Uniwersytet im. Adama Mickiewicza w Poznaniu
Wydział Matematyki i Informatyki

Michał Paszyn
Nr albumu: 334858

Wspomaganie wyszukiwania osób w sieciach społecznościowych

Supporting person searching in social networks

Praca magisterska
na kierunku INFORMATYKA

Promotor:
prof. UAM dr hab. Krzysztof Jassem

Poznań 2012

Moim Rodzicom

Oświadczenie

Ja, niżej podpisany **Michał Paszyn**, student Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt. **Wspomaganie wyszukiwania osób w sieciach społecznościowych** napisałem samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób.

Oświadczam również, że egzemplarz pracy dyplomowej w formie wydruku komputerowego jest zgodny z egzemplarzem pracy dyplomowej w formie elektronicznej.

Jednocześnie przyjmuję do wiadomości, że gdyby powyższe oświadczenie okazało się nieprawdziwe, decyzja o wydaniu mi dyplomu zostanie cofnięta.

Data Podpis autora pracy

Spis treści

Wstęp 7

1. Sieci społecznościowe 9
1.1. Wprowadzenie 9
1.2. Rodzaje serwisów społecznościowych 9
1.3. Serwisy społecznościowe w Polsce 10
1.4. Serwisy społecznościowe na świecie 12

2. Wyszukiwanie użytkowników w wybranych serwisach społecznościowych 15
2.1. Analiza zachowania użytkowników sieci społecznościowych 15
2.1.1. Brak polskich liter 15
2.1.2. Zdrobnienia i różne formy imienia 16
2.1.3. Różnice w zapisie i wymowie 16
2.1.4. Literówki 17
2.1.5. Zapożyczenia językowe 17
2.1.6. Stosowanie myślnika (łącznika) 18
2.1.7. Emigracja 18
2.1.8. Używanie drugiego imienia 19
2.1.9. Używanie nazwiska panieńskiego 19
2.2. Podsumowanie 19

3. Omówienie wybranych algorytmów fonetycznych 21
3.1. Algorytm Soundex 21
3.1.1. Opis działania algorytmu 21
3.1.2. Przykłady działania algorytmu dla angielskich nazwisk 22
3.1.3. Przykłady działania algorytmu dla polskich nazwisk 23
3.2. Algorytm Daitch-Mokotoff Soundex 23
3.2.1. Opis działania algorytmu 24
3.2.2. Przykłady działania algorytmu dla żydowskich nazwisk 26
3.2.3. Przykłady działania algorytmu dla polskich nazwisk 26
3.3. Algorytm Metaphone 27
3.3.1. Opis działania 27
3.3.2. Przykłady działania algorytmu dla angielskich nazwisk 29
3.3.3. Przykłady działania algorytmu dla polskich nazwisk 30

4. Autorski algorytm fonetyczny dla języka polskiego - *PolishSoundex* . . . 31
4.1. Opis działania 31
4.2. Przykłady działania algorytmu dla polskich nazwisk 32

4.3. Przykłady działania algorytmu dla anglojęzycznych nazwisk . . . 33

5. Opis implementacji systemu . . . 35
5.1. Baza imion i nazwisk . . . 35
5.2. Wyszukiwarka . . . 36

6. Porównanie systemu z wyszukiwarkami serwisów społecznościowych . . . 41
6.1. Wyniki ewaluacji . . . 41
6.2. Dalszy rozwój projektu . . . 42

Podsumowanie . . . 43

Dodatek A. Dokumentacja projektu magisterskiego . . . 45

Dodatek B. Kody źródłowe . . . 51

Bibliografia . . . 61

Wstęp

W niniejszej pracy magisterskiej przedstawione zostaną podstawowe informacje dotyczące coraz popularniejszych serwisów społecznościowych. Wraz ze wzrostem popularności wśród użytkowników Internetu zaczynają pojawiać się pewne problemy, które mogą utrudnić korzystanie z tych portali. W szczególności mowa tu o obsłudze imion, zdrobnień i nazwisk występujących w Polsce przez serwisy społecznościowe oraz zniekształcenia imion i nazwisk spowodowane przez codziennie zachowania internautów. Wyszczególnione zostaną główne problemy z wyszukiwaniem użytkowników oraz sposoby ich rozwiązania.

W dalszej części pracy przedstawione zostaną trzy najpopularniejsze obecnie algorytmy fonetyczne, czyli Soundex, Daitch-Mokotoff Soundex oraz Metaphone. Zaprezentowany zostanie także autorski algorytm fonetyczny dla języka polskiego, który stanie się podstawą do stworzenie serwisu internetowego **osobomat.pl**. Celem serwisu jest ułatwienie znalezienia danej osoby w serwisach społecznościowych Facebook.com i nk.pl. Przedstawiona zostanie także architektura serwisu i sposób jego wykonania.

Ostatnią część pracy stanowi porównanie stworzonego narzędzia z obecnymi rozwiązaniami. Zaproponowane zostaną dalsze możliwości rozwoju projektu.

W dodatku A znajduje się dokumentacja projektu magisterskiego. Dodatek B zawiera kody źródłowe skryptów w języku Python, które były niezbędne do zebrania danych na potrzeby niniejszej pracy.

Na płycie dołączonej do pracy znajduję się jej wersja elektroniczna, kody źródłowe skryptów oraz zrzut bazy danych.

Rozdział 1

Sieci społecznościowe

W niniejszym rozdziale przedstawione zostaną podstawowe informacje związane z sieciami społecznościowymi. Omówione zostaną rodzaje serwisów społecznościowych oraz sposoby wyszukiwania użytkowników na przykładach wybranych serwisów internetowych.

1.1. Wprowadzenie

W związku z ciągle zwiększająca się liczbą użytkowników Internetu zwiększa się także liczba osób chcących przynależeć do określonych sieci społecznościowych w obrębie serwisów społecznościowych. Według danych przedstawionych przez Megapanel z marca 2011 r.[1], aż 99,3 % internautów korzysta z sieci społecznościowych.

Na potrzebny niniejszej pracy przyjęto następujące definicje sieci społecznościowej oraz serwisu społecznościowego (za [1])[2]:

Definicja 1.1.1 Sieć społecznościowa *jest to struktura społeczna skupiająca ludzi połączonych między sobą wzajemnymi relacjami.*

Definicja 1.1.2 Serwis społecznościowy *jest to serwis internetowy umożliwiający utworzenie publicznego lub prywatnego profilu użytkownika, połączenie go relacjami z innymi użytkownikami oraz udostępnianie treści w obrębie serwisu.*

1.2. Rodzaje serwisów społecznościowych

Serwisy społecznościowe, podobnie jak pozostałe portale internetowe, można przypisać do określonych kategorii. Podział ten w dużej mierze zależy od przyjętych kryteriów grupowania. Ze względu na swoją specyfikę serwisy społecznościowe mogą należeć do więcej niż jednej kategorii.

Rodzaje serwisów społecznościowych:

1. **Ogólnego zastosowania**
   Serwisy te oferują swoim użytkownikom różne rodzaje funkcjonalności, nie skupiają się na wybranej tematyce. Użytkownicy najczęściej korzystają z możliwości publikowania

[1] http://www.pbi.org.pl/index.php/ida/2/?aktualnoscID=144&p=1
[2] Tłumaczenie własne

statusów, dodawania zdjęć, komentowania treści udostępnionych przez innych użytkowników a także wysyłania prywatnych wiadomości. Ze względu rozmiar i duży zasięg to przez ich pryzmat najczęściej ocenia się serwisy z innych kategorii. Przykładami sieci ogólnego zastosowania są serwisy: Facebook.com, Google+.

2. **Ukierunkowane na dzielenie się treściami**
   Głównym zadaniem tych serwisów jest umożliwienie użytkownikom udostępniania własnych materiałów np. filmów, muzyki, zdjęć oraz ocenianie i komentowanie ich przez pozostałych. Przykładami sieci ukierunkowanymi na dzielenie się treściami są serwisy: YouTube.com, wrzuta.pl, fotka.pl.

3. **Ukierunkowane na opiniowanie i recenzowanie**
   Dzięki tym serwisom osoby z nich korzystające mają możliwość wyrażenia swoich opinii na różne tematy począwszy do filmów i muzyki a na lekarzach i wykładowcach skończywszy. Przykładami sieci ukierunkowanymi na opiniowanie i recenzowanie są serwisy: Filmweb.pl, opiniuj.pl, Yelp.com.

4. **Skierowane do konkretnych grup społecznych** Serwisy te przeznaczone są do konkretnych grup społecznych chcących kontaktować się jedynie we własnym gronie, np. dla studentów, programistów czy artystów. Przykładami sieci skierowanymi do konkretnych grup społecznych są serwisy: Nasz-uam.pl, 9fingers.pl, digiart.pl.

5. **Ukierunkowane na poznawanie innych osób**
   W głównej mierze są to serwisy randkowe umożliwiające poznanie innych osób o podobnych zainteresowaniach. Większość tych serwisów udostępnia użytkownikom pewne funkcjonalności za dodatkową opłatą, np. nieograniczone wysyłanie wiadomości do innych użytkowników. Przykładami sieci ukierunkowanymi na poznawanie innych osób są serwisy: dopasowani.pl, Sympatia.pl.

1.3. Serwisy społecznościowe w Polsce

Pierwsze serwisy społecznościowe w Polsce zaczęły powstawać po roku 2000. Za pierwszy portal społecznościowy można uznać serwis fotka.pl, który został uruchomiony 14 lutego 2001 r.[3] Jego główną funkcjonalnością było ocenianie zdjęć innych użytkowników tego serwisu. Obecnie najpopularniejszym polskim serwisem społecznościowym, a drugim pod względem liczby użytkowników, jest nk.pl (dawniej Nasza-klasa.pl). Według danych Megapanel PBI/Gemius za miesiąc kwiecień 2012 r.[4] serwis posiada ok. 11 mln użytkowników. Uruchomiony został 11 listopada 2006 r., a jego głównym celem jest umożliwianie kontaktowania się ze znajomymi ze szkolnych czasów.

Portal nk.pl umożliwia użytkownikom założenie wirtualnej szkoły oraz klasy. Każdy może także zapisać się do dowolnej ilości klas i szkół, o ile taką możliwość udostępnił twórca danej klasy/szkoły.

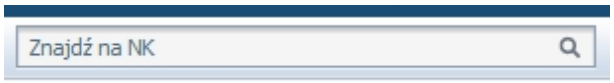
Na Rysunku 1.1 przedstawiony jest przykładowy profil klasy w serwisie nk.pl. Użytkownicy uczęszczający do tej klasy zapisani są w wirtualnym dzienniku. Dzięki temu pozostali użytkownicy mogą ich odnaleźć. Jednak odnajdywanie osób w ten sposób wymaga od osoby

[3] http://pl.wikipedia.org/wiki/Fotka.pl
[4] http://www.gemius.pl/pl/aktualnosci/2012-06-13/01

szukającej wiedzy na temat konkretnej szkoły oraz klasy. Nie każdy musi także być przypisany do klasy, do której chodził. Dlatego też, co stało się standardem wśród tego typu portali, serwis oferuje wyszukiwarkę użytkowników.

