Zastosowanie algorytmów pełnotekstowych w dopasowywaniu i uspójnianiu danych klientów

Piotr Kaczor[1]

Streszczenie: Współczesne systemy wykorzystujące bazy danych (zwłaszcza hurtownie danych) nie radzą sobie skutecznie z duplikatami danych klientów. Skutkiem wielokrotnego występowania jednej osoby fizycznej w bazach danych są, oprócz oczywistego zaburzenia porządku, liczne straty biznesowe wynikające m.in. z zafałszowania analiz biznesowych, czy rozsyłania redundantnych materiałów marketingowych. Niniejsza praca ma na celu zaprezentowanie i ocenę wydajności wybranych pełnotekstowych metod porównywania i dopasowywania różnych danych klientów – uspójniania bazy danych tak, by wskazanych duplikatów było możliwie jak najmniej.

Słowa kluczowe: dane klienta, dopasowywanie, uspójnianie, duplikacja

1. Wstęp

Klient klientowi nierówny – to stwierdzenie zna chyba każdy specjalista w dziedzinie handlu oraz każdy przeciętny klient, który dokonał choć jednego poważnego zakupu. Dziś jednak okazuje się ono być równie istotne w przypadku specjalistów od wyszukiwania informacji w bazach danych – choć pojmowanie słowa *nierówny* jest nieco inne. Dla sprzedawcy istotą jest tu stan majątkowy klienta, podczas gdy informatyk skupi się raczej na danych o kliencie[2] i różnicach w ich zapisie w różnych źródłach danych. W niniejszej pracy skupiono się jedynie na tym drugim.

Na początku warto uświadomić sobie, iż rozbieżności takie powstają w systemach informatycznych codziennie, w większości przypadków bez naszej wiedzy, a wynikają zazwyczaj z jednego z trzech podstawowych powodów:

- przez nieuwagę osoby wprowadzającej nowe dane osobowe, która nie zauważyła, że dana osoba już w spisie figuruje[3];
- w wyniku błędu bazy danych lub częściej aplikacji nad nią nadbudowanej (problemy wielodostępu, wyłączności zapisu, replikacji, itp.);
- z reguł biznesowych, podstaw prawnych, itp.

[1] Politechnika Warszawska, Instytut Automatyki i Informatyki Stosowanej, ul. Nowowiejska 15/19, 00-665 Warszawa, Polska
e-mail: pkaczor@elka.pw.edu.pl

[2] Za klienta uznawana jest zarówno osoba fizyczna, jak i przedsiębiorstwo (firma), dla którego wykonywana jest transakcja sprzedaży.

[3] Mamy tu do czynienia zarówno z duplikatami „pierwotnymi", tj. powstającymi w wyniku błędu osoby wprowadzającej dane, oraz „wtórnymi", tj. powstającymi w wyniku wprowadzenia poprawnych danych, w sytuacji gdy ich niepoprawna wersja już znajduje się w bazie danych.

Owe różnice oraz inne podobne problemy związane z jakością utrzymywanych danych, choć z pozoru niegroźne, kosztują amerykański przemysł ponad 600 miliardów dolarów rocznie![4] W Polsce sytuacja kształtuje się podobnie. Można zatem uznać, iż warto się temu zagadnieniu przyjrzeć nieco bliżej[5]. Niniejsza praca stanowi podsumowanie dwuletnich badań w tej dziedzinie, podaje pewne wskazówki oraz pełne algorytmy mające pomóc w dopasowaniu danych klientów, wykorzystujące nieużywane dotychczas w tej dziedzinie algorytmy porównywania pełnotekstowego.

2. Analiza zagadnienia

Analiza złożonego problemu, jakim jest uspójnianie danych klientów, wymaga uwzględnienia zagadnień z następujących dziedzin:

- bazy danych (postaci normalne, więzy deklaratywne);
- metody eksploracji danych (teoria reduktów, zależności funkcyjne);
- inżynieria wiedzy (zbiory przybliżone, logika rozmyta);
- leksykologia (słowniki, synonimy).

Dalsze podpunkty tej części pracy opisują powyższe zagadnienia, odnosząc je do problemu wykrywania duplikatów w bazach danych. We wszystkich założeniach, analizach, doświadczeniach, testach i wnioskach przyjęto, że mamy do czynienia z relacyjną (względnie relacyjno-obiektową), scentralizowaną bazą danych. Również aplikacja demonstrująca skuteczność proponowanych rozwiązań została napisana i przetestowana w scentralizowanym, jednorodnym środowisku programistycznym.

2.1. Bazy danych

Podczas wyszukiwania duplikatów danych dobrze jest przyjąć, że dane są co najmniej w trzeciej postaci normalnej. W przeciwnym przypadku będziemy mieli dodatkowo do czynienia z anomaliami i dane wprowadzane do bazy danych mogą okazać się niespójne, a ich poprawianie uciążliwe lub wręcz niemożliwe.

