Politechnika Warszawska
Wydzia Elektroniki i Technik Informacyjnych
Instytut Informatyki

Rok akademicki 2011/2012

Praca dyplomowa magisterska

inż. Piotr Jan Dendek

Rozróżnianie autorów dokumentów na podstawie metadanych

Opiekun pracy:
dr hab. inż. Piotr Gawrysiak

Ocena .....

.....
Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Specjalność: Informatyka – Inżynieria Systemów Informatycznych

Data urodzenia: 8 kwietnia 1987 r.

Data rozpoczęcia studiów: 1 października 2010 r.

Życiorys

Nazywam się Piotr Jan Dendek. Urodziłem się 08.04.1987r. w Łodzi. Po ukończeniu szkoły podstawowej i gimnazjum, kontynuowałem naukę w I Liceum Ogólnokształcącym im. Mikołaja Kopernika w Łodzi. W szkole średniej uczęszczałem do klasy o profilu nauk ścisłych. W październiku 2006r. rozpocząłem studia inżynierskie na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej na kierunku Informatyka. Po obronie pracy dyplomowej w 2010r. kontynuowałem kształcenie na studiach magisterskich.

.....

podpis studenta

Egzamin dyplomowy

Złożył egzamin dyplomowy w dn. .....

Z wynikiem .....

Ogólny wynik studiów .....

Dodatkowe wnioski i uwagi Komisji .....

.....

Streszczenie

W bibliotece wirtualnej staje się przed problemem rozróżnienia autorstwa prac osób noszących to samo imię i nazwisko. Celem niniejszej pracy jest stworzenie narzędzia do łączenia dokumentów w zbiory, które można przypisać tej samej osobie, na podstawie metadanych rozważanych utworów. Częścią tego zadania jest ocena istotności poszczególnych atrybutów artykułów w procesie rozróżniania autorstwa.

Rozwiązanie tej kwestii jest kluczowe dla przedstawienia w klarowny sposób dorobku danej osoby, bądź instytucji, w której pracuje. Jest to również punkt wyjścia dla innych zadań jak poszukiwanie kluczowych autorów, bądź dokumentów w danym temacie, czy usuwanie duplikatów prac.

Słowa kluczowe: *rozróżnianie autorstwa prac, ocena istotności atrybutów, maszyny wektorów nośnych, SVM, klasyfikacja, biblioteka wirtualna, uczenie z nadzorem, ekstrakcja danych,*

Author Name Disambiguation in documents from a virtual library

Abstract

Author Name Disambiguation is a process in which one distinguishes between several persons who share the same name. The aim of this work is to deliver a flexible, time effective tool, which utilizes metadata of compared documents to conduct this activity in the most potent manner. The other goal of the thesis is development of a solution to assess importance of metadata features on author name disambiguation process.

Archiving both goals is key condition in many other tasks, e.g. estimation of document's importance in given field of study, rating of personal impact on a subject, document deduplication or record linkage.

Key words: *author name disambiguation, feature importance assessment, support vector machines, SVM, classification, supervised learning, virtual libraries, information retrieval, computer science*

Spis treści

Spis treści

Rozdział 1

Wstęp

1.1 Tło

Liczba publikacji naukowych z roku na rok rośnie [40][30]. Wynika to z kilku czynników, jak choćby światowego wzrostu grupy osób z tytułami naukowymi[39], która biorąc pod uwagę grupę krajów zrzeszonych w Organizacja Współpracy Gospodarczej i Rozwoju (ang. Organization for Economic Co-operation and Development, OECD), powiększała się rok do roku o około 40%.

Podwyższeniu uległy też środki kierowane na rozwój i badania [32], zwiększyły się możliwości dystrybucji publikacji, np. za pośrednictwem stron, czy serwisów internetowych jak arXiv [25], ale też zakrojonego na niespotykaną w historii ludzkości poprawę możliwości technicznych i obliczeniowych maszyn. Przyczynia się do tego również włączenie się do naukowej sztafety państw takich jak Chiny, czy Indie. Zarazem podnosi się liczba ważnych ośrodków akademickich i ich wkład w naukę. Badać można nowe zagadnienia, ale też takie, które należało wcześniej odłożyć ze względu na brak potencjału technicznego.

W latach osiemdziesiątych XX w. liczba publikowanych artykułów na tydzień wynosiła 10000. Oznacza to, że dla zapoznania się z nimi wszystkimi badacz musiałby, rezygnując z innych swoich potrzeb, poświęcić na pojedynczy egzemplarz jedną minutę. Już wtedy oznaczało to, że jednostka ludzka nie jest w stanie nadążyć za tempem publikacji.

Biorąc to wszystko pod uwagę, należy się zastanowić, jak uczynić pracę z publikacjami naukowymi możliwie efektywną, przy tym mało pracochłonną. W obecnych czasach zapoznawanie się z artykułami z danej dziedziny jest ułatwione przez istnienie bibliotek wirtualnych. W idealnym przypadku pozwalają one nie tylko pozyskać sam artykuł, ale też w wygodny sposób przejść do cytowanych w dokumencie publikacji, czy też sprawdzić dorobek naukowy danego autora.

1.2 Przyczyny problemu

Aby osiągnąć ostatni z wymienionych celów, konieczne jest przeprowadzenie procesu rozróżniania autorstwa prac. Ponieważ wiele bibliotek wirtualnych nie posiada unikalnego kodu identyfikującego autorów, a prace bywają rozproszone po kilku bibliotekach, osiągnięcie takiego celu nie jest zadaniem trywialnym. Oczywistym pomysłem wydaje się wiązanie prac noszących takie same sygnatury autora.

Abstrahując od faktu posiadania jednego nazwiska przez wielu autorów (każdy kraj ma swoje najpopularniejsze nazwiska), pojawia się kwestia równoprawnych reprezentacji sygnatury jednego autora. Mogą one widnieć jako połączenie pierwszego imienia i nazwiska, wszystkich imion autora i nazwiska, czy też inicjałów imion i nazwiska. Należy też pamiętać o przydomkach, które mogą nosić autorzy, jak "młodszy", czy "trzeci".

Kolejnym aspektem bardzo mocno uwidocznionym na arenie międzynarodowej jest kwestia znaków diakrytycznych, pojawiających się w literach "ą", "ę", "ö", czy "ç". Otóż autorzy, z różnych przyczyn pisząc na arenie międzynarodowej nierzadko rezygnują dodatkowych oznaczeń, przez co sygnatury "J.Duszyński" i "J.Duszynski", czy "J.Więch" i "J.Wiech" mają reprezentować jedną osobę.

Idąc dalej, należy mieć na względzie możliwość zmiany nazwiska, która pojawia się np. przy okazji ślubu. Możliwa jest wtedy zarówno zmiana nazwiska, jak i dodanie do posiadanego dodatkowego członu połączonego z wcześniej posiadanym myślnikiem. W pewnych kręgach kulturowych można do imienia dodać imię ojca, z odpowiednią dla płci odmianą.

Ostatecznie, gdy rozważamy publikacje na arenie międzynarodowej, należy też wziąć pod uwagę alfabety inne niż łaciński, jak choćby alfabet arabski. Istnieje wiele możliwych równoprawnych transliteracji z niego do alfabetu łacińskiego. Przykładowo w łacińskiej wersji Koranu bardzo popularne jest wykorzystanie wyszukiwarek fonetycznych, które pozwalają odnaleźć różne warianty zapisu tych samych imion.

1.3 Definicja problemu

Problem rozróżniania nazw autorów w literaturze określany jest angielskim terminem **Author Name Disambiguation**. Zadanie do rozwiązania brzmi następująco: na podstawie danych dokumentu sprawdź, czy zadane prace zostały napisane przez tę samą osobę. Przy tym do danych zaliczyć można listę sygnatur autorów, tytuł, streszczenie, rok publikacji, słowa kluczowe, czy treść dokumentu.