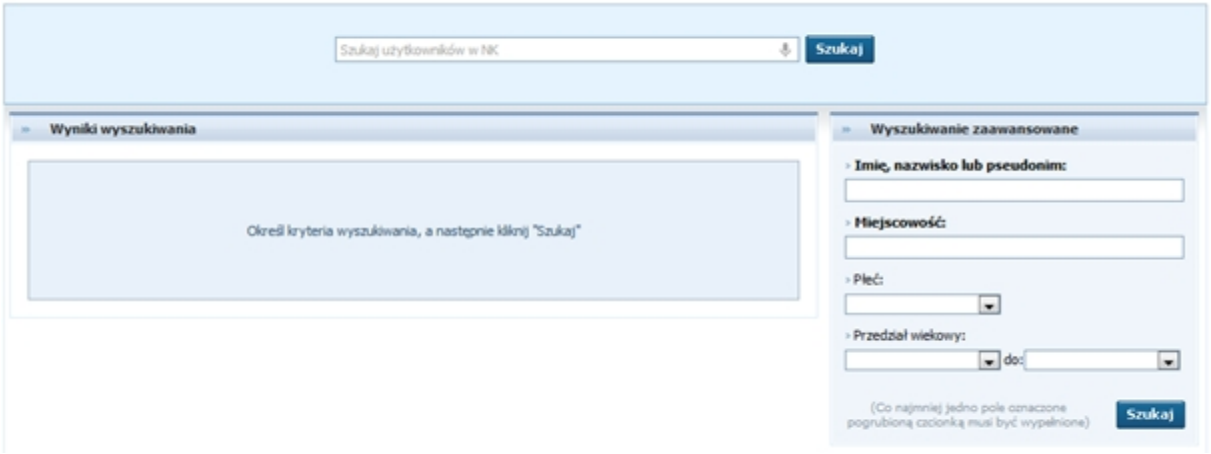
Rysunek 1.1: Profil klasy w serwisie nk.pl

Rysunek 1.2 przedstawia wyszukiwarkę portalu nk.pl. Umieszczona jest na stronie głównej serwisu w prawym górnym roku. Pozwala na wyszukiwanie osoby tylko po imieniu i nazwisku. W celu doprecyzowania kryteriów np. miasto lub wiek musimy przejść do wyszukiwarki zaawansowanej.



Rysunek 1.2: Wyszukiwarka w serwisie nk.pl

Rysunek 1.3 przedstawia wyszukiwarkę zaawansowaną, która umożliwia wyszukiwanie osób według imienia i nazwiska lub pseudonimu, miejscowości, płci oraz wieku.



Rysunek 1.3: Zaawansowana wyszukiwarka w serwisie nk.pl

Analiza wyszukiwania użytkowników serwisu nk.pl zostanie przedstawiona i omówiona w Rozdziale 2 niniejszej pracy.

1.4. Serwisy społecznościowe na świecie

Serwisy społecznościowe na świecie zaczęły pojawiać się kilka lat wcześniej niż w Polsce. Do pierwszych tego typu serwisów można zaliczyć amerykański ClassMates, który był pierwowzorem nk.pl. Uruchomiony został w 1995 r. i funkcjonuje do dzisiaj. Obecnie najpopularniejszym serwisem społecznościowym na świecie a także w Polsce jest portal Facebook. Ogólną liczbę kont szacuje się na ponad 900 milionów (!), z czego ok. 13,6 mln w samej Polsce (wg danych Megapanel PBI/Gemius za miesiąc kwiecień 2012 r.)[5]. Portal powstał w 2004 r. wyłącznie na potrzeby stworzenia sieci studentów Uniwersytetu Harvarda. Obecnie zaliczany jest do serwisów ogólnego przeznaczenia.

W przeciwieństwie do nk.pl Facebook nie oferuje użytkownikom możliwości tworzenia i zapisywania się do konkretnych klas. Możliwe jest jednak przypisanie danej szkoły do profilu użytkownika, lecz nie ma możliwości przeglądania innych osób, które uczyniły to samo poza „znajomymi" użytkownika. Z tego względu najpopularniejszą sposobem wyszukiwanie jest korzystanie z wbudowanej wyszukiwarki. Rysunek 1.4 przedstawia wygląd wyszukiwarki w serwisie Facebok.com

Rysunek 1.4: Wyszukiwarka w serwisie Facebook.com

Podobnie jak w przypadku nk.pl żeby doprecyzować wyniki wyszukiwania musimy skorzystać z zaawansowanej wyszukiwarki. Rysunek 1.5 przedstawia wygląd wyszukiwarki zaawansowanej. Umożliwia ona wyszukiwanie osób lokalizacji, wykształcenia lub miejsca pracy.

[5]http://www.gemius.pl/pl/aktualnosci/2012-06-13/01

Niestety nie można wyszukiwać osób według wieku lub płci, co oferuje serwis nk.pl.

Analiza wyszukiwania użytkowników serwisu Facebook.com zostanie przedstawiona i omówiona w Rozdziale 2 niniejszej pracy.

Rysunek 1.5: Zaawansowana wyszukiwarka w serwisie Facebook.com

Rozdział 2

Wyszukiwanie użytkowników w wybranych serwisach społecznościowych

W niniejszym rozdziale przedstawiona zostanie analiza zachowań użytkowników sieci społecznościowych odnosząca się do pisowni ich imion i nazwisk. Pokazane zostaną także rezultaty przykładowych wyszukiwań w serwisach społecznościowych Facebook.com i nk.pl, a także niedoskonałości wyszukiwarek tych serwisów.

2.1. Analiza zachowania użytkowników sieci społecznościowych

Celem zbadania zachowań użytkowników serwisów społecznościowych dokonano porównania danych przedstawionych w serwisach a ich poprawnym zapisem w języku polskim. W tym celu z listy najpopularniejszych polskich nazwisk[1] wybrano 200 najpopularniejszych rekordów. Umożliwiały one sprawdzenie obsługi przez serwisy:

- polskich znaków diakrytycznych: ą, ć, ę, ł, ń, ó, ś, ź, ż
- różnic w zapisie i wymowie
- myślników (łączników)

Następnie dla każdego nazwiska przeprowadzono wyszukiwanie w serwisie Facebook.com oraz nk.pl. Zebrane dane zostały opracowane manualnie. Kolejne podrozdziały opisują najważniejsze dostrzeżone zjawiska dla języka polskiego.

2.1.1. Brak polskich liter

Najczęściej występującym zjawiskiem wśród użytkowników serwisów społecznościowych jest brak stosowania polskich znaków diakrytycznych. Dotyczy to zarówno imion jak i nazwisk.

Przykłady:

- Michał – *Michal*,

[1] http://www.futrega.org/etc/nazwiska.html

- Jędrzej – *Jedrzej*,
- Zając – *Zajac*,
- Wójcik – *Wojcik*,
- Król – *Krol*.

W przypadku serwisu Facebook.com w wyniku zapytania, które zawiera polskie znaki diakrytyczne otrzymamy tylko listę osób, których dane w serwisie zawierają te znaki. Przy wyszukiwaniu bez polskich znaków otrzymamy zarówno listę osób z danymi zawierającymi polskie znaki jak i ich odpowiedniki bez polskich znaków np. *Wiśniewski – Wisniewsk*i. Dotyczy to 8 z 9 polskich liter zawierających znaki diakrytyczne. Wyjątek stanowi tutaj litera *ł* i jej odpowiednik *l*. Przy wyszukiwaniu, w którym użyjemy litery l, nie otrzymamy żadnych wyników zawierających literę *ł*. Jako przykład można przytoczyć imię *Łukasz*. Niepowodzeniem zakończy się wyszukiwanie osoby przy wprowadzeniu tekstu *Lukasz*.

W wyniku próby wyszukania w portalu nk.pl nie otrzymamy dwóch rodzajów odpowiedzi. Serwis dla wszystkich polskich liter zachowuje się w identyczny sposób jak serwis Facebook.com dla litery *ł*.

2.1.2. Zdrobnienia i różne formy imienia

Kolejnym często występującym zjawiskiem wśród użytkowników serwisów społecznościowych jest używanie zdrobnień lub innych form imion. Występuje to częściej w przypadku kobiet niż mężczyzn.

Przykłady:

- Katarzyna – *Kaśka*,
- Joanna – *Asia*,
- Kamila – *Kama*,
- Dariusz – *Darek*,
- Michał – *Michaś*.

Zarówno w przypadku serwisu Facebook.com jak i nk.pl wyszukiwanie wg oryginalnych imion osób, które podały inną formę lub zdrobnienie nie przyniesie rezultatów. Zwrócona zostanie lista osób, których dane dokładnie zgadzają się z danymi użytymi podczas wyszukiwania.

2.1.3. Różnice w zapisie i wymowie

W trakcie analizy danych zauważone zostały pewne nazwiska, które ze względu na charakter języka polskiego, posiadają różne warianty pisowni (często niepoprawne), a brzmiące tak samo.

Przykłady:

- Dąbrowski – *Dombrowski*,

- Wiśniewski – *Wiśnieski*,
- Mazur – *Mazór*,
- Grzyb – *Grzyp*,
- Czyż – *Czysz*.

Żaden z testowanych serwisów nie umożliwia poprawnego wyszukiwania użytkowników, dla których wymowie danych personalnych może odpowiadać kilka wersji zapisu.

2.1.4. Literówki

Osobom wyszukującym jak i wyszukiwanym zdarza się popełniać literówki. Można wyróżnić tutaj ich kilka rodzajów:

- pominięcie znaku
- wstawienie niewłaściwego znaku
- zamiana kolejności znaków

Przykłady:

- Tomek – *Tomk*,
- Skrzypczak – *Skrzyoczak*,
- Dariusz – *Daeiusz*,
- Krzysztof – *Krzysztfo*,
- Michalski – *Mihcalski*.

Zarówno serwis Facebook.com jak i nk.pl nie radzą sobie z wyszukiwaniem, gdy został pominięty znak we wprowadzonych danych.

Przy wstawieniu niewłaściwego znaku Facebook.com częściowo potrafi sobie z tym poradzić. Oprócz listy wyników przedstawia także propozycję prawidłowej pisowni. Portal nk.pl nie zwraca żadnych wyników w przypadku wstawienia niewłaściwego znaku.

Przy zamianie kolejności znaku sytuacja wygląda podobnie jak w przypadku wstawienia niewłaściwego znaku. Facebook.com częściowo obsługuje tego typu zapytania oraz sugeruje poprawną pisownię, natomiast nk.pl nie obsługuje tego typu zapytań.

2.1.5. Zapożyczenia językowe

Interesującym zjawiskiem występującym wśród osób korzystających z portali społecznościowych są zapożyczenia językowe. W głównej mierze tyczy się to imion, a rzadziej nazwisk użytkowników.

Przykłady:

- Katarzyna – *Kate*,

- Dawid – *David*,
- Michał – *Majk*,
- Tomasz – *Thomas*,
- Król – *King*.

