Politechnika Poznańska
Wydział Informatyki
Instytut Informatyki

Hurtownie danych i przetwarzanie analityczne

OCENA TECHNIK I NARZĘDZI DE-DUPLIKACJI DANYCH

Damian Litwin, 126167
Kasper Bojaruniec, 127221
Konrad Szymański, 127240
Mikołaj Leśny, 127218
Mikołaj Musidłowski, 127251
Piotr Kolasiński, 127110

Prowadzcy
prof. dr hab. inż. Robert Wrembel

Poznań, 2019

Spis treści

Spis treści **1**

1 Wprowadzenie **2**
1.1 Opis problemu . . . 2
1.2 Wykorzystywane zbiory danych . . . 2
1.3 Rozpartywane narzędzia . . . 3
1.4 Kryteria wyboru oraz wybrane narzędzia . . . 3

2 Opis projektu **4**
2.1 Opis danych testowych . . . 4
2.1.1 Zbiór danych Amazon - Google Products . . . 4
2.1.2 Zbiór wygenerowanych danych personalnych . . . 4
2.2 Opis generatora danych . . . 5
2.3 Oceniane cechy . . . 7
2.4 Przykłady użycia wybranych narzędzi . . . 9
2.4.1 Dedupe . . . 9
2.4.2 SERF . . . 11
2.4.3 Record Linkage . . . 12
2.4.4 Data Quality and Profiling . . . 13
2.4.5 Talend Open Studio . . . 16
2.4.6 JedAI . . . 20

3 Podsumowanie **25**
3.1 Ocena narzędzi . . . 25
3.2 Wybór najlepszego narzędzia . . . 26

Rozdział 1

Wprowadzenie

1.1 Opis problemu

W obecnych czasach większość ludzi posiada konta bankowe. Często zdarza się tak, że w celu uzyskania korzyści, pojedyncza osoba zakłada konta w kilku bankach jednocześnie. W biznesie bankowym mają miejsce fuzje lub wykupienia, w momencie których, klienci mniejszego banku przechodzą do banku większego. W efekcie może dojść do sytuacji, w której pojedyncza osoba ma w jednych banku więcej niż jedno konto. Takie zdarzenie wymaga połączenia kont w jedno. Jednak nie zawsze jest to takie oczywiste. Często bywa jednak, że dane tego samego klienta w dwóch jego kontach nie są identyczne, mimo, że reprezentują tą samą osobę. Spowodowane jest to tym, że jego dane uległy zmianie, zostały błędnie wprowadzone do systemu przez pomyłkę klienta lub pracownika banku, nie zostały wprowadzone wcale, lub też zostały przypadkowo usunięte z systemu. Przejrzenie wszystkich kont bankowych w celu wyszukania duplikatów jest procesem zbyt kosztownym, by został wykonany przez pracowników banku. Porządane jest zatem specjalistyczne oprogramowanie wykrywające w bazie danych duplikaty kont. Jednak duplikaty te nie muszą być identyczne, a dane mogą zawierać szumy. Celem tego projektu jest wyszukanie takiego oprogramowania, wybrania sześciu najbardziej obiecujących propozycji, przeprowadzenie na nich testów oraz przygotowanie zestawienia prezentującego wyniki.

1.2 Wykorzystywane zbiory danych

W sieci można wyszukać wiele zbiorów danych, które wykorzystywane są jako benchmarki w testach software'u stworzonego do de-duplikacji danych. Jednak nie można wyszukać zbioru zawierającego przykładowe danne klientów polskich banków. Może być to spowodowane np. RODO. Banki mimo istniejącego, coraz większego problemu z posiadaniem nadmiarowych danych, nie udostępniają w sieci nawet próbki tego, jak wygląda ich baza danych. W związku z tym problemem postanowiono do testów oprogramowania wykorzystać dwa zbiory danych. Pierwszym z nich jest benchmarkowy zbiór Amazon - Google Products Dataset. Jest on zbiorem produktów wystawianych w sklepach internetowych Amazon oraz Google Products (pliki przestały być dostępne w serwisie https://dbs.uni-leipzig.de/de, z którego zostały pierwotnie pobrane, jednak ich kopie zostały zapisane w repozytorium jednego z testowanych narzędzi: https://github.com/scify/JedAIToolkit/tree/mavenizedVersion/jedai-core/data/cleanCleanErDatasets). Zawiera on różne rekordy dla tych samych produktów wystawianych w tych serwisach internetowych. Produkty te, mimo że są identyczne, mogą posiadać wprowadzone inne nazwy, opisy, ceny i producentów. Dane te są obszerne i mocno zaszumione. Wraz z benchmarkiem dostarczany jest plik z prawidłowych dopasowaniem, w którym można znaleźć informacje, które rekordy z dwóch sklepów odpowiadają

tym samym produktom. Postanowiono przygotować różnież skrypt, który ma za zadanie wygenerować zbiór danych oraz zaszumić go, tak aby mógł przypominać przyjęte wyobrażenie zbioru danych w polskich bankach.

1.3 Rozpartywane narzędzia

Równolegle z przygotowaniami zbiorów testowych, wyszukiwano w sieci najbardziej obiecujące narzędzia, do przeprowadzenia na nich testów. W poniższej tabeli przedstawiono oprogramowanie, które zostało włączone w rozważania. Bardziej szczegółowe informacje o wybranch przez zespół projektowy narzędziach, można znaleźć w sekcji 2.4.

Nazwa	Forma	Bezpłatne	Metoda
Dedupe	Biblioteka do Pythona	Tak	Uczenie maszynowe
SERF	Biblioteka do Javy	Tak	Techniki porównywania
Record Linkage	Biblioteka do Pythona	Tak	Techniki porównywania
OSDQ	Aplikacja desktopowa	Tak	Logika rozmyta
Data Cleaner	Aplikacja desktopowa	Tak	Logika rozmyta
Talend Open Studio	Aplikacja desktopowa	Tak	Techniki porównywania
Duke	Biblioteka do Javy	Tak	Techniki porównywania
DemandTools	Aplikacja desktopowa	Tak	Techniki porównywania
Cloudingo	Aplikacja desktopowa	Nie	Techniki porównywania
Data Quality and Profiling	Aplikacja desktopowa	Tak	Techniki porównywania
JedAI	Aplikacja desktopowa	Tak	Techniki porównywania
WinPure	Aplikacja desktopowa	Nie	Techniki porównywania
DataMatch	Aplikacja desktopowa	Nie	Techniki porównywania

Rysunek 1.1: Rozważane narzędzia

1.4 Kryteria wyboru oraz wybrane narzędzia

Następnym krokiem było ustalenie przez zespół krytierów wyboru software'u do testów. W pierwszej fazie postanowiono odrzucić płatne rozwiązania, ze względu na to, że projekt ten nie posiada budżetu. Dzięki temu lista narzędzi zawężyła się z 13 do 10. Należało jeszcze odrzucić cztery z nich. Zespół postanowił postawić na różnorodne rozwiązania. Jeśli któreś z nich zostało napisane w tym samym języku i wykorzystywało te same sposoby wykrywania duplikatów, odrzucano gorzej oceniane na forach, z gorszą dokumentacją oraz z mniejszym wsparciem producenta lub użytkowników. Na podstawie tych kryteriów udało się wyłonić ostateczną szóstkę faworytów do przeprowadzenia testów:

- Dedupe
- SERF
- RecordLinkage
- Data Quality and Profiling
- Talend Open Studio
- JedAI

Rozdział 2

Opis projektu

2.1 Opis danych testowych

2.1.1 Zbiór danych Amazon - Google Products

Zbiór ten składa się z danych na temat produktów umieszczonych w sklepach Amazonu i Google'a. Zbiór ten jest opisany następującymi atrybutami: id, title, description, manufacturer, price. Niektóre produkty występują w obu zbiorach pomimo innych wartości w polach. Wraz ze zbiorem danych w projekcie został wykorzystany plik z dokładnym dopasowaniem, na podstawie którego wyniki narzędzi zostały sprawdzone oraz zostały dla nich policzone miary skuteczności. Na poniższym zrzucie ekranu zaprezentowany został fragment tego zbioru.

b000tdgyks	enemy territory: quake wars	enemy territory: quake wars r	aspyr media	49.99
b000pihtbk	weekly reader mastering elementary/middle school math learning		fogware publishing	19.99
b0002yiuns	sentinel: descendants in time	sentinel: descendants in time	dreamcatcher interactive	19.99
b000in8mj0	photostory on cd & dvd 5	magix photostory on cd and c	magix entertainment	19.99
b000cs3s2c	flash remoting 1 alp ret eng cd 2u	- marketing information: macr	adobe	3314.09
b00005bigp	shapes		school zone	9.99
b000h1df7w	dragon naturally speaking standard v9	dragon naturallyspeaking 9 (s	nuance communications inc.	99.99
b000p9cr66	mediarecover	mediarecover gives you the a	aladdin systems	29.99
b000j588g4	photo explosion 3.0	photo explosion 3.0	nova development	29.99
http://www.google.cc	learning quickbooks 2007	learning quickbooks 2007	intuit	38.99
http://www.google.cc	superstart! fun with reading & writing!	fun with reading & writing! is	designed to help kids learn to r	8.49
http://www.google.cc	qb pos 6.0 basic software	qb pos 6.0 basic retail mngm	intuit	637.99
http://www.google.cc	math missions: the amazing arcade adv	save spectacle city by disrupting	randall underling's plan to	12.95
http://www.google.cc	production prem cs3 mac upgrad	adobe cs3 production premiu	adobe software	805.99
http://www.google.cc	video studio 11 plus	corel video studio 11 plus is c	corel corporation	103.99
http://www.google.cc	edius pro 4	whether you are working with	canopus/grass valley	585.99
http://www.google.cc	qb pos 6.0 pro multi store sw	qb pos 6.0 pro multistore reta	intuit	1054.99

Rysunek 2.1: Fragment zbioru danych Amazon - Google Products

2.1.2 Zbiór wygenerowanych danych personalnych

Wygenerowany zbiór danych personalnych jest zbiorem, który w domyśle zespołu projektowego może wyglądać jak zbiór zduplikowanych danych w banku. Każdy klient jest opisany poprzez następujące atrybuty: nazwisko, imię, PESEL, data urodzenia, kod pocztowy, miasto, ulica, nr domu. Wszystkie dane dobierane są losowo. Na poniższym zrzucie ekranu przedstawiony został fragment wygenerowanego zbioru.

