Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych

Wprowadzenie do Eksploracji Danych Tekstowych w Środowisku WWW

Gromadzenie i przechowywanie przepisów kulinarnych przy użyciu ontologii

Sprawozdanie końcowe

Autorzy:
Maciej Suchecki
Michał Toporowski
Jacek Witkowski

Prowadzący:
dr inż. Piotr Andruszkiewicz

16 stycznia 2015

1 Treść zadania

Tytuł Gromadzenie i przechowywanie przepisów kulinarnych w ustrukturalizowany sposób, przy użyciu ontologii.

Opis

- ekstrakcja informacji ze stron z przepisami (Information Extraction)
- odwzorowanie wyekstrahowanych elementów na ontologię
- wstawienie instancji do ontologii/ew. modyfikacja ontologii (na poziomie pojęć, dodanie nowych pojęć)
- wykonywanie zapytań na ontologii

2 Definicja problemu

Przepisy kulinarne są dostępne w ogromnych ilościach w Internecie. Znalezienie interesujących nas przepisów jest bardzo owocne, bez względu na zdefiniowane wymagania – takie jak język przepisu, czy składniki potrzebne do jego wykonania. Z drugiej jednak strony, przepisy te są udostępniane w sposób silnie nieustruktruralizowany, a zatem poruszanie się w takim zbiorze oraz pobieranie z niego informacji staje się żmudne i trudne. Z tego powodu problem ekstrakcji wspomnianych przepisów z sieci Internet oraz ich przechowywanie w ustrukturalizowany sposób wydaje się ciekawy oraz ważny.

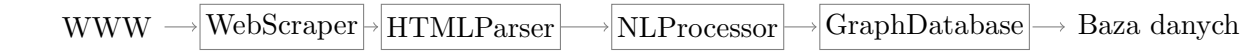
3 Opis rozwiązania

Rozwiązaniem tak postawionego problemu będzie program eksplorujący przepisy kulinarne znajdujące się w sieci Internet. Będzie on pobierał przepisy z wcześniej zdefiniowanych stron WWW. Ich lista będzie zdefiniowana w pliku tekstowym, przy czym nie będzie ona modyfikowalna przez użytkownika, z racji na konieczność definiowania różnych fragmentów kodu w zależności od struktury badanej strony. Moduły aplikacji, które będą brały udział w początkowej analizie przepisów będą silnie zależne od układu badanych stron z przepisami. Ponadto, dla każdej strony będzie zdefiniowana liczba przepisów, którą program ma z danej strony odczytać. Wartość ta będzie mogła być modyfikowana przez użytkownika.

Podczas działania programu wczytane przepisy będą poddawane działaniu procesora języka naturalnego, odwzorowywane na wcześniej zdefiniowaną ontologię, oraz wstawiane do niej jako nowe instancje. Wynikiem działania programu będzie wypełniona przepisami grafowa baza danych. Będzie można wykonywać na niej zapytania, pozwalające na ekstrakcję z niej ustrukturalizowanych danych przepisów.

4 Implementacja

Program będzie napisany w języku Python i będzie składał się z czterech modułów, widocznych na poniższym rysunku:



Opis każdego z modułów znajduje się w kolejnych podrozdziałach.

4.1 Moduł *WebScraper*

4.1.1 Wejście

Moduł będzie pobierał z pliku listę adresów stron, z których powinien pobrać przepisy – wraz z liczbą przepisów do pobrania z każdej strony.

4.1.2 Sposób działania oraz wykorzystywane biblioteki

Z pomocą bibliotek *requests* oraz *BeautifulSoup* moduł będzie pobierał cały kod HTML dotyczący przepisu, zapisując go w strukturze *RawRecipe*.

4.1.3 Wyjście

Wynikiem działania modułu *WebScraper* – przekazywanym do modułu *HTMLParser*, będzie zbiór obiektów typu *RawRecipe*, zawierających adres WWW przepisu, oraz jego kod HTML zawierający wszystkie dane jego dotyczące – takie, jak opis, nazwa, składniki itp.

