należało przy tym uważać, aby nie doprowadzić do „zapętlenia się" procesu transliteracji.

Obecna implementacja nie uwzględnia reguł niedeterministycznych postaci $s_1 \rightarrow s_2 \vee s_3$. Reguły tego typu upraszczane są do wersji deterministycznej $s_1 \rightarrow s_2$. Jedną z przyczyn tej decyzji jest fakt, że w większości zastosowań chcemy, aby transliteracja prowadziła do jednego wyniku, czyli działała w sposób deterministyczny.

W celu implementacji modułu transliteracji zgodnego z powyższym opisem przygotowany został prosty język EDSL (Embedded Domain Specific Language), pozwalający definiować reguły jak najbardziej zbliżone w swojej formie do oryginalnej postaci zadanej przez lingwistkę. EDSL został skonstruowany na bazie operatorów biblioteki Parsec (http://hackage.haskell.Org/package/parsec-3.1.2), popularnej biblioteki języka Haskell pozwalającej na definiowanie parserów dla szerokiej klasy gramatyk, również kontekstowych. Następująca lista przedstawia część udostępnionych przez powstałą bibliotekę transliteracji operatorów:

- **`>#>, #>`** — reguła transliteracji, przykładowo **`"ow" #> "ów"`**.
- **`.|, .|.`** — operator lub, pozwala po lewej stronie reguły transliteracji deklarować listę napisów: **`"a" .| "a" .|. "a" .|. "a." >#> "a".`**
- **`>+>`** — sekwencyjne połączenie dwóch reguł, pozwala np. na zadawanie prawego i lewego kontekstu: **`vowel >+> ("y" #> "j")`**

Ponieważ definiowanie reguł transliteracji sprowadza się w rzeczywistości do definiowania parserów, przygotowując zestaw reguł można wykorzystywać również funkcje zdefiniowane na poziomie samej biblioteki Parsec.

Opis modułu implementującego EDSL transliteracji można znaleźć pod adresem http: //hackage.haskell.org/packages/archive/hist-pl-transliter/latest/doc/html/NLP-HistPL-Transliter.html, natomiast zapisane przy pomocy zdefiniowanego EDSL- a reguły transliteracji dla dokumentów projektu IMPACT: https://github.com/kawu/hist-pl/blob/master/transliter/src/ NLP/HistPL/Transliter/Impact.hs.

7. Łączenie informacji z wielu słowników

Słownik zbudowany na podstawie tradycyjnej wersji książkowej zawiera hasła w ich formie podstawowej. Zastosowana procedura uzupełniania go o znalezione w tekście formy odmienione w niewielkim tylko stopniu pozwoliła na uzupełnienie brakujących

form fleksyjnych. W trakcie automatycznej analizy tekstu dawnego formy nieumieszczone w słowniku nie mogłyby być prawidłowo rozpoznane. Ponieważ część tych form pozostała niezmieniona w języku współczesnym, ustalając formę podstawową słowa na podstawie znalezionej w tekście formy odmienionej, postanowiliśmy odnosić się zarówno do słownika historycznego, jak i do słownika współczesnego. W sytuacji gdy w słowniku historycznym brakuje opisu odpowiadającego napotkanej w tekście formie, słownik współczesny może stanowić alternatywne źródło informacji o morfologicznych własnościach i formie podstawowej. Realizując ten schemat postępowania wykorzystaliśmy automatyczną metodę scalania słowników polegającą na przepisywaniu do haseł słownika historycznego form pochodzących z odpowiadających im haseł słownika współczesnego. W tak powstałym słowniku zachowana została informacja, że są to jedynie formy zapożyczone ze słownika współczesnego, aby nie dodać do słownika formy, dla której nie znaleźliśmy wystąpień w analizowanych tekstach. Jest jednak możliwe, że w przypadku gdy hasła opisujące ten sam leksem występują w obu słownikach, w słowniku historycznym opis ten jest niepełny i hasło może nie zawierać opisu niektórych form odmienionych. Fakt, że forma została odnaleziona wyłącznie w słowniku współczesnym, wcale nie musi świadczyć o tym, że nie mogłaby się znaleźć w słowniku historycznym — być może zostanie rozpoznana w tekście, który nie był brany pod uwagę w momencie uzupełniania słownika.

