Systemy Agentowe / Wstęp do Eksploracji Danych Tekstowych

Rozproszony system do pół-automatycznej budowy ontologii ze stron WWW

Dokumentacja projektu

Autorzy:
Maciej Gańko
Rafał Jagielski
Paweł Kaczyński

Opis zadania i założenia

Celem projektu jest stworzenie wielowątkowego systemu do równoległego przetwarzania tekstów witryn internetowych z zadanej domeny, na podstawie których zostaje przeprowadzony proces budowy ontologii. Ontologia jest opisem wiedzy znajdującej się na tychże stronach. Wynikiem działania programu jest plik w formacie .xml z wygenerowaną bazą wiedzy zgodnej z językiem OWL.

Środowisko

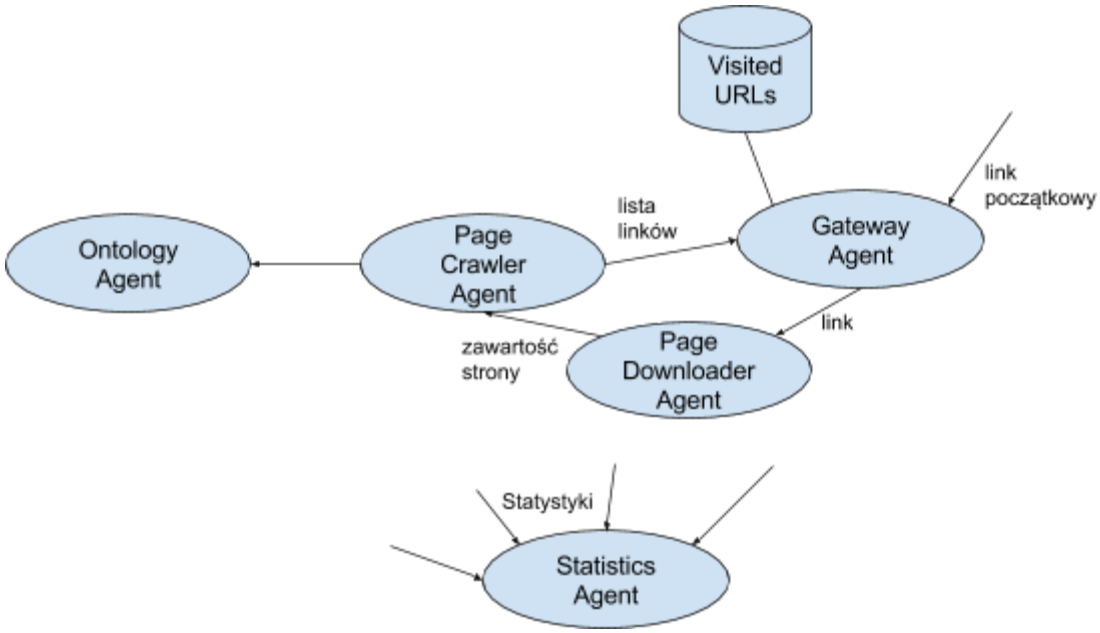
- IntelliJ IDEA + Maven
- Protege 2.0 (do wizualizacji ontologii)

Użyte biblioteki

- jade
- jsoup
- owlapi
- jmorfeusz

Architektura

Do implementacji systemu o architekturze agentowej została wybrana platforma programistyczna JAVA Agent DEvelopment Framework (JADE). Dostarcza ona wystarczającej abstrakcji systemu pozwalającej, poza przesyłaniem wiadomości, na uruchamianie poszczególnych agentów na różnych komputerach, klonowanie ich oraz wyłączanie. Umożliwia także poszukiwanie agentów realizujących określone zadanie poprzez usługę “directory facilitator”. Komunikacja między agentami jest zorganizowana tak, że poszukują oni innego agenta, który zrealizuje zlecane zadania. Wyróżnione są dwa tryby pracy agentów: niezależny od domeny oraz przypisany do domeny. Agenty niezależne od domeny (Page Crawler oraz Pade Downloader) mogą przyjmować zlecenia z dowolnych domen, więc są wybierane w pełni losowo spośród dostępnych. Takie poszukiwanie jest przeprowadzane dla każdego zlecenia osobno, więc nowe instancje poszczególnych agentów mogą pojawić się w każdej chwili odciążając już poprzednie instancje od pracy. Agenty przypisane do domeny (Gateway Agent oraz Ontology Agent) w jednej chwili mogą obsługiwać tylko konkretną domenę. Po przypisaniu do domeny zmieniają rejestrację swoich usług na takie, które jednoznacznie określają jaką domenę obsługują, dzięki czemu są jednoznacznie identyfikowane przez pozostałych. Na poniższym diagramie został przedstawiony schemat połączeń.



Opis zadań realizowanych przez poszczególnych agentów:

1. Gateway Agent. Jego zadaniem jest przyjęcie początkowego linku od użytkownika i rozpoczęcie procesu przeglądania stron. Parametrami wymaganymi są (poza urlem) ścieżka do pliku, w których zostanie zapisana ontologia oraz lista fraz, dla których ontologia będzie budowana. Gateway wyszukuje wolnego Ontology Agenta i rejestruje

go do budowy ontologii dla danej domeny. Podczas crawlowania pełni rolę walidatora linków sprawdzając ich poprawność oraz dbając o to, aby nie wychodzić poza początkową domenę oraz limit (jeśli taki ustalono).
2. Page Downloader Agent. Jest to stosunkowo prosty agent, który jest pośrednikiem w dostępie do stron internetowych. Jest bezpośrednio odpowiedzialny za ściąganie ich zawartości spod dostarczonych linków i przekazywanie ich Page Crawlerowi. Może być dowolnie klonowany.
3. Page Crawler. Jego zadaniem jest parsowanie kodu strony internetowej. Z jednej strony wyszukuje kolejne linki, które są dostarczane do Gateway Agenta, z drugiej strony przeprowadza rozpoznanie obiektów, na podstawie których zostanie przeprowadzona budowa ontologii. Drugi typ wiadomości jest przesyłany do agenta Ontology Agent. Szczegółowy opis sposobu selekcji słów do budowy ontologii znajduje się poniżej.
4. Statistics Agent - Jest to agent przyjmujący informacje statystyczne od pozostałych agentów i cyklicznie wypisujący je do konsoli przez działający w tle wątek.
5. Ontology Agent - Jest odpowiedzialny za budowanie ontologii z informacji dostarczonych od Page Crawlera. Jego szczegółowy opis znajduje się poniżej.