1.4 Efekty problemu

Brak powiązania dokumentów ze względu na autora rodzi różnorodne problemy. Nie jest możliwe przeglądanie publikacji jednej osobny, nie można zatem ocenić ani jej aktywności naukowej, ani sumarycznej aktywności instytucji, w której jest zatrudniona. Kolejny aspekt to określenie na ile dany człowiek tworzy istotne publikacje w zakresie poruszanego zagadnienia (ang. impact factor). Jest to szczególnie ważne z perspektywy użytkowników, którzy chcąc wdrożyć się w nowe zagadnienie, powinny śledzić główne, a w szczególności najczęściej cytowane, prace z danego tematu. Te bardzo często są tworzone przez kluczowych autorów. Innym zagadnieniem jest śledzenie i usuwanie duplikatów prac, które przy śledzeniu powiązania z autorem byłoby dużo prostsze. Temat ten pojawia się, np. wtedy kiedy chce się połączyć zbiory dwóch bibliotek wirtualnych. Dalszym ciekawym aspektem jest wyszukiwanie tzw."wschodzących gwiazd", tj. autorów, którzy zaczynają publikować ważne prace, m.in. współpracując z kluczowymi autorami.

1.5 Dotychczasowe próby rozwiązania problemu

Biorąc pod uwagę kluczową rolę znalezienia metody rozróżniania autorstwa prac w innych zagadnieniach, podjęto do tej pory wielokrotne próby rozwiązania go. Starano się do tego zaprząc różnorodne metody sztucznej inteligenci, bazujące na uczeniu z nadzorem oraz uczeniu bez nadzoru.

Han *et* al. [33] porównali rozwiązania bazujące na Naiwnym Klasyfikatorze Bayesa z klasyfikatorem korzystającym z Maszyn Wektorów Nośnych (ang. Support Vector Machines). Pedersen *et* al. [49] podobnie jak Mann i Yarowsky [43] oraz Han *et* al. [34] zaproponowali różnodorne metody klastrowania Galvez i Moya-Anegón [31] sprawdzali możliwość wykorzystania automatów skończonych, żeby zestandaryzować warianty nazw Pavelec *et* al. [48] starali się przypisać autorstwo traktując prace jako anonimowe i przypisując je do osób na podstawie stylometrii Levin i Heuser [41] starali się stworzyć najlepszy klasyfikator binarny z wykorzystaniem programowania genetycznego.

Wszystkie wymienione prace bazują na wskazanym przez eksperymentatorów zbiorze atrybutów. Autorzy zakładają w swoich pracach wybór atrybutów albo przez przekazanie pełnego posiadanego zestawu atrybutów, albo też ich subiektywną selekcję. Przy tym zakładają, że cechy są równie istotne, bądź też że ich waga jest zróżnicowana, przy czym różnicują je bazując na intuicji. Same wartości atrybutów w zależności od podejścia podlegają, lub też nie, przeskalowaniu. Przy tym pojawia się temat traktowania wartości pustych, wynikających z braku danych o poszczególnych atrybutach. Takie przypadki są bowiem na tyle powszechne, że nie można ich ignorować.

Dobór jednego z w/w podejść, ma niebagatelne znaczenie na jakość otrzymywanych wyników.

1.6 Cel pracy

Niniejsza praca jest częścią projektu Europejskiej Cyfrowej Matematycznej Biblioteki (ang. European Digital Math Library, EuDML) w ramach udziału w projekcie Interdyscyplinarnego Centrum Modelowania Matematycznego i Komputerowego Uniwersytetu Warszawskiego. Z uwagi na skalę przedsięwzięcia, zawierającego dziesiątki milionów publikacji do przetworzenia, zadanie rozróżniania autorstwa okazało się nietrywialne zarówno architektonicznie, jak i algorytmicznie.

Dlatego poniższa praca niesie dwie główne wartości:

1. stworzenie efektywnego rozwiązania dedykowanego systemom składującym duże ilości danych, opisanego przez Bolikowski i Dendek [17])
2. stworzenie narzędzia do analizy danych (opisanego przez Dendek *et al.* [27])
   a) oceny istotności atrybutów
   b) oceny wpływu przekształceń danych na efekt klasyfikacyjny.

Przy analizie danych w różnoraki sposób wykorzystywano narzędzie matematyczne jakim są Maszyny Wektorów Nośnych.

W pracy skupiono się na zagadnieniu uczenia z nadzorem. Możliwe było to dzięki połączeniu zbiorów pochodzących z EuDML oraz Zentralblatt MATH, gdzie drugi zbiór posiadał przypisane autorom unikalne identyfikatory. Zbadano dwa narzędzia służące do przechowywania danych (Sesame oraz bigdata®), oba będące realizacją magazynu trójkowego, pozwalającego w wygodny sposób przechowywać dane w formie grafowej. Ograniczono się do przebadania kilku możliwych przekształceń danych i kilku jąder przekształceń dla metody Maszyn Wektorów Nośnych. Nie było ambicją autora wyczerpujące zestawienie dostępnych klasyfikatorów, z uwagi na ich bardzo dużą liczbę. Podjęto się jednak porównana części z nich.

Przy powstaniu szkieletu do rozróżniania autorstwa pomocą nie do przecenienia, zarówno merytoryczną jak i praktyczną, okazał się dr Łukasz Bolikowski (ICM UW). Natomiast system ważenia wskazówek powstał przy współpracy z inż. Michałem Łukasikiem (ICM UW, MiM UW).

1.7 Opis dalszych rozdziałów

W rozdziale drugim prezentowany jest obecny stan wiedzy w dziedzinie rozróżniania autorstwa. W rozdziale trzecim przybliża się instrumenty matematyczne i informatyczne, na których bazują wykonane w pracy eksperymenty i narzędzia. W rozdziale czwartym definiuje się podstawowe pojęcia w rozróżnianiu autorstwa. Przybliża się też architekturę szkieletu do rozróżniania autorów, wraz z ogólnym opisem jego poszczególnych części. Rozdział piąty charakteryzuje trzy wykonane eksperymenty. Pierwszy to wyznaczania istotności miar podobieństwa między autorami. Drugi, mierzenie

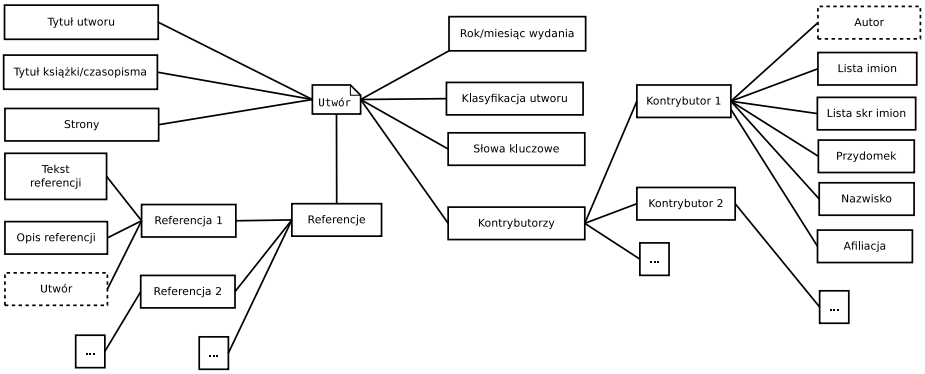
wpływu przekształceń danych i doboru jądra przekształcenia dla metody Maszyn Wektorów Nośnych na czas wykonania obliczeń i ich dokładność. Ostatni eksperyment porównywał implementacje rozróżniania autorstwa bazujące na magazynie trójek Sesame oraz bigdata®. Rozdział szósty zawiera wyniki opisanych w rozdziale piątym eksperymentów, zaś rozdział siódmy zawiera propozycje dalszych kierunków prac.

Rozdział 2

Obecny stan wiedzy

2.1 Sposoby opisu utworu

Istnienie bibliotek wirtualnych zbierających dane o utworach (książkach, artykułach) implikuje istnienie różnorodnych problemów. Aby się nad nimi pochylić należy wpierw wymienić jakie dane opisują dany utwór, a są to m.in.: imiona i nazwiska twórców, wydawnictwo (ang.publisher), miesiąc i rok publikacji, ewentualnie strony, na których w książce widnieje dany artykuł, fachowa klasyfikacja dokumentu (np. według klasyfikacji Matematycznej Klasyfikacji Zagadnienia [12], ang. Mathematics Subject Classification, MSC), słowa kluczowe i cytowania. Opis ten w postaci drzewa prezentuje rysunek 2.1. Można go przedstawić w różnorodny sposób, np. w standardzie BibTeX, bądź też z wykorzystaniem pliku XML. Warto zwrócić uwagę na to, w jaki sposób takie opisy powstają. Generalnie są one tworzone albo przez ludzi (pra-



Rysunek 2.1: Opis utworu. Informacje posiadane explicite widnieją w ramkach rysowanych linią ciągłą. Dane, które można odtworzyć (implicite obecne w bazie) przedstawione są w ramkach o linii przerywanej.

cowników biblioteki, autora utworu w trakcie wysyłania pracy) albo wydobywane z wykorzystaniem mechanizmów Wizualnego Rozpoznawania Znaków [56] (ang.Optical Character Recognition, OCR). Dzięki tak uzyskanym informacjom możliwe jest zbudowanie drzewa opisującego dany dokument. Należy przy tym zwrócić uwagę, że w ten sposób uzyskuje się niepełną informację o dokumencie, np. brakuje bezpośredniego połączeń do cytowanych dokumentów, nawet jeśli te obecne są w tej samej bibliotece. Dla wygody użytkownika przeglądającego dane biblioteki, ale też z uwagi na inne zastosowania ważne jest, aby takie powiązania odtworzyć.