W samym porównywaniu pomocne mogą niejednokrotnie okazać się klucze unikalne, jak np. numer PESEL. Wskazanie duplikatu na podstawie kluczy unikalnych może jednak stanowić problem, gdyż:

- wartości kluczy unikalnych nie są zazwyczaj wypełniane w bazie automatycznie (co jest ich niewątpliwą przewagą w stosunku do sztucznych kluczy głównych, gdyż daje nadzieję, że klucze te będą identyczne dla tych samych

[4] patrz Fryman (2004)

[5] Zwłaszcza, że obecne na rynku programy do usuwania duplikatów (np. Informatica, Warehouse Builder) wykorzystują jedynie niektóre prezentowane poniżej techniki głównie do czyszczenia danych (poprawiania i usuwania danych o nieprawidłowym formacie lub spoza słownika), rzadko zaś skupiają się na poszukiwaniu duplikatów w nieustandaryzowanej (np. ze względów polityki bezpieczeństwa firmy) bazie danych.

klientów, nawet w różnych bazach danych), ale w zamian podlegają błędom osób je wprowadzających[6];
- nie w każdej bazie danych muszą być przechowywane wartości takich kluczy, a w szczególności walidacja takich kluczy może być różna w różnych systemach (błędy lub braki projektowe, różne wymagania biznesowe, ustawa o ochronie danych osobowych).

2.2. Metody eksploracji danych

W nieco mniej trywialnych przypadkach, do identyfikowania duplikatów nadają się metody mające swe źródła w metodach eksploracji danych. Po podzieleniu tabeli na osobne kolumny okazuje się bowiem, iż wartości każdej z nich, potraktowane jako zwykły tekst (pozbawiony semantyki), dają się łatwo porównywać, pozwalając określić przy tym procentową, odpowiadającą ludzkiej percepcji miarę zgodności. Idealne zastosowanie znajdują w tym przypadku metody porównania gramowego (opisane w dalszej części pracy). Dopasowywanie jedynie na podstawie 100% zgodności w przypadku porównania gramowego nie jest jednak efektywne.

2.3. Inżynieria wiedzy

Podstawowym pojęciem, zrozumienie którego daje możliwość efektywnego wykrywania duplikatów jest zbiór rozmyty. Praktycznym zastosowaniem zbioru rozmytego może być określenie na podstawie preferencji użytkownika[7] w jakim stopniu dane krotki są nierozróżnialne (na ile, według przyjętej miary, można być pewnym, że zawierają semantycznie to samo) – wystarczy uzyskać wartości skrajne miary dla takiej zgodności a następnie zamodelować wartości pośrednie w przedziale pomiędzy wskazanymi przez niego preferencjami. Modelowanie sprowadza się z reguły do wyboru pewnej funkcji wielomianowej, podczas gdy określenie nawet najprostszej miary zgodności stanowi znacznie bardziej skomplikowany problem. Umiejętne połączenie teorii eksploracji tekstowej z teorią zbiorów rozmytych stanowi podstawę autorskiego algorytmu zaproponowanego w niniejszej pracy.

2.4. Leksykologia

W przypadkach, gdy wartości atrybutów nie pozwalają na bezpośrednie, jednoznaczne zidentyfikowanie na ich podstawie klienta (przykładowo w bazie istnieje *Jan Kowalski* i *Janek Kowalski*), a metody eksploracyjne zawiodą[8], przydatne mogą się okazać elementy leksykologii. Wystarczy, że A jest synonimem B, a można je

[6] Nawet poprawny numer PESEL nie gwarantuje poprawności przechowywanych danych – typowo stosowany jako „zaślepka" PESEL 22222222222 jest poprawny!

[7] Użytkownikowi zwykle łatwo jest podjąć skrajne decyzje, trudno natomiast podać jednoznaczne rozwiązanie w przypadkach bardziej typowych. Użycie zbiorów rozmytych pozwala na automatyczne podejmowanie decyzji w sytuacjach typowych na podstawie wiedzy o preferencjach użytkownika w określonych przez niego przypadkach skrajnych.

[8] Metody eksploracji tekstowej idealnie nadają się do porównywania co najmniej kilkuznakowych ciągów liter, często zawodzą natomiast przy tekstach krótkich, gdzie błąd w wyrazie ma duże znaczenie. Niemożliwe staje się również zastosowanie eksploracji tekstowej do pojedynczego wyrazu, w celu korekty danych.

będzie uznać za zgodne. Szczegółowe rozwiązania oparte o metody słownikowe przedstawiono w dalszej części pracy.