4.2 Moduł *HTMLParser*

4.2.1 Wejście

Moduł będzie przyjmował zbiór obiektów typu *RawRecipe*, przekazanych mu od modułu *WebScraper*.

4.2.2 Sposób działania oraz wykorzystywane biblioteki

Za pomocą biblioteki *BeautifulSoup* – służącej do parsowania dokumentów HTML – moduł będzie starał się wyekstrahować jak najwięcej przydatnych informacji dotyczących struktury przepisu z otrzymanego kodu HTML.

4.2.3 Wyjście

Wynikiem działania modułu *HTMLParser* – przekazywanym do modułu *NLProcessor*, będzie zbiór obiektów typu *ParsedRecipe*, zawierających pola:

- **url** – pole zawierające dokładny adres URL przepisu na stronie WWW
- **name** – pole zawierające nazwę przepisu
- **ingredients** – pole zawierające listę napisów opisujących poszczególne składniki przepisu (wraz z podanymi ilościami)
- **preparation** – pole zawierające właściwy tekst przepisu – inaczej, sposób przygotowywania danej potrawy
- **additional_attributes** – opcjonalne pole, zawierające słownik dodatkowych atrybutów, takich jak: czas przygotowywania, stopień trudności itp. Możliwe klucze dla słownika będą z góry zdefiniowane w programie.

4.3 Moduł *NLProcessor*

4.3.1 Wejście

Moduł będzie przyjmował zbiór obiektów typu *ParsedRecipe*, przekazanych mu od modułu *HTMLParser*.

4.3.2 Sposób działania oraz wykorzystywane biblioteki

Za pomocą biblioteki *NLTK* moduł będzie przekształcał tekst opisujący składniki przepisu na model obiektowy.

W ramach tego procesu wykonywane będą czynności takie, jak:

- lematyzacja
- filtrowanie słów nieistotnych
- wyodrębnienie z tekstu informacji o nazwach składników i ich ilości
- zbudowanie powiązań pomiędzy krokami przepisu a wykorzystywanymi składnikami

4.3.3 Wyjście

Wynikiem działania modułu *NLProcessor* – przekazywanym do modułu *GraphDatabase*, będzie zbiór obiektów typu *ProcessedRecipe*, zawierających pola:

- **url** – pole zawierające dokładny adres URL przepisu na stronie WWW
- **name** – pole zawierające nazwę przepisu
- **additional_attributes** – opcjonalne pole, zawierające słownik dodatkowych atrybutów, takich jak: czas przygotowywania, stopień trudności itp. Możliwe klucze dla słownika będą z góry zdefiniowane w programie.
- **ingredients** – pole zawierające zbiór składników w postaci obiektów zawierających następujące pola:
  - **name** – nazwa składnika
  - **amount** – ilość składnika (obiekt zawierający wartość oraz jednostkę)
- **preparation** - pole zawierające zbiór obiektów *PreparationStep* zawierających następujące pola:
  - **text** – opis kroku
  - **ingredients** – lista składników wykorzystywanych w danym kroku

4.4 Moduł *GraphDatabase*

4.4.1 Wejście

Moduł będzie przyjmował zbiór obiektów typu *ProcessedRecipe*, przekazanych mu od modułu *NLProcessor*.