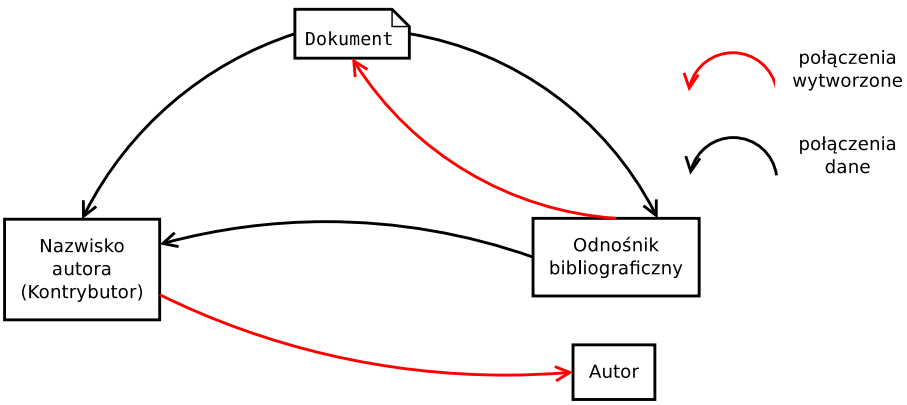
2.2 Związki między obiektami biblioteki wirtualnej

Na które połączenia należałoby zwrócić szczególną uwagę? Można wymienić kilka przykładów od odzyskiwania informacji o autorstwie (wytworzenia ID autora), przez wiązanie referencji z cytowanymi dokumentami, przez afiliacje (generowanie ID instytucji), kończąc na konferencjach, czy czasopismach.

Ostatecznie zatem można podzielić relacje między poszczególnymi w/w jednostkami na jawne i niejawne, co w odniesieniu do imienia i nazwiska autora, ID autora, dokumentu i odnośnika bibliograficznego przedstawia rysunek 2.2. Chodzi zatem o utrzymanie spójności danych tj. odzyskania brakujących połączeń. To również kwestia wykrywania ewentualnych duplikatów dokumentów, które mogłyby zaburzyć uzupełnianie połączeń. Możliwym byłoby bowiem, że część cytowań byłaby wiązana z dokumentem A, a część z jego kopią A'.

2.3 Używane definicje problemu

Wyżej wymienione kwestie nie są nowe. Wielokrotnie podejmowano je, nadawano im różne nazwy jak rozróżnianie nazwisk autorów (ang.author name disambiguation), łą-



Rysunek 2.2: Podział relacji między poszczególnymi danymi opisującymi dokument

czenie danych [29][16] (ang.record linkage), wiązanie cytowań [46][47] (ang.citation matching), niepewność tożsamości [47] (ang.identity uncertainity), usuwanie duplikatów [14][53] (ang.duplicate deletion). W części publikacji zauważono również zbieżność wymienionych zadań [42]. Proponowane są zarówno różnorodne metody ich rozwiązywania (uczenie z nadzorem, uczenie bez nadzoru [33]), jak i szkielety pozwalające na dowolny dobór algorytmu i rozpatrywanego obiektu np. szkielet PACE stworzony w ramach programu OpenAIRE+ [59].

Wymienione zadania mają różny poziom trudności, np. wiązanie cytowania z dokumentem jest w wielu przypadkach możliwe na podstawie umieszczonego w referencji Identyfikatora Obiektu Cyfrowego [13] (ang.Digital Object Identifier, DOI), a afiliacje scalać można wykorzystując postfiksy adresu e-mail autora (np. z adresu "jkowalski@elka.pw.edu.pl" użyć człon "elka.pw.edu.pl"). Z drugiej strony łączenie prac jednego autora może stanowić problem ze względu na różne reprezentacje tego samego imienia i nazwiska (np. "Jan Maria Kowalski", "J.Kowalski"), popularność danego nazwiska (np. "Kowalski", "Smith", "Han"). Dla ilustracji problemu weźmy nazwisko "Chen", które w bazie Springer-a łączy się z około 24000 dokumentami. Nie jest to przy tym najliczniejszy zbiór, bo ten w zbiorze Springera z lipca 2010 roku liczył sobie 35498 dokumentów. Dodatkowo należy wziąć pod uwagę różnorodne sposoby zapisu danego nazwiska.

2.4 Dotychczas sprawdzone rozwiązania

2.4.1 Przykładowe dobory atrybutów i miar podobieństwa obiektów

Bez względu na rodzaj rozpatrywanego do ewentualnego połączenia obiektu, wymagane jest aby był opisany pewnym zbiorem atrybutów, których wartości podlegają porównaniu między parami badanych instancji (przykłady atrybutów w odniesieniu do dokumentów opisano w rozdziale 2.1).

Przy badaniu zbieżności autorstwa w artykule [33] porównywanymi cechami były pełne imię i nazwisko (np. "James Nicholas Anderson"), afiliacja (np."Boston College") i pole badań (np."ekonomia"), zaś w [41] zaproponowano dodatkowo tytuł dokumentu, afiliacja, inicjał pierwszego imienia i nazwisko. Sprawdzano też, czy pierwsze imię jest oznaczone inicjałem, weryfikowano ilość zgodnych słów z tytułu. W artykule [59] porównywano imiona i nazwiska współautorów jako całość, ale też zliczano poszczególne zgodne ich człony między parą analizowanych obiektów oraz ilość takich samych liter. W artykule [27] zaproponowano porównywanie ilości wspólnych autorów, zgodność adresów e-mail, zgodność prefiksów adresu e-mail (dla adres "jkowalski@elka.pw.edu.pl" prefiksem jest "jkowalski"), zgodność kodów klasyfikacyjnych MSC, zgodność fraz kluczowych z listy słów kluczowych (np. "uczenie maszynowe" jako cały wyraz) oraz samych słów kluczowych (dla frazy "uczenie maszynowe" osobno rozpatrywane są słowa "uczenie" i "maszynowe"), ilość wspólnych referencji, różnicę lat między poszczególnymi publikacjami, zgodność Międzynarodowych Znormalizowa-

nych Numerów Wydawnictwa Ciągłego (ang. International Standard Serial Number, ISSN), bądź też Międzynarodowy Znormalizowany Numer Książki (ang. International Standard Book Number, ISBN).

2.4.2 Wykorzystywane algorytmy

W pracy [33] skupiono się na porównaniu autorstwa prac z wykorzystaniem Naiwnego Klasyfikatora Bayesa oraz Maszyn Wektorów Podpierających [23] (SVM). Drugi z algorytmów wykorzystywany był w implementacji SVM^{light} [38]. W artykule wykazano, że biorąc pod uwagę pojedyncze cechy, Naiwny Klasyfikator Bayesa lepiej wykorzystuje informację o nazwiskach współautorów niż SVM. W przypadku tytułu pracy i czasopisma SVM był nieznacznie lepszy (o ok.2%). Artykuł wskazuje, że obie metody najlepiej wykorzystują informacje o wspólnym autorstwie (dokładność 69,3% dla Naiwnego Klasyfikatora Bayesa i 64,3% dla SVM), słabiej dla nazwy czasopisma (odpowiednio 40,0% i 37,4%), a najgorzej dla tytułu artykułu (18,9%, 20,1%).