Zarówno wyszukiwarka serwisu Facebook.com jak i serwisu nk.pl nie potrafi dopasować obcej formy imienia do jego polskiego odpowiednika.

2.1.6. Stosowanie myślnika (łącznika)

Część polskich nazwisk to nazwiska dwuczłonowe, czyli takie, które składają się z dwóch części połączonych myślnikiem. Występują też rzadkie przypadki, gdy łącznik nie występuje.

Przykłady:

- *Żmuda-Trzebiatowska*,
- *Wnuk-Lipiński*,
- *Wantoch-Rekowski*,
- *Czeszejko-Sochacki*,
- *Frycz Modrzewski*.

Oba serwisy społecznościowe dobrze radzą sobie z nazwiskami dwuczłonowymi. Zwracają poprawne wyniki zarówno przy użyciu myślnika, bez jego użycia oraz tylko przy podaniu jednego członu nazwiska.

2.1.7. Emigracja

W związku z emigracją polskich obywateli do innych krajów pojawił się problem z nazwiskami kobiet. Dotyczy to krajów, w których języku nie występuje odmiana wyrazów przez przypadki. W celu uniknięcia problemów kobiety przyjmują formę rodzaju męskiego nazwiska w mianowniku.

Przykłady:

- *Anna Kowalski*,
- *Marta Wiśniewskii*,
- *Joanna Szymański*,
- *Kamila Lewandowski*,
- *Anna Nowicki*.

W przypadku obu testowanych serwisów żaden nie jest w stanie zwrócić dobrej odpowiedzi. Dzieje się tak zarówno, gdy szukane jest nieodmienione nazwisko, a w serwisie widnieje jako odmienione, a także w przeciwnym przypadku.

Problem	Facebook.com	nk.pl
Brak polskich liter	+	-
Zdrobnienia i różne formy imienia	-	-
Różnice w zapisie i wymowie	-	-
Literówki	+/-	-
Zapożyczenia językowe	-	-
Stosowanie myślnika (łącznika)	+	+
Emigracja	-	-
Używanie drugiego imienia	-	-
Używanie nazwiska panieńskiego	-	-

Tabela 2.1: Porównanie obsługi problemów w serwisach Facebook.com i nk.pl

2.1.8. Używanie drugiego imienia

Z różnych względów zdarza się, że osoby nie korzystają ze swojego pierwszego imienia, tylko podają drugie. Dla osób chcących je odnaleźć może to stanowić problem, ponieważ w serwisie społecznościowym mogą mieć podane pierwsze imię.

W przypadku gdy użytkownik poda swoje dwa imiona wyszukiwarki wbudowane w serwisy Facebook.com i nk.pl odnajdują bez problemów daną osobę. Niestety w przypadku podania tylko jednego imienia może okazać się to bardzo uciążliwe lub wręcz niemożliwe, np. przy bardzo popularnym nazwisku.

2.1.9. Używanie nazwiska panieńskiego

Kobiety z różnych powodów w sieci mogą figurować pod nazwiskiem panieńskim. Podobnie jak w przypadku drugiego imienia, gdy w serwisie podane są oba nazwiska, nie ma problemu z odnalezieniem konkretnej osoby. Trudności pojawiają się, gdy próbujemy odnaleźć kogoś pod zupełnie innym nazwiskiem, niż to podane w portalu. Sytuacja jest tutaj dużo trudniejsza niż w przypadku imienia, gdzie można spróbować wyszukiwania jedynie po samym nazwisku, gdyż praktycznie niemożliwe jest odnalezienie kogoś po samym imieniu.

2.2. Podsumowanie

Celem tego rozdziału było przeanalizowanie problemów, jakie mogą wystąpić przy próbie wyszukiwania osób w serwisach społecznościowych. Większość ze zbadanych problemów nie jest obecnie rozwiązana w wyszukiwarkach serwisów Facebook.com i nk.pl. Tabela poniżej przedstawia porównanie problemów w obu portalach społecznościowych. Zadowalające wyniki zostały oznaczone znakiem „+”, częściowo zadowalające – znakiem „+/-”, a niezadowalające – znakiem „-”.

W dalszej części pracy zostaną przedstawione rozwiązania mające na celu zniwelowanie problemów różnic w zapisie i wymowie, zdrobnień i różnych form imion oraz literówek przy korzystaniu z wyszukiwarek serwisów społecznościowych Facebook.com oraz nk.pl.

Rozdział 3

Omówienie wybranych algorytmów fonetycznych

Na potrzeby niniejszej pracy przyjęto następującą definicje algorytmu fonetycznego (za [2] i [7])[1]:

Definicja 3.0.1 Algorytm fonetyczny *jest to algorytm służący do indeksowania słów na podstawie ich wymowy. Litery lub grupy liter kodowane są za pomocą znaków alfanumerycznych.*

Algorytmy fonetyczne wykorzystywane są przy spisach ludności, badaniach genealogicznych a także w wyszukiwarkach. Umożliwiają porównywanie i wyszukiwanie wyrazów, które brzmią tak samo lub bardzo podobnie, a różnią się w zapisie. Większość z nich została stworzona na potrzeby anglojęzycznych wyrazów np. Soundex[11], Metaphone[6], New York State Identification and Intelligence System[7]. Istnieją także algorytmy opracowane pod kątem innych języków, np. Daitch-Mokotoff Soundex[4] dla żydowskich nazwisk, czy Kölner Phonetik dla języka niemieckiego[5]. Najpopularniejsze z nich posiadają gotowe implementacje w językach programowania i systemach bazodanowych. W poniższym rozdziale opisane zostaną trzy z nich.

3.1. Algorytm Soundex

Soundex jest jednym z najstarszych znanych algorytmów fonetycznych. Został opatentowany w latach 1918 i 1922. Zaprojektowany został, by móc porównywać anglojęzyczne nazwiska, które brzmią tak samo, a różnią się w zapisie. Dzięki swojej prostocie i niskiej złożoności obliczeniowej zyskał dużą popularność. Wbudowaną implementację algorytmu Soundex posiadają najpopularniejsze systemy bazodanowe (m.in. MySQL, PostgreSQL, MS SQL Server, Oracle) oraz języki programowania (PHP).

3.1.1. Opis działania algorytmu

Algorytm działa zgodnie z poniższym schematem:

1. Pierwsza litera wyrazu zostaje pierwszym wyrazem kodu

[1] Tłumaczenie własne

Liczba	Litera
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R

Tabela 3.1: Lista kodów Soundex

2. Spśród pozostałych liter usuwane są: *A*, *E*, *H*, *I*, *O*, *U*, *W*, *Y*
3. Pozostałym literom przypisuje się następujące liczby:
4. Spośród kolejnych wystąpień tego samego kodu usuwane są wszystkie poza pierwszym. W przypadku, gdy samogłoska rozdziela litery o tym samym kodzie, to kodowana jest ta po samogłosce.
5. Jeżeli pozostaje więcej niż trzy cyfry, to następne są usuwane. Jeżeli jest ich mniej niż trzy, to na końcu dodawane są zera.
6. Kodem Soundex wyrazu jest jego pierwsza litera i trzy uzyskane powyżej cyfry.

3.1.2. Przykłady działania algorytmu dla angielskich nazwisk

Przebieg działania algorytmu Soundex dla jednego z najpopularniejszych nazwisk w USA: *Williams* wygląda następująco[2]:

1. Pierwszym wyrazem kodu zostaje pierwsza litera nazwiska: **W**illiams.
2. Spośród pozostałych liter usuwane są: *A, E, H, I, O, U, W, Y*: W~~i~~ll~~ia~~ms = **Wllms**.
3. Zgodnie z Tabelą 3.1 literom przypisywane są odpowiednie cyfry: l – 4, m – 5, s – 2. Wllms = **W4452**.
4. Usuwane są podwójne wystąpienia tego samego kodu: W44**5**2 = **W452**.
5. Kod Soundex dla nazwiska *Williams* to **W452**.

Kody Soundex dla innych nazwisk występujących w USA:

- Willens – W452
- Thomas – T520
- Tomas – T520
- Smith – S530
- Smyth – S530
- Taylor – T460

[2]http://www.zgapa.pl/zgapedia/Najpopularniejsze_nazwiska_na_świecie.html

- Tullar – T460
- Davis – D120
- Davies – D120

3.1.3. Przykłady działania algorytmu dla polskich nazwisk

Przebieg działania algorytmu Soundex dla jednego z najpopularniejszych nazwisk w Polsce[3]: *Zawadzki* wygląda następująco:

1. Pierwszym wyrazem kodu zostaje pierwsza litera nazwiska: Zawadzki
2. Spośród pozostałych liter usuwane są: A, E, H, I, O, U, W, Y: Z~~awa~~dzk**i** = **Zdzk**
3. Zgodnie z Tabelą 3.1 literom przypisywane są odpowiednie cyfry: d – 3, z – 2, k – 2. Zdzk = **Z322**.
4. Usuwane są podwójne wystąpienia tego samego kodu: Z32~~2~~ = **Z32**.
5. W kodzie pozostały mniej niż trzy cyfry więc na końcu dopisywane jest 0: **Z320**.
6. Kodem Soundex dla nazwiska Zawadzki jest **Z320**.

Kody Soundex dla innych nazwisk występujących w Polsce:

- Zawacki – Z200
- Dąbrowski – D162
- Dombroski – D516
- Przybylski – P621
- Pszybylski – P214
- Mazur – M260
- Mazór – M260
- Zakrzewski – Z262
- Zakżewski – Z220

3.2. Algorytm Daitch-Mokotoff Soundex

Ulepszoną wersją algorytmu Soundex jest algorytm Daitch-Mokotoff Soundex. Został on stworzony przez dwóch żydowskich genealogów w 1985 r. ze względu na niską dokładność algorytmu Soundex dla słowiańskich i żydowskich nazwisk.

W odróżnieniu od wcześniej omówionego algorytmu D-M Soundex koduje wyraz jako sześć cyfr a nie jako literę i trzy cyfry. Dla każdego wyrazu może istnieć kilka różnych kodów, gdyż pewne kombinacje liter posiadają kilka alternatywnych kodowań.

[3] http://www.futrega.org/etc/nazwiska.html

3.2.1. Opis działania algorytmu

Daitch-Mokotoff Soundex działa według poniższego schematu:

1. Wyrazy kodowane są za pomocą sześciu cyfr. Każdej literze lub grupie liter odpowiada cyfra z Tabeli 3.2 W przypadku *CH, CK, C, J, RZ i RS* kodowane są dwa alternatywne przypadki.