nazwisko	imie	pesel	data urodzenia	kod pocztowy	miasto	ulica	nr domu
GUNIA-SOŁTAN	LORRY	97082446290	1997-08-24	91-179	SADOWIE	ŁUKASZEWSKIEGO	291
LICHTARSKA	SELMA	78091974186	1978-09-19	41-706	PRZEMĘT	LIPOWA	339
POŻYCZKI	NICKY	54031836121	1954-03-18	78-201	SOKOŁÓW MAŁOPOLSKI	ANDERSA	11
KOWALCZYK-WÓJCIK	HILDAGARD	95100289567	1995-10-02	20-751	NIECHLÓW	NADWARCIAŃSKA	140
WÓJCIK-KOWALCZYK	ARLEE	76091093476	1976-09-10	35-304	BRODNICA	KLEEBERGA	83
KIERBLEWSKI	ROCKY	37121159399	1937-12-11	26-900	ODRZYWÓŁ	BRZEZINA	251
CIEGERT	SABA	27110205143	1927-11-02	41-700	LISIA GÓRA	KOŚCIELNA	265
JÓRZAK	PATTIE	27042788344	1927-04-27	41-500	ZBROSŁAWICE	MAKUSZYŃSKIEGO	472
SELWESTROWICZ	TOMI	16062227089	1916-06-22	35-302	MIĄCZYN	ZARATAJKA	424
PRAWIC	ALDRICH	61092106873	1961-09-21	20-851	MORZESZCZYN	MAGNOLII	28
KRYSMAN	ROSALEEN	04092748692	1904-09-27	03-201	RADOMYŚL NAD SANEM	SASANKI	422
WISTA	MICHELE	98113037064	1998-11-30	51-647	ROGOWO	KRÓTKA	169
TASDEMIR	ZONNYA	50050584330	1950-05-05	69-200	IWONICZ-ZDRÓJ	WOLNOŚCI	259
KULENKO	JACKLYN	51032701952	1951-03-27	42-600	DOMANIÓW	OŻAROWSKA	24

Rysunek 2.2: Fragment wygenerowanego zbioru danych personalnych

2.2 Opis generatora danych

Generator danych składa się z dwóch części: skryptu tworzącego rekordy oraz skryptu zaszumiającego dane. W pierwszym etapie z plików CSV zawierających listy wartości danego parametru losowane są wartości, które następnie sklejane są w jeden rekord. Szansa powstania duplikatu jest tak mała, została pominięta w rozważaniach. Następnie skrypt, którym dane są zaszumiane przyjmuje następujące parametry:

- % duplikatów (parametr duplicated_rows)
- % szansy usunięcia PESELu w duplikacie (parametr swap_chance)
- % szansy zamiany wartości pól w duplikacie (parametr swap_chance)
- % niepoprawnych znaków w nazwisku i imieniu (parametr typo_percent)

Aby uruchomić generator należy wywołać skrypt wraz z odpowiednimi parametrami. Jeśli chcemy zduplikować 30% wierszy, mieć 50% szans na usunięcie numeru PESEL i zamianę pól oraz 70% znaków z nazwisku i imieniu na niewłaściwym miejscu, to wywołanie będzie wyglądać następująco:

python scramble_set.py --duplicated_rows=30 --swap_chance=50 --typo_percent=70

Tak przygotowany generator umożliwia przeprowadzenie testów na tym samym zbiorze z różnymi stopniami zaszumienia.

Kod skryptu zaszumiającego dane:

```
import csv
import random
import click
from math import ceil


def swap_letters(text, first, second):
    text_l = list(text)
    text_l[first], text_l[second] = text_l[second], text_l[first]
    return ''.join(text_l)
```

```
def scramble(swap, percent, chance):
    chance = 100 - chance/100
    swap = swap/100
    source = '../datasets/dataset_m.csv'
    pairs = f'../datasets/pairs_{swap}.csv'
    dest = source.replace('.csv', f'_scrambled_{swap}.csv')
    with open(source, 'r', encoding='utf8') as source_f:
        with open(dest, 'w', encoding='utf8') as dest_f:
            with open(pairs, 'w', encoding='utf8') as pairs_f:
                csv_reader = csv.reader(source_f, delimiter=';')
                for line in csv_reader:
                    line.append('id')
                    dest_f.write(';'.join(line) + '\n')
                    break
                id = 0
                lines = []
                # Randomize order of rows
                for line in csv_reader:
                    lines.append(line)
                random.shuffle(lines)
                # Take x% of rows to duplicate
                for line in lines[:int(len(lines)*percent)]:
                    new_line = line.copy()

                    # Swap % of letters in name, surname, pesel, city, street
                    for i in range(ceil(len(new_line[0]) * swap)):
                        a, b = random.sample(range(0, len(new_line[0]) - 1), 2)
                        new_line[0] = swap_letters(new_line[0], a, b)
                    for i in range(ceil(len(new_line[1]) * swap)):
                        try:
                            a, b = random.sample(range(0, len(new_line[1]) - 1), 2)
                            new_line[1] = swap_letters(new_line[1], a, b)
                        except:
                            new_line[1] = swap_letters(new_line[1], 0, 1)
                    for i in range(ceil(len(new_line[2]) * swap)):
                        a, b = random.sample(range(0, len(new_line[2]) - 1), 2)
                        new_line[2] = swap_letters(new_line[2], a, b)
                    for i in range(ceil(len(new_line[5]) * swap)):
                        a, b = random.sample(range(0, len(new_line[5]) - 1), 2)
                        new_line[5] = swap_letters(new_line[5], a, b)
                    for i in range(ceil(len(new_line[6]) * swap)):
                        a, b = random.sample(range(0, len(new_line[6]) - 1), 2)
                        new_line[6] = swap_letters(new_line[6], a, b)
                    # Swap name and surname
                    if random.randint(0, 100) > chance:
                        new_line[0], new_line[1] = new_line[1], new_line[0]
```

```
            # Swap city and street
            if random.randint(0, 100) > chance:
                new_line[5], new_line[6] = new_line[6], new_line[5]
            # Remove PESEL
            if random.randint(0, 100) > chance:
                new_line[2] = ''
            # Change home number
            if random.randint(0, 100) > chance:
                new_line[7] = str(int(new_line[7]) + random.randint(0, 10))
            if random.randint(0, 100) > chance:
                new_line[3] = str(int(new_line[3][:4]) + random.randint(0, 10)) + new_line[3][4:]
            # Write original and new line to output file
            new_line.append(str(id))
            pairs_f.write(str(id) + ';' + str(id + 1) + '\n')
            dest_f.write(';'.join(new_line) + '\n')
            id += 1
            line.append(str(id))
            id += 1
            dest_f.write(';'.join(line) + '\n')
        for line in lines[int(len(lines) * percent):]:
            id += 1
            line.append(str(id))
            dest_f.write(';'.join(line) + '\n')


@click.command()
@click.option('--duplicated_rows', help='Procent zduplikowanych wierszy')
@click.option('--swap_chance', help='Szansa na zamiane pol')
@click.option('--typo_percent', help='Procent zamienionych liter w nazwisku i imieniu')
def main(typo_percent, duplicated_rows, swap_chance):
    scramble(typo_percent, duplicated_rows, swap_chance)


if __name__ == "__main__":
    main()
```

2.3 Oceniane cechy

W celu porównania narzędzi oraz wyboru najlepszego z nich ustalono kryteria ich oceny. W kwestii subiektywnych odczuć członków zespołu podczas pracy z przydzielonymi do nich narzędziami postanowiono wyróżnić trzy cechy:

- Łatwość instalacji w skali od 1 do 10
- Łatwość konfiguracji w skali od 1 do 10
- Łatwość obsługi w skali od 1 do 10

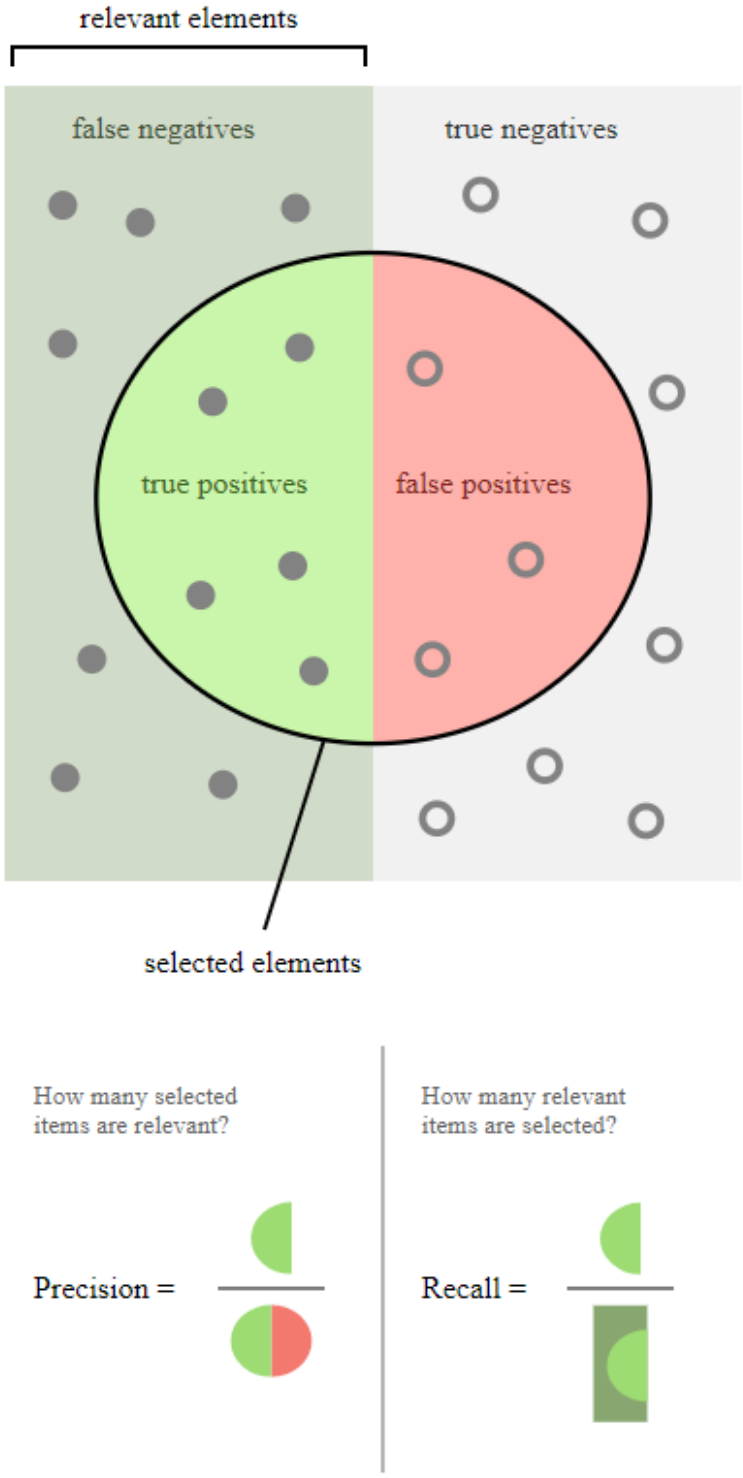
Do porównania jakości i skuteczności wybranych rozwiązanań zostały zaprojektowane testy na obu przygotowanych zbiorach danych. Każde narzędzie zostało przetestowane na zbiorze danych Amazon - Google Products w najlepszej, odszukanej przez danego członka zespołu, konfiguracji dla tego zbioru. Jeśli chodzi o wygenerowany przez skrypt zbiór danych personalnych, testy zostały przeprowadzone przy następujących parametrach wejsciowych skryptu:

- 30% duplikatów
- 70% szans na usunięcie PESELu duplikatu
- 70% szans na zamianę wartości pól duplikatu
- 10%, 30%, 50%, 70% niepoprawnych znaków w każdym polu duplikatu

Skuteczność software'ów została zmierzona na podstawie ogólnoprzyjętych miar benchamrkowych:

- Precission - miara przymująca wartości od 0 do 1, wskazująca jaka część wybranych elementów jest trafnych, obliczana wzorem: true positives/(true positives + false positives)
- Recall - miara przymująca wartości od 0 do 1 wskazująca jaka część trafnych elementów jest wybrana, obliczana wzorem: true positives/(true positives + false negatives)
- F1-score - miara przymująca wartości od 0 do 1, biorąca pod uwagę precission i recall w równej wadze, obliczana wzorem: 2*precission*recall/(precission + recall)

Na rysunku 2.3 została umieszczona graficzna prezentacja miar precision i recall.



Rysunek 2.3: Graficzna prezentacja miar precision i recall. Źródło: wikipedia.org

2.4 Przykłady użycia wybranych narzędzi

2.4.1 Dedupe

1. Instalacja Wymagania: Python 3
   Główne repozytorium projektu Dedupe: https://github.com/dedupeio/dedupe

Narzędzie funkcjonuje jako biblioteka do języka Python. Aby z niej skorzystać należy skorzystać z wybranego instalatora pakietów (np. pip) do pobrania oraz zainstalowania. Aby z niej skorzystać należy zaimportować w skrypcie pakiet dedupe.

2. Konfiguracja i przykład użycia

   a) Pierwszym krokiem jest zaimportowanie biblioteki w celu jej późniejszego użycia. Należy to zrobić w standardowy dla Pythona sposób poprzez słowo kluczowe import dedupe.

   b) Następnie należy wczytać dane, które będą przetwarzane. Docelowo powinny one się znaleźć w postaci tablicy, gdzie indeks odpowiada indeksowi rekordu, a wartością jest słownik zawierający wpisy, gdzie kluczem jest nazwa cechy, a wartością wielkość danej cechy. Przykładowy wiersz powinien ostatecznie wyglądać w następujący sposób:

```
data[724] = {'nazwisko': 'Kowalski',
             'imie': 'Jan',
             'pesel': 89072209876,
             'data_urodzenia': '19890722',
             'kod_pocztowy': '60323',
             'miasto': 'Poznan',
             'ulica': 'Ratajczaka',
             'nr_domu': '33A'}
```

   Dane można pozyskać w dowolny sposób, który umożliwi nam utworzenie wspomnianego słownika. W tym przypadku zostało użyte wczytywanie z pliku oraz odpytanie zdalnej bazy danych PostgreSQL za pomocą biblioteki psycopg2. Dane nie powinny zawierać cudzysłowów i innych wyszczególnionych znaków specjalnych, których usunięcie leży w gestii użytkownika.

   c) Następnie należy zdefiniować model danych. W formie listy słowników trzeba podać nazwy kolumn, które będą wykorzystywane w procesie deduplikacji, typy tych kolumn (w tym przypadku String - dla krótkich ciągów znaków, Text - dla długich ciągów znaków, Price - dla typów numerycznych oraz czy dana kolumna może zawierać wartości puste.

```
{'field' : 'nazwisko', 'type': 'String'},
{'field' : 'imie', 'type': 'String'},
{'field' : 'pesel', 'type': 'Price', 'has missing' : True},
{'field' : 'data_urodzenia', 'type': 'String', 'has missing' : True},
{'field' : 'kod_pocztowy', 'type': 'String', 'has missing' : True},
{'field' : 'miasto', 'type': 'String', 'has missing' : True},
{'field' : 'ulica', 'type': 'String', 'has missing' : True},
{'field' : 'nr_domu', 'type': 'String', 'has missing' : True},
```

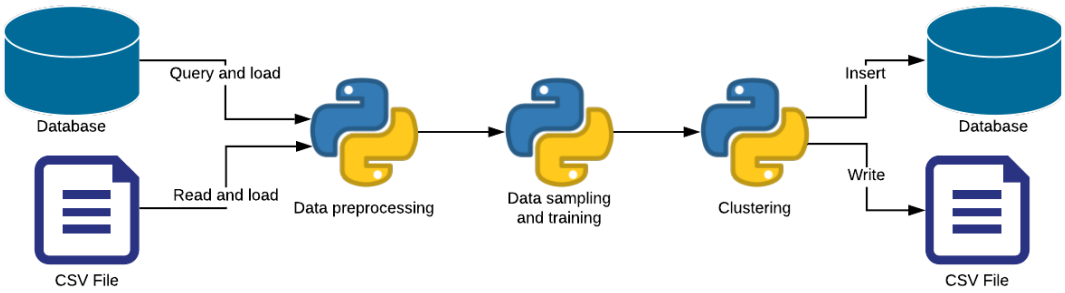
   d) Kolejnym krokiem należy zdefiniować wielkość próbki treningowej oraz rozpocząć trening. Użytkownikowi zostaną przedstawione pary rekordów, które będzie musiał poetykietować. Program wyświetli w konsoli dwa rekordy wraz z wartościami wskazanych wcześniej kolumn dla danego rekordu, na podstawie tych danych należy wprowadzić odpowiedź:

- y - dane rekordy są duplikatem,
- n - dane rekordy nie są duplikatem,
- u - nie da się jednoznacznie stwierdzić czy są duplikatem,
- f - aby zakończyć etykietowanie,
- p - aby powrócić do poprzedniej pary.

Zaleca się znalezienie co najmniej po 10 przykładów dla klasy pozytywnej i negatywnej. Poetykietowane przykłady mogą zostać zapisane w pliku do poźniejszego użycia, aby uniknąć etykietowania przy każdym uruchomieniu programu.

e) Następnie, Dedupe zaaplikuje zdobyte informacje dla reszty danych oraz zwróci rekordy w postaci klastrów, które odpowiadają jednej klasie obiektu/osoby. Dalsze przekształcenie danych jest możliwe za pomocą innych bibliotek i mechanizmów języka Python.

f) Otrzymane wyniki można zapisać do pliku, bazy danych lub przetworzyć w dowolny sposób, który umożliwia język i dostępne biblioteki. W tym przypadku zostają one zapisane do pliku w postaci par indeksów plików, które zostały zidentyfikowane jako duplikaty.



Rysunek 2.4: Schemat przetwarzania w Dedupe

2.4.2 SERF

1. Instalacja
   Wymagane elementy do poprawnego działania biblioteki:

   - Java 6+
   - biblioteka SERF
   - biblioteka commons-io
   - biblioteka commons-lang3
   - biblioteka opencsv (opcjonalne)

   Strona główna projektu: http://infolab.stanford.edu/serf
   Narzędzie funkcjonuje jako biblioteka do języka programowania Java. W celu instalacji, należy utworzyć nowy projekt aplikacji Java, a następnie zaciągnąć wszystkie powyższe biblioteki do projektu.

2. Konfiguracja i przykład użycia

a) Pierwszym krokiem jest ściągnięcie wymaganych bibliotek w formacie jar, a następnie zaimportowanie ich do projektu.

b) W następnym kroku należy zaimplementować obsługę wczytywania danych. Można to zrobić na dwa sposoby:

- przekonwertowanie pliku csv na plik xml
- dodanie obsługi do wczytywania plików csv i zamianę tych danych na typ wymagany dla biblioteki SERF.

c) Następnie należy zaimplementować główny serwis aplikacji do komunikacji z biblioteką. Można tutaj posłużyć się przykładowym serwisem udostępnianym na stronie głównej projektu.

d) W następnej kolejności trzeba stworzyć klasę implementującą interfejs MatcherMerger. Jest to podstawowa klasa konfiguracyjna. Odpowiada za to jakie kolumny mają być porównywane.

e) Po stworzeniu klasy implementującej interfejs MatcherMerger, należy dla każdej porównywanej kolumny stworzyć klasę Matchera implementującego interfejs AtomicMatch. Odpowiada za to jak mają być porównywane ze sobą pola.

f) W ostatnim kroku należy dodać plik konfiguracyjny dla danego datasetu. Przykładowy wygląd pliku można znaleźć na stronie głównej projektu. Należy w nim ustawić parametry takie jak:

- nazwa pliku wyjściowego (plik xml)
- nazwa stworzonej klasy z interfejsem MatcherMerger
- wagi (0-1) branych pod uwagę kolumn

g) Po implementacji można rozpocząć przetwarzania danych. Po jego zakończeniu zostanie wygenerowany wyjściowy plik xml z rezultatem wyników oraz w oknie konsoli pojawi się liczba wierszy przed i po zastosowaniu narzędzia.