2.5. Konkluzja

Można dojść do wniosku, że przy zastosowaniu dość znacznej, ale ograniczonej, liczby środków i algorytmów będziemy w stanie w pełni zautomatyzować proces czyszczenia i uspójniania danych klientów. Praktyka pokazuje jednak, że konkretne algorytmy znajdują zastosowanie jedynie w konkretnych przypadkach, natomiast praktycznie niemożliwe jest użycie ich w każdej sytuacji. Ponieważ zaprezentowane algorytmy nie są w stanie zagwarantować pełnej skuteczności, co najmniej faza ostatnia, związana z fizycznym usuwaniem redundatnych danych z bazy, musi nadal być nadzorowana i zatwierdzana przez „nieomylnego" człowieka. Dzięki poniższym algorytmom prostsze natomiast powinno okazać się zapobieganie powstawaniu redundantnych danych klientów i ich zidentyfikowanie w istniejących systemach.

3. Algorytmy

Opisane w tym rozdziale algorytmy mają na celu rozwiązanie dwóch podstawowych problemów:

- wykrywanie istnienia odpowiednich danych w bazie podczas próby wpisania nowego klienta – przypadek pozbawiony możliwości bezpośredniej konfiguracji, zorientowany na wydajność algorytmu;
- przeszukiwanie całej bazy danych i znalezienie wielokrotnie wpisanych klientów – przypadek rozbudowany o pełne możliwości konfiguracyjne, zorientowany na dokładność algorytmu.

Pomimo na pozór sprzecznych wymagań stawianych przez powyższe podpunkty, doświadczenia pokazują, że istnieją algorytmy na tyle dobre, by sprostać wymaganiom tak dokładności, jak i wydajności[9] (naturalnie wszystko zależy od rozmiaru bazy). Algorytmy takie radzą sobie przy tym z większośią poniższych problemów:

- dodatkowe spacje, kropki, myślniki, itp.;
- różna wielkość liter;
- literówki (dodane, pominięte, zmienione lub zamienione pojedyncze litery, w tym znaki narodowe),
- opcjonalne występowanie pewnych zapisów, np. *Pan, Sz.P.*, itp.

3.1. Techniki wstępnego przetwarzania

Techniki wstępnego przetwarzania mają na celu przekształcenie danych wejściowych do postaci reprezentacji umożliwiającej ich technicznie prostsze i dokładniejsze wzajemne porównywanie.

[9] Warto zauważyć, że porównywanie wykonywane przy wprowadzaniu nowych danych do bazy jest na dobrą sprawę jednym z wielu porównań wykonywanych podczas wyszukiwania duplikatów w już istniejącej bazie.

3.1.1. Reguły interpunkcyjne

We wszystkich współczesnych językach naturalnych (w tym w języku polskim), a nawet w większości języków sztucznych (np. programowania) można doszukać się dwóch grup znaków przestankowych – takich, które na ogół rozdzielają wyrazy od siebie oraz takich, których występowanie na ogół sugeruje pewien bliższy związek pomiędzy wyrazami. Do pierwszej grupy należy zaliczyć przede wszystkim: spacje, nawiasy okrągłe, znaki matematyczne (z wyjątkiem znaku minus, utożsamianego niekiedy z myślnikiem), ukośniki, cudzysłowy, przecinki, średniki, dwukropki, itp. Do grupy drugiej należą za to: kropki, myślniki, tzw. małpy, apostrofy, tyldy, itp. Co za tym idzie, przy usuwaniu dodatkowej interpunkcji pierwsze z tych znaków warto zastępować znakiem spacji, podczas gdy drugie po prostu eliminować z napisu. Ten prosty zabieg pozwala usunąć zbędne znaki interpunkcyjne, pozostawiając napis w dostosowanej do dalszego porównywania formie.

3.1.2. Znaki narodowe

Kolejnym problemem są znaki narodowe. Ze względu na sposób uzyskiwania znaków narodowych na standardowej klawiaturze (zwykle ALT+klawisz), użytkownicy, wprowadzając dane do systemu, co jakiś czas (niektórzy być może zawsze) popełniają literówki, wstawiając zamiast znaku narodowego jego łaciński odpowiednik. Aby doprowadzić do ujednolicenia wszystkich takich zapisów, należy wszystkie znaki narodowe obligatoryjnie zastąpić ich odpowiednikami łacińskimi.

3.1.3. Metody leksykograficzne

Metody leksykograficzne to kolejny zbiór prostych algorytmów pozwalających z kolei na usuwanie zbędnych, nieznaczących zapisów (typu *Pan, mgr*). Po zbudowaniu odpowiedniego słownika znanych zapisów[10], samo usuwanie jest już intuicyjne i polega na pozbyciu się zapisu, który występuje w słowniku, z aktualnie opracowywanej nazwy. Należy przy tym rozważyć dwa sposoby usuwania takich zapisów:

- przed wykonaniem porównania skrótowego – usunięciu podlegają zapisy niewchodzące w skład skrótów, np. *spółka akcyjna, szanowny pan* oraz wszystkie dozwolone ich formy skrótowe *s.a., SzP*, w dowolnej konfiguracji interpunkcyjnej;
- po porównaniu skrótowym, lecz przed porównaniem tekstowym – usunięciu podlegają zapisy wchodzące w skład skrótów, lecz niewnoszące wiele w semantykę nazwy, np. *przedsiębiorstwo, szkoła*;

