Wyznaczanie reprezentacji preferencji uniwersalych

Bartosz Górski Marcin Kaczyński Paweł Sokołowski

13 stycznia 2013

1 Opis zadania

Szablon do wypełnienia. Jak ktoś chce napisać jakiś konkretny rozdział to niech się przy nim wpisze - może być w komentarzu. Jak nie to się później rozdzieli zadania.

Przykład wykorzystania bibliografii[1]. W bibliografii wpisy są z jakiegoś mojego wcześniejszego raportu, będą zmienione. Kompilacja:

- pdflatex raport (chyba musi być)
- bibtex raport
- pdflatex raport (x2)

Jak ktoś nie umie pisać w texie, to niech prześle plain text w notaniku.

2 Założenia

3 Dane wejściowe i wyjściowe

Dane wejściowe są przekazywane do aplikacji w postaci następujących plików:

- plik zawierający zbiór danych
- plik z opisem relacji

Obsługiwanym formatem plików zawierających zbiór danych jest Csv. Wymagane jest aby w jednej z kolumn przechowywane były dane określające do której klasy należy dany wiersz. W drugim pliku znajduje się opis relacji pomiędzy nimi.

4 Szczegóły implementacji

Implementacja została przygotowana w języku C# i działa na platformie .NET Framework 4.0. Implementacja związana z algorytmem została podzielona na 4 projekty: Algorithm, DAL, HashTree i pomocniczy projekt Common.

W projekcie Algorithm znajdują się wszystkie klasy i interfejsy w bezpośredni sposób związane z interfejsem. Projekt DAL zawiera implementacje dostępu do danych, a więc w tym przypadku parsowanie plików z danymi do postaci wykorzystywanej w algorytmie. W projekcie HashTree znalajdują się interfejsy i implementacja drzewa mieszającego. Projekt Common zawiera wspólne klasy wykorzystywane w pozostałych projektach. W projekcie Common znajdują się niemal jedynie obiekty DTO. Obiekty te same w sobie nie wykonują operacji, dlatego nie będą szczegółowo opisywane.

W celu zapewnienia elastyczności skorzystano w projekie ze wzorca Dependency Injection. Zgodnie z wzorcem, na implementację algorytmu składa się szereg interfejsów, których odpowiedzialności przestawiaja się następująco:

Za główny interfejs można uznać *IAlgorithm*, a jego podstawowym zadaniem jest znajdowanie preferencji uniwersalnych. Interfejs jest implementowany przez klasy *Generators* i *ModifiedApriori*. Podstawowa różnica pomiędzy klasami polega na tym, że klasa Generators do obliczania minimalnych preferencji wykorzystuje informację o generatorach do odrzucania większej liczby zbiorów kandydujących. Obie implementacje operują na wierszach typu *Row*/*SimpleRow*, które są odwzorowaniem danych w postaci wygodnej do prowadzenia obliczeń. Do znajdywania zbiorów kandydujących wykorzystywany jest interfejs *ICandidatesGenerator*. Za znajdowanie podzbiorów wspieranych w transakcjach odpowiada *IHashTree*. Wczytywanie danych odbywa się za pomocą *IDataManager*, a konwersja wyników do postaci wygodnej dla użytkownika za pomocą *IResultsConverter*.

Ogólnie przebieg działania całego algorytmu składa się z następujących kroków:

1. wczytanie i parsowanie danych,
2. wykonanie obliczeń,
3. parsowanie wyników do czytelnej postaci

Wykonanie obliczeń (2) przebiega wg następującego schematu:

W kolejnych iteracjach znajdywane są nowe zbiory kandydujące, dla wszystkich zbiorów obliczane są dwa liczniki mówiące o tym jakie wsparcie mają te zbiory w klasie transakcji spełniających i nie spełniających relację. Aby zapewnić wydajne obliczanie wartości omawianych liczników, wykorzystywane jest drzewo mieszające tworzone w każdej iteracji dla zbioru kandydatów. Po obliczeniu liczników podejmowana jest decyzja o tym, czy dany kandydat jest minimalnym wzorcem, czy należy go odrzucić lub analizować dalej. Na tym etapie (opcjonalnie) podejmowana jest decyzja czy któryś z podzbiorów danego zbioru kandydującego jest generatorem. Jeżeli tak, taki zbiór nie będzie analizowany dalej.

5 Wyniki

5.1 Wyniki podstawowe

Tutaj będzie opis wyników dla danych z artykułu.