4.4.2 Sposób działania oraz wykorzystywane biblioteki

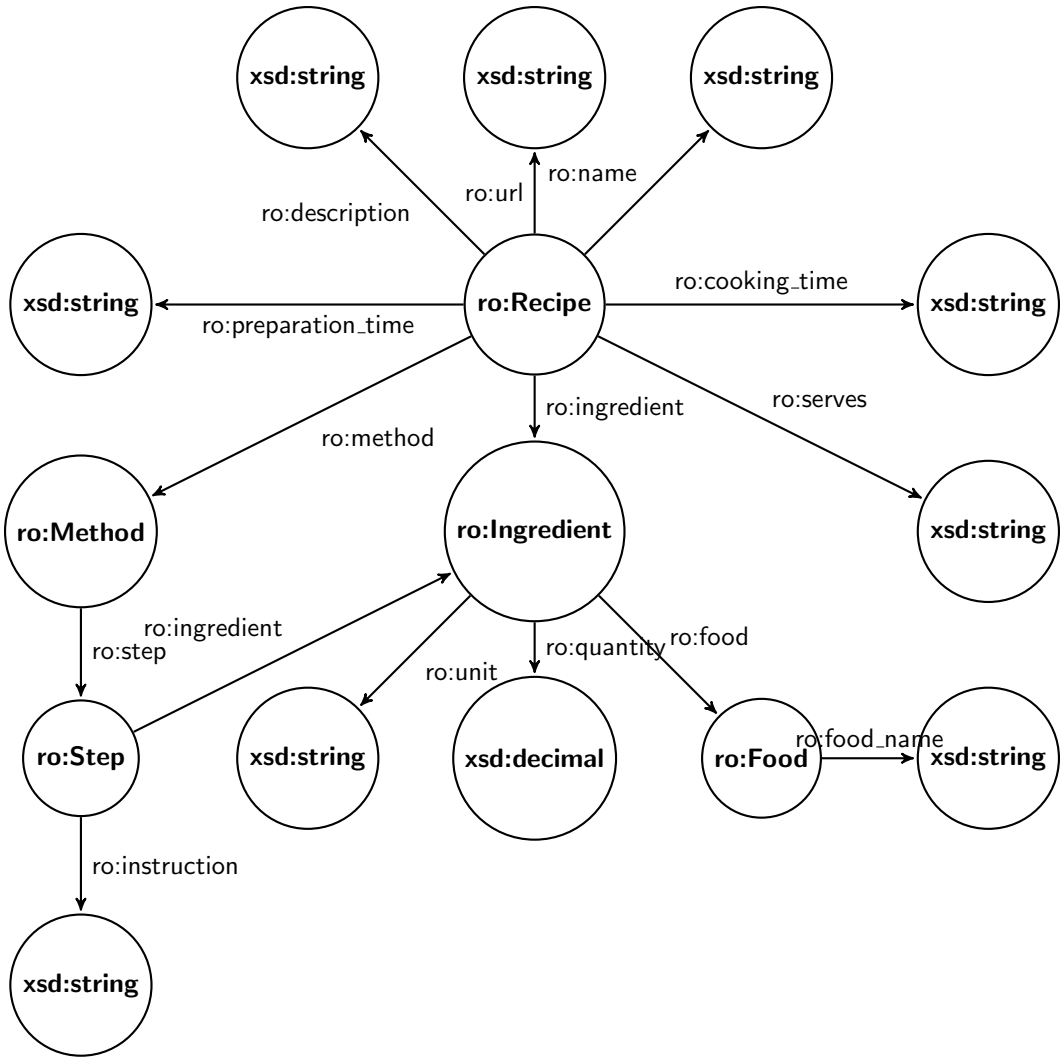
Za pomocą biblioteki *rdflib* oraz zdefiniowanej przez autorów ontologii, moduł utworzy z przepisów grafową bazę danych. Po zachowaniu wszystkich przepisów, baza danych będzie zachowana w pliku o formacie *RDF*, aby można było wykonywać na niej zapytania.

4.4.3 Wyjście

Wynikiem działania modułu będzie grafowa baza danych zapisana w pliku. Do bazy będzie można się odwoływać poprzez zapytania predefiniowane w skrypcie języka *Python*.

5 Ontologia

Na potrzeby projektu została stworzona następująca ontologia ro (Recipe Ontology):



Rysunek 1: Ontologia utworzona na potrzeby projektu.

Została ona wykorzystana do zapisywania przepisów w bazie danych. Plik z definicją ontologii w formacie *Turtle* znajduje się pod ścieżką *rextractor/db/recipes_ontology.ttl*.

6 Instrukcja obsługi

6.1 Uruchomienie programu

TODO Maciek

6.2 Wykonywanie przykładowych zapytań

TODO Maciek

Wypisanie dostępnych przepisów

Wypisanie dostępnych składników

Wypisanie przepisów zawierających dany składnik

Wypisanie przepisów, które da się zrealizować z listy podanych składników

7 Testy

7.1 Testy end-to-end

TODO Maciek i Michał
Testy end-to-end polegają na uruchomieniu całego procesu od pobrania przepisów z sieci, poprzez parsowanie i przetwarzanie języka naturalnego, aż po zapis do grafowej bazy danych.