3.2. Algorytmy selekcji danych

Algorytmy selekcji danych służą ograniczeniu zakresu danych poddawanych dalszemu przetwarzaniu. Algorytmy te mają na celu jak najszybsze określenie, które z par wyrazów zdecydowanie nie są identyczne, bez dokonywania pełnej oceny

[10] Budowa słownika zależy od dziedziny, w której słownik będzie zastosowany, i zajmuje z reguły od kilku dni w sytuacjach oczywistych, do nawet kilku miesięcy.

ich zgodności. Poniżej zaproponowano autorskie podejście do zagadnienia selekcji danych. Wskazane sugestie wynikają z obserwacji zachowań człowieka przy porównywaniu danych oraz z analizy działania algorytmów porównywania i pomimo braku konkretnego uzasadnienia teoretycznego okazują sie być bardzo pomocne w eliminacji zbędnych porównań. Warto podkreślić, że ich zastosowanie prowadzi do wyników „na ogół identycznych" z porównywaniem bez wykorzystania tych algorytmów. Istnieje jednak szansa, że zastosowanie jednego bądź kilku z nich ograniczy zbiór wynikowy. Omawiane algorytmy to:

- warunek na długość – jeżeli dwa napisy różnią się długością o więcej niż 10 znaków – są zdecydowanie różne;
- warunek na zawieranie – jeżeli napis A ma co najmniej 9 znaków, wybieramy z niego 2 pierwsze i 1 ostatni trigram; jeżeli żaden z tych trigramów nie występuje w wyrazie B – wyrazy są zdecydowania różne.

3.3. Algorytmy porównywania nazw

Algorytmy porównywania nazw służą do określania procentowej zgodności pomiędzy dwoma nazwami. Zgodność może dotyczyć zarówno syntaktyki napisu (zapis fonetyczny, wymowa, odmiany), jak i jego semantyki (synonimy, wyrazy bliskoznaczne). Opisane algorytmy są zasadniczą częścią prezentowanego rozwiązania stanowiącą o jego skuteczności i efektywności.

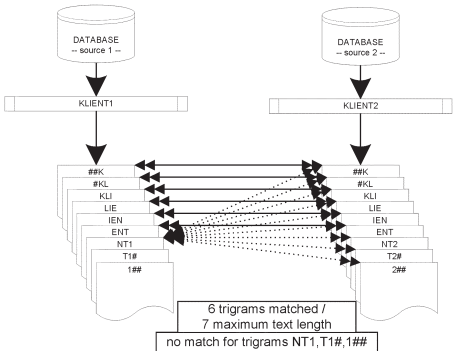
3.3.1. Porównywanie trigramowe

Porównywanie trigramowe[11] jest prostą koncepcją umożliwiającą stwierdzenie zgodności dwóch wyrazów (wyraz A, wyraz B) jedynie na podstawie ich budowy (kolejności i rodzaju liter wchodzących w jego skład). Rozpoczynamy od podzielenia wyrazu A na podciągi 3–literowe w ten sposób, by każda z liter wyrazu stanowiła początek dokładnie jednego takiego podciągu (np. wyraz 'klient' dzielimy na 'kli','lie','ien','ent'), tworząc tzw. trigramy (rysunek 1).

Nastepnie wewnątrz wyrazu B poszukujemy trigramów wyrazu A. Liczba wystąpień trigramów wyrazu A w wyrazie B, podzielona przez długość dłuższego z wyrazów, daje miarę prawdopodobieństwa, że oba wyrazy są identyczne. Dodatkowym krokiem przed podziałem na trigramy powinno być dodanie do wyrazów A i B na początku i na końcu dokładnie dwóch znaków specjalnych (najlepiej odpowiadających zastosowanym separatorom wyrazów – zwykle spacji). Dzięki temu każda litera wyrazu A występowała będzie w dokładnie trzech trigramach, co sprawi, że wynik powinien lepiej odzwierciedlać rzeczywistość.

Zalety porównania trigramowego stanowią: prosta implementacja, duża skuteczność w pomijaniu literówek oraz możliwość określenia procentowej zgodności wyrazów. Podstawową wadą jest natomiast czas porównywania (który dla dłuższych wyrazów przyjmuje znaczące wielkości) oraz niska skuteczność w przypadku wystąpienia opcjonalnych wyrazów typu: *Pan, mgr.*

[11] patrz Crochemore (1994), Kowalczyk (1999)