Artykuł [59] porównuje nie tylko algorytmy dzielenia zbioru roboczego imion i nazwisk na podgrupy łączone z tym samym autorem. Na wstępie wspomniane są różne podejścia do samego tworzenia grup roboczych (nazywanego tworzeniem bloków lub krócej - blokowaniem). Wymienia się możliwość ich dzielenie z użyciem części nazwy kontrybutora przez porównywanie samego nazwiska, inicjału pierwszego imienia i całego nazwiska, bądź też inicjałów imienia i nazwiska. Wskazuje się przy tym, że co prawda uzyskane dzięki ostatniej metoda wyniki będą lepsze niż przy porównywaniu pierwszej litery imienia oraz całego nazwiska (z uwagi na niewrażliwość na literówki w nazwiskach, bądź dodawane do nazwiska dodatkowe człony, jak w przypadku kobiet dodających nazwisko męża do swojego nazwiska), ale uzyskane bloki będą dużo większe. Inną możliwością jest potraktowanie każdego członu jako osobnego tokenu i grupowanie kontrybutorów, którzy mają choć jeden wspólny token. Takie podejście ma kilka minusów, m.in. nakładanie się na siebie zbiorów roboczych, co ma szczególne znaczenie w przypadku popularnych imion, bądź długich nazw, np. "Juan David Gonzalez Cobas El-Nasr" albo obecności inicjałów, np. "V. S. P. Srivastava". Kolejnym podejściem jest wykorzystanie N-gramu nazwiska połączonego w jeden ciąg znaków. N-gram to podciąg zadanego ciągu znaków mający długość N. Np. "Jan-Kowalski" ma 8 N-gramów o długości 4, 4-$gram_{JanKowalski} = \{$JanK, anKo, nKow, Kowa, owal, wals, alsk, lski$\}$. Uzyskuje się przez to duże bloki, z uwagi na podobieństwo N-gramów imienia z N-gramami z nazwiska, np. dla "David R. Johnson" i "F. Barr-David"

Aby policzyć podobieństwo między kontrybutorami posłużono się metodami Naiwnego Klasyfikatora Bayesa, SVM, miarą cosinusową, współczynnikiem częstości pojęcia do odwróconej częstości dokumentu (ang. Term Frequency Inverse Document Frequency, TFIDF), Jacarda, Jaro, Jaro-Winklera. Autorzy chwalą się udowodnieniem, że wykorzystanie do procesu rozróżniania autorstwa informacji o cytowaniach, przyspiesza go i podnosi dokładność o 50%.

W pracy [41] postawiono sobie za cel wykorzystanie programowania genetycznego do wytworzenia miary podobieństwa, która dawałaby najlepsze rezultaty przy dzieleniu imion i nazwisk na bloki autorów. Uzyskana miara zwracała współczynnik decyzyjny postaci {"Ten sam autor"/"Prawda", "Różni autorzy"/"Fałsz"}, a uzyskana dokładność sięgnęła 78,5%. Poniższy wzór prezentuje finalną wersję klasyfikatora:

$$\begin{aligned}\textbf{Levin-Heuser}(c_1, c_2) = \;& (IsAbbrev = 1 \wedge NameSim \geqslant 0.94 \wedge MinRQ \leqslant 1) \\ & \vee (TitleSim \geqslant 0.39 \wedge MaxRQ \leqslant 5 \wedge NameSim \geqslant 0.87 \\ & \wedge NumTitleWords \geqslant 2 \wedge IniLastName = 1) \\ & \vee (TitleWordSim \geqslant 0.23 \vee NameSim \geqslant 0.87 \wedge IsAbbrev = 1 \\ & \wedge VenueSim \geqslant 0.35 \wedge MaxRQ \leqslant 3) \\ & \vee (VenueSim \geqslant 0.67 \wedge MaxRQ \leqslant 3 \wedge NameSim \geqslant 0.98) \\ & \vee (IsAbbrev = 1 \wedge NameSim \geqslant 0.45 \wedge NumTitleWords \geqslant 4 \\ & \wedge MinRQ \leqslant 2 \wedge IniLastName = 1 \wedge RE = 0) \\ & \vee (IniLastName = 1 \wedge RE = 1 \wedge MaxRQ \leqslant 9) \\ & \vee (RE = 1 \wedge NameSim \geqslant 0.72) \\ & \vee (RS \geqslant 2 \wedge IniLastName = 1)\end{aligned}$$

2.4.3 Dotychczasowe dostępne implementacje

Ponieważ problem rozróżniania autorstwa jest podejmowany od pewnego czasu, zaproponowano do tej pory kilka jego implementacji. Jedną z nich jest Academic Researcher Social Network Search [8] [54] (Arnetminer). Narzędzie to oferuje zarówno informację o kolekcji dokumentów danego autora wraz ze wskazaniem ich indeksu Hirscha [36], jak również rozróżnienie współpracowników danego autora na współautorów, dyplomantów i promotorów.

Inne narzędzia o zbliżonej funkcjonalności to m.in. CiteSeer [18] i Google Scholar [45]

Rozdział 3

Wykorzystane koncepcje

3.1 Maszyny Wektorów Nośnych (SVM)

Rozwiązania w uczeniu maszynowym można podzielić na uczenie z nadzorem i bez nadzoru [22]. Adresują one problemy odpowiednio, względem których posiadamy wszystkie dane jakie mogą wystąpić oraz takie, gdzie przetwarzane dane mogą zawierać informacje nowe. Przykładem uczenia z nadzorem jest przydzielenie obserwacją etykiety z pewnego zamkniętego zbioru. Z drugiej strony mamy problem klastrowania, tj. podziału zbioru na podgrupy, nie znając ich oczekiwanej liczby.

Skupmy się na uczeniu z nadzorem. Bazowo mamy do dyspozycji przykłady uczące opisane wartościami ich atrybutów wraz z atrybutem decyzyjnym. Wartość atrybutu decyzyjnego należy do z góry określonego zbioru. Zadanie polega na wytworzeniu takiego modelu, który byłby w stanie odtworzyć właściwą wartość atrybutu decyzyjnego, nie znając go. Wytrenowany model ma zatem za zadanie zaklasyfikować obiekt do jednej z kategorii. Obecnie istnieje wiele metod rozwiązujących problem klasyfikacji, a należą do nich m.in. drzewa decyzyjne [50], naiwny klasyfikator bayesowski [52], algorytm k-najbliższych sąsiadów [24], sieci bayesowskie [35], czy maszyny wektorów nośnych [19](z ang. Support Vector Machines, SVM). Ostatnia z nich jest stosunkowo młoda, bo zaproponowano ją w 1992r. Prace nad nią trwały dalej pod kierunkiem Vladimira Vapnika [21]. Oprócz problemu klasyfikacji wykorzystywano ją również do rozwiązania problemów regresji [28], bądź grupowania [15].

3.1.1 Klasyfikacja liniowa

W najprostszej wersji metody SVM, czyli klasyfikatorze liniowym, chodzi o przyporządkowanie przykładów do jednej z dwóch klas na podstawie ich położenia w przestrzeni p-wymiarowej, biorąc pod uwagę położenie w niej obiektów. Niech dana będzie próbka ucząca przykładów, gdzie $i \in I$ to indeksy tych przykładów, $\mathbf{x}_i$ wektor

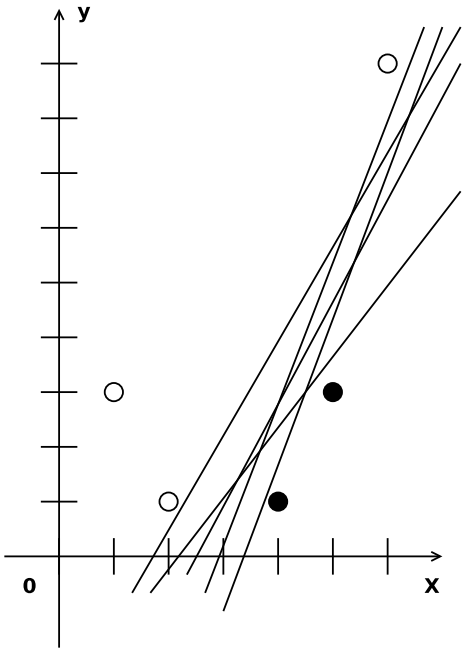
wartości atrybutów, a y_i atrybutem decyzyjnym ze zbioru {-1,1} wskazującym, do której grupy należy dany przykład. Wtedy:

$$\forall_{i \in I} \, (\mathbf{x}_i, y_i) : y_i \in \{-1, 1\}, \mathbf{x}_i \in R^p \tag{3.1}$$

Niech przykłady będą liniowo separowalne tj. niech zadane obiekty trenujące da się rozdzielić w przestrzeni przez hiperpłaszczyznę H

$$\mathbf{a}^T \mathbf{x}_i + b = 0$$

Oczywiście można poprowadzić takich hiperpłaszczyzn nieskończenie wiele co przedstawia rysunek 3.1.



