Zadanie laboratoryjne nr 2

Historyczny Facebook, czyli od Arystotelesa do Einsteina

Celem zadania jest stworzenie mechanizmu wspomagającego import danych do różnych ontologii (samo słowo *ontologia* może być dla wielu nowe i nie do końca zrozumiałe; jego znaczenie powinno stać się odrobinę jaśniejsze po lekturze tej specyfikacji). Skoncentrujemy się tutaj na budowie pewnej szczególnej, uproszczonej ontologii zawierającej dane postaci historycznych. Źródłem tych danych powinny być artykuły poświęcone osobom w angielskiej wersji Wikipedii. W naturalny sposób wewnętrzne odnośniki Wikipedii tworzą sieć powiązań między artykułami. Istotą zadania będzie przyjrzenie się bliżej zbudowanemu w ten sposób grafowi połączeń dla postaci historycznych.

W uproszczeniu zadanie polegać więc będzie na zaimplementowaniu narzędzia, które sklasyfikuje artykuły z przygotowanego podzbioru Wikipedii i rozstrzygnie czy dany artykuł poświęcony jest osobie. Następnie, dla każdego takiego artykułu, należy pobrać nazwy artykułów osób z nią powiązanych. Jesteśmy zainteresowani wykonywaniem prostych zapytań eksplorujących wspomniany graf połączeń między osobami. Chcielibyśmy na przykład poznać odpowiedź na pytanie ile "uścisków dłoni" dzieli Arystotelesa od Einsteina i jak wygląda najkrótsza ścieżka między tymi dwiema postaciami.

Zadanie inspirowane jest projektem DBPedia, którego główny cel stanowi udostępnienie danych elektronicznej encyklopedii, ale w ustrukturyzowanej formie - "przyjaznej" dla programów automatycznie przetwarzających takie dane. My skupimy się jednak jedynie na części klasyfikującej artykuły i tworzącej byty należące do ontologii, zwane dalej *encjami* (encja może np. reprezentować osobę). Do minimum zredukujemy natomiast część związaną z wyodrębnianiem *metadanych* tych encji na podstawie artykułów (np. z treści artykułu można wyekstrahować też datę i miejsce urodzenia danej osoby; my ograniczymy się tylko do znalezienia powiązań tej osoby z innymi). Zupełnie pomijamy kwestię zapisu ustrukturyzowanych danych, czyli encji, w pamięci nieulotnej. Wprawdzie zbudowanie takiej trwałej bazy danych jest w zasadzie wpisane w definicję DBPedii, ale my dla uproszczenia przyjmujemy, że encje będą istnieć jedynie w pamięci programu. Architektura zaimplementowanego rozwiązania nie powinna jednak wykluczać możliwości jego rozszerzenia przez użytkowników o te funkcjonalności.

Co trzeba zaimplementować?

Należy zaprojektować i zaimplementować bibliotekę umożliwiającą importowanie artykułów z pliku będącego statycznym zrzutem zawartości Wikipedii (lub jej fragmentem). Częścią biblioteki powinien być również system klasyfikatorów decydujących czy dany artykuł powinien być reprezentowany przez obiekt(y) encji czy pominięty. Ponadto biblioteka powinna udostępniać funkcjonalność wybrania pewnych danych z takiego artykułu (np.

tytułów artykułów z nim powiązanych) - implementacja tej operacji należy do użytkownika biblioteki.

Dodatkowo należy udostępnić program `WikiImporter.java` prezentujący możliwości biblioteki. Jego działanie można podzielić na dwie fazy.

W fazie pierwszej zadaniem programu jest wykonanie importu (z podanego przez użytkownika pliku) tych artykułów, które odpowiadają osobom (w tym celu należy stworzyć odpowiednią implementację klasyfikatora artykułów). W wyniku otrzymujemy skierowany graf powiązanych ze sobą encji odpowiadających osobom. Jako metadane każdej takiej encji pobierane są: nazwy artykułów osób powiązanych z daną osobą (indukujące graf) oraz nazwy kategorii, do których należy artykuł o tej osobie (więcej o tym w sekcji "Wikipedia i klasyfikacja artykułów").