Selekcja i przetwarzanie tekstu ze stron www

Przetwarzanie strony odbywa się za pomocą parsera JSoup. Page Crawler wykonuje ekstrację zawartości wszystkich elementów dokumentu HTML o danym tagu: <p>, <a> oraz wszystkie nagłówki (<h1> do <h6>). Dla danego elementu pobierany jest tekst i przekazywany do analizy w klasie Language Analyzer.

Language Analyzer zaczyna pracę od przeanalizowania listy szukanych fraz - ponieważ mają to być obiekty w naszej ontologii, weryfikuje, że są to poprawne rzeczowniki i znajduje ich formy podstawowe (tzw. lemma).

Language Analyzer otrzymuje na wejściu bloki tekstu i przetwarza je. Pierwszym etapem jest wykonanie analizy morfologicznej przy pomocy analizatora morfologicznego Morfeusz. Dla każdego słowa w danej formie zwraca wszystkie możliwe poprawne gramatycznie użycia, np. Dla “mam” będzie to czasownik “mam” od “mieć” lub rzeczownik “mam” (od liczby mnogiej słowa “mama”). Dla każdego przypadku identyfikuje część mowy i w zależności od niej liczbę, osobę, rodzaj, przypadek. Analyzer stara się wybrać użycie odpowiadające poszukiwanemu.

Następnym etapem, wykonywanym, jeśli w parsowanym bloku znajdzie się (w dowolnej liczbie, przypadku, itd.) któraś z szukanych fraz, jest analiza reguł. Analyzer posiada listę reguł rozszerzających abstrakcyjną klasę Rule. Każda z reguł utrzymuje na wejściu listę wyników zwróconych przez Morfeusza i pozycję znalezionej frazy. Reguła może wędrować po liście wyników, by znajdować wystąpienia odpowiednich łańcuchów słów i na ich podstawie identyfikować elementy do dodania do ontologii. Znalezione elementy umieszczane są w

kolekcjach analizatora - ObjectProperties (gdzie property może mieć postać obiekt-atrybut lub obiekt-cecha-wartość cechy) lub ObjectSubclasses.

System zawiera dwie podstawowe reguły:

- IsRule - wyszukuje wystąpienia łańcucha:
  <objekt> <forma czasownika być lub zostać> <rzeczownik lub przymiotnik>.
  Jeśli trzecim elementem jest przymiotnik (np. Burek jest czarny), dodaje atrybut dla obiektu, jeśli rzeczownik (np. Burek jest psem), dodaje podklasę.
- PropertyRule - wyszukuje wystąpienia rzeczownika lub przymiotnika przed bądź po szukanej frazie. Np. dla łańcucha “czarny Burek” doda obiektowi “Burek” atrybut “czarny”. Analogicznie dla pozostałych przypadków.
  Dla łańcucha <rzeczownik> <szukany obiekt> chcieliśmy dodać podklasę (“pies Burek”), ale powodowało to dodawanie do ontologii wielu niepoprawnych elementów (np. “Wskoczywszy do **budy Burek** położył się spać”). Taka reguła musiałaby być bardziej skomplikowana i uzwględniać co najmniej deklinację obu rzeczowników.

Nie wszystkie typy zależności w ontologiach są obsługiwane, ale można jest bardzo łatwo doimplementować pisząc własne reguły i dodając je do Language Analyzera.

Budowa ontologii

Do budowy ontologii została wykorzystana biblioteka OWL API w wersji 4.2.3, dostępna na licencji open source. Wybór języka OWL był podyktowany elastycznością, popularnością i skutecznością w konstruowaniu semantyki internetu. Budowane ontologie są zapisywane do plików .xml zgodnie z powszechnie przyjętym formatem, a potem wizualizowane w programie Protege 2.0.
Klasą wystawiającą interfejs do tworzenia ontologii jest OntologyManager. Wykorzystuje ona bezpośrednio funkcje dostępne w OWL API. Zdefiniowane zostały następujące metody:

- createNewOntology()
- addClass()
- addSubclass()
- addObjectPropertyAssertion()
- addDataPropertyAssertion()
- addClassAssertion()
- addPropertyDataRange()
- addPropertyDomain()
- addSubObjectProperty()
- setPropertyType()
- setDifferentIndividuals()
- setDisjointClasses()
- createClassExpression()

Ten zestaw metod pozwala na dość złożone definiowanie klas, literałów, hierarchii, zależności i ograniczeń dla obiektów należących do danej bazy wiedzy. Oczywiście nie jest to pełny zakres możliwości, jakie oferuje wykorzystana biblioteka, ale do tego zadania w zupełności wystarczy.

Za pomocą zdefiniowanych reguł przetwarzania tekstu, zostały wyodrębione klasy oraz hierarchia dla przekazanych fraz kluczowych. Poniżej przedstawione zostały przykładowe wywołania aplikacji wraz z wygenerowanymi ontologiami.

Testy

W poniższych punktach zostały krótko opisane pozyskane ontologie oraz dołączone zostały zrzuty ekranu z programu Protege (hierarchia) oraz narzędzia WebVOWL do wizualizacji ontologii w formie grafów.