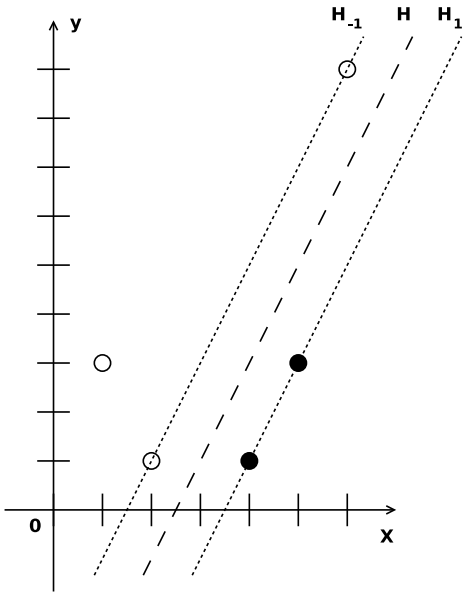
Rysunek 3.1: Przykładowe hiperpłaszczyzny oddzielające obiekty kategorii -1 (reprezentowane przez białe punkty), od obiektów kategorii 1 (czarne punkty).

Niech istnieją dwie hiperpłaszczyzny równoległe do H, tj. H_1 i H_{-1}, gdzie w H_1 zawiera się najbliższe do H obiekty z klasy 1 (jak opisuje wzór 3.2), a w H_-1 najbliższe obiekty z klasy -1 (jak opisuje wzór 3.3). Wtedy

$$\mathbf{w}^T \mathbf{x}_i + b = 1 \tag{3.2}$$

$$\mathbf{w}^T \mathbf{x}_i + b = -1 \tag{3.3}$$

Co więcej niech hiperpłaszczyzna H znajduje się w równej odległości od H_1 i H_{-1}. Taką sytuację obrazuje rysunek 3.2. Widać na nim przykłady należące do klasy -1 i 1, reprezentowane odpowiednio przez obiekty białe i czarne.Linią wykropkowaną oznaczono hiperpłaszczyzny H_1, H_{-1}, zaś przerywaną hiperpłaszczyznę H



Rysunek 3.2: Hiperpłaszczyzna separująca H, równoległa do hiperpłaszczyzn H_{-1} i H_1. H_{-1} i H_1 opierają się na obiektach należących odpowiednio do klasy -1 i 1.

Wynika z tego, że pomiędzy H_1, a H_{-1} nie ma ani jednego obiektu. H będzie zlokalizowana w tej samej odległości zarówno od H_1 jak i H_{-1}. Wszystkie obiekty spełniają warunki 3.4, 3.5, tj.

$$\mathbf{w}^T\mathbf{x}_i + b \geq 1, \quad gdy \quad y_i = 1 \tag{3.4}$$

$$\mathbf{w}^T\mathbf{x}_i + b \leq -1, \quad gdy \quad y_i = -1 \tag{3.5}$$

Można je ujednolicić do warunku 3.6.

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0 \tag{3.6}$$

Normę euklidesową dla dowolnego wektora $\mathbf{z}$ opisuje wzór

$$||\mathbf{z}|| = \sqrt{\mathbf{z}^T\mathbf{z}} = \sqrt{z_1^2 + \cdots + z_n^2} \tag{3.7}$$

Przy tym odległość punktu $\mathbf{x}_i$ od hiperpłaszczyzny H opisuje wzór

$$d(\mathbf{x}_i, H) = \frac{|\mathbf{w}^T\mathbf{x}_i + b|}{||\mathbf{w}||} \tag{3.8}$$

Konsekwentnie hiperpłaszczyzny H, H_1 oraz H_{-1} są odległe od punktu zero o

$$d(\mathbf{0}, H) = \frac{|b|}{||\mathbf{w}||} \tag{3.9}$$

$$d(\mathbf{0}, H_1) = \frac{|1 - b|}{||\mathbf{w}||} \tag{3.10}$$

$$d(\mathbf{0}, H_{-1}) = \frac{|-1-b|}{||\mathbf{w}||} \tag{3.11}$$

Natomiast odległość między hiperpłaszczyznami H_1 i H_{-1} to

$$d(\mathbf{0}, H_1) + d(\mathbf{0}, H_{-1}) = \frac{2}{||\mathbf{w}||} \tag{3.12}$$

Powyższy wzór opisuje kluczową wartość z punktu widzenia opisywanej metody. Otóż chcąc otrzymać najlepszy możliwy klasyfikator staramy się, aby odległość hiperpłaszczyzn H_1 i H_{-1} była jak największa. Dzięki temu minimalizuje się możliwość pomyłki w późniejszej klasyfikacji. Zamiast maksymalizować formułę 3.12 można równoważnie zająć się minimalizacją $||\mathbf{w}||$, $\frac{||\mathbf{w}||}{2}$, bądź też korzystając z nieujemności $||\mathbf{w}||$ zająć się minimalizacją $\frac{||\mathbf{w}||^2}{2}$, tj. $\frac{\mathbf{w}^T\mathbf{w}}{2}$, przy zachowaniu ograniczenia ze wzoru 3.6. Finalnie otrzymujemy zadanie następującej postaci

$$\begin{aligned} \min \frac{\mathbf{w}^T\mathbf{w}}{2} \\ p.o. \\ y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0 \end{aligned} \tag{3.13}$$

Dobrze znanym przykładem rozwiązywania problemu minimalizacji jest metoda mnożników Lagrange'a. Biorąc pod uwagę równanie 3.13 i towarzyszące mu ograniczenie otrzymujemy następującą funkcję Lagrange'a

$$\begin{aligned} L(\mathbf{w}, b, \alpha, \mathbf{x}) &= \frac{\mathbf{w}^T\mathbf{w}}{2} - \sum_{i \in I} \alpha_i[(\mathbf{x}_i^T\mathbf{w} + b) \cdot y_i - 1] \\ &= \frac{\mathbf{w}^T\mathbf{w}}{2} - \sum_{i \in I} \alpha_i \cdot \mathbf{x}_i^T\mathbf{w} \cdot y_i \\ &- \sum_{i \in I} \alpha_i \cdot b \cdot y_i + \sum_{i \in I} \alpha_i \end{aligned} \tag{3.14}$$

W powyższym równaniu α jest wektorem nieujemnych współczynników Lagrange'a. Problem, który należy rozwiązać sprowadza się do znalezienia punktu siodłowego, gdzie osiągnięte zostanie maksimum funkcji, przy jednoczesnej minimalizacji **w** oraz b i maksymalizacji α. Aby taki punkt siodłowy mógł istnieć warunkiem koniecznym w myśl twierdzenia Karusha-Kuhna-Tuckera jest zerowanie się gradientu funkcji Lagrange'a względem wektora **w**, pochodnej cząstkowej względem b oraz spełnienie warunków

$$\forall_{i \in I}\, \alpha_i[(\mathbf{x}_i^T\mathbf{w} + b) \cdot y_i - 1] = 0 \tag{3.15}$$

Sprawdźmy najpierw warunek na zerowanie się gradientu funkcji Lagrange'a

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \tag{3.16}$$

$$\begin{aligned} \mathbf{w} - \sum_{i \in I} y_i \cdot \alpha_i \cdot \mathbf{x}_i = 0 \\ \mathbf{w} = \sum_{i \in I} y_i \cdot \alpha_i \cdot \mathbf{x}_i \end{aligned} \tag{3.17}$$

Następnie przejdźmy do warunku na zerowanie się pochodnej cząstkowej po wartości b

$$\frac{\partial L}{\partial b} = 0 \tag{3.18}$$

$$\sum_{i \in I} \alpha_i \cdot y_i = 0 \tag{3.19}$$

Należy uwzględnić wymagania 3.17, 3.19 we wzorze 3.14. Poszczególne kroki prezentują się w następujący sposób

$$\frac{1}{2}(\sum_{i \in I} y_i \cdot \alpha_i \cdot \mathbf{x}_i)^2 - (\sum_{i \in I} y_i \cdot \alpha_i \cdot \mathbf{x}_i)^2 + \sum_{i \in I} \alpha_i \tag{3.20}$$

$$W(\alpha, \mathbf{x}) = \sum_{i \in I} \alpha_i - \frac{1}{2} \sum_{i,j \in I} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \mathbf{x}_i^T \mathbf{x}_j \tag{3.21}$$

$$\begin{aligned} W(\alpha, \mathbf{x}) = \sum_{i \in I} \alpha_i - \frac{1}{2} \sum_{i,j \in I} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot K(\mathbf{x}_i, \mathbf{x}_j) \\ dla\, K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \end{aligned} \tag{3.22}$$

Celem jest zatem znalezienie wartości takich, że

$$\max_{\alpha} W(\alpha, \mathbf{x}) \tag{3.23}$$

Przy tym z warunku 3.15 wynika, że współczynnik α_i może być inny niż zero tylko jeżeli i jest indeksem wektora podpierającego, tj. $\forall_{i \in SV} \alpha_i$, gdzie SV to zbiór indeksów wektorów podpierających. Wzór 3.22, będący przekształceniem wzoru 3.21 podkreśla fakt, że dla znalezienia wektora α_i ważny jest iloczyn wektorów $\mathbf{x}_i$,$\mathbf{x}_j$ nie zaś same ich wartości. Otwiera to możliwość wykorzystania innego niż liniowe przekształcenia $K(\mathbf{x}_i, \mathbf{x}_j)$.