Program `WikiImporter` przyjmuje w linii poleceń jeden parametr - ścieżkę do pliku XML (patrz: sekcja "Format danych wejściowych" poniżej) zawierającego artykuły z Wikipedii, np. może być wywołany w ten sposób:

```
java WikiImporter enwiki-20140402-pages-articles-mini.xml
```

Po zakończeniu fazy pierwszej program wypisuje odpowiedni komunikat informujący ile encji zostało zaimportowanych. Następnie wczytuje ze standardowego wejścia zapytania użytkownika. Każde zapytanie podane jest w osobnej linii, składa się z 3 napisów rozdzielonych pojedynczymi spacjami, tzn. zapisane jest w formacie:

```
<filtr> <osoba_1> <osoba_2>
```

gdzie

1. `<filtr>` określa metodę filtrowania grafu powiązań; parametr ten może przyjąć jedną z dwóch wartości: `all` albo `physicist`;
2. `<osoba_1>` oraz `<osoba_2>` są nazwami artykułów dwóch osób.

Dla każdego takiego zapytania program powinien wypisywać na standardowe wyjście długość najkrótszej ścieżki między wierzchołkiem `<osoba_1>` a wierzchołkiem `<osoba_2>` (o ile te osoby te są reprezentowane w grafie; w przypadku braku choć jednej z nich należy wypisać stosowny komunikat) i nazwy artykułów kolejnych osób na tej ścieżce.

Wczytanie znacznika końca pliku zamiast poprawnego zapytania powinno powodować zakończenie programu.

Wymagane w zadaniu przeszukiwanie grafu może być zaimplementowane dowolną metodą (np. BFS z jednym wierzchołkiem startowym), przy czym proces ten powinien odbywać się na grafie, który został dodatkowo przefiltrowany. I tak: w przypadku, gdy parametr `<filtr>` w zapytaniu ma wartość `all` przeszukiwany jest cały oryginalny graf powiązań. Natomiast jeśli parametr ten został ustawiony na wartość `physicist` to żądamy, by znaleziona ścieżka przechodziła tylko przez osoby, które były fizykami. Pozostałe wierzchołki nie powinny być rozpatrywane. To czy dana osoba zajmowała się fizyką rozstrzygane jest na podstawie nazw

kategorii, do których przypisany był artykuł Wikipedii odpowiadający tej osobie (patrz: sekcja "Wikipedia i klasyfikacja artykułów" poniżej).

Interakcja między użytkownikiem i programem uruchomionym na pliku z testowymi danymi (patrz: sekcja "Format danych wejściowych" poniżej) mogłaby więc wyglądać następująco:

```
all Aristotle Albert_Einstein
***
Path length: 2
Aristotle
Immanuel Kant
Albert Einstein
***
all Marie_Curie Carl_Friedrich_Gauss
***
Path length: 3
Marie Curie
Kazimierz Żorawski
Marian Rejewski
Carl Friedrich Gauss
***
physicist Marie_Curie Carl_Friedrich_Gauss
***
Path length: 4
Marie Curie
Henri Poincaré
Isaac Newton
Pierre-Simon Laplace
Carl Friedrich Gauss
***
```

Znaki _ w tytułach artykułów podawanych jako parametry `<osoba_1>` oraz `<osoba_2>` należy zastąpić pojedynczą spacją. Wszystkie dopasowania napisów przy konstruowaniu grafu powiązań i przeszukiwaniu go powinny się odbywać *bez* uwzględniania wielkości liter. W tym celu przed wykonaniem operacji porównania najlepiej przekształcić napisy za pomocą metody `toLowerCase()` w klasie `String`. Jeśli zarówno parametr zapytania jak i tytuł artykułu Wikipedii zostaną przekształcone w ten sposób to przyjmujemy, że znalezienie w grafie danej osoby jest możliwe tylko wtedy, gdy oba napisy są *identyczne* - nie przewidujemy częściowych dopasowań.