1) wikipedia.pl/Pole

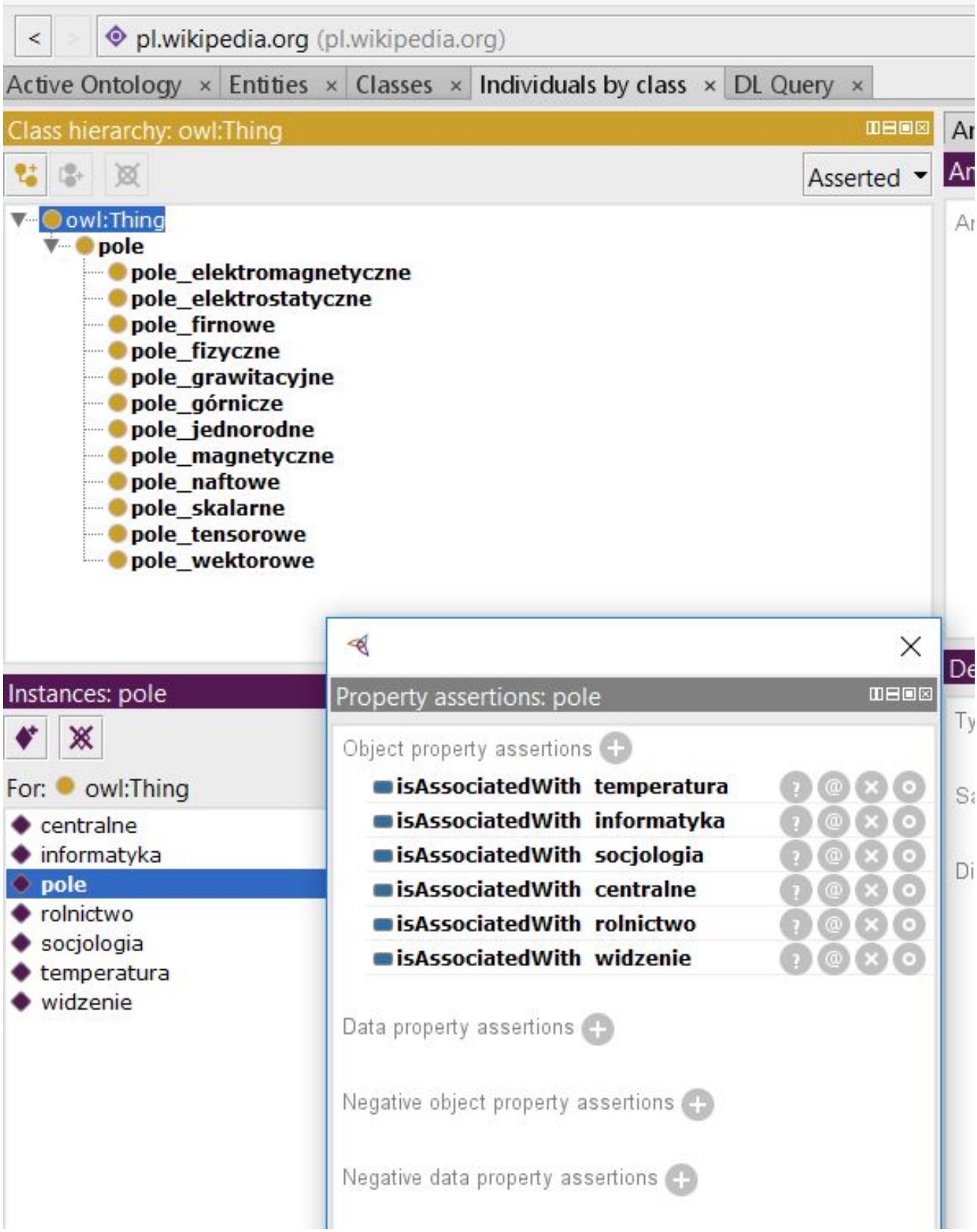
Frazy: pole
Wikipedia jest dobrym źródłem wszelakich definicji i różnych kategorii dla danych pojęć, dlatego od niej zaczęło się testowanie. Hasła w tej domenie często mogą odnosić się do więcej niż jednego artykułu - jest więc to dobre źródło podklas dla szukanych fraz. Ponadto, wikipedia dostarcza informację do jakiej dziedziny dany artykuł należy. Wykorzystaliśmy to do określenia atrybutów “isAssociatedWith”, które wskazywały na jedną ze znalezionych domen pojęciowych.