Źródłem informacji o formach współczesnych, z którego korzystaliśmy w ramach projektu, był program Morfeusz oraz PoliMorf (http://zil.ipipan.waw.pl/PoliMorf). Morfeusz pozwala na określenie potencjalnych form podstawowych i odpowiadających im części mowy (czyli, de facto, leksemów) dla danej formy, ale wersja programu istniejąca w momencie przeprowadzania opisywanych prac nie oferowała funkcjonalności odwrotnej, czyli określenia wszystkich form odmienionych danego leksemu. Funkcjonalności tej wymaga jednak zastosowana metoda. Z tego powodu podczas scalania słowników wykorzystujemy dane pochodzące bezpośrednio ze słownika PoliMorf, przekształcone za pomocą modułów opracowanej biblioteki automatów minimalnych (http://hackage.haskell.org/package/dawg) pozwalającej na przechowywanie słowników morfologicznych w możliwie małej przestrzeni pamięci operacyjnej komputera.

Słownik PoliMorf zapisany w postaci dwóch automatów (jednego mapującego formy odmienione na formy podstawowe, drugiego — odwrotnie) wykorzystujemy do przeprowadzenia scalenia, czyli określenia dla każdego hasła leksykalnego słownika historycznego odpowiadających mu haseł słownika współczesnego. Proces ten jest zaimplementowany w sposób następujący:

- dla każdej formy odmienionej hasła leksykalnego ze słownika historycznego określane są wszystkie współczesne hasła leksykalne, które zawierają taką formę,
- odrzucane są te hasła współczesne, które mają część mowy niezgodną z hasłem historycznym.

Tak powstały słownik, zawierający zarówno formy historyczne, jak i współczesne, może posłużyć np. do oznaczania tekstów historycznych informacjami słownikowymi. Scalona wersja słownika historycznego została zaimplementowana w postaci bibliote-

ki języka Haskell, której opis można znaleźć pod adresem http://hackage.haskell.org/package/hist-pl-lexicon/docs/NLP-HistPL-Lexicon.html.

8. Klastrowanie słów

Próbując rozszerzać słownik o nowe hasła, można posłużyć się metodą znajdowania zbioru odmienionych form jednego słowa wyznaczonego na podstawie ich podobieństwa. Niniejszy rozdział opisuje pokrótce wyniki prac nad grupowaniem zbioru słów w leksemy. Proces ten jest przydatny w kontekście języków o małej liczbie zasobów, czyli w szczególności dla dawnej polszczyzny.

Poniżej opisujemy dwie przetestowane metody podziału zbioru słów na leksemy. Obie metody należą do klasy metod bez nadzoru. Pierwsza z nich opiera się na metodzie klastrowania DBSCAN (Ester i in. 1996) oraz na odległości edycyjnej, która służy za miarę podobieństwa pomiędzy poszczególnymi słowami języka. Szczegółowy opis tej metody można znaleźć w rozdziale 8.1.

Druga metoda stanowi implementację algorytmu opisanego w pracy: Janicki 2012. Klastrowanie odbywa się tutaj w oparciu o informację wzajemną (ang. mutual information) pomiędzy różnymi grupami sufiksów. Metoda ta daje całkiem satysfakcjonujące wyniki między innymi dla współczesnego języka polskiego — autor wspomnianego artykułu uzyskał dla polskiego dokładność na poziomie 96% oraz pełność na poziomie 79%. Opis implementacji tej metody i jej wykorzystanie do grupowania form dla języka historycznego znajduje się w rozdziale 8.2.

8.1. Metoda 1: DBSCAN + odległość edycyjna

Na pierwszą z przetestowanych metod grupowania leksemów składają się dwa elementy:

- algorytm klastrowania DBSCAN,
- odległość edycyjna stanowiąca miarę odległości pomiędzy poszczególnymi słowami zbioru wejściowego.