Tabela 3.2: Lista kodów Daitch-Mokotoff Soundex

Litery	Alternatywa	Początek nazwiska	Przed samogłoską	Pozostałe przypadki
NK - nie jest kodowane				
AI	AJ, AY	0	1	NK
AU		0	7	NK
A		0	NK	NK
B		7	7	7
CHS		5	54	54
CH	Użyj KH (5) i TCH (4)			
CK	Użyj K (5) i TSK (45)			
CZ	CS, CSZ, CZS	4	4	4
C	Użyj K (5) i TZ (4)			
DRZ	DRS	4	4	4
DS	DSH, DSZ	4	4	4
DZ		4	4	4
D	DT	3	3	3
EI	EJ, EY	0	1	NK
EU		1	1	NK
E		0	NK	NK
FB		7	7	7
F		7	7	7
G		5	5	5
H		5	5	NK
IA	IE, IO, IU	1	NK	NK
I		0	NK	NK
J	Użyj Y (1) i DZH (4)			
KS		5	54	54
KH		5	5	5
K		5	5	5
L		8	8	8
MN				66
M		6	6	6
NM				66
N		6	6	6
OI	OJ, OY	0	1	NK
O		0	NK	NK
P	PF, PH	7	7	7
Q		5	5	5
RZ, RS	Użyj RTZ (94) i ZH (4)			
R		9	9	9

Tabela 3.2 – *Kontynuacja*

Litery	Alternatywa	Początek nazwiska	Przed samogłoską	Pozostałe przypadki
SCHTSCH	SCHTSH, SCHTCH	2	4	4
SCH		4	4	4
SHTCH	SHCH, SHTSH	2	4	4
SHT	SCHT, SCHD	2	43	43
SH		4	4	4
STCH	STSCH, SC	2	4	4
STRZ	STRS, STSH	2	4	4
ST		2	43	43
SZCZ	SZCS	2	4	4
SZT	SHD, SZD, SD	2	43	43
SZ		4	4	4
S		4	4	4
TCH	TTCH, TTSCH	4	4	4
TH		3	3	3
TRZ	TRS	4	4	4
TSCH	TSH	4	4	4
TS	TTS, TTSZ, TC	4	4	4
TZ	TTZ, TZS, TSZ	4	4	4
T		3	3	3
UI	UJ, UY	0	1	NK
U	UE	0	NK	NK
V		7	7	7
W		7	7	7
X		5	54	54
Y		1	NK	NK
ZDZ	ZDZH, ZHDZH	2	4	4
ZD	ZHD	2	43	43
ZH	ZS, ZSCH, ZSH	4	4	4
Z		4	4	4

2. Litery *A, E, I, J, O, U, Y* są kodowane tylko na początku wyrazu. W pozostałych przypadkach są ignorowane za wyjątkiem przypadku, gdy dwa z nich tworzą parę występującą przed samogłoską

3. Jeżeli sąsiadujące litery mogą tworzy kodowane ciągi różnej długości, wtedy wybierany jest kod najdłuższego ciągu.

4. W przypadku, gdy sąsiadujące litery mają taki sam kod używany jest tylko jeden z nich. Wyjątek stanowią tutaj textitMN i NM, których litery kodowane są oddzielnie.

5. Jeżeli na wejściu pojawia się więcej niż jeden wyraz, ciągi są łączone i traktowane jako jeden wyraz.

6. Jeżeli pozostaje mniej niż sześć cyfr, to na końcu dodawane są zera.

3.2.2. Przykłady działania algorytmu dla żydowskich nazwisk

Przebieg działania algorytmu Daitch-Mokotoff Soundex dla jednego z najpopularniejszych nazwisk w USA *Williams* wygląda następująco:

1. Zgodnie z Tabelą 3.2 literom przypisywane są odpowiednie kody: W – 7, I – nie jest kodowane, L – 8, L – 8, I – nie jest kodowane, A – nie jest kodowane, M – 6, S – 4 = **78864**
2. Usuwane są podwójne wystąpienie tego samego kodu obok siebie: 788**8**64 = **7864**
3. W wygenerowanym kodzie jest mniej niż sześć cyfr, więc na końcu dopisywane są zera: **786400**
4. Kodem Daitch-Mokotoff Soundex dla nazwiska Williams jest **786400**

Kody Daitch-Mokotoff Soundex dla innych nazwisk występujących w USA:

- Willens – 786400
- Thomas – 364000
- Tomas – 364000
- Smith – 463000
- Smyth – 463000
- Taylor – 389000
- Tullar – 389000
- Davis – 374000
- Davies – 374000

3.2.3. Przykłady działania algorytmu dla polskich nazwisk

Przebieg działania algorytmu Daitch-Mokotoff Soundex dla jednego z najpopularniejszych nazwisk w Polsce: *Zawadzki* wygląda następująco:

1. Zgodnie z Tabelą 3.2 literom przypisywane są odpowiednie kody: Z – 4, A – nie jest kodowane, W – 7, A – nie jest kodowane, DZ – 4, K – 5, I – nie jest kodowane = **4745**.
2. W wygenerowanym kodzie jest mniej niż sześć cyfr, więc na końcu dopisywane są 0: **474500**.
3. Kodem Daitch-Mokotoff Soundex dla nazwiska Zawadzki jest **474500**.

Dla nazwiska *Zawacki*, które posiada alternatywne wersje kodowania, algorytm działa w następujący sposób:

1. Zgodnie z Tabelą **??** literom przypisywane są odpowiednie kody: Z – 4, A – nie jest kodowana, W – 7, A – nie jest kodowana, C – kodowana jest jako dwie alternatywne wersje: K (kod 5) i TZ (kod 4), K – w pierwszym przypadku nie jest kodowana ponieważ, gdy dwie sąsiadujące litery mają ten sam kod używana jest tylko jedna z nich, natomiast w drugim przypadku litera kodowana jest jako 5.

2. W wygenerowanych kodach jest mniej niż sześć cyfr, więc na końcu dopisywane są zera: **475000** i **474500**.

3. Kodami Daitch-Mokotoff Soundex dla nazwiska Zawacki są odpowiednio kody **475000** i **474500**.

Kody Daitch-Mokotoff Soundex dla innych nazwisk występujących w Polsce:

- Dąbrowski – 379745
- Dombroski – 367945
- Przybylski – 747845, 794784
- Pszybylski – 747845
- Mazur – 649000
- Mazór – 649000
- Zakrzewski – 454745, 459474
- Zakżewski – 457450

3.3. Algorytm Metaphone

Algorytm Metaphone został opracowany przez Lawrence'a Philipsa i opublikowany w 1990 r. W odróżnieniu od algorytmu Soundex uwzględnia informacje dotyczące różnic i niespójności w angielskiej pisowni i wymowie, a także stanowi przybliżoną fonetyczną reprezentację słowa. Tak samo jak w przypadku Soundex-a istnieją wbudowane implementacje algorytmu w systemach bazodanowych i językach programowania.

Wyrazy kodowane są za pomocą 16 spółgłosek: *0 (th), B, X (sh, ch), S, K, J, T, F, H, L, M, N, P, R, W, Y*. Występują także samogłoski: *A, E, I, O, U*, ale wyłącznie na początku kodu. W odróżnieniu od dwóch wcześniej omawianych algorytmów uzyskany kod może być różnej długości.

3.3.1. Opis działania

1. Jeżeli dwie takie same litery występują obok siebie, jedna z nich jest usuwana. Wyjątek stanowi litera C

2. Jeżeli wyraz zaczyna się na *KN-, GN-, PN-, AE-, WR-* to usuwana jest pierwsza litera.

3. Jeżeli wyraz zaczyna się na *X-* to zamieniany jest na *S*, jeżeli zaczyna się na *WH-* to zamieniany jest na *W*.

4. Pozostałe znaki zamieniane są zgodnie z Tabelą 3.3:

Tabela 3.3: Lista kodów Metaphone

Litera	Kod	Uwagi
NK - nie jest kodowane		
A	A	Jeżeli występuje na początku wyrazu
	NK	W pozostałych przypadkach
B	B	Jeżeli nie występuje na końcu wyrazu po literze M
C	X	Jeżeli zawiera się w –CIA- lub –CH-
	S	Jeżeli zawiera się w –CI-, -CE-, -CY-
	NK	Jeżeli zawiera się w –SCI-, -SCE- lub –SCY-
	K	W przeciwnym przypadku, włączając w to –SCH-
D	J	Jeżeli zawiera się w –DGE-, -DGY- lub –DGI-
	T	W pozostałych przypadkach
E	E	Jeżeli występuje na początku wyrazu
	NK	W pozostałych przypadkach
F	F	-
G	NK	Jeżeli zawiera się w –GH- i nie jest na końcu lub przed samogłoską. Jeżeli zawiera się w –GN lub –GNED. Jeżeli zawiera się w –DGE-
	J	Jeżeli jest przed I, E lub Y i jeżeli nie zawiera się w GG
	K	W pozostałych przypadkach
H	NK	Jeżeli jest po samogłosce i następna nie jest samogłoska lub po –CH-, -SH-, -PH-, -TH-, -GH-
	H	W pozostałych przypadkach
I	I	Jeżeli występuje na początku wyrazu
	NK	W pozostałych przypadkach
J	J	-
K	NK	Jeżeli występuje po C
	K	W pozostałych przypadkach
L	L	-
M	M	-
N	N	-
O	O	Jeżeli występuje na początku wyrazu
	NK	W pozostałych przypadkach
P	F	Jeżeli występuje przed H
	P	W pozostałych przypadkach
Q	K	-
R	R	-
S	X	Jeżeli występuje przed H lub zawiera się w –SIO- lub –SIA-
	S	W pozostałych przypadkach

Tabela 3.3 – *Kontynuacja*

Litera	Kod	Uwagi
T	X	Jeżeli zawiera się w –TIA- lub –TIO-
	0	Jeżeli występuje przed H
	NK	Jeżeli zawiera się w –TCH-
	T	W pozostałych przypadkach
U	U	Jeżeli występuje na początku wyrazu
	NK	W pozostałych przypadkach
V	F	-
W	NK	Jeżeli nie występuje po samogłosce
	W	W pozostałych przypadkach
X	KS	-
Y	NK	Jeżeli nie występuje po samogłosce
Z	Y	Jeżeli występuje po samogłosce
Z	S	-

3.3.2. Przykłady działania algorytmu dla angielskich nazwisk

Przebieg działania algorytmu Metaphone dla jednego z najpopularniejszych nazwisk w USA: *Williams* wygląda następująco:

1. Usuwana jest jedna z dwóch takich samych liter stojących obok siebie: Wiłliams = **Wiliams**.
2. Zgodnie z Tabelą 3.3 literom przypisywane są odpowiednie kody: W – W, I – nie jest kodowana, L – L, I – nie jest kodowana, A – nie jest kodowana, M – M, S – S = **WLMS**
3. Kodem Metaphone dla nazwiska Williams jest **WLMS**.

Kody Metaphone dla innych nazwisk występujących w USA:

- Willens – WLNS
- Thomas – 0MS
- Tomas – TMS
- Smith – SM0
- Smyth – SM0
- Taylor – TLR
- Tullar – TLR
- Davis – TFS
- Davies – TFS

3.3.3. Przykłady działania algorytmu dla polskich nazwisk

Przebieg działania algorytmu Soundex dla jednego z najpopularniejszych nazwisk w Polsce: *Zawadzki* wygląda następująco:

1. Zgodnie z Tabelą 3.3 literom przypisywane są odpowiednie kody: Z – S, A – nie jest kodowana, W – W, A – nie jest kodowane, D – T, Z – S, K – K, I – nie jest kodowane = **SWTSK**.
2. Kodem Metaphone dla nazwiska Zawadzki jest **SWTSK**.