Decyzja o tym jak ma wyglądać podział na konkretne klasy, powiązania między nimi, jakie sygnatury będą mieć metody, etc. należy w całości do autora biblioteki/programu. Szczególną uwagę należy jednak zwrócić na elastyczność rozwiązania tak, by możliwe było wygodne modyfikowanie mechanizmu działania programu importującego dane (np. zmiana algorytmu klasyfikującego czy artykuł opisuje osobę; więcej o tym w sekcji "Wikipedia i klasyfikacja artykułów" poniżej).

Klasy powinny być podzielone na odpowiednie podpakiety będące częścią pakietu `pl.edu.mimuw.wikiontology.<nr indeksu>`, gdzie za `<nr indeksu>` należy podstawić własne dane (chodzi o numer indeksu łącznie z inicjałami, np. kd209238).

Wikipedia i klasyfikacja artykułów

Format danych wejściowych

W projekcie Wikipedia udostępnione są okresowo wykonywane statyczne zrzuty zawartości całej elektronicznej encyklopedii. Zrzuty te mają postać jednego ~~dużego~~ gigantyczneo pliku XML zawierającego treści wszystkich artykułów (bez zachowania porządku alfabetycznego). Najbardziej aktualny zrzut dostępny jest pod tym adresem[1]. Na potrzeby tego zadania przygotowany został jednak plik będący jedynie niewielkim podzbiorem tego ogromnego archiwum. Dostępny jest tutaj:

http://www.mimuw.edu.pl/~kdr/wikiontology/enwiki-20140402-pages-articles-mini.xml

Plik zawiera 150 artykułów, w tym 75 poświęconych osobom. Wizualizcję grafu powiązań między tymi osobami można obejrzeć tutaj (potrzebna wtyczka w przeglądarce umożliwiająca uruchamianie appletów Javowych):

http://www.mimuw.edu.pl/~kdr/wikiontology/

Każdy artykuł w powyższym pliku XML ujęty jest w znaczniki `<page>` oraz `</page>`. Wewnątrz tych znaczników znajdują się znaczniki `<title>` oraz `<text>` zawierające odpowiednio: tytuł artykułu oraz jego treść. Pozostałe znaczniki występujące w pliku są nieistotne.

W obrębie znaczników `<text>` obowiązuje uproszczona składnia wiki - zawartość tego pola jest identyczna ze źródłem widocznym podczas edycji artykułu na stronach Wikipedii (na podstawie źródła silnik Wikipedii generuje stronę HTML dla danego artykułu). Specjalne znaczenie w składni wiki mają fragmenty ujęte w podwójne nawiasy `{{ … }}` oraz `[[ … ]]`. Te pierwsze pełnią rolę szablonów (patrz: przykład szablonu w sekcji “Klasyfikatory” poniżej), a na podstawie zawartości drugich generowane są wewnętrzne odnośniki między artykułami (np. `[[Albert Einstein|Einstein]]` zostanie przekształcone na odnośnik Einstein prowadzący do artykułu o tytule Albert Einstein).

[1] UWAGA: Archiwum jest bardzo dużych rozmiarów: skompresowane zajmuje ponad 10GB, po rozpakowaniu - prawie 48GB.

Klasyfikatory

Biblioteka powinna udostępniać możliwość zbudowania klasyfikatora rozstrzygającego czy dany artykuł powinien być dalej przetwarzany czy też zostać pominięty. Klasyfikator dokonuje "oceny" na podstawie treści artykułu i może brać pod uwagę różne kryteria.