DBSCAN jest algorytmem, w którym dwa słowa p i q są umieszczane w tym samym klastrze wtedy i tylko wtedy, gdy łączy je ścieżka $w_1, w_2, \ldots, w_n$ dowolnej długości $n > 0$ (przy czym $w_1 = p$ oraz $w_n = q$) taka, że dla każdej pary (w_i, w_{i+i}) słowo w_{i+i} znajduje się w zbiorze sąsiadów słowa w_i lub odwrotnie, w_i znajduje się w zbiorze sąsiadów słowa w_{i+1}. Zbiór sąsiadów słowa w definiujemy w sposób następujący:

$$N(w) = s(\{v : v \in W, dist(w, v) < \varepsilon\})$$

$$s(X) = \begin{cases} X, & |X| \geq minPts \\ \emptyset, & \text{w przeciwnym przypadku} \end{cases}$$

gdzie W to zbiór wszystkich słów, ε to maksymalna odległość pomiędzy słowami należącymi do tego samego klastra, *minPts* to minimalna liczba słów w zbiorze sąsiadów (zapobiega powstawaniu małych klastrów i w ten sposób zmniejsza wrażliwość na

„szum" danych wejściowych), a *dist* to odgórnie zdefiniowana funkcja odległości pomiędzy słowami zbioru W.

Zgodnie z intencjami autorów algorytmu funkcja *dist* powinna być poprawnie zdefiniowaną odległością (w szczególności warunek ten jest spełniony przez tradycyjną odległość Levenshteina), ale w praktyce algorytm zadziała również, gdy funkcja *dist* nie spełnia pewnych warunków odległości (np. nie jest symetryczna). Dzięki temu możliwe jest testowanie metody DBSCAN w kontekście rozszerzonej wersji odległości edycyjnej, np. takiej, w której koszt operacji edycji zależy od odległości operacji od końca wyrazu. W przeprowadzonych eksperymentach wykorzystaliśmy, opisaną w rozdziale 5.2, miarę odległości zdefiniowaną wcześniej dla podzadania uzupełniania słownika o formy podobne.

Implementacja powyższej metody grupowania słów znajduje się w repozytorium https: //github.com/kawu/hist-pl/tree/master/cluster-lang. Narzędzie ma postać pakietu języka Haskell i dostarcza program umożliwiający przeprowadzenie klastrowania zadanego na wejściu tekstu. Szczegóły dotyczące instalacji i sposobu używania narzędzia można znaleźć w pliku README znajdującym się w podanym wyżej repozytorium kodu.

8.1.1. Zalety

Warto zauważyć, że użycie algorytmu DBSCAN w kontekście grupowania słów jest pod wieloma względami naturalne. Po pierwsze, odległość edycyjna stanowi naturalną miarę podobieństwa pomiędzy słowami. Po drugie, podczas obliczaniu kolejnych klastrów przeprowadzana jest operacja wyszukiwania słów znajdujących się w odległości co najwyżej ε od zadanego słowa, a operacja ta, dzięki zastosowaniu metod wyszukiwania przybliżonego, może być przeprowadzona w sposób efektywny zarówno pod względem szybkości, jak i wymagań pamięciowych komputera. Operacja ta jest dostępna za pośrednictwem biblioteki wyszukiwania przybliżonego adict (http://hackage.haskell.org/package/adict) i działa względem minimalnego automatu słów, dzięki czemu jest mało wymagająca pamięciowo. Naiwna implementacja tego algorytmu — przejście po wszystkich parach słów $v \neq w$ zbioru W i określenie tych, dla których — działałaby w czasie kwadratowym względem rozmiaru zbioru wejściowego, co dla dużych zbiorów danych byłoby rozwiązaniem nie do przyjęcia.

Inną zaletą metody DBSCAN jest to, że nie wymaga podawania na wejściu odgórnie ustalonej liczby grup, co w przypadku zbioru słów byłoby bardzo niewygodne ze względu na potencjalnie bardzo dużą i trudną do oszacowania ich liczbę. W związku z tym niektóre bardziej popularne metody, np. algorytm k-średnich (ang. k-means) lub jego odmiana — algorytm k-medoids, nie byłyby odpowiednie.