Wracając do równania 3.23, można przekształcić je do postaci

$$\begin{aligned} \max_{\alpha} -\frac{1}{2} \alpha^T Q \alpha + f^T \alpha \\ p.o \\ y^T \alpha = A \\ 0 \leq \alpha \leq B \end{aligned} \tag{3.24}$$

gdzie $Q_{ij} = y_i y_j \mathbf{x}_i^T \mathbf{x}_j$, $\mathbf{f}$=$\mathbf{1}$, $\alpha = [\alpha_1, \cdots, \alpha_n]$, $A = 0$, $B \to \infty$. Ostatecznie otrzymujemy rozwiązanie α^0 problemu 3.23, dzięki któremu można uzyskać wartości $\mathbf{w}^0$ oraz b^0

$$\mathbf{w}^0 = \sum_{i \in I} \alpha_i^0 y_i \mathbf{x}_i = \sum_{j \in SV} \alpha_j^0 y_j \mathbf{x}_j \tag{3.25}$$

$$b^0 = \frac{1}{N_{SV}} \sum_{j \in SV} (\frac{1}{y_j} - \mathbf{x}_j^T) \mathbf{w}^0 \tag{3.26}$$

Optymalna hiperpłaszczyzna $f(\mathbf{x}_i)$ jest opisana wzorem $f(\mathbf{x}_i) = \mathbf{x}_i^T \mathbf{w}^0 + b^0$, zaś funkcja decyzyjna to $signum(f(\mathbf{x}_i))$.

3.1.2 Klasyfikacja liniowa - uogólnienie

W rozdziale 3.1.1 przedstawiono sposób postępowania z danymi separowalnymi liniowo tj. takimi, że dane z jednej kategorii da się oddzielić od tych z drugiej. Nie we wszystkich przypadkach taki warunek jest do spełnienia. Dlatego panowie Vapnik i Cortes rozwinęli omówioną metodę Support Vector Classification (SVC), w Cost Support Vector Classification (C-SVC), czyli tzw. metodę miękkiego marginesu.

Opiera się ona na zastąpieniu równań 3.4, 3.5 odpowiednio przez

$$\mathbf{w}^T \mathbf{x} + b \geq 1 - \xi_i, \quad gdy \quad y_i = 1 \tag{3.27}$$

$$\mathbf{w}^T \mathbf{x} + b \leq -1 + \xi_i, \quad gdy \quad y_i = -1 \tag{3.28}$$

przy czym

$$\xi_i \geq 0, i \in I \tag{3.29}$$

Powyższe krócej zapisuje się przez

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) + \xi_i - 1 \geq 0 \tag{3.30}$$

ξ_i to składnik umożliwiający łamanie ograniczenia na nieprzebywanie żadnego obiektu między hiperpłaszczyznami H_1 i H_{-1}, więcej - pozwala również przebywać obiektowi zaklasyfikowanemu do grupy 1 w strefie grupy -1. Uwzględnienie składnika ξ we wzorze 3.13 prowadzi do następującego zadania

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \frac{\mathbf{w}^T \mathbf{w}}{2} + C \sum_{i \in I} (\xi_i)^2 \\ p.o \\ \forall_{i \in I}\, y_i(\mathbf{w}^T \mathbf{x} + b) + \xi_i - 1 \geq 0 \\ \forall_{i \in I}\, \xi_i \geq 0 \end{aligned} \tag{3.31}$$

Występujący powyżej współczynnik C jest związany z karaniem przykładów z grupy 1, bądź grupy -1, które przekraczają hiperpłaszczyznę - odpowiednio H_1, bądź H_{-1}.

Pierwszy człon funkcji celu 3.31, tak jak w przypadku SVC odpowiada za maksymalizację marginesu, drugi zaś za zmniejszenie przekroczeń marginesów przez przykłady.

Drogą przekształceń z poprzedniego rozdziału otrzymujemy zadanie postaci

$$\begin{aligned} W(\alpha, \mathbf{x}) = \sum_{i \in I} \alpha_i - \frac{1}{2} \sum_{i,j \in I} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot K(\mathbf{x}_i, \mathbf{x}_j) \\ dla\, K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \cdot \mathbf{x}_j \\ p.o. \\ \sum_{i \in I} \alpha_i \cdot y_i = 0 \\ \forall_{i \in I} 0 \leq \alpha_i \leq C \end{aligned} \tag{3.32}$$

Pamiętajmy też, że dla $C \to \infty$ zadanie C-SVC zamienia się w klasyczne zadanie SVC.

3.1.3 Klasyfikacja nieliniowa

W rozdziale 3.1.1 przyjęto, że zbiór obserwacji musi być liniowo separowalny, zaś w rozdziale 3.1.2 osłabiono ten warunek dla znalezienia najlepszej możliwej hiperpłaszczyzny w zbiorze nieseparowalnym liniowo. Jednak istnieją dane, dla których możliwe jest znalezienie innego, lepszego niż liniowe rozwiązanie. W zastosowaniach takich zamiast szukać hiperpłaszczyzny, celem jest odkrycie hiperpowierzchni separującej, będącej w dwóch wymiarach krzywą, a w trzech powierzchnią.

Warto przy tym zauważyć, że jeżeli możliwe jest znalezienie takiej hiperpowierzchni, to istnieje szansa znalezienia przekształcenia $\Phi(\mathbf{x}_i)$, w którym takowa hiperpowierzchnia zmieni się w hiperpłaszczyznę. Gdyby dało się znaleźć takie Φ, możliwym byłoby jego wykorzystanie wzoru 3.22 dla

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

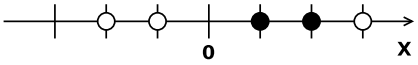
Wtedy poszukiwaną optymalną hiperpłaszczyzną byłoby $f(\Phi(\mathbf{x}_i))$, a odzwierciedlałby ją wzór $f(\Phi(\mathbf{x}_i)) = \Phi(\mathbf{x}_i^T)\mathbf{w}^0 + b^0$. Funkcja decyzyjną byłoby zaś $signum(f(\Phi(\mathbf{x}_i)))$.

Co istotne znajomość samego przekształcenia Φ nie jest konieczna do odnalezienia hiperpłaszczyzny.