W zadaniu (jako część programu `WikiImporter`) należy przygotować klasyfikator uwzględniający dwa kryteria "bycia osobą" (wystarczy, że jedno z nich będzie spełnione, by artykuł został zakwalifikowany jako artykuł poświęcony osobie):

1. Obecność szablonu `{{Persondata}}`. Jedna z inicjatyw projektu Wikipedia ma na celu uzupełnienie wszystkich artykułów poświęconych osobom i dodanie do ich źródeł specjalnego szablonu Persondata. Szablon ten zawiera podstawowe metadane takiej osoby. Np. w artykule poświęconym Arystotelesowi można odnaleźć następujący fragment:

```
{{Persondata
|NAME = Aristotle
|ALTERNATIVE NAMES = Ἀριστοτέλης (Greek)
|SHORT DESCRIPTION = [[Greek philosophy|Greek]] philosopher
|DATE OF BIRTH = 384 BC
|PLACE OF BIRTH = [[Stagira (ancient city)]]
|DATE OF DEATH = 322 BC
|PLACE OF DEATH = [[Chalcis]]
}}
```

   Szablon Persondata choć nie jest widoczny dla użytkownika w artykule w wersji HTML to może być przydatny w automatycznym przetwarzaniu artykułów. W zadaniu nie jest wymagane zaimplementowanie parsowania metadanych tego szablonu - wystarczy jedynie stwierdzić, że taki szablon jest obecny w źródle artykułu.

2. Przynależność artykułu do odpowiedniej kategorii. Większość artykułów opisujących osoby została uzupełniona o szablon Persondata. Dotyczy to jednak angielskiej wersji Wikipedii. Natomiast wersja polska nie posiada odpowiednika tego szablonu. Dlatego wskazane jest udostępnienie innego kryterium rozstrzygającego czy artykuł poświęcony jest osobie. Każdy artykuł na Wikipedii może zostać przypisany do wielu kategorii (widocznych w stopce artykułu w wersji HTML), czyli grupy artykułów powiązanych tematycznie. Grupy te są tworzone i nazywane niezależnie przez wikipedystów. Przypisanie artykułu do kategorii odbywa się przez dodanie odpowiedniej informacji ujętej w nawiasy `[[...]]` do źródła takiego artykułu. Przykładowo w artykule o Arystotelesie występuje następujący fragment:

```
[[Category:384 BC births]]
[[Category:322 BC deaths]]
[[Category:4th-century BC philosophers]]
[[Category:4th-century BC writers]]
[[Category:Academic philosophers]]
[[Category:Acting theorists]]
[[Category:Ancient Greek mathematicians]]
```

Racjonalnym założeniem jest przyjęcie, że artykuł należący do kategorii postaci "`<dowolny rok> births`" lub "`<dowolny prefiks> philosophers`" opisuje osobę. W zadaniu należy zaproponować kilka (ok. 5) kategorii (konkretnych lub uogólnionych tzn. np. postaci "`<dowolny rok> births`") i zaimplementować sprawdzenie czy artykuł przypisany jest do jednej z nich.

Użytkownik biblioteki powinien mieć możliwość zdefiniowania innych, własnych kryteriów i ich składanie (por. opisany wyżej łańcuch kryteriów Persondata i kategorii). Pomocnicze funkcjonalności, takie jak: wyodrębnienie ze źródła strony szablonów czy kategorii, powinny być zaimplementowane jako część biblioteki.

Encje odpowiadające osobm tworzone w programie `WikiImporter` powinny zawierać nazwy kategorii do których przypisana jest dana osoba. Przyjmujemy, że osoba ta jest fizykiem (a więc jest uwzględniania przy przeszukiwaniu grafu z parametrem `<filter>` ustawionym na wartość `physicist`) wtedy i tylko wtedy, gdy nazwa co najmniej jednej z tych kategorii zawiera napis "`physicist`" występujący w dowolnym miejscu (można tutaj skorzystać z metody `contains` w klasie `String`).