7.2 Testy modułu *NLProcessor*

Do modułu NLProcessor zostały utworzone testy weryfikujące poprawność wydobywania danych z tekstu w języku naturalnym. Testy zostały wykonane w następujący sposób:

- jako dane wejściowe zostały przekazane obiekty typu *ParsedRecipe*
- zdefiniowano oczekiwany wynik przetworzenia danych przepisów (w postaci obiektów typu *ProcessedRecipe*)
- uruchomiono moduł NLProcessor z danymi parametrami wejściowymi i zweryfikowano, czy wynik jest zgodny z oczekiwanym.

Wynik Dla testowanych przepisów, 78% składników zostało przetworzonych całkowicie zgodnie z oczekiwaniami tj. każdy element listy składników miał identyczną nazwę, ilość oraz jednostkę, jak w danych oczekiwanych.
W pozostałych przypadkach zaobserwowano następujące niezgodności:

1. została wydobyta jedynie część nazwy składnika (np. *fillet* zamiast *salmon fillet*)
2. część nazwy składnika została sklasyfikowana jako nazwa jednostki lub vice versa
3. rozpoznano dany fragment tekstu jako opis jednego składnika, gdy w rzeczywistości zawierał opis dwóch

Przyczyny problemów:

- niepoprawna klasyfikacja części mowy przez bibliotekę NLTK w niektórych zdaniach
- płynność i różnorodność języka naturalnego utrudniająca przewidzenie struktury tekstu

Warto jednak zauważyć, że pomimo powyższych problemów, zdania o typowej strukturze są przetwarzane na ogół dobrze, zaś niezgodności typu 1. i 3. zazwyczaj nie dyskwalifikują wyniku tj. może być on w dalszym ciągu użyteczny.

7.3 Wydajność

TODO Maciek - napisać, ile wczytano przepisów, w jakim czasie i jaki był średni czas na przepis

8 Wnioski

TODO Michał i Maciek
W skrócie:

- Python świetnie się nadaje do procesowania tekstu, web scrapingu itp.

- Narzędzia z biblioteki NTLK znacznie ułatwiają przetwarzanie języka naturalnego w Pythonie, jednak nie są idealne i czasem produkują niepoprawne wyniki.
- Słabe wsparcie dla ontologii i OWL dla Pythona
- Implementacja aplikacji wydobywającej wiedzę ze stron webowych wymaga dostosowania do konkretnej strony i konkretnego stylu.
- ???

8.1 Ewentualne zastosowania aplikacji

TODO Maciek i Michał

- jako generator contentu dla strony z przepisami
- jako narzędzie dla dietetyka (wykonywanie zapytań typu znajdź przepis wykorzystujący pomidory, bo zdrowe itp.)
- jako narzędzie pomagające odpowiedzieć na pytanie "Co można zrobić ze składników znajdujących się aktualnie w lodówce?"
- ???

8.2 Rozwój aplikacji

TODO Maciek

Architektura aplikacji wspiera jej rozbudowę.
W skrócie:

- można dopisywać kolejne strony
- można zrobić WebScrapera wielowątkowo – większa wydajność
- można dopisać inny backend – np. wrzucać przepisy do zwykłej bazy relacyjnej
- można pokusić się o mechanizm zapobiegający ponownego wstawiania istniejących przepisów – wtedy można uruchamiać aplikację cyklicznie jako crawler

Ponadto, jakość kodu została zapewniona poprzez stosowanie się do standardu PEP8 oraz stosowania narzędzia pylint. Kod był regularnie analizowany tym narzędziem – gotowy projekt uzyskał ocenę

9 Podsumowanie

Praca nad projektem była dla nas ciekawa i rozwijająca, z wielu powodów. Po pierwsze, mieliśmy okazję poznać język Python – jak pisze się w nim większe aplikacje obiektowe. Poza tym, zdobyliśmy wiedzę nt. procesowania tekstu, web scrapingu oraz ontologii. TODO coś jeszcze?