2.4.3 Record Linkage

1. Instalacja
   Do poprawnego działania biblioteki wymagane są następujące elementy:

   - Python 3
   - biblioteka Pandas
   - biblioteka Scikit-learn
   - biblioteka Scipy
   - biblioteka Jellyfish

   Główne repozytorium projektu: https://github.com/J535D165/recordlinkage
   Narzędzie funkcjonuje jako biblioteka do języka Python. Aby z niej skorzystać należy skorzystać z wybranego instalatora pakietów (np. pip używając polecenia pip install recordlinkage) do pobrania oraz zainstalowania. Następnie można jej używać w kodzie tworzonych skryptów.

2. Konfiguracja i przykład użycia

a) Pierwszym krokiem jest zaimportowanie biblioteki w celu jej późniejszego użycia. Należy to zrobić w standardowy dla Pythona sposób poprzez słowa kluczowe:
import pandas
import recordlinkage

b) Następnym krokiem jest wczytanie danych (np. z pliku csv) w postaci obiektu typu DataFrame będącego częścią biblioteki Pandas, ponieważ omawiana biblioteka operuje bezpośrednio na tego typu obiektach.

c) Po wczytaniu danych wymagane jest zdefiniowanie sposobu indeksowania danych. Można to zrobić w oparciu o wybraną kolumnę, której jesteśmy pewni, że nie ma w niej duplikatów. Jeśli takiej nie ma, to należy wybrać opcję łączącą rekordy w iloczyn kartezjański. Jest to jednak opcja bardzo nieefektywna obliczeniowo i przetwarzanie może trwać długo.

d) Kolejnym krokiem, po zdefiniowaniu sposobu indeksowania rekordów jest zdefiniowanie sposobu porównywania poszczególnych kolumn. Biblioteka udostępnia wiele różnych kryteriów jak na przykład odległość Levenshteina dla pól tekstowych. Możemy też zdefiniować wagi, lecz jest to ustawienie opcjonalne i domyślnie każda kolumna jest równoważna.

e) Ostatnim elementem wymagającym zdefiniowania jest próg, który służy do decydowania czy dany rekord jest duplikatem, czy nie. Aby wynik był miarodajny należy uwzględnić wagi zdefiniowane w poprzednim punkcie.

f) Można rozpocząć przetwarzanie. Po jego ukończeniu dla każdego rekordu zostanie wygenerowany wynik, który jest porównywany z progiem i dzięki czemu otrzymujemy zbiór obiektów będący duplikatami.

2.4.4 Data Quality and Profiling

1. Instalacja

   Program wymaga Java JDK/JRE 7 lub nowszej. Ze strony https://sourceforge.net/projects/dataquality/ lub http://www.arrahtech.com/getosdq pobieramy plik .zip z narzędziem i go rozpakowujemy. Uruchamiamy program za pomocą pliku runprofiler.bat (w przypadku korzystania z systemu Windows) lub runprofiler.sh(w przypadku systemów UNIX).

2. Konfiguracja i przykład użycia

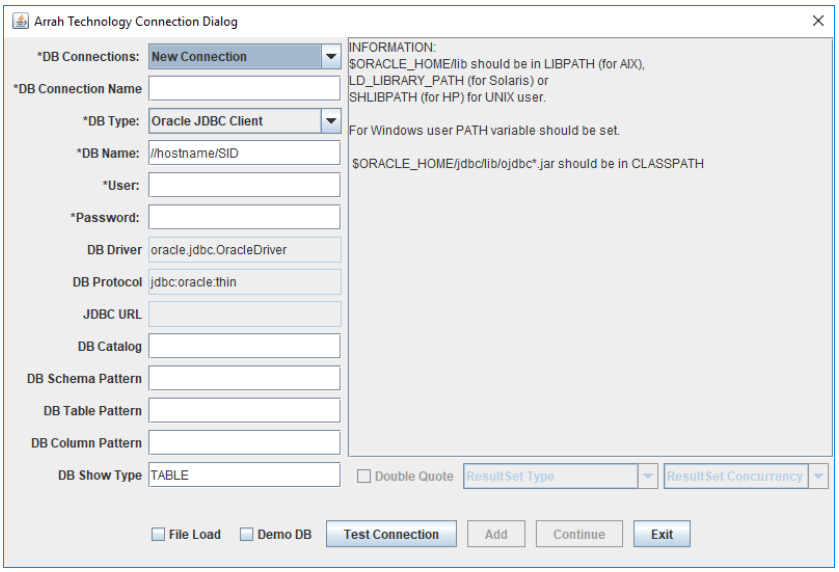
   a) Po uruchomieniu pliku runprofiler zobaczymy ekran startowy(rys. 2.5). W naszym przypadku działamy na plikach .csv, więc wybieramy opcję "Demo DB" by skorzystać z wbudowanej bazy danych.

   b) Następnie w pasku menu wybieramy opcję
   $Tools \rightarrow RecordMatch \rightarrow SingleFileMatch$(rys. 2.6).
   A następnie wybieramy plik z rozszerzeniem .csv (jako seperator ustawiamy przecinek) z danymi do deduplikacji.

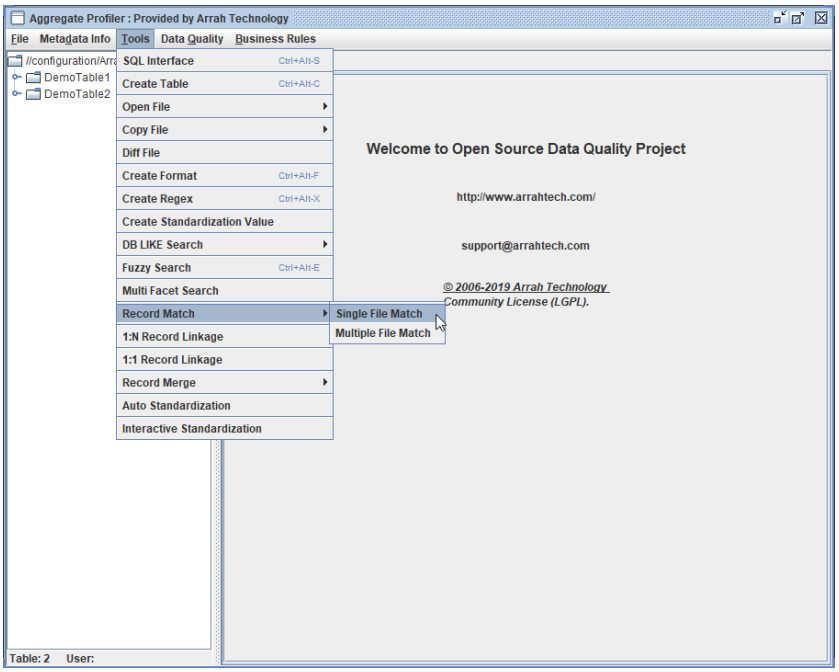
   c) Po wyborze pliku pojawi się ekran z podglądem jego zawartości (rys. 2.7). Przechodzimy do konfiguracji klikając przycisk "Next".

   d) Konfiguracja polega na:

      - wybraniu atrybutów na podstawie, których oceniane będzie podobieństwo



Rysunek 2.5: Ekran startowy



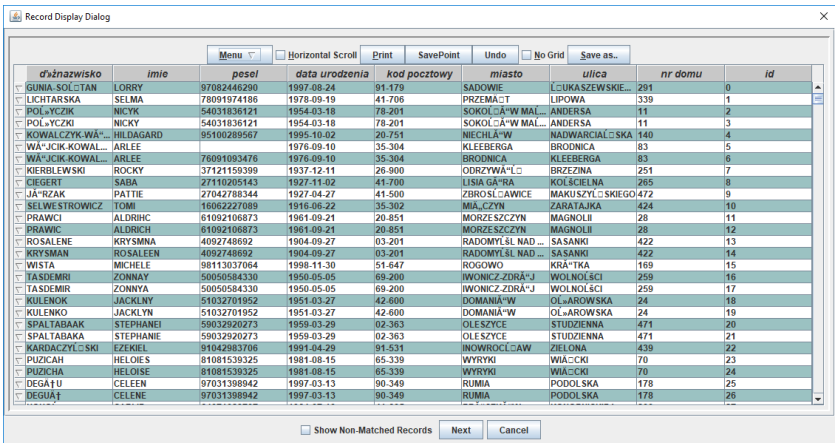
Rysunek 2.6: Wybór narzędzia deduplikacji

- przyporządkowaniu porównywanych atrybutów - w przypadku pojedynczego pliku wybór jest oczywisty, gdy porównujemy dwa pliki już nie,
- wybraniu odpowiedniego algorytmu oceniającego podobieństwo atrybutów,
- wagi podobieństw danych atrybutów.

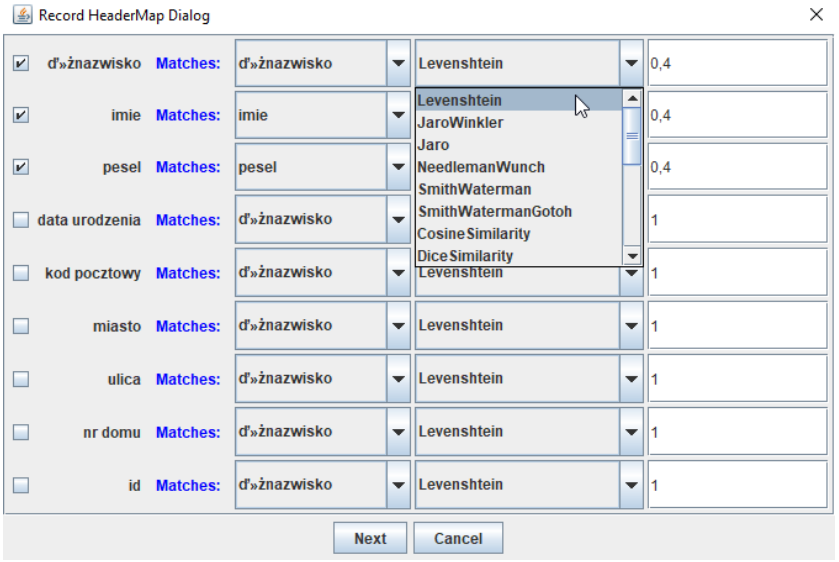
Przykładowa konfiguracja rys. 2.8.

e) Po zakończeniu przetwarzania zobaczymy ekran wynikowy(rys. 2.9), w którym zawarte są informacje o zgrupowaniach odpowiednich rekordów - rekordy należące do tej samej grupy uznane za duplikaty.