Przydatne narzędzia

Parser XML

Do przetworzenia pliku XML z zawartością Wikipedii wskazane jest użycie jednego z istniejących parserów (niepolecana jest próba stworzenia od podstaw własnego parsera). Zalecamy skorzystanie z jednej z gotowych implementacji takiego prostego parsera (SAX parser, czyli Simple API for XML) dostępnego w rozszerzeniu standardowej biblioteki klas Javy. Cenne może być zapoznanie się z krótkim tutorialem oraz jego omówienie na laboratorium.

Wyrażenia regularne

Wyrażenia regularne[2] to niezwykle przydatne narzędzie w przetwarzaniu tekstów. Ich wykorzystanie *nie* jest wymagane w rozwiązaniu tego zadania, ale mogą one okazać się pomocne np. w wyszukiwaniu wystąpień kategorii czy wewnętrznych odnośników Wikipedii w źródłach artykułów.

Standardowa implementacja wyszukiwania za pomocą wyrażeń regularnych dostępna jest w bibliotece klas Javy w pakiecie java.util.regex.

Poniższy fragment kodu wypisuje nazwy kolejnych kategorii (ujętych w nawiasy `[[...]]`) występujące w podanym tekście:

[2]Wyrażenia regularne to formalne pojęcie z teorii języków. W tym zadaniu wyrażenia regularne to po prostu sposób zapisu wzorca do wyszukiwania. Na potrzeby tego zadania wystarczy zrozumieć przykład.

```
String text = …;
Pattern pattern = Pattern.compile("\\[\\[Category:([^\\]]*)\\]\\]");
Matcher matcher = pattern.match(text);
while (matcher.find()) {
      System.out.println(matcher.group(1));
}
```

W tym przykładzie zmienna `pattern` zawiera wyszukiwany wzorzec postaci `[[Category:...]]`. Ze uwagi na fakt, iż znaki `[` oraz `]` mają specjalne znaczenie w składni wyrażeń regularnych, to każde wystąpienie jednego z tych znaków w wyszukiwanym wzorcu należy poprzedzić znakiem `\` (który również należy poprzedzić kolejnym znakiem `\` ze względu na specjalne znaczenie `\` w literałach napisowych w Javie). Fraza `[^\\]]*` pozwala na dopasowanie dowolnie długiego ciągu znaków innych niż `]`. Ujęcie tej frazy w nawiasy `(` oraz `)` umożliwia odwołanie się do dopasowanego ciągu znaków - w tym przypadku dopasowana nazwa kategorii dostępna jest jako wynik wywołania `matcher.group(1)`. Wielokrotne wywołanie `matcher.find()` powoduje iteracyjne dopasowywanie wzorca do kolejnych jego wystąpień w tekście.

Jakie założenia można przyjąć?

Można przyjąć następujące upraszczające założenia:

1. Plik XML z artykułami Wikipedii ma poprawną strukturę (poprawna struktura znaczników, każdy artykuł oznaczony odpowiednimi znacznikami w sposób opisany w sekcji “Wikipedia i klasyfikacja artykułów”; każdy artykuł posiada tytuł i treść). Nie trzeba wykonywać dodatkowej walidacji.
2. Zapytania wprowadzane w programie `WikiImporter` będą zawsze zgodne z podanym formatem.
3. Powiązania między artykułami zadane są tylko i wyłącznie przez wewnętrzne odnośniki Wikipedii, tzn. przez frazy ujęte w nawiasy `[[...]]`. Inne wystąpienia np. nazwisk osób w treści artykułu, ale poza tymi nawiasami nie powinny być rozpatrywane. Należy również pamiętać, że artykuły w przygotowanej “minimalistycznej” wersji Wikipedii mogą zawierać odnośniki do artykułów, które nie znalazły się w pliku XML - takie odnośniki można również zignorować (na dowolnym etapie, np. przy przeszukiwaniu grafu).
4. W przypadku istnienia wielu ścieżek równej długości między dwiema osobami program `WikiImporter` może wypisać dowolną z nich. W przypadku, gdy ścieżka nie istnieje program powinien wypisać stosowny komunikat.
5. Program `WikiImporter` będzie posiadał do dyspozycji wystarczającą ilość pamięci, by jednocześnie przechować *wszystkie* zaimportowane encje. Należy jednak założyć, że pamięć będzie zbyt mała, by *jednocześnie* pomieścić treść *wszystkich artykułów* (tzn. łącznie z treścią) z pliku. Wprowadzenie takiego ograniczenia ma następującą motywację. Odczyt z pliku XML jest zaimplementowany jako część biblioteki, natomiast decyzja o tym jak wygląda “cykl życia” pojedynczych encji importowanych przez bibliotekę należy do użytkownika tej biblioteki. Dążymy do tego, by biblioteka była możliwie uniwersalna, a więc powinna być przygotowana do przetwarzania