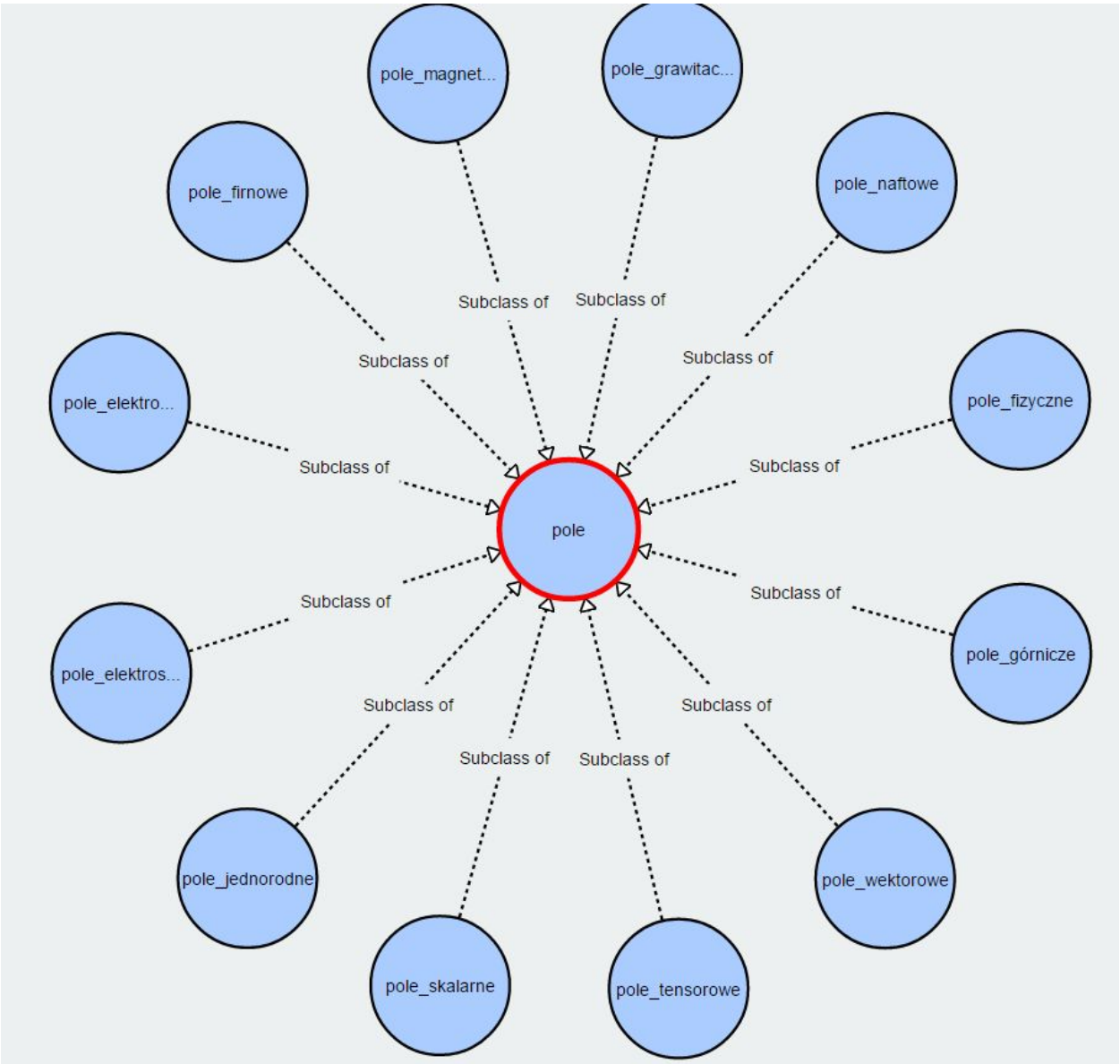
2) komputerswiat.pl

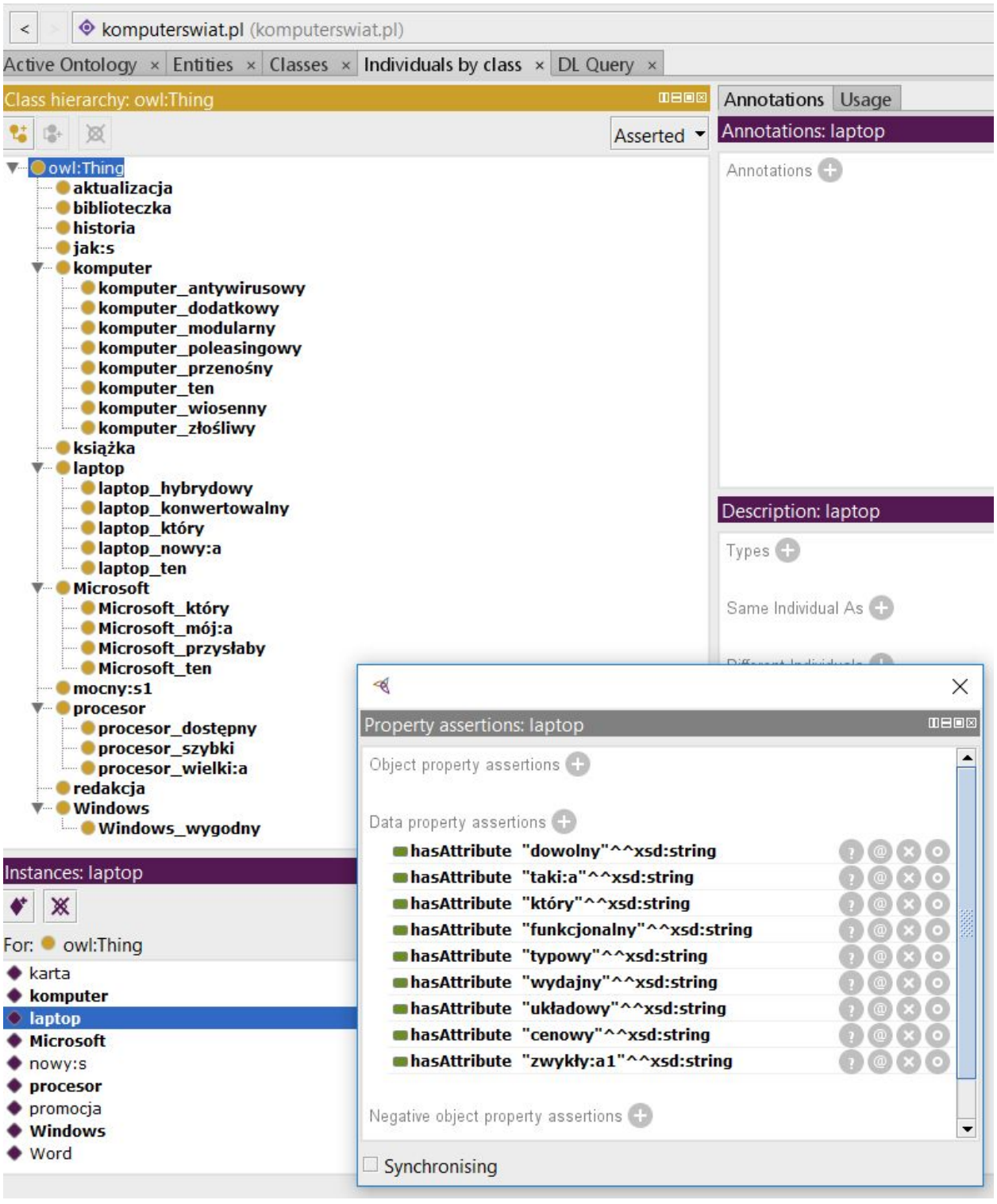
Frazy: komputer, procesor, windows, linux, laptop, microsoft
Strona komputerswiat jest idealną kopalnią wiedzy na temat elektroniki i oprogramowania. W tym teście chcieliśmy sprawdzić jak system poradzi sobie z wieloma frazami jednocześnie. Udało się uzyskać zarówno hierarchię klas jak i kilka przykładowych atrybutów. Nie wszystkie frazy zostały odnalezione - w tym wypadku brakuje hasła “linux”. Warto nadmienić, że mimo iż wyszukiwane są słowa nie występujące w języku polskim, ich przetwarzanie nie jest ograniczane, gdyż są to frazy na których nam zależy.