8.1.2. Wady

Wykorzystanie metody klastrowania DBSCAN do grupowania słów ma jedną istotną wadę. Metoda ta nie gwarantuje w żaden sposób, że powstałe klastry będą jednolite i często występuje sytuacja, w której do jednego klastra trafia kilka leksemów podobnych do siebie jedynie „na styku".

Dla przykładu, niech $v \in W$ i $w \in W$ będą dwoma słowami należącymi do różnych leksemów języka oraz niech $dist(v,w) < \varepsilon$. Jeśli słowa te posiadają odpowiednio duże zbiory sąsiadów (o rozmiarze > *minPts*), to odpowiadające im klastry zostaną połączone nawet wtedy, gdy dla wszystkich pozostałych par słów (v',w') takich, że $v' = v$ lub $w' = w$, należących do poszczególnych leksemów.

Powyższa sytuacja występuje w praktyce bardzo często. Przykładem mogą być leksemy Abba, Abbas, Abbasyda, Abbot oraz Abbott, które w wyniku uruchomienia powyższego algorytmu przy *minPts* = 3, odległości edycyjnej zależnej od pozycji, oraz parametrze ε również zależnym od pozycji, zostały połączone w jeden klaster. Łatwo sprawdzić, że średnia odległość edycyjna pomiędzy słowami leksemu Abba jest dużo mniejsza niż średnia odległość edycyjna pomiędzy słowami tego leksemu a słowami należącymi do pozostałych leksemów, ale metoda DBSCAN nie bierze tej własności pod uwagę.

Potencjalnym rozwiązaniem tego problemu byłby postprocessing, w którego ramach otrzymane przy użyciu metody DBSCAN wyniki były ponownie dzielone — każdy klaster niezależnie — na podgrupy przy użyciu metody, która bierze pod uwagę spójność klastrów, np. algorytmu k-medoids.

8.2. Metoda 2: metoda sufiksowa

Druga z przetestowanych metod opiera się na informacji wzajemnej pomiędzy grupami sufiksów. Obszerny opis tej metody można znaleźć w: Janicki 2012, tutaj podamy jedynie podstawowe informacje na jej temat.

W pierwszej fazie algorytmu obliczane są częstości występowania sufiksów różnych długości (maksymalna rozpatrywana długość stanowi parametr algorytmu). Jedynie sufiksy o odpowiedniej częstości w zbiorze danych (minimalna częstość również stanowi parametr, w przeprowadzonych testach ustawiony na 0.001) są dalej rozpatrywane.

Przestrzeń zdarzeń elementarnych definiujemy jako zbiór prefiksów występujących w zbiorze danych. Prefiksy te reprezentują potencjalne „trzony" poszczególnych leksemów języka. Przestrzeń można łatwo obliczyć na podstawie drzewa Trie lub automatu minimalnego przechowującego zbiór słów wejściowych. Program udostępniony przez autora artykułu implementujący opisywaną metodę wykorzystuje tutaj właśnie drzewa Trie. Implementacja wykorzystana w testach klastrowania dawnej polszczyzny opiera się na automatach minimalnych, dzięki czemu powinna być bardziej wydajna pamięciowo.

W kolejnym kroku definiujemy informację wzajemną $I(X,Y)$, która mierzy zależność pomiędzy rozłącznymi zbiorami sufiksów X i Y (formalnie: zmiennymi losowymi reprezentującymi te zbiory). Powiemy, że prefiks p przyjmuje sufiks s, jeżeli $p + s$ należy do zbioru wejściowego W, gdzie + stanowi konkatenację słów. Gdy informacja o tym, które sufiksy ze zbioru X przyjmuje losowo wybrany prefiks p, pozwala z dużą pewnością określić, które sufiksy ze zbioru Y prefiks p przyjmuje (i odwrotnie: infor-

macja o tym, które sufiksy ze zbioru *Y* są przyjmowane losowo przez prefiks *p*, pozwala z dużą pewnością określić, które sufiksy ze zbioru *X* przyjmuje *p*), wartość informacji wzajemnej jest relatywnie wysoka.