5.2 Wyniki dla złożonych danych

A tutaj wyniki dla normalnych danych. Na pewno musi być zbiór z samochodami, ten który był na którymś lab.

6 Wnioski

A Podręcznik użytkownika

Do uruchomienia aplikacji wymagany jest zainstalowany .NET Framework 4.0. Nie jest konieczna instalacja dodatkowych programów, bibliotek lub komponentów.

Do obsługi algorytmu został przygotowany prosty graficzny interfejs, opierający się na idei instalatora. W kolejnych krokach użytkownik wybiera kolejne dopuszczalne opcje i zatwierdza je klikając przycisk Next. Możliwe jest cofnięcie się do poprzedniego widoku za pomocą przycisku Prev. Kolejne widoki wyglądają następująco:

W kroku pierwszym1 należy wbrać pliki z danymi, klikając w przyciski Wybierz. Pliki wybiera się wykorzystując standardowy dialog wyboru plików z systemu Windows. Nie ma ograniczenia na rozszerzenie pliku.

Plik z danymi powinien być poprawnym pod względem budowy plikiem csv. W pliku powinny być kolejne wiersze w których w każdym znajduje się taka sama liczba kolumn oddzielonych określonym separatorem.

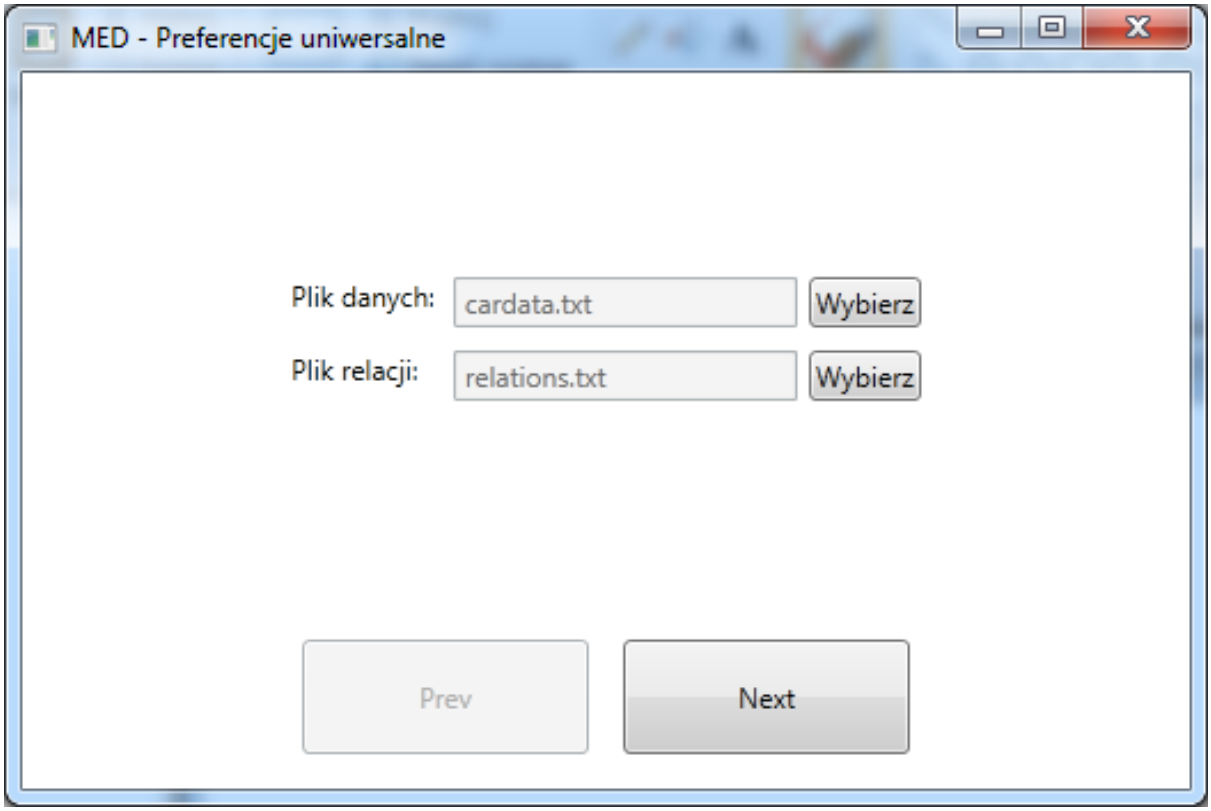
Plik z relacjami składa się z wielu linii. W każdej z nich znajduje się wpis postaci:

```
a<b
```

Oznacza to, że klasa **a** jest w ralacji z klasą **b**. Dla przykładowego grafu relacji (2) plik powinien być postaci:

```
Y<V
Z<V
X<Y
X<Z
U<W
```

W kroku drugim3 należy wybrać parametry algorytmu. Dopuszczono wybór samego algorytmu - możliwe jest wykonanie algorytmu w wersji wykorzystującej oraz nie wykorzystującej faktu o podzbiorach będących generatorami.