Kody Metaphone dla innych nazwisk występujących w Polsce:

- Zawacki – SWK
- Dąbrowski – TBRSK
- Dombroski – TMRSK
- Przybylski – PRBLSK
- Pszybylski – PSSBLSK
- Mazur – MSR
- Mazór – MSR
- Zakrzewski – SKRSSK
- Zakżewski – SKSK

Rozdział 4

Autorski algorytm fonetyczny dla języka polskiego - *PolishSoundex*

Przedstawione w poprzednim rozdziale algorytmy fonetyczne nie są przystosowane w pełni do polskich nazwisk. Ponadto, przedstawione algorytmy oferują kodowanie stratne przez co utracone zostają informacje o wyrazie, którego niestety nie można później odtworzyć. W przypadku poniższej pracy był to czynnik wykluczający je z użycia. Zastosowanie ich pociągałoby za sobą konieczność utworzenia bazy wszystkich użytkowników wraz z przypisanymi im kodami wygenerowanymi przez algorytmy fonetyczne. Tylko w taki sposób można byłoby porównywać je z danymi wprowadzonymi przez użytkownika. Ze względu na specyfikę serwisów społecznościowych i rozmiar danych jest to niemożliwe.

W pracach nad systemem podjęto się próby stworzenia algorytmu fonetycznego, który dostosowany byłby do potrzeb polskich użytkowników oraz wykorzystywał kodowanie bezstratne umożliwiające odtworzenie kodowanego wyrazu. Ponadto, postuluje się, aby algorytm był prosty w implementacji.

W poniższym rozdziale przedstawiony jest autorski algorytm fonetyczny dla języka polskiego PolishSoundex

4.1. Opis działania

Algorytm działa w następujący sposób:

1. Wszystkie litery wyrazu zamieniane są na małe litery.
2. Poszczególnym literom przypisuje się następujące kody:

Tabela 4.1: Lista kodów PolishSoundex

Kod	Litery	Uwagi
A1	dzki, cki, tski	-
A2	ski, wski, fski	-
B1	ą, on, om	-
C1	ę, em, en	-
D1	au, ał	-
E1	szcz	-

Tabela 4.1 – *Kontynuacja*

Kod	Litery	Uwagi
E2	icz	-
E3	cz	-
F1	ch, h	-
G1	ó, u	-
H1	yż, ysz	Jeżeli występuje na końcu nazwiska
H2	prz, psz	-
H3	rz, ż	-
I1	yb, yp	Jeżeli występuje na końcu wyrazu
J1	k, g	Jeżeli występuje na końcu wyrazu
K1	ci, ć, dź	-
L1	ś, si	Jeżeli nie występuje na końcu wyrazu
M1	ń, ni	-

4.2. Przykłady działania algorytmu dla polskich nazwisk

Przebieg działania algorytmu dla jednego z najpopularniejszych nazwisk w Polsce: *Zawadzki* wygląda następująco:

1. Wszystkie litery wyrazu zamieniane są na małe litery: **zawadzki**.
2. Zgodnie z tabelą 2 literom przypisywane są odpowiednie kody: z – z (bez zmian), a – a (bez zmian), w – w (bez zmian), a – a (bez zmian), dzki – A2 = **zawaA2**.
3. Kodem PolishSoundex dla nazwiska Zawadzki jest **zawaA2**.

Kody PolishSoundex dla innych nazwisk występujących w Polsce:

- Zawacki – zawaA2
- Dąbrowski – dB1broA1
- Dombroski – dB1broA1
- Przybylski – G2ybylA1
- Pszybylski – G2ybylA1
- Mazur – MazF1r
- Mazór – MazF1r
- Zakrzewski – ZakG1eA1
- Zakżewski – ZakG1eA1

4.3. Przykłady działania algorytmu dla anglojęzycznych nazwisk

Przebieg działania algorytmu PolishSoundex dla jednego z najpopularniejszych nazwisk w USA: *Williams* wygląda następująco:

1. Wszystkie litery wyrazu zamieniane są na małe litery: **williams**.
2. Zgodnie z tabelą 2 literom przypisywane są odpowiednie kody: w – w, i – i, l – l, l – l, i – i, a – a, m – m, s – s (wszystkie bez zmian) = **williams**.
3. Kodem PolishSoundex dla nazwiska Williams jest **williams**.

Kody PolishSoundex dla innych nazwisk występujących w USA:

- Willens – willens
- Thomas – tE1omas
- Tomas – tomas
- Smith – smitE1
- Smyth – smytE1
- Taylor – taylor
- Tullar – tF1llar
- Davis – davis
- Davies – davies

Rozdział 5

Opis implementacji systemu

Na potrzeby niniejszej pracy powstał serwis internetowy **osobomat.pl**. Za jego pomocą użytkownicy Internetu mogą skorzystać z wyszukiwarki użytkowników dwóch najpopularniejszych portali społecznościowych w Polsce – Facebook.com i nk.pl (Nasza-klasa). Skorzystanie z serwisu **osobomat.pl** nie wymaga posiadania konta w wyżej wymienionych serwisach.

W niniejszym rozdziale opisane są poszczególne etapy tworzenia systemu i wykorzystane technologie.

5.1. Baza imion i nazwisk

Pierwszym etapem budowy serwisu **osobomat.pl** było utworzenie dwóch baz danych *Names* i *Surnames* zawierających odpowiednio imiona i zdrobnienia oraz nazwiska a także odpowiadające im nazwy wygenerowane za pomocą algorytmu fonetycznego PolishSoundex. W tym celu powstały dwa crawlery **DiminutivesCrawler** i **SurnamesCrawler** napisane w języku Python wykorzystujące dodatkowe moduły MySQLdb oraz BeautifulSoup.

Zadaniem crawlera **DiminutivesCrawler** było zebranie ze strony[1] listy wszystkich imion wraz z odpowiadającymi im zdrobnieniami. Łącznie zebrane zostały **3063** imiona. W przypadku gdy co najmniej jedno ze zdrobnień było identyczne w różnych grupach, wtedy grupy te łączone były w jedną. Wybrano takie rozwiązanie ponieważ, gdy dwóm imieniom odpowiada jedno zdrobnienie, potencjalny użytkownik powinien otrzymać dane dotyczące obydwu imion a system obliczyć odpowiednie prawdopodobieństwa. Po przetworzeniu przez algorytm fonetyczny zebranych danych otrzymano **14 131** rekordów. Kolejnym krokiem było wygenerowanie wszystkich kombinacji wyrazów z pominiętymi polskimi znakami diakrytycznymi, które występowały w nich w podstawowej wersji. Wśród zebranych imion było **8179**, w których występowały polskie znaki diakrytyczne. Po wykonaniu tej czynności powstało kolejne **11 784** nowych. Łączna liczba imion wynosi **26 037**. Każdy z nich został przypisany do odpowiadającej mu grupy, tak by można było powiązać imię z właściwymi zdrobnieniami np. imiona Joanna i Asia zostały przypisane do grupy o numerze *102*. Powstało **1818** grup.

Zadaniem drugiego crawlera **SurnamesCrawler** było zebranie ze strony[2] listy najpopularniejszych nazwisk w Polsce. Łącznie zebrane zostało **20 000** nazwisk. Analogicznie jak w poprzednim przypadku, dla każdego zastosowano algorytm fonetyczny, co dało **83 673** rekordów. Wśród nich **44 869** zawierało polskie znaki diakrytyczne. Następnie wygenerowano

[1] http://mamdziecko.interia.pl/ksiega-imion
[2] http://www.futrega.org/etc/nazwiska.html

Opis	Imiona	Nazwiska
Liczba grup	1818	19 707
Liczba zebranych danych	3063	20 000
Liczba rekordów po zastosowaniu algorytmu fonetycznego	14 131	83 673
Łączna liczba rekordów	26 037	144 218
Razem	**170 255**	

Tabela 5.1: Podsumowanie zebranych danych

wszystkie kombinacje bez polskich znaków diakrytycznych. Efektem końcowym jest baza zawierająca **144 218** rekordów. Każdy z nich został przypisany do właściwej grupy np. nazwiska Nowakowski i Nowakoski zostały przypisane do grupy *21*. Takie rozwiązanie pozwala wiązać ze sobą oryginalne nazwiska i te wygenerowane przez algorytm fonetyczny. Jest także niezbędne do obliczania wag (prawdopodobieństw). Powstało **19 707 grup**.

Drugim etapem było zebranie z Internetu danych umożliwiających obliczanie wag poszczególnych imion i nazwisk. W tym celu powstał crawler **WeightChecker** napisany w języku Python. Za pomocą wyszukiwarki Google (wyszukiwanie dokładne dla języka polskiego) zebrane zostały informacje o ilości wyników zwracanych dla każdego imienia i nazwiska.

5.2. Wyszukiwarka

Ostatnim etapem było utworzenie warstwy prezentacyjnej serwisu przeznaczonej dla użytkowników. Do tego celu wybrany został język skryptowy PHP z zainstalowanym dodatkiem PDO do obsługi bazy danych.
Logotyp serwisu powstał przy pomocy *Web2.0 Title Generator*.[3]

Rysunek 5.1: Strona główna serwisu osobomat.pl

[3]http://www.webestools.com/web20-title-generator-logo-title-maker-online-web20-effect-reflect-free-photoshop.html

W celu odnalezienia danej osoby użytkownik musi podać imię i nazwisko a także wybrać społeczność wśród której chce szukać. Dostępne są opcje:

- Nk.pl (Nasza-klasa)
- Facebook.com
- Wszystkie – umożliwiająca wyszukanie wśród obu serwisów jednocześnie

Opcjonalnie można także podać miejscowość.

Na podstawie wprowadzonych danych w bazie zostają odszukane odpowiadające im rekordy. Gdy zostają znalezione, obliczane jest prawdopodobieństwo według którego sortowane są wyniki (od największego do najmniejszego). Wyświetlana lista ograniczona jest do 10 pierwszych wyników o niezerowym prawdopodobieństwie. W przypadku gdy dane nie zostaną odnalezione w bazie, na ich podstawie generowane są wersje przez algorytm fonetyczny. Te z kolei zostają dodane do bazy danych.

Na Rysunku 5.2 przedstawiony jest wygląd strony: wyniki wyszukiwania.

Rysunek 5.2: Prezentacja wyników wyszukiwania

Serwis umożliwia także dodanie własnych propozycji zdrobnień dla imion. Przed pojawieniem się w bazie danych muszą zostać zaakceptowane przez moderatora. Rysunek 5.3 przedstawia formularz dodawania własnych zdrobnień.