3. Dodatkowe informacje ze strony projektu:

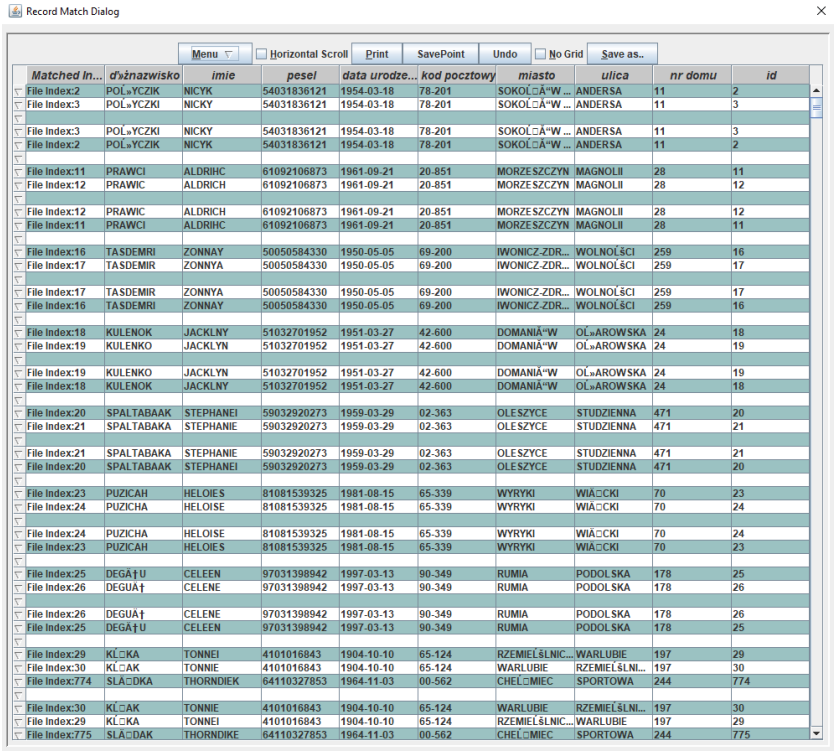


Rysunek 2.7: Podgląd pliku



Rysunek 2.8: Wybór mapowań atrybutów, algorytmu dopasowania oraz wag dopasowań

- Instrukcja instalacji: http://www.arrahtech.com/docs/installation˙guide.html
- Instrukcja użytkowania: http://www.arrahtech.com/docs/profiler˙user˙guide.html



Rysunek 2.9: Ekran wynikowy

2.4.5 Talend Open Studio

1. Instalacja

   Ze strony https://www.talend.com/download/ pobieramy program Master Data Management, poprzez wybranie opcji “Download Free Tool”.

   Następnie wybieramy opcję Windows Download (w przypadku systemu operacyjnego Windows/Linux) lub Mac Download (w przypadku Mac OS).

   Pobrany zostanie plik .zip który należy wypakować.

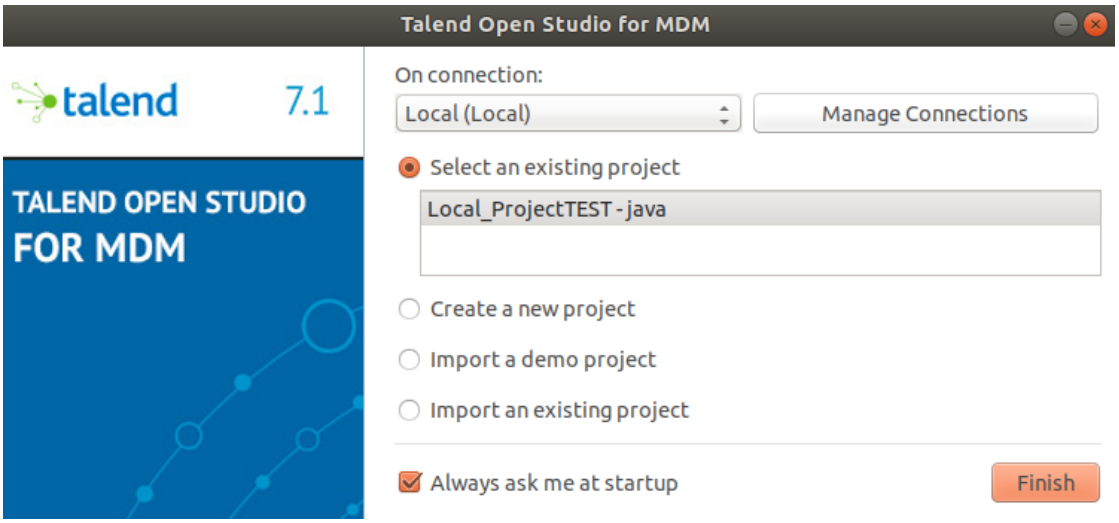
2. Konfiguracja i przykład użycia

Po rozpakowaniu wchodzimy w terminal, a w nim w folder “TOS_MDM-Studio-20181026_1147-V7.1.1/”, w nim uruchamiamy odpowiedni plik dla naszego systemu (np. “./TOS_MDM-linux-gtk-x86_64”).
Tworzymy lub wybieramy projekt.

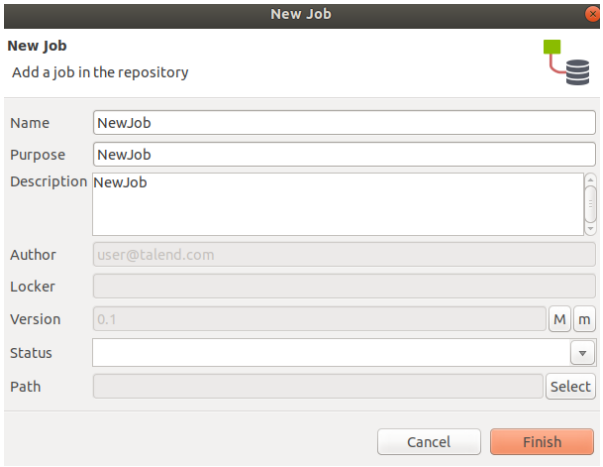
Talend służy do wielu operacji eksploatacji danych. Aby rozpocząć nowy proces należy najpierw stworzyć nowy “Job” (W oknie Repository → PPM na opcji Job Designs → create job). Wypełniamy trzy pola jak na rysunku, po czym klikamy Finish.

Nasze dane do przeanalizowania muszą znajdować się w bazie danych. Następnie tworzymy nowe połączenie:

1. Rozwijamy opcje w oknie repository w menu “Metadata”.
2. klikamy PPM na DB Connections I wybieramy opcje “create connection”.



Rysunek 2.10: Talend Open Studio



Rysunek 2.11:

3. Jak przy tworzeniu nowego “Job’a” podajemy Name, Purpose oraz Description I klikamy Next.

4. Wybieramy rodzaj bazy danych, uzupełniamy dane I klikamy Finish.

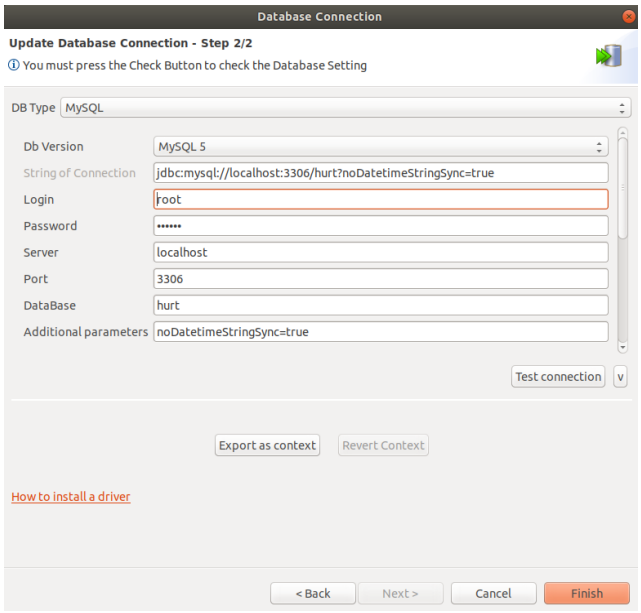
Następnie utworzony DBConnection “przeciągamy” na pole robocze, zostanie wyświetlone menu z wyborem bazy danych, więc wybieramy odpowiadające naszej bazie (rys. 2.4).

Resztę komponentów budujemy przy użyciu palety. Wpisujemy nazwę komponentu i przeciągamy go na pole robocze.