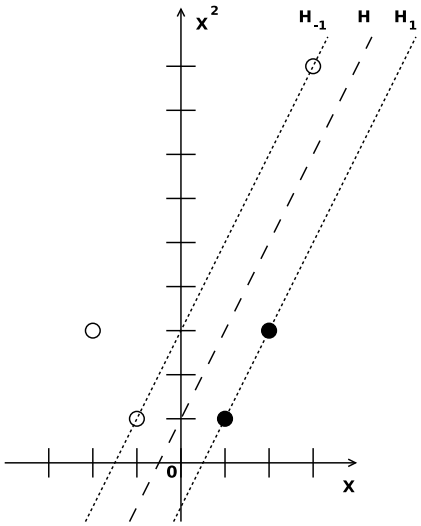
W pewnych okolicznościach dysponując przekształceniem $K(\mathbf{a}, \mathbf{b})$, można odtworzyć Φ. Weźmy za przykład weźmy obiekty $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2)$, $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2)$ i przekształcenie $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^2$. Wtedy

$$\begin{aligned} K(\mathbf{a}, \mathbf{b}) &= (\mathbf{a}^T \mathbf{b})^2 \\ &= \mathbf{a}_1^2 \mathbf{b}_1^2 + 2\mathbf{a}_1 \mathbf{a}_2 \mathbf{b}_1 \mathbf{b}_2 + \mathbf{b}_1^2 \mathbf{b}_2^2 \\ &= (\mathbf{a}_1^2, \sqrt{2}\mathbf{a}_1 \mathbf{a}_2, \mathbf{a}_2^2)^T (\mathbf{b}_1^2, \sqrt{2}\mathbf{b}_1 \mathbf{b}_2, \mathbf{b}_2^2) \\ \Phi(\mathbf{x}) &= (\mathbf{x}_1^2, \sqrt{2}\mathbf{x}_1 \mathbf{x}_2, \mathbf{x}_2^2) \end{aligned} \tag{3.33}$$

Za przykład przydatności przekształceń liniowych niech posłużą obiekty **x**={-2,-1,1,2,3} o parametrach decyzyjnych **y**={-1,-1,1,1,-1}. Mamy wtedy do czynienia z sytuacją przedstawioną na rysunku 3.3, rozwiązaną dzięki przekształcenia kwadratowemu, co obrazuje rysunek 3.4.



Rysunek 3.3: Obiekty nieseparowalne liniowo w przestrzeni jednowymiarowej.



Rysunek 3.4: Obiekty przeniesione z przestrzeni jednowymiarowej do dwuwymiarowej.

Przekształceniami popularnie wykorzystywanymi są przekształcenia

- wielomianowe – $K(x_i, x_j) = (\gamma x_i^T x_j + c)^d$
- radialne – $K(x_i, x_j) = e^{-\gamma|x_i - x_j|^2}$
- sigmoidalne – $tanh(\gamma x_i^T x_j + c)$

gdzie wymienione zmienne oznaczają:

- d – stopień wielomianu
- γ – współczynnik gamma
- c – wyraz wolny

3.2 Klastrowanie hierarchiczne

W rozdziale 3.1 rozróżniono uczenie z nadzorem i bez nadzoru, wskazano również zadanie klasyfikacji jako przykład uczenia z nadzorem. Przykładem uczenia bez nadzoru jest problem klastrowania. Rzecz sprowadza się do podzielenia obiektów wejściowych na klastry, tj. zbiory obiektów, gdzie żaden z obiektów nie występuje w dwóch różnych klastrach jednocześnie. Co więcej każdy z obiektów klastra jest bardziej podobny do klastra, do którego należy niż do innych klastrów. Do dokonania takiego podziału wykorzystuje się miarę wzajemnego podobieństwa $\sigma(x_i, x_j)$ między obiektami x_i, x_j.

Wyróżniamy zasadniczo dwa główne rodzaje klastrowania: wstępujące (ang.bottom-up) i zstępujące (ang.top-down). Dla klastrowania zstępującego w każdym kroku dokonywany jest podział wybranego zbioru na n (np.n=2) podzbiorów, aż do osiągnięcia pewnego warunku zatrzymania klastrowania, np. spójności uzyskanych zbiorów albo zadanej ilości klastrów. Przykładem klastrowania zstępującego jest problem dyskretyzacji zbioru wartości, np. przekształcenie wartości wskazań termometru z przedziału [36,45] do 6 wartości.

W przypadku klastrowania wstępującego mamy do czynienie z podejściem ód szczegółu do ogółu". Bazowo uznaje się, że każdy obiekt stanowi osobny klaster. Następnie w każdym kroku sprawdza się, która para klastrów jest do siebie najbardziej podobna i łączy się ją w jeden klaster.

Występuje przy tym wiele możliwych wariantów klastrowania opierających się o różne warunki podobieństwa klastrów, a najbardziej popularne to:

1. Klastrowanie proste (ang.Single Linkage Clustering, złożoność $\mathcal{O}(N^2)$)
2. Klastrowanie bazujące na średnich (ang.Average Linkage Clustering, złożoność $\mathcal{O}(log(N) \cdot N^2)$)
3. Klastrowanie pełne (ang. Complite Linkage Clustering, złożoność $\mathcal{O}(log(N) \cdot N^2)$)

Decydując o mierze podobieństwa należy mieć na względzie zarówno charakter klastrowanych danych jak również złożoność obliczeniową związaną z daną strategią.

3.2.1 Naiwna implementacja SLC

Prosta implementacja SLC ma złożoność obliczeniową rzędu $\mathcal{O}(N^3)$. Mając macierz podobieństwa, C oraz miarę podobieństwa $\sigma(\mathbf{x}_i, \mathbf{x}_j)$ obiektów $\mathbf{x}_i, \mathbf{x}_j$, w każdym kroku wybieramy najpodobniejsze dwa klastry ($\mathcal{O}(N^2)$), po czym łączymy je ze sobą. Jeden z klastrów jest włączany w drugi, co jest odnotowane w tablicy I, przez deaktywację (przypisanie wartości 0). Określa się również podobieństwo uzyskanego klastra o składowych $\mathbf{x}_i, \mathbf{x}_j$ do innych klastrów, $\mathbf{x}_k$. Wartością tą jest maksymalna wartość spomiędzy $\sigma(\mathbf{x}_i, \mathbf{x}_k)$, $\sigma(\mathbf{x}_j, \mathbf{x}_k)$. Proces łączenia powtarza się, aż do powiązania wszystkich klastrów ($\mathcal{O}(N)$), następnie zwracana jest lista kolejnych złączeń, A, w kolejności od pierwszego do ostatniego.

Implementację takiego procesu prezentuje program 1.

Program 1 NaiwneKlastrowanieProste$(d_1, d_2, \cdots, d_n)$

```
1: for n ← 1 to N do
2:    for i ← 1 to N do
3:       C[n][i] ← σ(d_n, d_i)
4:    end for
5:    I[n] ← 1
6: end for
7: A ← []
8: for k ← 1 to N-1 do
9:    ⟨i, m⟩ ← arg max_{⟨i,m⟩:i≠m∧I[m]=1} C[i, m]
10:   A.DOLACZ(⟨i, m⟩)
11:   for j ← 1 to N do
12:      C[i][j] ← σ(i, m, j)
13:      C[j][i] ← σ(i, m, j)
14:   end for
15:   I[m] ← 0
16: end for
17: return A
```

3.2.2 Efektywna implementacja SLC

W efektywnej implementacji SLC korzysta do puli informacji dołącza się tablicę $NajblizszyKlaster$ zawierającą indeks najbliższego klastra, wraz z wartością tego podobieństwa. Po inicjalizacji, następuje iteracyjne wybieranie złączeń ($\mathcal{O}(N)$). W każdej iteracji wyszukiwane jest największe podobieństw między klastrami ($\mathcal{O}(N)$). Zapamiętuje się przy tym, który klaster należy do takiej pary i sprawdza w tablicy $NajblizszyKlaster$ z jakim klastrem jest powiązany. Po tym jeden z klastrów (i_i) jest włączany w drugi (i_j), co odzwierciedla przypisanie w tablicy I, na pozycji i_i wartości i_j. Dokonywane jest też scalenie wartości podobieństwa klastrów i_i oraz i_j, wybierając przy tym największe podobieństwo do innych klastrów i_k, tj. $max(\{\sigma(i_i, i_k), \sigma(i_j, i_k)\})$. W kolejnych iteracjach mogą brać udział tylko klastry,

które nie zostały włączone w inne, tj. takie gdzie $I[n] = n$. Na koniec zwracana jest tablica wykonywanych złączeń, A, w kolejności od pierwszego do ostatniego.

Implementację takiego procesu prezentuje program 2.