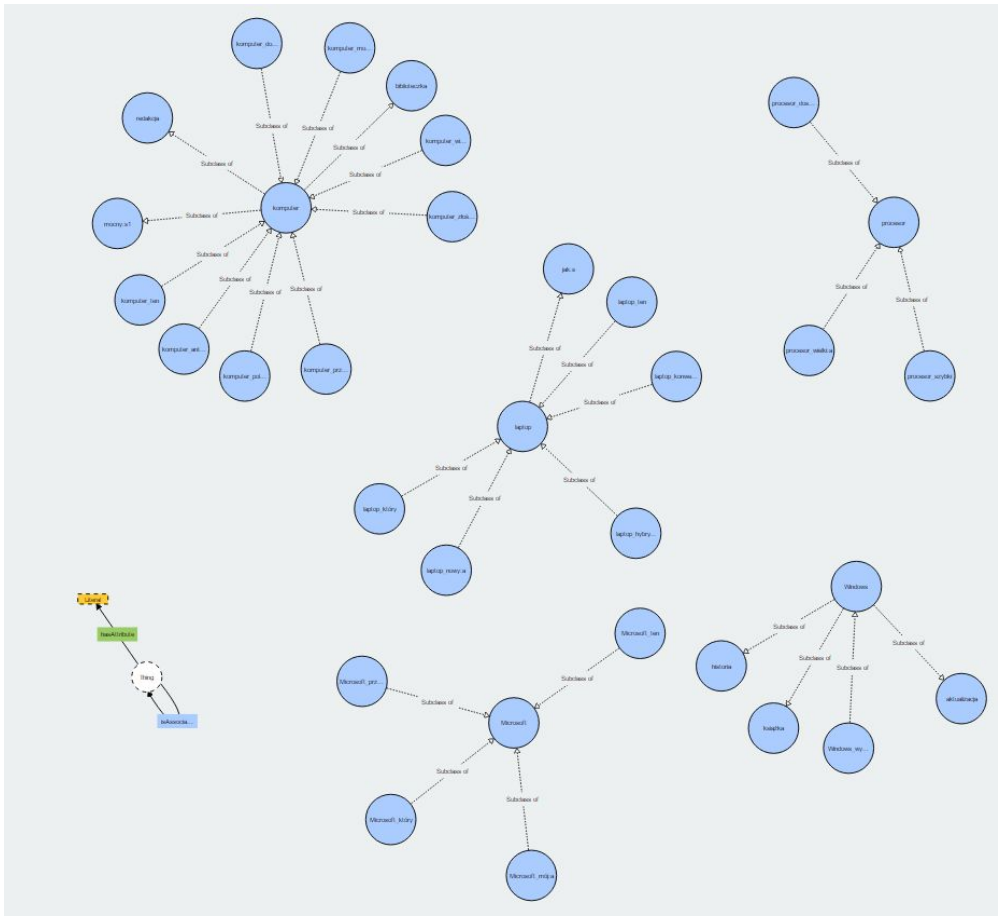
3) ztm.pl

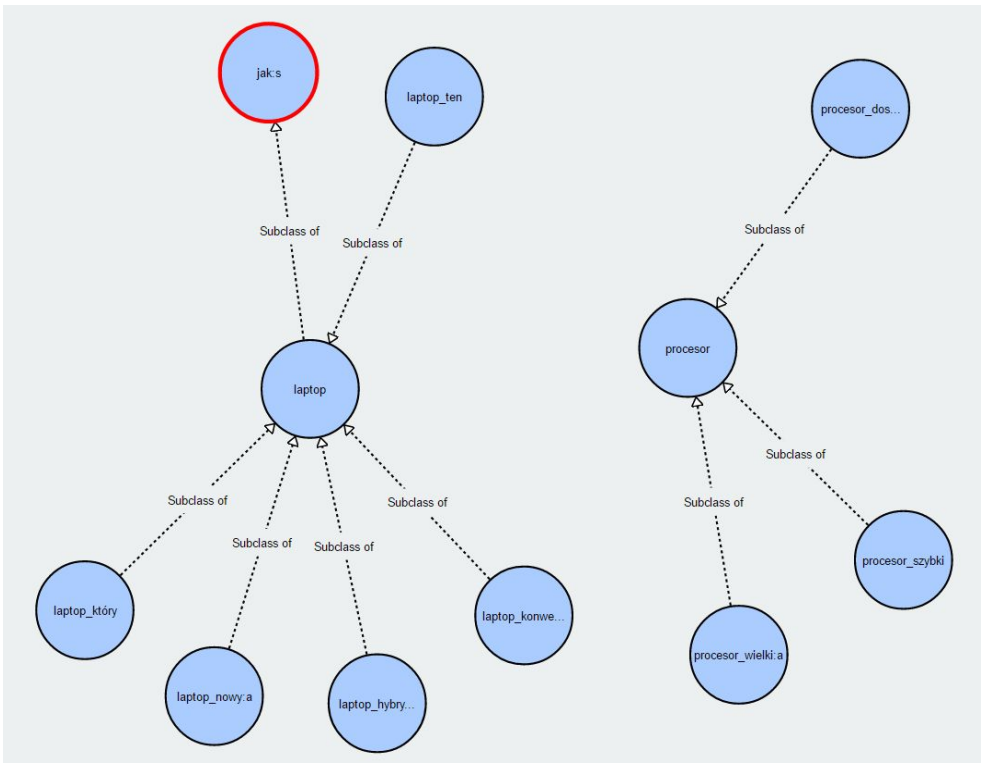
Frazy: bilet
Test ilościowy - sprawdzone zostało ile różnych postaci klas, podklas i cech aplikacja jest w stanie znaleźć.