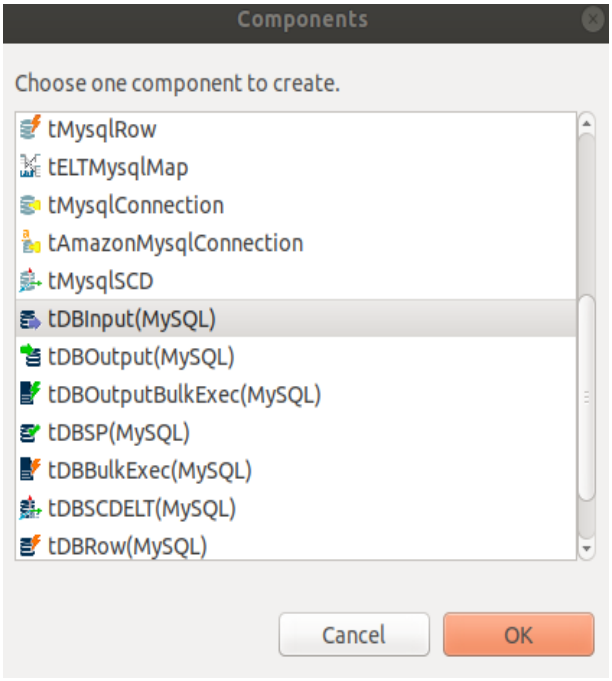
Aby zdeduplikować dane należy zbudować następujący przepływ:

DBInput, po nim bloczek UniqRow, z którego rozchodzą się osobno dla odfiltrowanych danych i osobno dla duplikatów gałęzie LogRow→FileOutputDelimited

Należy także utworzyć schemat naszej tabeli z bazy. Robimy to rozwijając DB Connections (Repository), klikając PPM na stworzonym połączeniu i następnie Retrive Schema. Następnie



Rysunek 2.12: Talend Open Studio

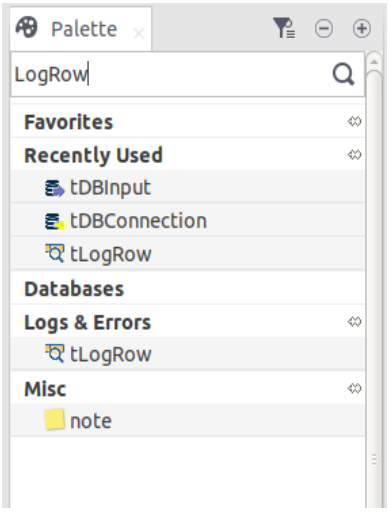


Rysunek 2.13: Talend Open Studio

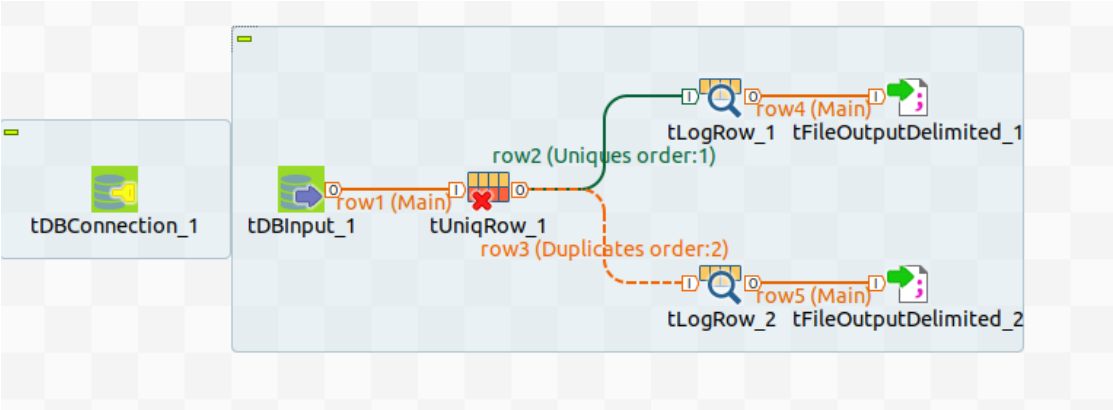
postępujemy pokolei w kreatorze, wybieramy Use Name Filter, w kolejnym oknie wybraną bazę danych i tabelę. W kolejnym oknie podajemy nazwę schematu, wybieramy na której tabeli ma bazować i klikamy Retrieve Schema. W załadowanym schemacie możemy zmienić nazwy (początkowo bedę takie jak na bazie), wybrać opcję Nullable dla każdej tabeli, oraz wybrać klucze (Rys. 2.7)

Następnie należy skonfigurować pokolei każdy bloczek:

1. DBInput → w Basic Settings podajemy dane do logowania na bazę danych, wykorzystywaną tabelę oraz wybieramy wcześniej stworzony schemat



Rysunek 2.14: Talend Open Studio



Rysunek 2.15: Talend Open Studio

2. UniqRow → w Basic settings wybieramy kluczowe atrybuty (możemy także wybrać opcję Case Sensitive), w zaawansowanych opcjach możemy wybrać kilka dodatkowych opcji jak ignorowanie zer dla dużych liczb, czy wykrywanie tylko jednego duplikatu dla każdej krotki.

UWAGA! W pełnej, płatnej wersji programu mamy bloczek FuzzyUniqRow, która zamiast porównywania stringów może używać jednego z trzech algorytmów odległość Levenshteina, Metaphone (oparty na algorytmie fonetycznym do indeksowania według ich wymowy) lub podwójny Metaphone (udoskonalony poprzedni algorytm).

3. LogRow → znów wybieramy schemat oraz w jakim formacie mamy mieć zwrócone wyniki

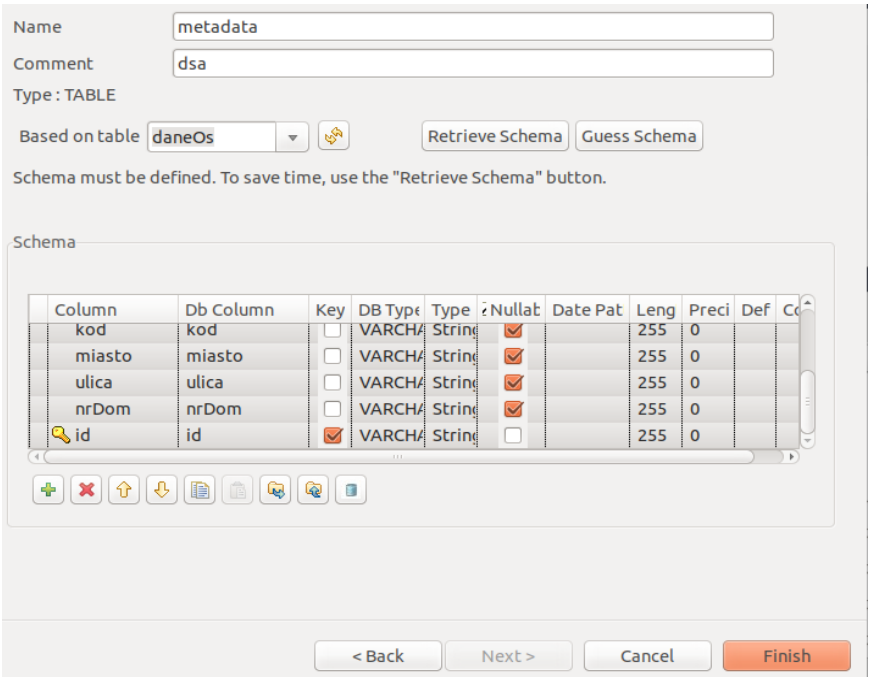
UWAGA! W LogRow możemy edytować schemat i wybrać które kolumny chcemy mieć pokazane w wynikach (np. Samo id elementów).

4. FileOutputDelimited →wybieramy miejsce zapisu pliku oraz jego nazwę oraz separatory i schemat.

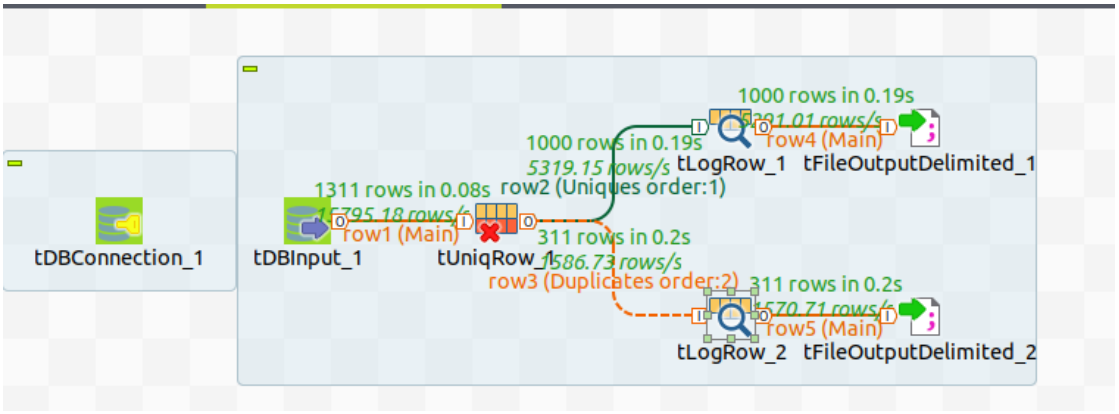
UWAGA! W jeśli plik istnieje dostaniemy błąd i plik nie zostanie nadpisany (po każdym uruchomieniu należy usunąć plik lub zmienić jego nazwę w programie).

Następnie klikamy RUN

Odpowiednie pliki .csv z wynikami możemy znaleźć w podanych wcześniej ścieżkach.



Rysunek 2.16: Talend Open Studio



Rysunek 2.17: Talend Open Studio

2.4.6 JedAI

1. Instalacja

   Wymagania:

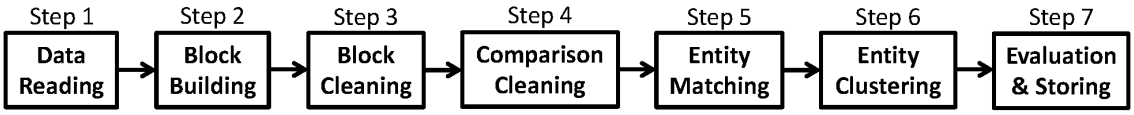
   - GIT
   - Maven
   - JDK

   Instalacja oprogramowania ogranicza się do sklonowania repozytorium: https://github.com/scify/JedAIToolkit.git oraz wywołania w katalogu głównym projektu komend:

   - mvn clean package
   - java -jar jedai-ui/target/jedai-ui-version.jar

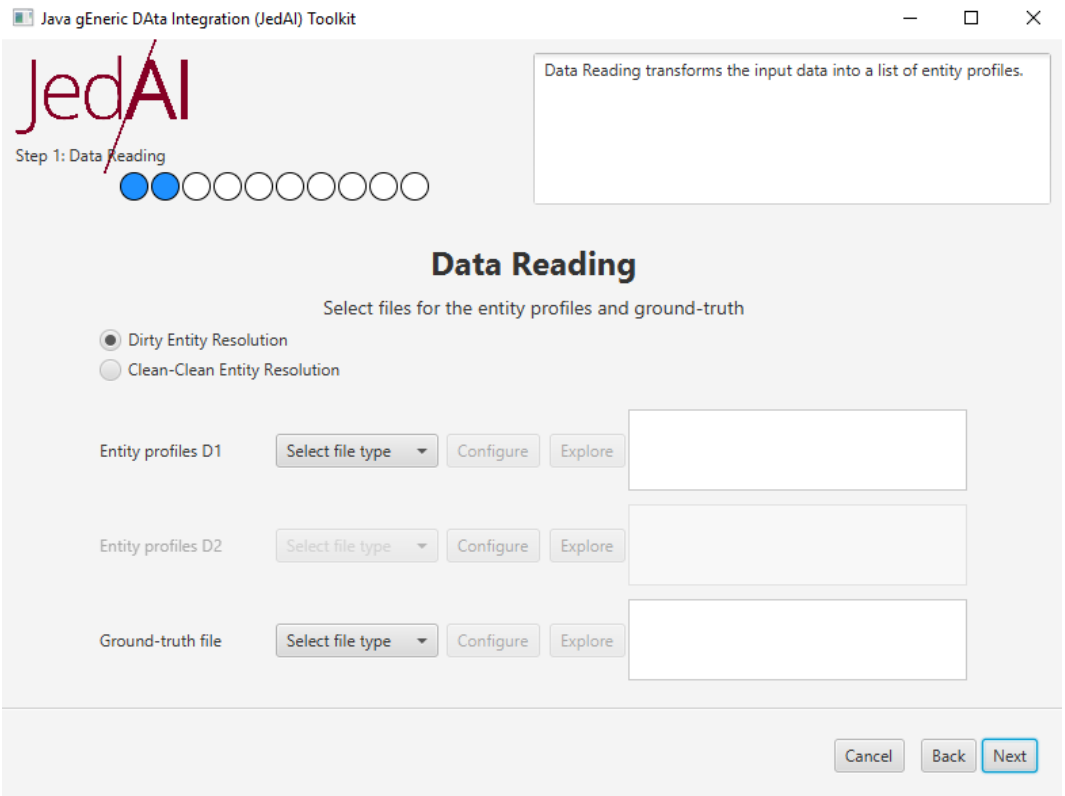
2. Przykład użycia

Workflow JedAI składa się z 7 kroków zaprezentowanych na poniższym schemacie. Każdy z nich został dokładnie opisany w dokumentacji JedAI dostępnej pod adresem: https://github.com/scify/JedAIToolkit. Optymalna praca z tym programem wymaga od użytkownika zapoznania się z wszystkimi oferowanymi metodami na każdym kroku przepływu. Jednakże oprogramowanie oferuje zestaw domyślej konfiguracji, która nie wymaga od użytkownika żadnej wiedzy i pozwala szybko rozpocząć pracę, a zapoznanie się z dokumentacją jest wymagane w momencie gdy wyniki nie są zadowalające. Optymalna konfiguracja oprogramowania jest zależna od specyfiki zbioru danych, na których będzie ono testowane.

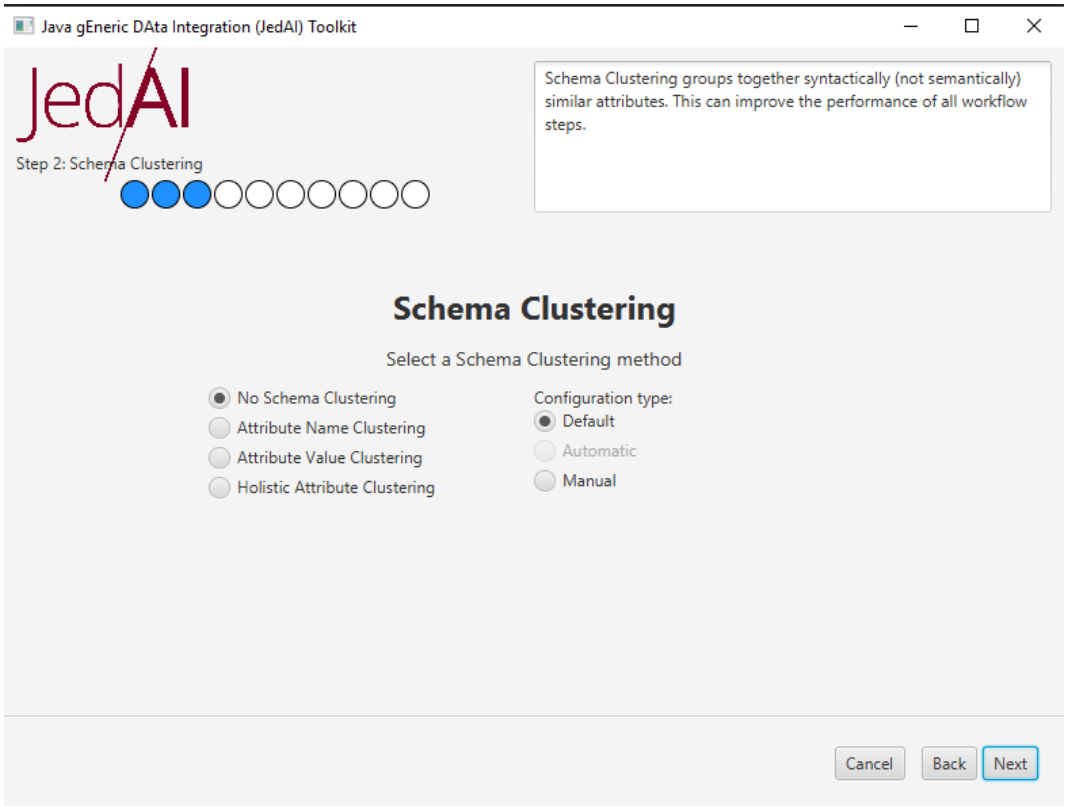


Rysunek 2.18: Przepływ pracy w JedAI

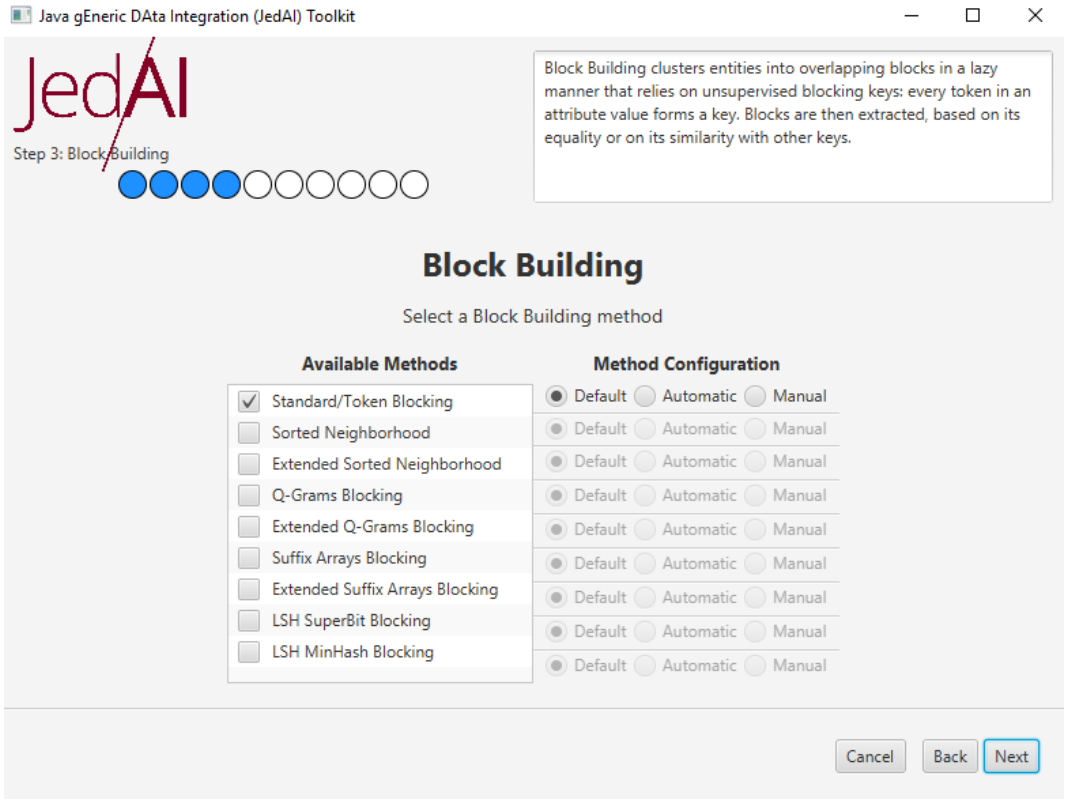
Poniżej przedstawione zostały przykładowe zrzuty ekranu przedstawiające etapy pracy z oprogramowaniem.