dużych zbiorów danych, w tym pełnej zawartości Wikipedii. Użytkownik biblioteki w tym zadaniu przewiduje jej użycie jedynie do tworzenia niewielkich ontologii - cały graf może więc być przechowany w pamięci. W przypadku innych zastosowań użytkownik może być zmuszony do bardziej oszczędnego gospodarowania pamięcią.

6. Możliwy jest dwuprzebiegowy import danych połączony z modyfikacją encji przechowywanych w pamięci. W pierwszym przebiegu rozstrzygane jest czy artykuł opisuje osobę, ale reprezentujący tę osobę obiekt/obiekty może zawierać wszystkie odnośniki do artykułów (niekoniecznie tylko tych reprezentujących osoby). W drugim przebiegu, na podstawie stworzonego już zbioru osób, lista tych odnośników może zostać przefiltrowana tak, by zawierała już tylko odnośniki do artykułów poświęconych osobom. Takie rozwiązanie nie jest wymagane - dopuszczalne jest przechowywanie w obiektach encji wszystkich odnośników. Algorytm realizujący przeszukiwanie grafu osób powinien jednak uwzględniać (ale i ignorować) obecność takich odnośników do artykułów, które nie są reprezentowane w grafie powiązań.

Na co zwracać uwagę, czyli uwagi końcowe

1. Program `WikiImporter` testowany będzie na wskazanym w sekcji “Format danych wejściowych” pliku ze 150 artykułami (z oczywistych względów bezcelowa byłaby próba importu do pamięci artykułów z pełnej wersji Wikipedii). Prosimy o zadbanie, by odpowiedzi programu na przykładowe zapytania wymienione w sekcji “Co trzeba zaimplementować? były zgodne z podanymi.
2. Biblioteka i programy demonstracyjne powinny być stabilne i zapewniać obsługę typowych błędów (wypisanie odpowiedniego komunikatu na standardowe wyjście diagnostyczne i kontynuowanie działania bądź, jeśli to nie jest możliwe, zakończenie). Do takich typowych błędów należą (nie jes to rzecz jasna pełna lista): brak pliku XML z artykułami na ścieżce podanej w parametrze programu `WikiImporter`, brak encji odpowiadającej podanej w zapytaniu osobie, (okazjonalne) niezgodności ze składnią wiki w obrębie pojedynczego artykułu (w takiej sytuacji dopuszczalne jest dowolne sensowne rozwiązanie, np. pominięcie takiego artykułu, bądź pobranie niekompletnych metadanych).
3. Należy unikać duplikacji kodu - klasy, które mogą być potencjalnie użyte w różnych programach importujących i przetwarzających zaimportowane dane powinny być zaimplementowane jako części biblioteki.

Możliwe jest dowolne rozszerzenie, a nawet modyfikacja niniejszej specyfikacji. Każda taka zmiana musi być jednak uzgodniona indywidualnie z prowadzącym grupę laboratoryjną.

Termin wysyłania rozwiązań upływa dnia: 9 czerwca.