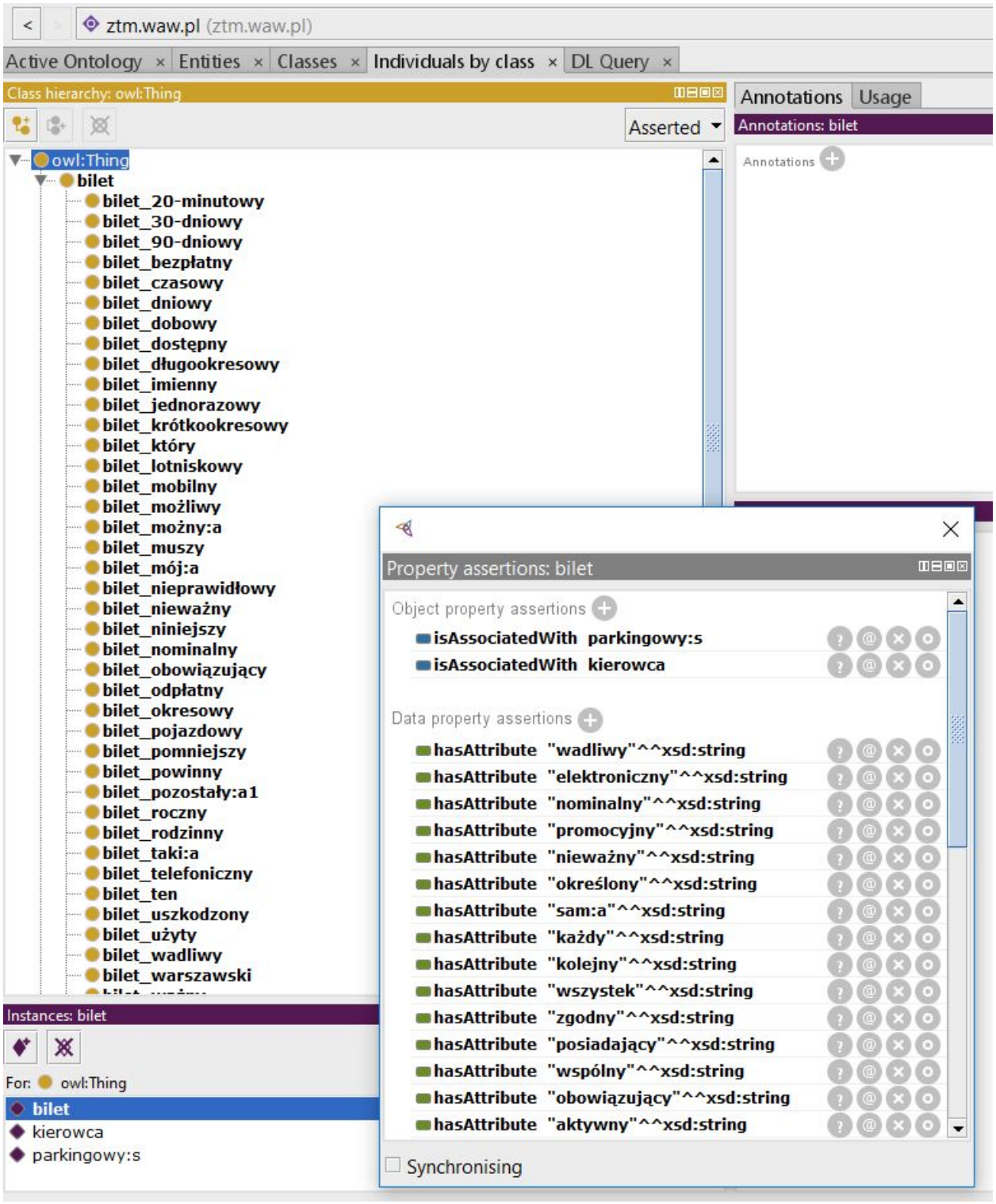

pl.wikipedia.org (pl.wikipedia.org)
Active Ontology
Entities
Classes
Individuals by class
DL Query
Class hierarchy: owl:Thing
Asserted
owl:Thing
pole
pole_elektromagnetyczne
pole_elektrostatyczne
pole_firnowe
pole_fizyczne
pole_grawitacyjne
pole_górnicze
pole_jednorodne
pole_magnetyczne
pole_naftowe
pole_skalarne
pole_tensorowe
pole_wektorowe
Instances: pole
For: owl:Thing
centralne
informatyka
pole
rolnictwo
socjologia
temperatura
widzenie
Property assertions: pole
Object property assertions
isAssociatedWith temperatura
isAssociatedWith informatyka
isAssociatedWith socjologia
isAssociatedWith centralne
isAssociatedWith rolnictwo
isAssociatedWith widzenie
Data property assertions
Negative object property assertions
Negative data property assertions