Rysunek 5.3: Formularz dodawania zdrobnień

Moderator ma możliwość akceptacji lub odrzucenia proponowanych zdrobnień. Gdy zdrobnienie zostanie zaakceptowane nastąpi jedna z poniższych akcji:

- W przypadku, gdy w bazie nie istnieje dane imię i zdrobnienie, oba zostają dodane do bazy danych i przypisane do tej samej grupy.
- W przypadku gdy w bazie istnieje imię wprowadzone przez użytkownika do bazy zostaje dodane tylko zdrobnienia i przypisane do grupy odpowiadającego mu imienia.

Rysunek 5.4 przedstawia wygląd panelu moderatora.

Rysunek 5.4: Panel moderatora

Rozdział 6

Porównanie systemu z wyszukiwarkami serwisów społecznościowych

W poniższym rozdziale przedstawione zostaną wyniki ewaluacji systemu osobomat.pl na tle dwóch pozostałych serwisów społecznościowych, czyli Facebook.com i nk.pl. Omówione także zostaną możliwości rozwoju projektu oraz jego potencjalne zastosowania.

6.1. Wyniki ewaluacji

Celem porównania działania systemu osobomat.pl z wyszukiwarkami wbudowanymi przez serwis nk.pl (Nasza-klasa) i Facebook.com przeprowadzone zostało badanie wśród 5 osób. Testerzy za zadanie otrzymali wyszukanie 5 osób, których imiona i nazwiska zawierały:

- Polskie znaki diakrytyczne lub,
- Posiadały zdrobnienia lub,
- Mogły różnić się od wymowy

Osoby te jednocześnie musiały być znane testerom, tak by sprawdzić jaka jest faktyczna skuteczność wyszukiwarek w odniesieniu do zarejestrowanych użytkowników.

Testerzy zostali poproszeni o wyszukanie każdej z pięciu osób w trzech serwisach: Nk.pl, Facebook.com oraz osobomat.pl. Dla poszczególnej osoby w każdym serwisie wykonano trzy wyszukiwania:

1. Bez polskich znaków diakrytycznych
2. Ze zdrobnieniem imienia
3. Z poprawną pisownią

Na podstawie wyżej wymienionych prób osoby testujące wystawiły ocenę w skali od 0 (najgorzej) do 5 (najlepiej) biorąc pod uwagę:

- Dokładność wyszukiwania

- Czas poświęcony na odszukanie danej osoby

Łącznie wystawionych zostało 75 ocen: po 15 przez każdego testera (po 5 na każdy serwis). Wyniki testu zostały przedstawione w tabeli 6.1.

Ocena	nk.pl	Facebook.com	osobomat.pl
Ocena 0	5	2	2
Ocena 1	1	1	0
Ocena 2	6	2	1
Ocena 3	3	3	4
Ocena 4	4	7	9
Ocena 5	6	10	9
Średnia ocena	**2,72**	**3,68**	**3,76**

Tabela 6.1: Wyniki testu

Według osób uczestniczących w badaniu serwis **osobomat.pl** prezentuje się lepiej w porównaniu z wyszukiwarkami wbudowanymi w dwa testowane serwisy społecznościowe. Zdecydowaną przewagę widać nad portalem nk.pl, który nie potrafi wyszukać osób bez podania identycznej pisowni. Użytkownicy mogą jednak jeszcze ustawić swoje pseudonimy, które często odpowiadają formie imienia bez polskich znaków lub są zdrobnieniem. Z kolei serwis Facebook.com w podstawowej wyszukiwarce na stronie głównej umożliwia wyszukiwania dynamiczne, które przy podaniu tylko części imienia lub nazwiska prezentuje wyniki. Pomimo takich możliwości testowanych portali serwis **osobomat.pl** okazał się skuteczniejszy. Autorski algorytm fonetyczny dla języka polskiego oraz zastosowane rozwiązania okazały się odpowiednie.

6.2. Dalszy rozwój projektu

Obecna wersja systemu obsługuje jedynie dwa najpopularniejsze portale społecznościowe w Polsce. Oprócz nich są także inne, które skupiają inne grupy odbiorców np. portal Goldenline.pl przeznaczony dla profesjonalistów i osób szukających pracy. Wprowadzenie obsługi dodatkowych portali rozszerzyłoby zasięg serwisu, co wpłynęłoby na jego atrakcyjność i zwiększyłoby grono potencjalnych odbiorców.

Dodatkowo należałoby dopracować algorytm fonetyczny dla języka polskiego, tak by nie generował pewnych nieprawdopodobnych przypadków a także nie był nastawiony tylko na imiona i nazwiska.

Kolejną funkcjonalnością, która poszerzyłaby możliwości serwisu osobomat.pl mogłaby stać się ocena wyników wyszukiwania przez użytkowników. Dzięki niej możliwy stałby się wzrost dokładności prezentowanych treści i generowanych kombinacji.

Wszystkie wymienione wyżej posunięcia przyczyniłby się do podstaw stworzenie kompletnej internetowej wyszukiwarki osób.

Podsumowanie

Celem pracy było przedstawienie możliwości wyszukiwarek wbudowanych w serwisy społecznościowe, zaproponowanie możliwych usprawnień a także stworzenie narzędzia, które wykorzysta te rozwiązania.

W tym celu zaszła potrzeba utworzenia algorytmu fonetycznego dla języka polskiego. Został on zaimplementowany wraz z innymi rozwiązaniami w postaci serwisu internetowego **osobomat.pl**. Dzięki niemu możliwe zostało wyszukiwanie osób, w stosunku do których nie mamy pewności co do poprawnej pisowni ich imienia i nazwiska. Możliwe jest także wyszukiwanie po zdrobnieniach, bez polskich liter a także z dwóch serwisów naraz.

Według przeprowadzonego testu serwis sprawdza się lepiej od podstawowych wyszukiwarek serwisów społecznościowych. Posiada także pewne możliwości rozwoju.

Dodatek A. Dokumentacja projektu magisterskiego

Autor: Michał Paszyn
Temat: Wspomaganie wyszukiwania osób w sieciach społecznościowych

1. **Początek**

   1.1. **Wizja projektu**

   Celem projektu jest stworzenie aplikacji internetowej – wyszukiwarki osób korzystającej z danych serwisów społecznościowych (Facebook.com, nk.pl). Umożliwi ona wyszukiwanie użytkowników, nawet gdy zapis ich nazwiska różni się od wymowy, nie zastosowano polskich znaków diakrytycznych, użyto zdrobnienia imienia lub wystąpiła literówka. Wyszukiwarka będzie zwracać użytkownikowi listę najbardziej prawdopodobnych użytkowników wraz z odnośnikami do ich profili.

   1.2. **Uzasadnienie istnienia projektu**

   Wyszukiwarki występujące w serwisach społecznościowych nie obsługują wyszukiwania, gdy zapis różni się od wymowy danego imienia i nazwiska, a użytkowników próbuje odszukać osobę według wymowy nazwiska. Nie radzą sobie dobrze także ze zdrobnieniami oraz niektórymi typami literówek. Sprawia to, że czasami znalezienie danej osoby kończy się niepowodzeniem. W ramach projektu stworzona aplikacja będzie obsługiwać ww. zjawiska.

   1.3. **Główne cele projektu**

   Głównym celem projektu jest stworzenie wyszukiwarki użytkowników wybranych serwisów społecznościowych (Facebook, nk.pl). Niezbędne do tego będzie:

   - Opracowanie i implementacja algorytmu fonetycznego dla języka polskiego
   - Stworzenie bazy zdrobnień imion polskich

   1.4. **Kluczowe rezultaty dla projektu**

   - Opracowanie i implementacja algorytmu fonetycznego dla języka polskiego
   - Wykorzystanie API serwisu Facebook oraz modułu cURL celem dostępu do wyszukiwarek serwisów społecznościowych

1.5. Członkowie zespołu oraz interesariusze

- Zespół: Michał Paszyn
- Główny zainteresowany: prof. UAM dr hab. Krzysztof Jassem

1.6. Wymagane zasoby

- Dostęp do API serwisu Facebook
- Serwer z obsługą PHP5, modułem cURL, PDO i bazą MySQL
- Komputer z zainstalowaną obsługą języka Python i modułami BeautifulSoup oraz MySQLdb

2. Planowanie

2.1. Wymagania funkcjonalne i opis zakresu projektu

Wymaganie	Zamiana imion i nazwisk na wersje fonetyczne
Opis	Imiona, ich zdrobnienia oraz nazwiska pozyskane uprzednio z Internetu zamieniane są na odpowiadające im wersje fonetyczne. Wygenerowane dane dodawane są do bazy danych.
Dane wejściowe	Imiona i nazwiska
Dane wyjściowe	Wersje fonetyczne imion i nazwisk
Warunek początkowy	Imiona i nazwiska w postaci pojedynczych rekordów
Stan końcowy	Baza danych zawiera pogrupowane imiona i nazwiska

Wymaganie	Przypisanie wag poszczególnym imionom
Opis	Poszczególnym wersjom imion i nazwisk w bazie danych, przypisywane są wagi na podstawie liczby wystąpień w Internecie uzyskanych za pomocą wyszukiwarki Google.
Dane wejściowe	Imiona i nazwiska z bazy danych.
Dane wyjściowe	Prawdopodobieństwa wystąpienia poszczególnych wyrazów.
Warunek początkowy	Baza danych zawiera pogrupowane imiona i nazwiska.
Stan końcowy	Imiona i nazwiska w bazie danych mają przypisane wagi odpowiadające prawdopodobieństwu wystąpienia w Internecie.

Wymaganie	Wyszukanie najbardziej prawdopodobnych użytkowników
Opis	Po wprowadzeniu przez użytkownika danych szukanej osoby obliczane są prawdopodobieństwa możliwych kombinacji imion i nazwisk. Na tej podstawie wyszukiwane są osoby o największym prawdopodobieństwie.
Dane wejściowe	Imię i nazwisko szukanej osoby
Dane wyjściowe	Lista najbardziej prawdopodobnych użytkowników
Warunek początkowy	Użytkownik podał imię i nazwisko szukanej osoby
Stan końcowy	Użytkownik otrzymuje listę najbardziej prawdopodobnych użytkowników serwisów społecznościowych

Wymaganie	Wyszukiwanie w dwóch serwisach społecznościowych
Opis	Użytkownik otrzymuje wyniki pochodzące jednocześnie z dwóch serwisów społecznościowych.
Dane wejściowe	Imię i nazwisko szukanej osoby.
Dane wyjściowe	Lista najbardziej prawdopodobnych użytkowników obydwu serwisów społecznościowych.
Warunek początkowy	Użytkownik podał imię i nazwisko szukanej osoby.
Stan końcowy	Użytkownik otrzymuje listę najbardziej prawdopodobnych użytkowników pochodzących z obydwu serwisów społecznościowych

Wymaganie	Dodanie zdrobnień do bazy
Opis	Użytkownik może dodać do bazy danych nowe zdrobnienia imion
Dane wejściowe	Zdrobnienie imienia.
Dane wyjściowe	Nowe zdrobnienie przypisane do imienia.
Warunek początkowy	Lista imion.
Stan końcowy	W bazie danych dodane zostaje zdrobnienie dla danego imienia (grupy)

Zakres projektu:

- Algorytm fonetyczny dla języka polskiego
- Skrypt implementujący algorytm fonetyczny dla języka polskiego
- Baza imion polskich i odpowiadających im zdrobnień

2.2. Zagrożenia oraz sposoby ich zminimalizowania

Zagrożenie nr 1: Nieukończenie projektu w terminie
Sposób zminimalizowania: Dobra organizacja pracy
Wpływ na powodzenie projektu: Wysoki

Zagrożenie nr 2: Nieukończenie algorytmu fonetycznego
Sposób zminimalizowania: Ograniczenie reguł algorytmu
Wpływ na powodzenie projektu: Wysoki

Zagrożenie nr 3: Zablokowanie dostępu do wyszukiwarek serwisów społecznościowych
Sposób zminimalizowania: Ograniczenie ilości zapytań
Wpływ na powodzenie projektu: Wysoki

Zagrożenie nr 4: Niezadawalająca szybkość wyszukiwania
Sposób zminimalizowania: Optymalizacja aplikacji i ograniczenie dokładności
Wpływ na powodzenie projektu: Średni

2.3. Plan pracy

Data	Zakres działań
4.03	Dopracowanie algorytmu fonetycznego
11.06	Dodanie obsługi zdrobnień imion
25.06	Ostateczne poprawki oraz testy aplikacji
28.08	Finalna wersja aplikacji

3. Zatwierdzenie

Projekt został zatwierdzony przez Promotora 12 września 2012 r.