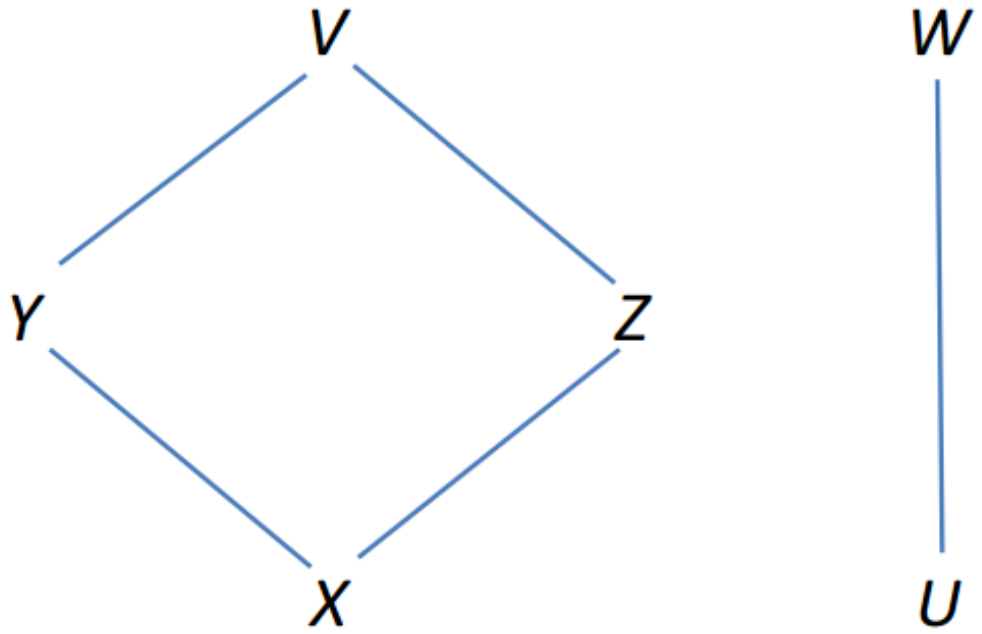
Rysunek 1: Wybór plików z danymi

W tym kroku należy również wybrać separator wykorzystywany w pliku danych, oraz określić indeks kolumny która zawiera klasę do której przypisany jest dany wiersz.