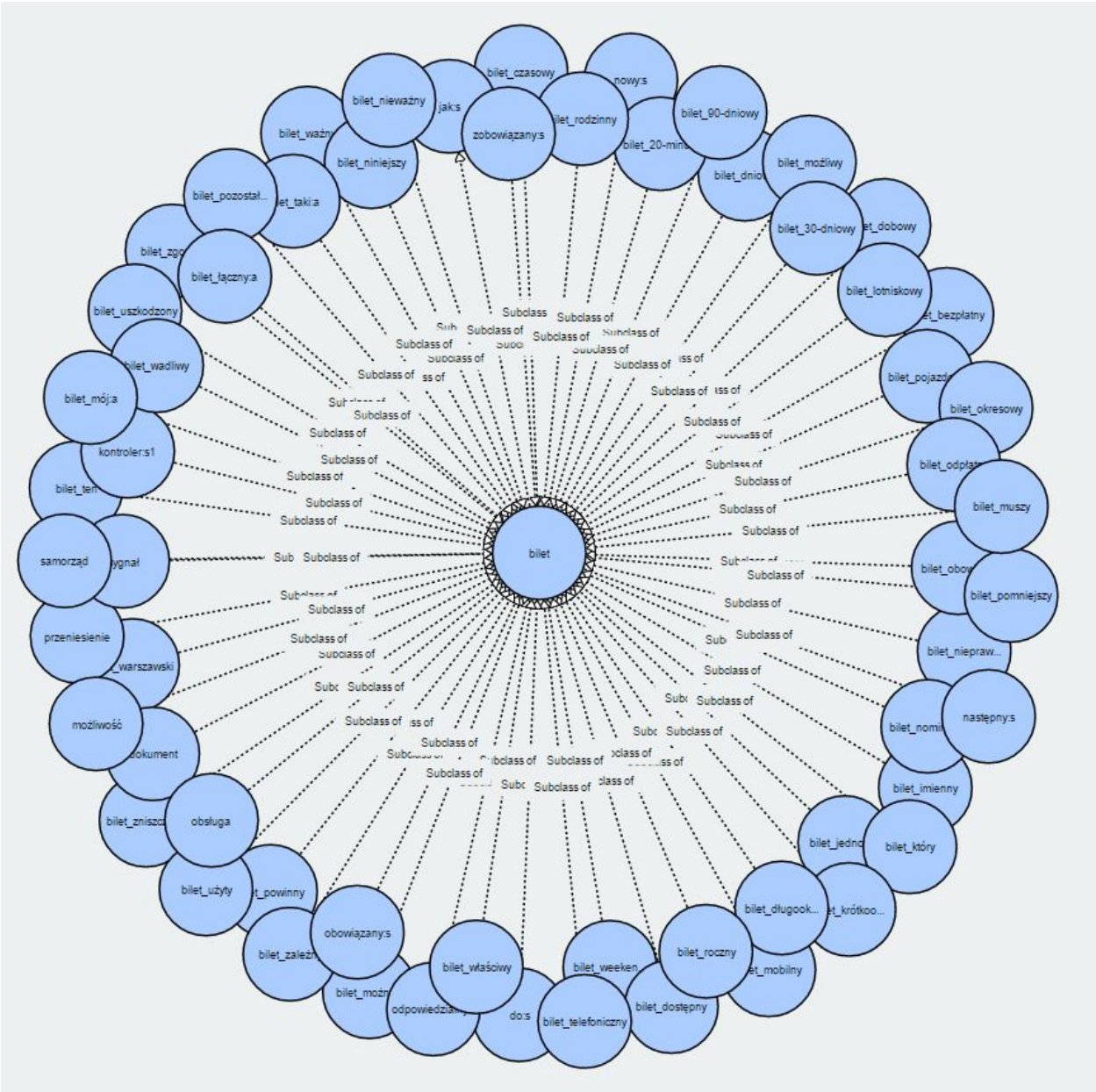

pole_magnet...
pole_grawitac...
pole_firnowe
pole_naftowe
Subclass of
Subclass of
Subclass of
Subclass of
pole_elektro...
pole_fizyczne
Subclass of
Subclass of
pole
Subclass of
Subclass of
pole_elektros...
pole_górnicze
Subclass of
Subclass of
Subclass of
Subclass of
pole_jednorodne
pole_wektorowe
pole_skalarne
pole_tensorowe