Rysunek 1. Koncepcja trigramów

3.3.2. Porównywanie skrótowe

Porównywanie skrótowe[12] zdecydowanie różni się od prezentowanego porównania trigramowego. Swoim charakterem zbliżone jest nieco do słownikowych metod wyszukiwania wyrazów podobnych, metod skojarzeniowych. Z tego względu jest zdecydowanie mniej precyzyjne[13]. Porównywanie skrótowe służy do wskazania dwóch nazw, z których jedna stanowi skrócony zapis drugiej. Przy jednoczesnym całkowitym pozbawieniu nazw jakiejkolwiek semantyki podejście takie może prawadzić do błędnych wyników (np. *BPH* to raczej nie to samo co *Bezpieczne Pośrednictwo Handlowe*). Aby sprawdzić, czy nazwa A jest zapisem skrótowym B, należy sprawdzić, czy uda się na podstawie pierwszych liter członów nazwy B (pomijając być może niektóre z nich) uzyskać słowo A. Można przy tym rozpatrywać algorytmy „bez nawrotów" [14] oraz „z nawrotami". Warto zauważyć, iż porównywanie skrótowe nie jest wykonywane w przypadku, gdy liczba członów nazwy A przekracza jedność (kilka wyrazów nie stanowi zazwyczaj skrótu). Mimo to czasami zdarza się, że skrót występuje obok kilku wyrazów (zapisów, zazwyczaj *adhoc* pozbawionych semantyki, np. *spółka cywilna*), które w takiej sytuacji należy uprzednio usunąć.

3.3.3. Ograniczenia algorytmów porównywania nazw

Wybór dowolnego algorytmu transformacji, czy porównywania nazw klientów, z dowolnie ustawionym poziomem zgodności, niestety nie jest w stanie zapewnić, że znalezione dwa zapisy istotnie opisują jednego i tego samego klienta. Nazwa klienta nie jest bowiem jego unikalnym identyfikatorem. Należy zatem poszukiwać innych, dodatkowych danych o kliencie tak, by zminimalizować prawdopodobieństwo znalezienia dwóch nazw nie będących parą oryginał-duplikat. Doskonale do tego celu

[12] patrz Antos (2002)

[13] Skojarzenie jest znacznie trudniejsze do zdefiniowania, niż prosta relacja identyczności. Trudno jest bowiem stwierdzić, iż A kojarzy się z B w 50%.

[14] Porównywanie skrótowe bez nawrotów polega na dopasowywaniu kolejnych liter skrótu A do początkowych liter wyrazu B, bez uwzględnienia możliwości pominięcia jednego pasującego wyrazu z korzyścią dla całkowitej zgodności A i B.

nadają się szczegółowe dane personalne (data urodzenia, pesel), czy też dane adresowe. Niestety, jak łatwo zauważyć, przy porównywaniu dodatkowych danych obserwujemy te same problemy, co w przypadku porównywania nazw (konieczność czyszczenia danych, niska wydajność, itp.). Dodatkowa niska wiarygodność takich danych (liczne błędy, wynikające z nieuwagi wprowadzających lub wykorzystania sprzecznych źródeł informacji o kliencie) oraz niska ich dostępność (braki w danych) sprawiają, iż uwzględnianie danych personalnych, czy adresowych staje się trudne. Do ich analizy należy zastosować kompleksowe algorytmy porównywania.

3.4. Kompleksowe porównywanie – główny algorytm autorski

W pierwszej kolejności użytkownik powinien określić dwa podstawowe parametry wykorzystywane w logice rozmytej:

- próg niezgodności (ozn. *inf*) – procentowa zgodność, poniżej której dwa wyrazy uznawane są za zdecydowanie różne;
- próg zgodności (ozn. *sup*) – procentowa zgodność, powyżej której dwa wyrazy uznawane są za identyczne.

Naszym zadaniem będzie określenie (na podstawie tak zdefiniowanych parametrów) dla której pary klientów z zadanego zbioru[15] zachodzi relacja zgodności[16]. W tym celu porównujemy dwie nazwy klientów (najpierw skrótowo, potem pełnotekstowo – wybieramy większy uzyskany poziom zgodności). Jeżeli poziom zgodności nazw jest mniejszy od przyjętego progu niezgodności – rekordy nie są zgodne. W przeciwnym przypadku kontynuujemy porównywanie dla wszelkich dostępnych danych adresowo-personalnych, po dokonaniu normalizacji dotychczas uzyskanej zgodności do zakresu 0-*sup*%. Za każdą zgodność powyżej ustalonego progu zgodności – przyznajemy dodatkowe procenty zgodności[17] dla całego rekordu. Ostateczny wynik normalizujemy do przedziału 0-100%. Jeżeli znormalizowana wartość przekracza próg zgodności – rekordy uznawane są za zgodne. Schemat blokowy opisanego wyżej algorytmu zamieszczono poniżej (rysunek 2).