4. Wykonanie

4.1. Opis przeglądu stanu projektu

Aplikacja realizuje swoje podstawowe zadania. W serwisach społecznościowych wyszukiwane są osoby na podstawie prawdopodobieństw obliczanych w oparciu o dane zawarte w bazie danych.

5. Zamykanie

5.1. Ocena użytkowników i promotora

Projekt został pozytywnie oceniony przez Promotora i użytkowników.
Tabela **??** przedstawia wyniki oceny użytkowników

Ocena	nk.pl	Facebook.com	osobomat.pl
Ocena 0	5	2	2
Ocena 1	1	1	0
Ocena 2	6	2	1
Ocena 3	3	3	4
Ocena 4	4	7	9
Ocena 5	6	10	9
Średnia ocena	**2,72**	**3,68**	**3,76**

Tabela A.1: Wyniki testu

5.2. Umiejętności nabyte podczas tworzenia projektu

Podczas tworzenia projektu nauczyłem się:

- korzystania z API serwisu Facebook
- uzyskiwania informacji ze stron za pomocą biblioteki cURL
- podstawowych pojęć z zakresu Information
- wyszukiwania artykułów naukowych
- składania tekstów w LaTeX-u

5.3. **Stan projektu (nieukończone funkcjonalności)**

Projekt został ukończony.

6. **Odbiór projektu**

Projekt został odebrany 12 września 2012 r.

Dodatek B. Kody źródłowe

Listing B.1: Kod źródłowy crawlera DiminutiveCrawler

```
1  # −*− coding: utf−8 −*−
2  #DiminutiveCrawler.py
3
4  import urllib2
5  import re
6  import time
7  import random
8  import MySQLdb
9  from PolishSoundex import PolishSoundex
10
11 class DiminutiveCrawler:
12     def __init__(self, host, user, password, database):
13         #inicjalizacja
14         self.db = MySQLdb.connect(host, user, password, database)
15         self.soundex = PolishSoundex()
16
17     def __del__(self):
18             #zamykanie polaczenia
19             self.db.close()
20
21     def crawl(self, url):
22         user_agents = ['Mozilla/5.0 (X11; U; Linux i686) Gecko/20071127 Firefox/2.0.0.11',
23         'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 \
24 (KHTML, like Gecko) Chrome/21.0.1180.79 Safari/537.1',
25         'Mozilla/5.0 (Macintosh; U; PPC; ja−JP; rv:1.0.1) Gecko/20020823 Netscape/7.0',
26         'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1',
27         'Opera/9.80 (Windows NT 6.1; WOW64; U; pl) Presto/2.10.289 Version/12.01',
28         'Opera/9.80 (X11; Linux i686; U; pl) Presto/2.8.131 Version/11.11',
29         'Mozilla/5.0 (compatible; Konqueror/4.3; Windows) KHTML/4.3.0 (like Gecko)',
30         'Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)',
31         'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)',
32         'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/534.55.3 \
33 (KHTML, like Gecko) Version/5.1.3 Safari/534.53.10']
34         naglowki = {'User−Agent' : random.choice(user_agents)}
35         try:
36             data = urllib2.Request(url, None, naglowki)
37             response = urllib2.urlopen(data)
38             self.html = response.read().decode('utf8')
39         except:
40             print u'łąBd łąpoczenia'
41
42     def get_names(self):
```

```
43          #pobieranie imienia
44          name = re.compile('<title>(.+)</title>').findall(self.html)
45          for tmp in name:
46              imie = tmp.split()
47          imie = imie[0].capitalize()
48
49          all_names = [imie]
50
51      #pobieranie zdrobnien imienia
52          diminutives = re.compile(u'(?<=<b>ZDROBNIENIA:</b> )(.+)(?=<b>INNE FORMY:</b>) \
53  |(?<=<b>ZDROBNIENIA:</b> )(.+)(?=<b>OBCE FORMY:</b>)|(?<=<b>ZDROBNIENIA:</b> ) \
54  (.+)(?=<b>Forma)|(?<=<b>ZDROBNIENIA:</b> )(.+)(?=<b> Forma)|(?<=<b>ZDROBNIENIA:</b> ) \
55  (.+)(?=</p>)').findall(self.html)
56          print diminutives
57          if diminutives:
58              for i in diminutives[0]:
59                  if len(i) > 0:
60                      tmp = i.split('.')
61              for i in tmp[0].split(','):
62                  all_names.append(i.strip())
63          else:
64              print u'Brak ńzdrobnie dla imienia', imie
65
66          #sprawdzenie w bazie danych (grupa)
67          cursor = self.db.cursor()
68          check = 0
69          new_names = []
70          for i in all_names:
71              for j in self.soundex.generate_codes(i):
72                  new_names.append(j.capitalize())
73          all_names = new_names
74
75          for i in all_names:
76              print u'ęlmi:', i
77              try:
78                  cursor.execute("set names utf8")
79                  sql = "SELECT `group` FROM `names` WHERE name='%s'" % i.encode('utf-8')
80                  cursor.execute(sql)
81                  data = cursor.fetchall()
82                  if len(data) != 0:
83                      if data[0][0] != check:
84                          check = data[0][0]
85              except:
86                  self.db.rollback()
87
88          #dodanie imion do istniejacej grupy
89          if check > 0:
90              for i in all_names:
91                  try:
92                      cursor.execute("set names utf8")
93                      sql = "INSERT INTO `names` (`group`, `name`) VALUES ('%d','%s')" % (check, i.encode('utf-8'))
94                      cursor.execute(sql)
95                      self.db.commit()
96                      print 'Dodano:', i
97                  except:
```

```
98                      self.db.rollback()
99
100         #dodanie imion do nowej grupy
101         else:
102             sql = "SELECT MAX(`group`) FROM names"
103             cursor.execute(sql)
104             max = cursor.fetchone()
105
106             if max[0] is None:
107                 max = 0
108             else:
109                 max = max[0]
110
111             for i in all_names:
112                 try:
113                     #print i.decode('utf-8')
114                     cursor.execute("set names utf8")
115                     sql = "INSERT INTO `names` (`group`, `name`) VALUES ('%d','%s')" % (max + 1, i.encode('utf-8
116                     cursor.execute(sql)
117                     self.db.commit()
118                     print 'Dodano:', i
119                     #dodanie imienia do indeksu
120                     cursor.execute("set names utf8")
121                     sql = "INSERT INTO `names_index` (`group`, `name`) VALUES ('%d','%s')" % (max + 1, imie.enco
122                     cursor.execute(sql)
123                     self.db.commit()
124                 except:
125                     self.db.rollback()
126
127 #-----------------------------------------------------------------------#
128 if __name__ == '__main__':
129     crawler = DiminutiveCrawler("host","user","password","db")
130     for i in range(263320, 266384):
131         url = 'http://mamdziecko.interia.pl/ksiega-imion/imie-michal,id,%d' % i
132         crawler.crawl(url)
133         crawler.get_names()
```

Listing B.2: Kod źródłowy crawlera SurnamesCrawler

```
# −∗− coding: utf−8 −∗−
#DiminutiveCrawler.py

import urllib2
import re
import random
import time
import MySQLdb
from BeautifulSoup import BeautifulSoup
from PolishSoundex import PolishSoundex

class SurnamesCrawler:
    def __init__(self, host, user, password, database):
        #inicjalizacja
        self.db = MySQLdb.connect(host, user, password, database)
        self.soundex = PolishSoundex()

    def __del__(self):
            #zamykanie polaczenia
            self.db.close()

    def crawl(self, url):
        user_agents = ['Mozilla/5.0 (X11; U; Linux i686) Gecko/20071127 Firefox/2.0.0.11',
        'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 \
(KHTML, like Gecko) Chrome/21.0.1180.79 Safari/537.1',
        'Mozilla/5.0 (Macintosh; U; PPC; ja−JP; rv:1.0.1) Gecko/20020823 Netscape/7.0',
        'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1',
        'Opera/9.80 (Windows NT 6.1; WOW64; U; pl) Presto/2.10.289 Version/12.01',
        'Opera/9.80 (X11; Linux i686; U; pl) Presto/2.8.131 Version/11.11',
        'Mozilla/5.0 (compatible; Konqueror/4.3; Windows) KHTML/4.3.0 (like Gecko)',
        'Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)',
        'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)',
        'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/534.55.3 \
(KHTML, like Gecko) Version/5.1.3 Safari/534.53.10']
        naglowki = {'User−Agent' : random.choice(user_agents)}
        try:
            data = urllib2.Request(url, None, naglowki)
            response = urllib2.urlopen(data)
            self.html = response.read().decode('iso−8859−2')
            self.soup = BeautifulSoup(self.html)
        except:
            print u'łąBd łąpoczenia'

    def get_surnames(self):
        #pobranie nazwisk
        all_names = self.soup.findAll('pre', text=True)[13].split()
        cursor = self.db.cursor()

        for i in range(len(all_names)):
            if i % 2:
                check = 0
                new_names = []

                #generowanie innych form
```

```
55                  for j in self.soundex.generate_codes(all_names[i]):
56                      new_names.append(j.capitalize())
57                  print new_names
58                  #sprawdzenie w bazie grupy
59                  for i in new_names:
60                      try:
61                          cursor.execute("set names utf8")
62                          sql = "SELECT `group` FROM `surnames` WHERE surname='%s'" % i.encode('utf-8')
63                          cursor.execute(sql)
64                          data = cursor.fetchall()
65                          if len(data) != 0:
66                              if data[0][0] != check:
67                                  check = data[0][0]
68                      except:
69                          self.db.rollback()
70
71                  #dodanie nazwiska do istniejacej grupy
72                  if check > 0:
73                      for i in new_names:
74                          try:
75                              cursor.execute("set names utf8")
76                              sql = "INSERT INTO surnames (`group`, `surname`) \
77  VALUES ('%d','%s')" % (check, i.encode('utf-8'))
78                              cursor.execute(sql)
79                              self.db.commit()
80                              print 'Dodano:', i
81                          except:
82                              self.db.rollback()
83
84                  #dodanie nazwiska do nowej grupy
85                  else:
86                      sql = "SELECT MAX(`group`) FROM surnames"
87                      cursor.execute(sql)
88                      max = cursor.fetchone()
89
90                      if max[0] is None:
91                          max = 0
92                      else:
93                          max = max[0]
94                      print max
95                      for i in new_names:
96                          try:
97                              cursor.execute("set names utf8")
98                              sql = "INSERT INTO surnames (`group`, `surname`) \
99  VALUES ('%d','%s')" % (max + 1, i.encode('utf-8'))
100                             cursor.execute(sql)
101                             self.db.commit()
102                             print 'Dodano:', i
103                         except:
104                             self.db.rollback()
105
106 #--------------------------------------------------------------------------#
107 if __name__ == '__main__':
108     url = 'http://www.futrega.org/etc/nazwiska.html'
109     crawler = SurnamesCrawler("host","user","password","db")
```

```
110     crawler.crawl(url)
111     crawler.get_surnames()
```