Algorytm Janickiego opiera się na założeniu, że jeśli dwa rozłączne zbiory sufiksów *X* i *Y* charakteryzują się wysoką informacją wzajemną (większą niż zadany na wejściu algorytmu parametr κ), to reprezentują ten sam paradygmat morfologiczny, co w praktyce oznacza, że po doklejeniu sufiksów z obu powyższych zbiorów – *X* oraz *Y* — do zadanego prefiksu *p* otrzymamy zbiór form należących do tego samego leksemu. W oparciu o powyższą przesłankę konstruowana jest relacja równoważności pomiędzy słowami zbioru wejściowego, a klasy abstrakcji tej relacji stanowią wyjściowe klastry algorytmu grupowania słów.

Metoda sufiksowa jest dobrze dostosowana do języków fleksyjnych, w których poszczególne formy leksemów wyrażają się poprzez doklejanie sufiksów, szczególnie jeśli te (sufiksy) można pogrupować w typowe szablony odmiany. Jest to cecha, której metoda opisana w rozdziale 8.1 nie posiada.

Oczywistą wadą metody jest to, że nie jest ona w stanie modelować żadnych innych form odmiany poza dodawaniem sufiksów. Chociaż wada ta jest fundamentalna, paradoksalnie w praktyce może wpływać pozytywnie na wyniki działania metody sufiksowej, ponieważ zapobiega rozpoznawaniu fałszywych relacji pomiędzy podobnymi do siebie, lecz różniącymi się na poziomie prefiksów słowami. W wielu przypadkach słowa takie należą do różnych leksemów, nawet jeśli z punktu widzenia odległości edycyjnej są do siebie bardzo podobne.

Potencjalną wadą metody sufiksowej jest jej wrażliwość na rzadkość zbioru danych wejściowych. W kontekście języka dawnej polszczyzny zbiór słów wejściowych generowany jest na podstawie zbioru tekstów historycznych. Zbiór ten nie jest jednak na tyle pokaźnych rozmiarów, aby dla każdego napotkanego w tekście słowa w pozostałej części zbioru tekstów znajdowały się pozostałe formy odmiany tego słowa. I rzeczywiście, opisywana tutaj metoda działa wyraźnie lepiej na zbiorze słów wydobytym ze słownika PoliMorf (po uprzednim usunięciu podziału na leksemy) niż na tekstach historycznych. Intuicyjnie metoda opisana w rozdziale 8.1 powinna być mniej wrażliwa na rzadkość danych wejściowych, lecz w praktyce i tak osiągała wyniki niższej jakości na danych historycznych niż metoda sufiksowa.

Opracowana implementacja algorytmu grupowania słów dostępna jest w repozytorium pod adresem https://github.com/kawu/lexeme-clustering, a opis jej użycia można znaleźć w pliku README tego repozytorium.

9. Podsumowanie

W artykule przedstawione zostały wyniki prac nad opracowaniem struktury słownika polszczyzny dawnej oraz zamiany przykładowego słownika tradycyjnego na słownik elektroniczny. Zaproponowano także metodę uzupełniania słownika o nowe, znalezione w tekstach oryginalnych formy fleksyjne. Wszystkie opracowane narzędzia są ogól-

nie dostępne i mogą być wykorzystane przy tworzeniu innych słowników, natomiast udostępnienie danych uzależnione jest od możliwości uzyskania praw do ich rozpowszechnienia. W przyszłości planowane są rozmowy na temat włączenia do budowanej struktury innych zasobów leksykalnych.

Przetworzenie słownika tradycyjnego na wersję elektroniczną wymaga sporego nakładu pracy. Zarówno wyniki OCR, jak i automatycznej konwersji zapisu oryginalnego na strukturę LMF zawierają sporo błędów wynikających z niedostosowania konwencji przyjętych przez autora słownika do potrzeb automatycznego przetwarzania. Wobec braku narzędzi i zasobów dedykowanych polszczyźnie dawnej praktycznie użyteczne może być wspomaganie prac przy budowie słowników przez wykorzystanie narzędzi opracowanych dla języka współczesnego i automatycznych metod znajdowania form zbliżonych. Jednak ich wyniki muszą być ręcznie poprawione ze względu na stosunkowo duży procent popełnianych błędów.