4. Zaimplementowane rozwiązanie problemu

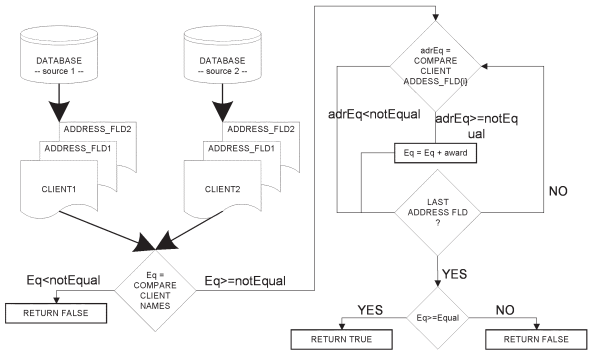
W rozwiązaniu wykorzystano autorski algorytm porównywania kompleksowego, wsparty większością wymienionych powyżej algorytmów wstępnego przetwarzania. Warto zauważyć, iż wszystkie powyższe rozważania były czysto teoretyczne, nie związane z żadną konkretną implementacją. Podczas testów zdecydowano się natomiast na następującą konfigurację środowiska:

- aplikacja testowa została napisana w środowisku Microsoft .NET, w architekturze trójwarstwowej;

[15] Zakładamy, że zbiór jest już zmodyfikowany w oparciu o wszystkie zaprezentowane powyżej algorytmy wstępnego przetwarzania.

[16] Relacja zgodności pomiędzy dwoma zestawami danych klientów zachodzi, gdy owe dane można uznać za należące fizycznie tego samego klienta.

[17] Liczba dodatkowych procentów może być przykładowo równa stosunkowi liczb (1-*sup*)/liczba kolumn adresowo-personalnych.



Rysunek 2. Podejście autorskie w porównywaniu rekordów

- dane z bazy danych aplikacja odczytywała z bazy i przechowywała w RAM (choć przewidziane jest umożliwienie jej pracy w innych trybach);
- testy wykonywane były na komputerze klasy PC, z procesorem klasy Pentium IV i pamięcią 1GB RAM, w środowisku MS-Windows 2000;
- aplikacja korzystała za pośrednictwem protokołu ODBC z danych w bazie Oracle 8i, umieszczonej na tym samym komputerze;
- ze względów wydajnościowych oraz bezpieczeństwa danych aplikacja nie dokonywała modyfikacji w bazie danych (jedynie pojedynczy pełen odczyt danych w chwili uruchomienia porównania).

4.1. Testowanie rozwiązania

4.1.1. Dane testowe

Testy przeprowadzono na rzeczywistym zbiorze testowym składającym się z ponad 2.000 rekordów, zawierającym dane firm, składającym się w około 5% z danych o wartości null (przy czym występowało kilka rekordów dla których wszystkie pola poza nazwą przyjmowały wartość null). Dane zostały dokładnie oznaczone przez właściciela bazy danych pod kątem istniejących w nich duplikatów. Zaznaczone duplikaty są niekiedy bardzo specyficzne i opierają się w dużej mierze na wiedzy eksperckiej (np. firma globalna i jej przedstawiciel w Polsce zarejestrowane pod tym samym adresem), stąd wyniki pracy można uznać nawet za nieco zaniżone (dla danych typu imię + nazwisko wyniki byłyby znacznie lepsze).

4.1.2. Przyjęte miary

Miary przyjęte podczas testów powinny określać zarówno efektywność, jak i jakość prezentowanych algorytmów, a dzięki temu ich użyteczność w zastosowaniach przemysłowych. Za takie uznano:

Lp	Zapisy	Skrotowe	Tekstowe	Max zgodnosc	Min zgodnosc	Rozmycie	Czas	Rekordow	Precyzja	Pokrycie
1	-	-	pobiezne	80	50	T	00:00:21.750	16	100,00%	12,90%
2	przed+po	-	pobiezne	80	50	T	00:00:26.906	23	100,00%	18,55%
3	przed+po	pobiezne	pobiezne	70	50	T	00:00:26.093	61	54,10%	26,61%
4	-	pobiezne	pobiezne	80	50	T	00:00:22.781	19	89,47%	13,71%
5	przed+po	pobiezne	pobiezne	80	50	T	00:00:28.796	47	51,06%	19,35%
6	-	pobiezne	skrocone	80	50	T	00:00:35.875	967	5,89%	45,97%
7	przed	pobiezne	skrocone	80	50	T	00:00:36.906	400	14,75%	47,58%
8	po	pobiezne	skrocone	80	50	T	00:00:36.718	1193	5,11%	49,19%
9	przed+po	pobiezne	skrocone	80	50	T	00:00:37.593	206	32,04%	53,23%
10	przed+po	z nawrotami	pobiezne	80	50	T	00:00:29.796	329	11,25%	29,84%
11	przed+po	z nawrotami	skrocone	80	50	T	00:00:39.765	486	16,26%	63,71%
12	przed+po	z nawrotami	pelne	80	50	T	00:01:35.015	490	16,33%	64,52%
13	przed+po	bez nawrotow	pelne	80	50	T	00:01:30.609	241	32,78%	63,71%
14	przed+po	pobiezne	pelne	80	50	T	00:01:31.468	211	32,23%	54,84%
15	przed+po	-	pelne	80	50	T	00:01:33.218	187	35,83%	54,03%
16	przed+po	bez nawrotow	skrocone	80	50	T	00:00:37.921	237	32,91%	62,90%
17	przed+po	bez nawrotow	skrocone	80	30	T	00:00:38.625	651	13,21%	69,35%
18	przed+po	bez nawrotow	skrocone	80	30	N	00:00:38.734	627	11,48%	58,06%
19	przed+po	bez nawrotow	skrocone	70	30	N	00:00:38.015	748	11,76%	70,97%
20	przed+po	bez nawrotow	skrocone	70	70	T	00:00:38.984	212	34,91%	59,68%
21	przed+po	bez nawrotow	skrocone	80	80	T	00:00:37.718	81	65,43%	42,74%
22	przed+po	bez nawrotow	skrocone	90	90	T	00:00:38.953	46	80,43%	29,84%
23	przed+po	bez nawrotow	skrocone	100	100	T	00:00:39.343	27	100,00%	21,77%