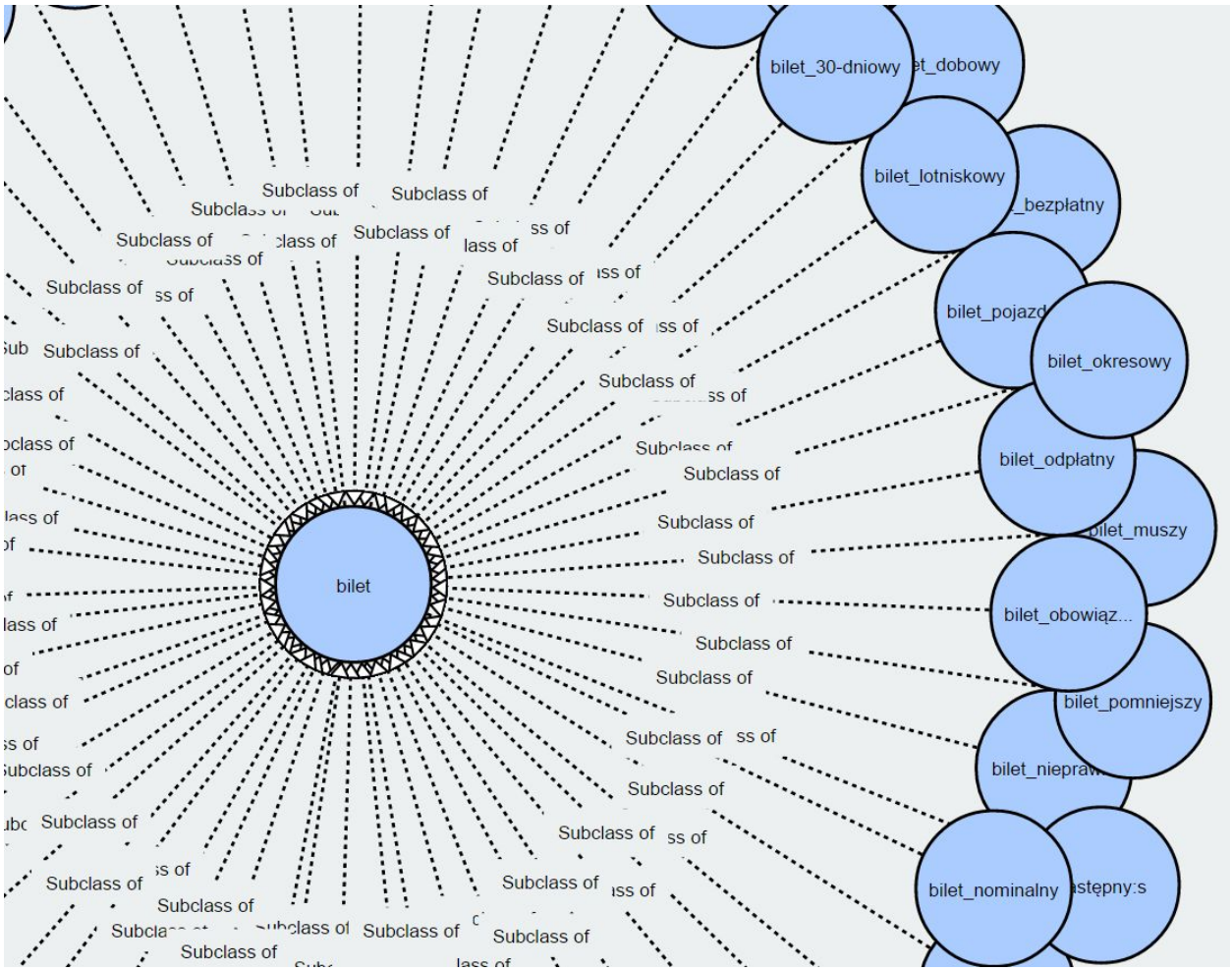

komputerswiat.pl (komputerswiat.pl)
Active Ontology
Entities
Classes
Individuals by class
DL Query
Class hierarchy: owl:Thing
Asserted
owl:Thing
aktualizacja
biblioteczka
historia
jak:s
komputer
komputer_antywirusowy
komputer_dodatkowy
komputer_modularny
komputer_poleasingowy
komputer_przenośny
komputer_ten
komputer_wiosenny
komputer_złośliwy
książka
laptop
laptop_hybrydowy
laptop_konwertowalny
laptop_który
laptop_nowy:a
laptop_ten
Microsoft
Microsoft_który
Microsoft_mój:a
Microsoft_przysłaby
Microsoft_ten
mocny:s1
procesor
procesor_dostępny
procesor_szybki
procesor_wielki:a
redakcja
Windows
Windows_wygodny
Annotations
Usage
Annotations: laptop
Annotations
Description: laptop
Types
Same Individual As
Property assertions: laptop
Object property assertions
Data property assertions
hasAttribute "dowolny"^^xsd:string
hasAttribute "taki:a"^^xsd:string
hasAttribute "który"^^xsd:string
hasAttribute "funkcjonalny"^^xsd:string
hasAttribute "typowy"^^xsd:string
hasAttribute "wydajny"^^xsd:string
hasAttribute "układowy"^^xsd:string
hasAttribute "cenowy"^^xsd:string
hasAttribute "zwykły:a1"^^xsd:string
Negative object property assertions
Synchronising
Instances: laptop
For: owl:Thing
karta
komputer
laptop
Microsoft
nowy:s
procesor
promocja
Windows
Word




jak:s
laptop_ten
procesor_dos...
Subclass of
Subclass of
Subclass of
procesor
laptop
Subclass of
Subclass of
Subclass of
Subclass of
laptop_który
laptop_nowy:a
laptop_hybry...
laptop_konwe...
procesor_wielki:a
procesor_szybki


ztm.waw.pl (ztm.waw.pl)
Active Ontology
Entities
Classes
Individuals by class
DL Query
Class hierarchy: owl:Thing
Asserted
owl:Thing
bilet
bilet_20-minutowy
bilet_30-dniowy
bilet_90-dniowy
bilet_bezpłatny
bilet_czasowy
bilet_dniowy
bilet_dobowy
bilet_dostępny
bilet_długookresowy
bilet_imienny
bilet_jednorazowy
bilet_krótkookresowy
bilet_który
bilet_lotniskowy
bilet_mobilny
bilet_możliwy
bilet_możny:a
bilet_muszy
bilet_mój:a
bilet_nieprawidłowy
bilet_nieważny
bilet_niniejszy
bilet_nominalny
bilet_obowiązujący
bilet_odpłatny
bilet_okresowy
bilet_pojazdowy
bilet_pomniejszy
bilet_powinny
bilet_pozostały:a1
bilet_roczny
bilet_rodzinny
bilet_taki:a
bilet_telefoniczny
bilet_ten
bilet_uszkodzony
bilet_użyty
bilet_wadliwy
bilet_warszawski
Annotations
Usage
Annotations: bilet
Annotations
Property assertions: bilet
Object property assertions
isAssociatedWith parkingowy:s
isAssociatedWith kierowca
Data property assertions
hasAttribute "wadliwy"^^xsd:string
hasAttribute "elektroniczny"^^xsd:string
hasAttribute "nominalny"^^xsd:string
hasAttribute "promocyjny"^^xsd:string
hasAttribute "nieważny"^^xsd:string
hasAttribute "określony"^^xsd:string
hasAttribute "sam:a"^^xsd:string
hasAttribute "każdy"^^xsd:string
hasAttribute "kolejny"^^xsd:string
hasAttribute "wszystek"^^xsd:string
hasAttribute "zgodny"^^xsd:string
hasAttribute "posiadający"^^xsd:string
hasAttribute "wspólny"^^xsd:string
hasAttribute "obowiązujący"^^xsd:string
hasAttribute "aktywny"^^xsd:string
Synchronising
Instances: bilet
For: owl:Thing
bilet
kierowca
parkingowy:s


bilet
Subclass of
bilet_czasowy
zobowiązany:s
bilet_rodzinny
bilet_90-dniowy
bilet_20-min
bilet_możliwy
bilet_30-dniowy
bilet_lotniskowy
bilet_okresowy
bilet_muszy
bilet_pomniejszy
bilet_następny:s
bilet_imienny
bilet_który
bilet_długook...
bilet_roczny
bilet_dostępny
bilet_telefoniczny
bilet_właściwy
obowiązany:s
bilet_użyty
obsługa
bilet_zniszcz
możliwość
przeniesienie
samorząd
bilet_mój:a
bilet_wadliwy
bilet_uszkodzony
bilet_łączny:a
bilet_pozostał...
bilet_nieważny
bilet_niniejszy
bilet_ważny



Linki

Kod źródłowy jest dostępny na githubie:
https://github.com/R-J-J/SAG
Do analizowania tekstów w języku polskim wykorzystaliśmy Słownik Gramatyczny Języka Polskiego stworzony przez Prof. Zygmunta Saloniego z Uniwersytetu Warszawskiego:
http://sgjp.pl/
Oraz oparty na nim Analizator Morfologiczny Morfeusz
http://sgjp.pl/morfeusz/
Oznaczenia części mowy i pozostałych informacji o analizowanych słowach pochodzą stąd:
http://www.ipipan.waw.pl/~wolinski/publ/znakowanie.pdf