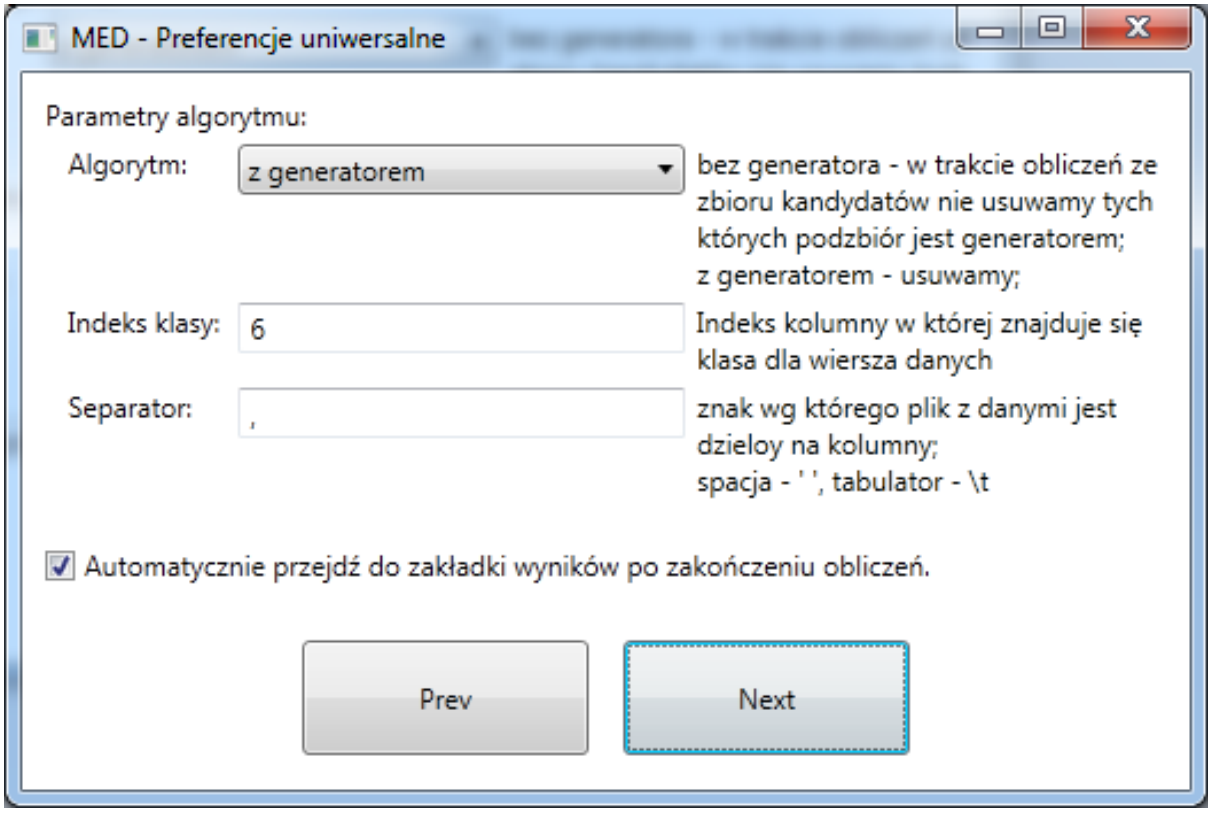
Krok trzeci4 to widok prezentowany w trakcie wykonywania obliczeń. W widoku wyświetlane są informacje diagnostyczne mówiące o tym które zbiory kandydujące zostały uznane za minimalne wzorce kontrastowe, które zostały odrzucone a które zostaną poddane dalszej analizie. Jeżeli w poprzednim kroku zaznaczono opcję Automatycznie przejdź do zakładki wyników po zakończeniu obliczeń, to po znalezieniu wszystkich wzorców program wyświetli widok wyników5.

Literatura

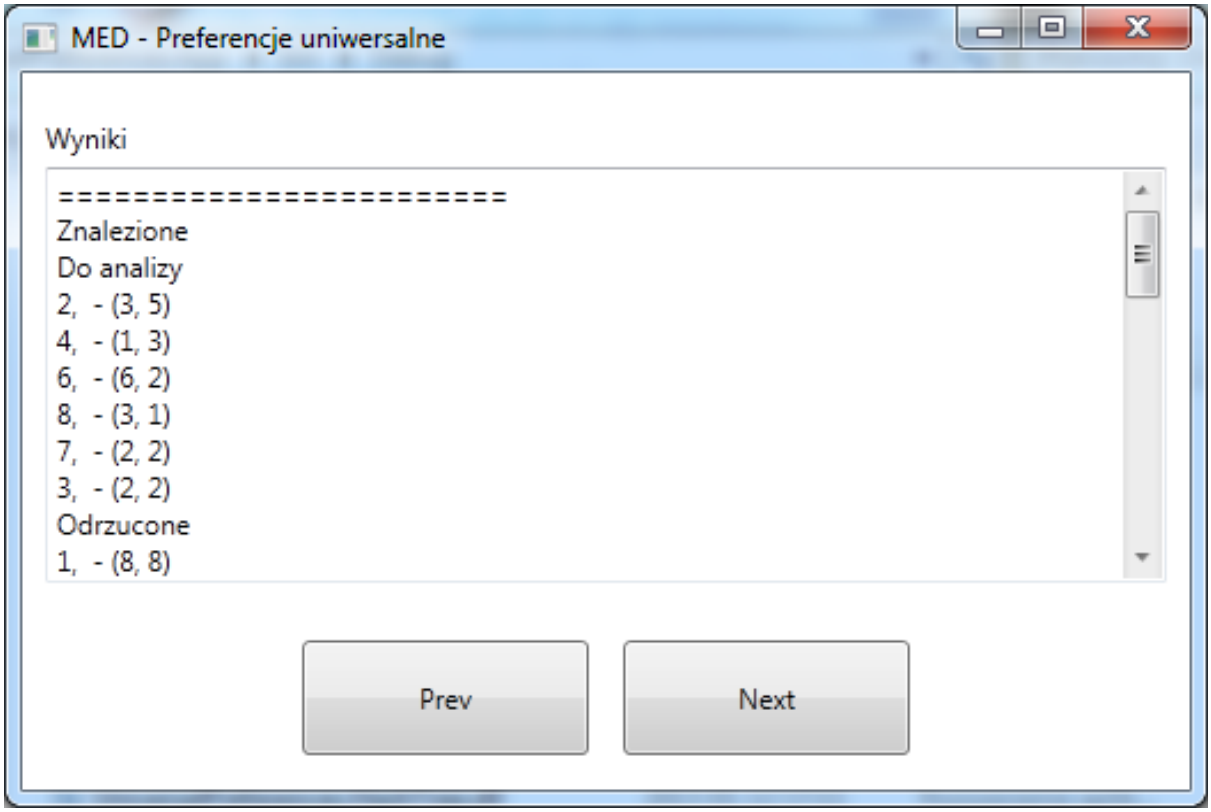
[1] R. Mihalcea and D. I. Moldovan. Ez.wordnet: Principles for automatic generation of a coarse grained wordnet. *In Proc. of 14th Intel Florida Artifacial Intelligence Research Society Conf*, pages 454–458, 2001.