Tabela 1. Wyniki testów

- średni czas przetwarzania (ang. *average timing*) – całkowity czas przeszukiwania całej bazy danych w poszukiwaniu duplikatów[18];
- precyzja (ang. *precision*) – stosunek liczby znalezionych par duplikatów do liczby wszystkich znalezionych par;
- pokrycie (ang. *coverage*) – stosunek liczby znalezionych par duplikatów do liczby wszystkich par duplikatów w bazie danych.

4.1.3. Wyniki

Ze względu na objętość artykułu, przedstawiono zestawienie jedynie najciekawszych scenariuszy testowych, prezentuje je tabela 1. Warto zwrócić uwagę, iż w zbiorze testowym aż 13% duplikatów stanowiły oczywiste powtórzenia[19] (wiersz 1) – wystarczyło usunąć polskie znaki, uporządkować interpunkcję oraz wielkości liter. Po dodatkowym usunięciu zbędnych zapisów (wiersz 2), oczywistych powtórzeń było już 18%. Dodanie porównania pełnotekstowego (wiersz 15), czy dowolnej wersji porównania skrótowego (wiersze 12-14) skutecznie poprawiło ten rezultat, zmniejszając jednak drastycznie precyzję otrzymywanych wyników. Spadek precyzji należy tłumaczyć głównie przez zastosowanie niskiego progu zgodności (80%) dla danych testowych oraz stosunkowo duże braki w danych[20]. Próby manipulacji zadanymi współczynnikami zgodności prezentują wiersze 20-23. Jak łatwo zauważyć wybór progu na poziomie 80% istotnie nie zapewniał wysokiej skuteczności, po-

[18] Testy na średni czas przetwarzania wykonano kilkukrotnie przy tym samym doborze algorytmów, odrzucając pierwszy uzyskany wynik (każde kolejne odwołanie do bazy było szybsze ze względu na odwołania do bufora, zamiast do danych, przez SZBD), a otrzymane wyniki uśredniono arytmetycznie.

[19] Przez oczywiste powtórzenie rozumie się parę rekordów, które są nierozróżnialne ze względu na zawarte w nich dane.

[20] Algorytm uznaje niewypełnioną (NULL) wartość kolumny za zgodną w 100% z pozostałymi wartościami w tej kolumnie.

zwalał jednak na wychwycenie znacznej liczby duplikatów[21]. Warto też zauważyć, iż wprowadzenie porównywania skrótowego z nawrotami spowodowało w najlepszym przypadku (wiersze 12-13) zaledwie 1% wzrost pokrycia przy jednoczesnym znacznym obniżeniu precyzji. Wynika z tego, iż dokładniejsze metody analizy nie zawsze muszą okazać się lepsze[22] (zwłaszcza że czas porównania był dłuższy dla tak niewielkiego zbioru o całe 5 sekund). Usunięcie rozmycia przy wyznaczaniu poziomu zgodności (wiersz 18) pogorszyło jedynie otrzymany wynik. Godną uwagi jest również różnica w czasie przetwarzania pomiędzy wersją porównywania pełnotekstowego skróconą[23], a pełną (wiersze 11-12). W wersji pełnej czas przetwarzania był blisko 3 razy dłuższy, co przy poprawie pokrycia i precyzji o niecałe 1% sugeruje możliwość zastąpienia metody pełnej przez skróconą w zastosowaniach przemysłowych.

Za największe osiągnięcie niniejszej pracy należy natomiast uznać uzyskanie:

- blisko 100% precyzji, pozwalając na zautomatyzowanie procesu deduplikacji (przy zachowaniu dość dobrego pokrycia rzędu 20% [24]) w bazach danych;
- pokrycia rzędu 60% (niedopasowane pozostają rekordy, które bez wprowadzenia synonimizacji nie mogą być uznane za identyczne), co przy blisko 30% precyzji daje użytkownikom spore szanse na przejrzenie wygenerowanej listy duplikatów w rozsądnym czasie i podjęcie decyzji o usunięciu duplikatów.