Listing B.3: Kod źródłowy crawlera WeightChecker

```
# −*− coding: utf−8 −*−
#WeightChecker.py

import urllib2
import urllib
import re
import random
import time
import urlparse
import MySQLdb
from BeautifulSoup import BeautifulSoup

class WeightChecker:
    def __init__(self, host, user, password, database):
        #inicjalizacja polaczenia
        self.db = MySQLdb.connect(host, user, password, database)

        def __del__(self):
            #zamykanie polaczenia
            self.db.close()

    def crawl(self, name):
        user_agents = ['Mozilla/5.0 (X11; U; Linux i686) Gecko/20071127 Firefox/2.0.0.11',
        'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 \
(KHTML, like Gecko) Chrome/21.0.1180.79 Safari/537.1',
        'Mozilla/5.0 (Macintosh; U; PPC; pl−PL; rv:1.0.1) Gecko/20020823 Netscape/7.0',
        'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1',
        'Opera/9.80 (Windows NT 6.1; WOW64; U; pl) Presto/2.10.289 Version/12.01',
        'Opera/9.80 (X11; Linux i686; U; pl) Presto/2.8.131 Version/11.11',
        'Mozilla/5.0 (compatible; Konqueror/4.3; Windows) KHTML/4.3.0 (like Gecko)',
        'Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)',
        'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)',
        'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/534.55.3 \
(KHTML, like Gecko) Version/5.1.3 Safari/534.53.10',
        'Mozilla/5.0 (X11; U; Linux x86_64; pl−PL) AppleWebKit/534.10 (KHTML, like Gecko) \
Ubuntu/10.10 Chromium/8.0.552.237 Chrome/8.0.552.237 Safari/534.10',
        'Mozilla/5.0 (X11; U; Linux i686; pl−PL; rv:1.9.0.8) Gecko/2009033017 GranParadiso/3.0.8',
        'Opera/9.80 (Windows NT 6.1; U; pl) Presto/2.7.62 Version/11.01',
        'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; SLCC2; \
Media Center PC 6.0; InfoPath.3; MS−RTC LM 8; Zune 4.7)',
        'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; \
.NET CLR 2.0.50727; .NET CLR 1.1.4322; InfoPath.2; .NET CLR 3.5.21022; .NET CLR 3.5.30729; \
MS−RTC LM 8; OfficeLiveConnector.1.4; OfficeLivePatch.1.3; .NET CLR 3.0.30729)']
        naglowki = {'User−Agent' : random.choice(user_agents)}
        q = u'"%s"' % name
        q = urllib.urlencode({ 'as_epq' : q.encode('utf−8') })
        url = u'https://www.google.pl/search?hl=pl&as_q=&'+q+'&as_oq=&as_eq=\
&as_nlo=&as_nhi=&lr=lang_pl&cr=&as_qdr=all&as_sitesearch=&as_occt=any&safe=\
images&as_filetype=&as_rights='
        try:
            data = urllib2.Request(url, None, naglowki)
            response = urllib2.urlopen(data)
            self.html = response.read()
            self.soup = BeautifulSoup(self.html)
```

```
55              return self.check()
56          except urllib2.HTTPError, e:
57              print 'Error:', e.code
58
59      def check(self):
60          #sprawdzenie ilosci wynikow w google
61          stats = self.soup.find('div', {'id' : 'resultStats'})
62          if stats:
63              if len(stats) > 0:
64                  stat = re.compile(u'łOkoo(.+)ówynikw').findall(stats.contents[0])
65                  if stat:
66                      return stat[0].strip().replace(',','')
67                  elif re.compile(u'óWynikw: (.+)').findall(stats.contents[0]):
68                      return re.compile(u'óWynikw: (.+)').findall(stats.contents[0])[0].strip().replace(',','')
69                  else:
70                      return re.compile(u'(.+) wynik').findall(stats.contents[0])[0].strip().replace(',','')
71              else:
72                  return 0
73
74          elif self.soup.find('p', {'style' : 'padding-top:.33em'}):
75              stat = self.soup.find('p', {'style' : 'padding-top:.33em'}).contents[0]
76              return 0
77          else:
78              return 0
79
80      def get_weights(self):
81          #sprawdzenie ilosci grup
82          cursor = self.db.cursor()
83          try:
84              cursor.execute("set names utf8")
85              sql = "SELECT `id`,`surname` FROM `surnames`"
86              cursor.execute(sql)
87              list = cursor.fetchall()
88          except:
89              self.db.rollback()
90
91          weight = []
92          all = 0
93
94          print len(list)
95          for name in list:
96              time.sleep(10)
97              ile = int(self.crawl(name[1].decode('utf-8')))
98              cursor.execute("set names utf8")
99              sql = "UPDATE `surnames` SET `weight`='%d' WHERE `id`='%d'" % (ile, name[0])
100             cursor.execute(sql)
101             self.db.commit()
102             print name[0], name[1].decode('utf-8'), ile
103
104 #--------------------------------------------------------------------------------#
105 if __name__ == '__main__':
106     crawler = WeightChecker("host","user","password","db")
107     crawler.get_weights()
```

Listing B.4: Implementacja algorytmu PolishSoundex w języku Python

```
# −*− coding: utf−8 −*−
#PolishSoundex.py

class PolishSoundex:
    #PolishSoundex
    def __init__(self):
        self.kody = ['A1','A1','A1','A2','A2','A2','B1','B1','B1','C1','C1','C1','D1','D1',
'E1','E2','E3','F1','F1','G1','G1','H1','H1','H2','H2','H3','H3','I1','I1','J1','J1','K1','K1','K1','L1','L1','M1','M1']
        self.znaki = ['dzki','cki','tski','wski','fski','ski',u'ą','on','om',u'ę','em','en','au',u'ła',
'szcz','icz','cz','ch','h',u'ó','u',u'ży','ysz','prz','psz','rz',u'ż','yb','yp','k','g','ci',u'ć',u'źd',u'ś','si',u'ń','ni']

    def replace_last(self, string, old, new):
        pos = string.rfind(old)
        string = string[0:pos] + new
        return string

    def replace_without_last(self, string, old, new):
        pos = string.rfind(old)
        string = string[0:pos].replace(old, new, pos) + old
        return string

    def name2code(self, name):
        name = name.lower()
        for i in range(len(self.znaki)):
            if i >= 21 and i <= 22:
                if self.znaki[i] == u'ży' and name[−2:] == u'ży':
                    name = self.replace_last(name, self.znaki[i], self.kody[i])
                elif self.znaki[i] == 'ysz' and name[−3:] == 'ysz':
                    name = self.replace_last(name, self.znaki[i], self.kody[i])
            elif i >= 27 and i <= 30:
                if self.znaki[i] == 'yb' and name[−2:] == 'yb':
                    name = self.replace_last(name, self.znaki[i], self.kody[i])
                elif self.znaki[i] == 'yp' and name[−2:] == 'yp':
                    name = self.replace_last(name, self.znaki[i], self.kody[i])
                elif self.znaki[i] == 'k' and name[−1:] == 'k':
                    name = self.replace_last(name, self.znaki[i], self.kody[i])
                elif self.znaki[i] == 'g' and name[−1:] == 'g':
                    name = self.replace_last(name, self.znaki[i], self.kody[i])
            elif i == 34:
                if self.znaki[i] == u'ś' and name[−1:] == u'ś':
                    name = self.replace_without_last(name, self.znaki[i], self.kody[i])
                else:
                    name = name.replace(self.znaki[i], self.kody[i])
            elif i == 35:
                if self.znaki[i] == 'si' and name[−2:] == 'si':
                    name = self.replace_without_last(name, self.znaki[i], self.kody[i])
                else:
                    name = name.replace(self.znaki[i], self.kody[i])
            else:
                name = name.replace(self.znaki[i], self.kody[i])
        return name

    def code2name(self, code):
        #code musi byc przekazany jako lista
```

```
55          for i in range(len(code)):
56              for j in range(len(self.kody)):
57                  code.append(code[i].replace(self.kody[j], self.znaki[j]))
58          code = list(set(code))
59          code.sort()
60
61          tmp = []
62          for i in range(len(code)):
63              for j in range(len(self.kody)):
64                  if self.kody[j] in code[i]:
65                      tmp.append(code[i])
66          tmp = list(set(tmp))
67          tmp.sort()
68
69          if tmp == code:
70              return self.code2name(code)
71          else:
72              code = list(set(code)-set(tmp))
73              return code
74
75      def generate_codes(self, name):
76          return self.code2name([self.name2code(name)])
```

Bibliografia

[1] Boyd, D. M. & Ellison, N. B., *Social Network Sites: Definition, History and Scholarship*, Journal of Computer-Mediated Communication, 13(1), 2007.

[2] Dictionary of Algorithms and Data Structures, *Phonetic Coding*, http://xlinux.nist.gov/dads/HTML/phoneticCoding.html

[3] *Facebook API manual*, http://developers.facebook.com/docs/guides/web/

[4] JewishGen, *Daitch-Mokotoff algorithm*, http://www.jewishgen.org/infofiles/soundex.html

[5] Wikipedia, *Kölner Phonetik*, http://de.wikipedia.org/wiki/Kölner_Phonetik

[6] Wikipedia, *Metaphone algorithm*, http://en.wikipedia.org/wiki/Metaphone

[7] Taft R.L., *Name Search Techniques*, New York State Identification and Intelligence System

[8] Wikipedia, *Phonetic algorithm*, http://en.wikipedia.org/wiki/Phonetic_algorithm

[9] *PHP Manual*, http://php.net/manual/pl/index.php

[10] *Python Manual*, http://docs.python.org/

[11] Russell R. and Odell M., *Soundex*, Patent 01 261 167, 1918.