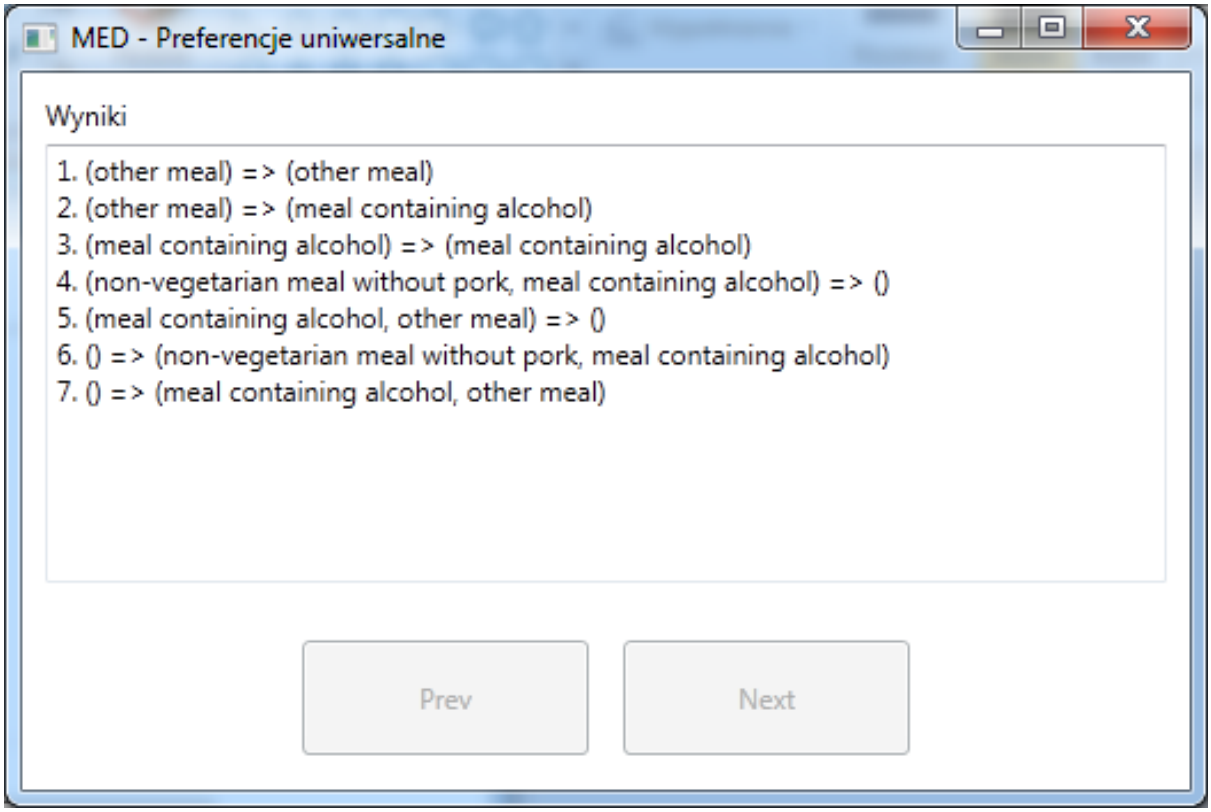
Rysunek 2: Przykładowy graf relacji



Rysunek 3: Wybór parametrów



Rysunek 4: Widok obliczeń



Rysunek 5: Widok wyników