5. Podsumowanie

Jak wynika z prezentowanych wyników, istnieją całkiem efektywne algorytmy wyszukiwania duplikatów klientów oparte na porównywaniu pełnotekstowym. Algorytmy te cechują się dużą skutecznością zarówno w zakresie pokrycia, jak i precyzji, a tym samym mogą być wykorzystywane (naturalnie przy odpowiedniej dla każdego przypadku konfiguracji) zarazem jako narzędzia automatycznej korekty danych jak i wyszukiwania duplikatów w istniejącej bazie danych.

Ich główną wadą jest natomiast wydajność (czas przetwarzania), która dla większych zbiorów danych jest narazie nieakceptowalna. Porównywanie rekordów metodą *każdy z każdym*[25], przy liczbie rekordów powyżej 200.000, wykonywałoby się blisko 36 godzin, a powyżej kilku milionów jeszcze dłużej (rzędu tygodni, czy miesięcy). Oczywiście można dyskutować, czy w rozwiązaniu posłużono się maksymalnie zoptymalizowanym kodem programu, czy wybrano dostatecznie szybką maszynę testową, jakie w końcu było obciążenie tej maszyny innymi pracami w trakcie

[21] Eksperymentalne wyznaczanie współczynników zgodności pozostawia się użytkownikowi, który musi zdecydować jaki poziom precyzji i pokrycia jest dla niego satysfakcjonujący

[22] Dzięki zastosowaniu metody „z nawrotami" liczba wykrytych duplikatów wzrosła bowiem blisko dwukrotnie, ale wskazane zostały w większości nazwy zdecydowanie różne, choć możliwe do interpretowania jako skróty i ich rozwinięcia.

[23] Metodę skracania omówiono w ramach algorytmów selekcji danych w niniejszej pracy.

[24] Pokrycie mogłoby być znacznie wyższe w przypadku uprzedniego wyczyszczenia danych – w testach operowano na danych niewyczyszczonych.

[25] W rzeczywistości wykonywane jest porównanie pierwszy z kolejnymi, drugi z kolejnymi, ... bez porównań wstecznych. Zastosowanie efektywniejszych algorytmów jest w tym przypadku niemożliwe ze względu na brak uszeregowania krotek według miary zgodności (tylko przy porównywaniu dwóch krotek da się wyznaczyć miarę zgodności między nimi).

procesu porównywania – niemniej jednak przy tej złożoności problemu (klasy N^2) usprawniając każdy z wymienionych elementów jesteśmy w stanie jedynie przesunąć granicę wykonalności obliczeń nieco wzwyż, nie jesteśmy natomiast w stanie rozwiązać problemu, zwłaszcza dla dużych baz danych.

Istnieje natomiast możliwość zmodyfikowania algorytmu o dodatkowy krok – grupowanie, co pozwoliłoby na ograniczenie liczby porównań, a tym samym skrócenie czasu dopasowywania. Dobór odpowiednich algorytmów grupowania oraz rozszerzenie obecnej funkcjonalności aplikacji pozostaje zatem przedmiotem kontynuacji prac badawczych.

Literatura

Antos, M. (2002) *Analiza algorytmów porównywania i wyszukiwania tekstów w bazach danych w zastosowaniu do katalogów bibliotek Politechniki Warszawskiej.* Praca Magisterska, Instytut Automatyki i Informatyki Stosowanej Politechniki Warszawskiej, Warszawa.

Crochemore, M. and Rytter, W. (1994) *Text algorithms.* Oxford University Press, USA.

Fryman, H. (2004) *Wrestling Value from All that Data.* Article from Informatica ViewPoint (`www.informatica.com`).

Gaździk, J. and Musiał, E. (2004) *Internetowy leksykon geomatyczny.* Polskie Towarzystwo Informacji Przestrzennej.

Kowalczyk, A. (1999) *Porównywanie opisów bibliograficznych w oparciu o algorytmy tekstowe, odporne na błędy.* Praca Magisterska, Instytut Automatyki i Informatyki Stosowanej Politechniki Warszawskiej, Warszawa.

Nguyen, T. and Fisher, T. (2005) *The case for ETL – merging data warehousing and data quality, because bad data costs companies money.* SAS Institute and DataFlux Corporation.

Usage of fullscan text mining algorithms in customer matching

Todays systems using databases (especially data warehouses) do not efficiently cope with client data duplication. The impact of one person existing more than once in the database are, despite the natural order disturbances, numerous business losses resulting mostly from falsified business analysis or redundant marketing offer delievery. This article presents and valuates efficiency of some fullscan text comparison approaches in customer matching algorithms helping to reduce the number of duplicates to minimum.