10. Podziękowania

Niniejsza praca powstała w ramach projektu SYNAT (Utworzenie uniwersalnej, otwartej, repozytoryjnej platformy hostingowej i komunikacyjnej dla sieciowych zasobów wiedzy dla nauki, edukacji i otwartego społeczeństwa wiedzy, SP/I/1/77065/10). Zestaw reguł do transliteracji tekstów wykorzystany przy konwersji danych projektu IMPACT na czcionki współczesne opracowała Ewa Rodek, w opracowywaniu zawartości hasła słownikowego uczestniczyła Katarzyna Głowińska, w pracach nad zebraniem i usystematyzowaniem uzyskanego zbioru tekstów danych główny udział miały Monika Kresa oraz Gabriela Świrska, która również dokonała większości ręcznych poprawek w ostatecznej wersji słownika.

Literatura

Ester M., Kriegel H.P., Sander J., Xu X., 1996, A density-based algorithm for discovering clusters in large spatial databases with noise, [w:] Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), AAAI Press, s. 226–231.

Francopoulo G., red., 2013, LMF Lexical Markup Framework, Wiley, Londyn.

Francopoulo G., Bel N., George M., Calzolari N., Monachini M., Pet M., Soria C., 2009, Multilingual resources for nlp in the lexical markup framework (lmf), Language Resources and Evaluation 43 (1), s. 57–70.

Janicki M., 2012, A Lexeme-Clustering Algorithm for Unsupervised Learning of Morphology, [w:] SKIL 2012 — Dritte Studentenkonferenz Informatik Leipzig, red. J. Schmidt, T. Riechert, S. Auer, t. 34 serii Leipziger Beiträge zur Informatik, LIV, Leipzig, Best Paper Award, s. 37–47.

Mykowiecka A., Głowińska K., Rychlik P., Waszczuk J., 2011, Construction of an electronic dictionary on the base of a paper source, [w:] Proceedings of the 5th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, red. Z. Vetulani, Poznań, s. 506–510.

Mykowiecka A., Rychlik P., Waszczuk J., 2011, Building an electronic dictionary of old Polish on the base of the paper resource, [w:] Proceedings of the Workshop on Adaptation of Language Resources and Tools for Processing Cultural Heritage at LREC 2012, European Language Resources Association, s. 16–21.

Reczek S., 1968, Podręczny słownik dawnej polszczyzny, Ossolineum, Wrocław.

Reynaert M., 2008, Non-Interactive OCR Post-Correction for Giga-Scale Digitization Projects, [w:] CICLing'08 Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing, LNCS 4919, Springer, s. 617–630.

Waszczuk J., 2013, A representation of an old Polish dictionary designed for practical applications, [w:] IIS, red. M. A. Kłopotek, J. Koronacki, M. Marciniak, A. Mykowiecka, S. T. Wierzchoń, t. 7912 serii Lecture Notes in Computer Science, Springer, s. 151–156.

Woliński M., 2006, Morfeusz — a practical tool for the morpho-logical analysis of Polish, [w:] Intelligent Information Processing and Web Mining, red. M.A. Kłopotek, S.T. Wierzchoń, K. Trojanowski, Advances in Soft Computing, Berlin, Springer-Verlag, s. 503–512.

SUMMARY

Defining the structure and tools supporting construction of the dictionary of Old Polish

Keywords: electronic dictionary, LMF, Old Polish, automatic text processing, digitalization.
Słowa kluczowe: słownik elektroniczny, LMF, polszczyzna niewspółczesna, automatyczne przetwarzanie tekstów, digitalizacja.

In this paper we present works on an electronic version of a dictionary of Old Polish. As a starting point of the construction of the dictionary we adopted an existing dictionary in a paper form. We describe all subsequent stages of the process of converting the paper source into fully functional electronic form. These stages consist in scanning the paper source followed by OCR and correction of its results, converting the data into LMF format and enriching the dictionary with information which was given indirectly or which can be obtained from other sources. In particular we describe the method of assigning part of speech names to lexicon entries.