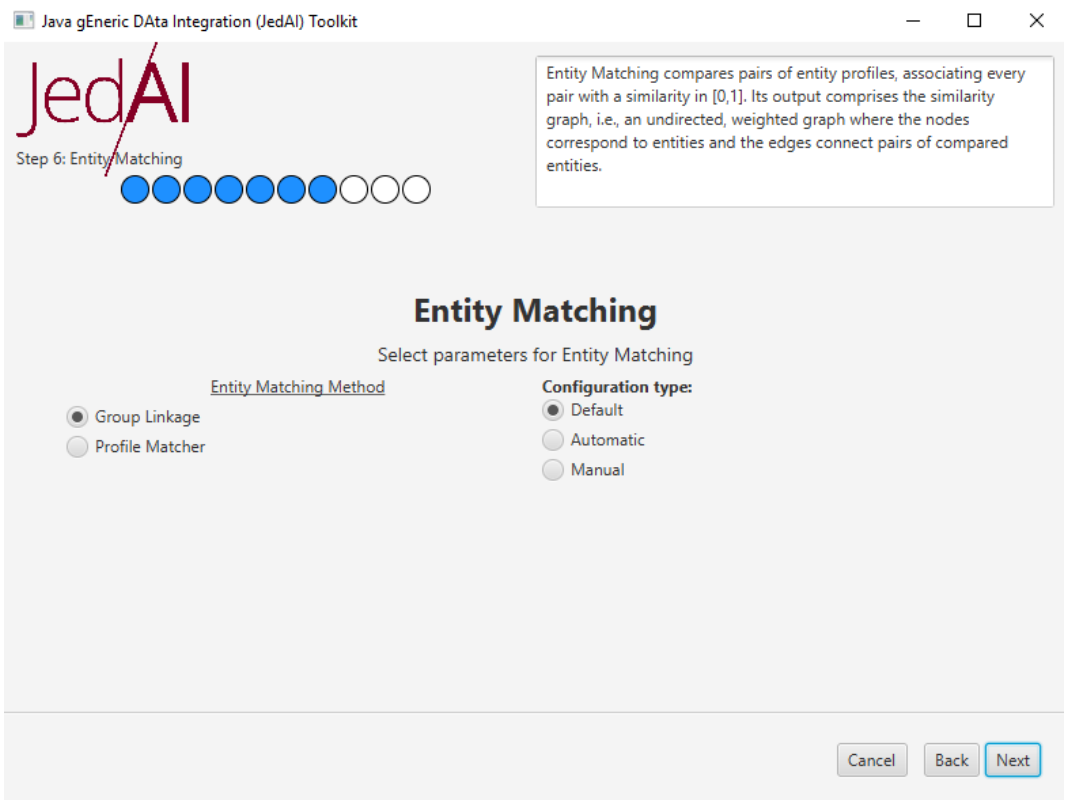
Rysunek 2.19: Ekran wyboru plików



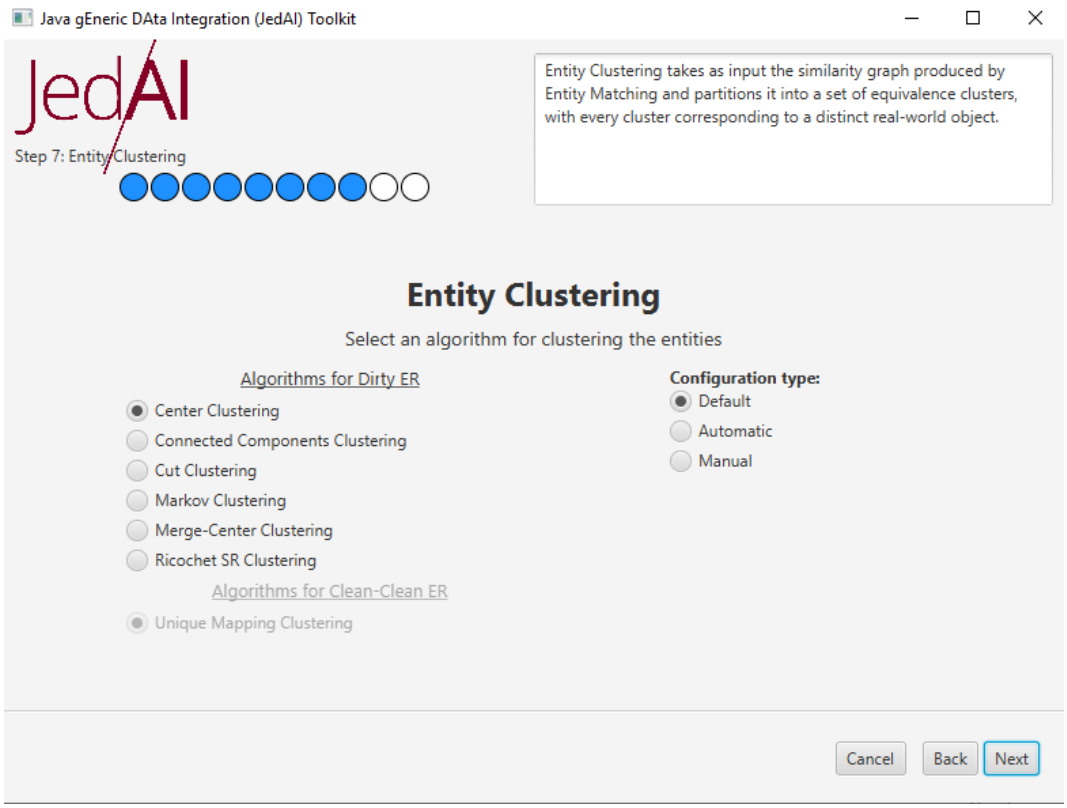
Rysunek 2.20: Ekran wyboru metody grupowania danych



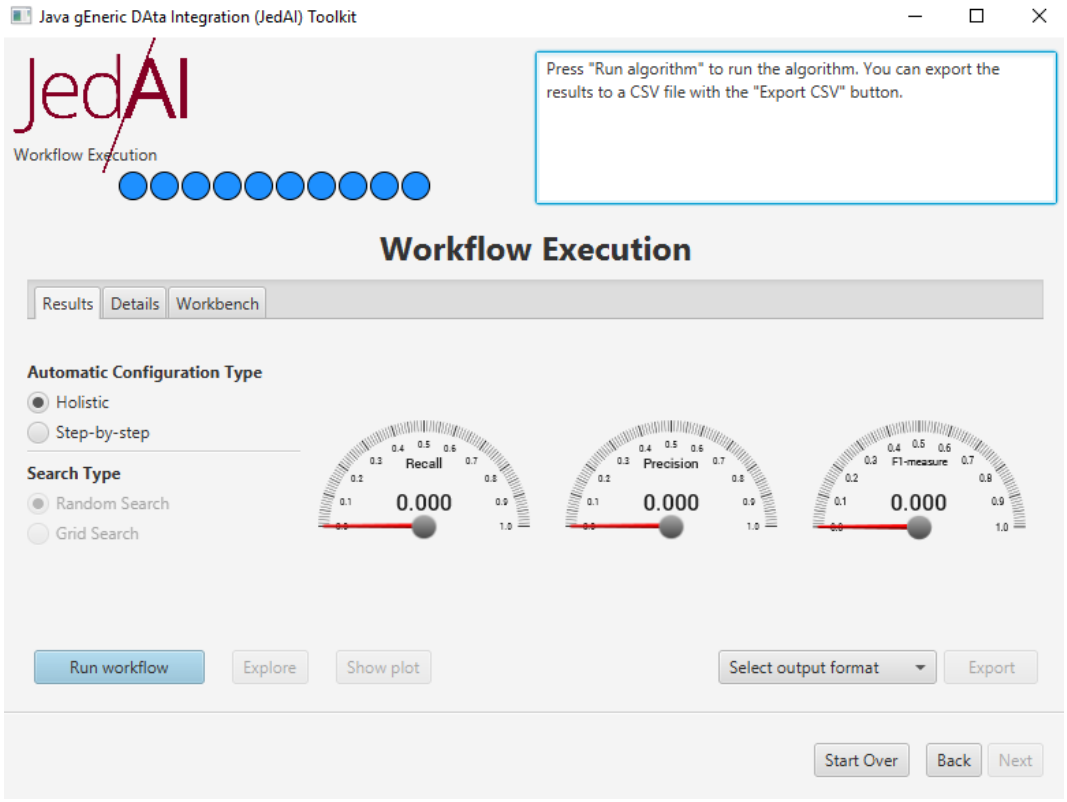
Rysunek 2.21: Ekran wyboru metod budowania bloków



Rysunek 2.22: Wybór metody porównywania rekordów



Rysunek 2.23: Ekran wyboru metody grupowania rekordów



Rysunek 2.24: Ekran końcowy

Rozdział 3

Podsumowanie

3.1 Ocena narzędzi

Na podstawie doświadczeń i odczuć zebranych przez zespół projektowy podczas realizacji projektu, została sporządzona tabela, zawierająca subiektywną ocenę każdego z rozpatrywanych narzędzi.

Nazwa	Instalacja	Konfiguracja	Obsługa
JedAI	8	7	7
Data Quality and Profiling	9	5	3
Record Linkage	8	5	2
SERF	8	4	7
Talend Open Studio	9	8	7
Dedupe	8	6	6

Rysunek 3.1: Subiektywna ocena narzędzi

Po przeprowadzeniu testów na wybranych narzędziach, zgodnie z planem pomierzono ich skuteczność na zbiorze wygenerowanych danych personalnych (test został opisany w seksji 2.3, zmienną w testach był % niepoprawnych znaków w polu). Poniżej zaprezentowane zostały uzyskane rezultaty.

		10%	30%	50%	70%
dedupe	precission	1	1	1	0,99
	recall	0,7	0,73	0,74	0,92
	F1-score	0,82	0,84	0,85	0,95
record linkage	precission	0,92	0,94	0,94	0,9
	recall	0,45	0,39	0,3	0,25
	F1-score	0,6	0,56	0,45	0,4
talend	precission	0,95	0,94	0,92	0,91
	recall	0,3	0,26	0,2	0,16
	F1-score	0,57	0,49	0,37	0,29
serf	precission	0,14	0,11	0,07	0,07
	recall	0,4	0,32	0,22	0,24
	F1-score	0,2	0,16	0,11	0,11
jedai	precission	0,92	0,86	0,78	0,71
	recall	0,76	0,69	0,26	0,25
	F1-score	0,83	0,77	0,39	0,26
data quality	precission	0,85	0,78	0,6	0,29
	recall	0,08	0,07	0,02	0,01
	F1-score	0,15	0,13	0,04	0,02

Rysunek 3.2: Skuteczność narzędzi

3.2 Wybór najlepszego narzędzia

Z uzyskanych wyników wnioskować można, że Dedupe został bezkonkurencyjnym zwycięzcą. Uzyskiwał najlepsze wyniki, mimo ogromnych szumów w zbiorze. Wybór najlepszego narzędzia jednak nie jest trywialny. 70% niepoprawnych znaków w polach to przypadek ekstremalny, wręcz taki, który raczej w rzeczywistej sytuacji nie wystąpi. Przy mniejszych wartościach tego parametru pozostałe narzędzia nie wypadły wiele gorzej niż Dedupe. Warto wspomnieć, że Dedupe nie posiada interfejsu i wymaga od użytkownika umiejętności programowania. Wybór najlepszego oprogramowania do deduplikacji z testowanych podczas tego projektu jest silnie zależny od zbioru danych, z którym oprogramowanie będzie pracować oraz od umiejętności i znajomości tematu przez użytkownika.