Program 2 EfektywneKlastrowanieProste$(d_1, d_2, \cdots, d_n)$

```
1: for n ← 1 to N do
2:    for i ← 1 to N do
3:       C[n][i].podob ← σ(d_n, d_i)
4:       C[n][i].indeks ← i
5:    end for
6:    I[n] ← n
7:    NajblizszyKlaster[n] ← arg max_{X∈{C[n][i]:n≠i}} X.podob
8: end for
9: A ← []
10: for n ← 1 to N-1 do
11:    i_1 ← arg max_{i:I[i]=i} NajblizszyKlaster.podob
12:    i_2 ← I[NajblizszyKlaster[i_1].indeks]
13:    A.DOLACZ(⟨i_1, i_2⟩)
14:    for i ← 1 to N do
15:       if I[i] = i ∧ i ≠ i_1 ∧ i_2 then
16:          C[i_1][i].podob ← C[i][i_1].podob ← max(C[i_1][i], C[i_2][i])
17:       end if
18:       if I[i] = i_2 then
19:          I[i] = i_1
20:       end if
21:    end for
22:    NajblizszyKlaster[i_1] ← arg max_{X∈{C[i_1][i]:I[i]=i∧i≠i_1}} X.podob
23: end for
24: return A
```

3.2.3 Uwagi

Algorytmy 1., 2. opisują przebieg pełnego procesu klastrowania. Oczywiście można ustalić pewien próg, T, który przerywa proces klastrowania jeżeli największe podobieństwo jest niższe od niego. Jest to o tyle istotne, że łączenie klastrów niepodobnych do siebie może nie stanowić istotnej informacji dla użytkownika.

Wykorzystana w nich metoda wyliczania podobieństwa utworzonego klastra do innych klastrów, $\sigma(x_i, x_j, x_k)$ pociąga za sobą pewne konsekwencje, tj. tzw. tworzenie łańcuchów. Otóż przy łączeniu sprawdzeniu podlega tylko podobieństwo dwóch najbliższych sobie obiektów z rozpatrywanych klastrów. W ten sposób łatwo może dojść do sytuacji, gdzie tworzy się bardzo niespójny klaster. Jeżeli chce się uniknąć tworzenia łańcuchów lepiej skorzystać z innej metody wyliczania podobieństwa,

jak porównywanie środków ciężkości klastrów, albo porównywanie najdalszych sobie obiektów w obrębie zadanych klastrów.

3.3 Baza trójek

Większość naszej wiedzy opiera się na opisie danego podmiotu w pewnym jego aspekcie, np. Jan jest synem Pawła, kwadrat jest prostokątem, etc. Stwierdzenia takie są uporządkowaną trójką

$$\langle podmiot, relacja, obiekt \rangle$$

. Jednocześnie o ile obiekt identyfikowany jest unikalnym id, nic nie stoi na przeszkodzie, aby obiekt w jednej relacji stał się podmiotem w następnej. W takiej bazie możemy też pokusić się o stworzenie pewnych reguł np. gdy

$$\langle A, ojciec, B \rangle \wedge \langle B, rodzic, C \rangle \Rightarrow \langle A, dziadek, C \rangle$$

Można też odwrócić relację, tj.

$$\langle A, rodzic, B \rangle \Rightarrow \langle B, dziecko, A \rangle$$

Obecnie istnieje standard opisujący przechowywanie danych w trójkowy sposób a jest to Szkielet Opisu Zasobów [3] (ang. Resource Description Framework, RDF). W RDF przyjmuje się, że podmiot oraz relacja (inaczej predykat) powinny być przedstawione jako Ujednolicony Identyfikator Zasobów (Uniform Resource Identifier, URI) zdefiniowany w dokumencie RFC 2396 [7]. Obiektem może być zarówno URI jak też dowolny literał, bądź liczba. Warto zwrócić uwagę na to, że z poszczególnych trójek łatwo można stworzyć grafowy opis zadanego problemu, przy czym węzłami się podmiot i obiekt, zaś krawędziami się predykaty. Obecnie istnieje wiele implementacji baz danych przechowujących dane spełniające standard RDF jak 3store, AllegroGraph, bigdata, Jena, Sesame, czy Virtuoso [2], różniących się zarówno językiem w jakim zostały stworzone (C, Java, Common Lisp), jak i językami zapytań (m.in. SPARQL [5] (ang.Simple Protocol And RDF Query Language), SeRQL [6] (ang. Sesame RDF Query Language)). Dzięki nim można zarówno otrzymać konkretne wartości z wyszukanych grafów, jak również całe grafy. W niniejszej pracy obiektem zainteresowania są Sesame oraz bigdata®.

3.3.1 Sesame

Sesame [4] jest otwartą, zaimplementowaną w języku programowania Java, bazą umożliwiającą składowanie danych spełniających standard RDF oraz wyszukiwanie ich. Baza danych może być składowana zarówno na dysku (ang.Native Store) jak też w pamięci (ang.in-Memory Store). Możliwy jest również eksport danych w kilku standardach (m.in. RDF, N3). Można łączyć się z nią za pośrednictwem interfejsu programowania aplikacji (ang.Aplication Programming Interface,API) RDF SAIL (ang.

RDF Storage And Inference Layer, RDF SAIL, SAIL). Umożliwia składowanie do 70 milionów trójek [2] .

3.3.2 bigdata®

bigdata® [1] jest komercyjną bazą danych zaimplementowaną w języku Java. Podobnie jak Sesame składowane dane spełniają standard RDF, a programy mogą łączyć się za pośrednictwem SAIL API. Umożliwia składowanie do 12.7 miliardów trójek [2]. Jakkolwiek produkt ten jest komercyjny, można z niego korzystać na Generalnej Publicznej Licencji w wersji 2 [11] (General Public Licence version 2, GPL v2).

Twórcy bazy opisują ją jako skonstruowaną do efektywnego składowania i odpytywania, o wbudowanych wysoko efektywnych mechanizmach pracy równoległej.

3.4 Map/Reduce

W roku 2004 pracownicy Google Jeffrey Dean i Sanjay Ghemawat zaproponowali model obliczeń wielkoskalowych bazujących na metodzie MapReduce [26]. Najprościej rzecz ujmując proponuje się podzielenie obliczeń na dwie fazy. Na każdym etapie węzły przetwarzające polecenie przyjmują i zwracają listę par $\langle klucz, warto \rangle$. Istotnym założeniem jest, znane np. z języka Scala, czyste (ang.pure) wykonanie funkcji, tj. brak zmiennych globalnych ulegających modyfikacji w trakcie wykonania którejkolwiek funkcji, które wpływają na sposób obliczeń oraz niezmienność danych w trakcie wykonania obliczeń.

W pierwszej fazie - mapowania - problem główny jest dekomponowany przez węzeł główny na podproblemy przydzielane węzłom roboczym mapującym. Następnie węzły robocze mapujące wykonują obliczenia, których wyniki są przekazywane do węzłów roboczych redukujących. W kolejnej fazie - redukcji - dokonywana jest konsolidacja wyników.

Dla ustalenia uwagi weźmy przykład grupowania ID dokumentów wg. tożsamości autora. Wejściem jest ⟨klucz:identyfikator dokumentu, wartość: sparsowane dane dokumentu⟩. Węzeł roboczy mapujący zwraca pary ⟨klucz:nazwisko autora dokumentu, wartość: sparsowane dane dokumentu⟩, zaś węzeł roboczy redukujący dla każdego klucza (nazwiska autora) klastruje powiązane z nim dokumenty emitując ⟨klucz:ID autora, wartość: listę ID powiązanych z autorem dokumentów⟩. Dzięki tak pozyskanym danym użytkownik końcowy korzystając z danego dokumentu będzie mógł się odpytać o inne dokumenty napisane przez tego samego autora.

Najistotniejszym aspektem metody MapReduce jest niezmienność danych wejściowych w trakcie dokonywania operacji, a co zatem idzie możliwość wykonania niezależnie od siebie obliczeń. Dla zadań, które da się zrównoleglić można uzyskać rezultat w dużo krótszym czasie - modelowo N razy krótszym, gdzie N to ilość wykorzystanych węzłów roboczych. Co więcej, jeżeli jakiś węzeł ulegnie awarii i wynik jego obliczeń zostanie utracony możliwe jest przydzielenie tego zadania do innego węzła

roboczego. W tym aspekcie niezwykle ważna jest dbałość o węzeł główny, przechowujący informacje o tym, którym węzłom roboczym przydzielono poszczególne obliczenia.

W związku z ogromnym rozwojem maszyn wielordzeniowych model MapReduce cieszy się ogromną popularnością. Znane są jego liczne implementacje, m.in. cieszący się dużą popularnością Apache Hadoop [9] dostępny na licencji Apache License 2.0 [10].

Rozdział 4

Opis szkieletu do rozróżniania autorstwa

4.1 Definicje

Na wstępie tego rozdziału warto usysytematyzować słownik pojęć, używanych w dalszej części pracy. Podstawową omawianą jednostką jest **dokument**. Typowo dokument to artykuł, któremu przypisany jest unikalny w rozpatrywanej kolekcji numer identyfikacyjny (ID). Oprócz numery ID dostępne są metadane dokumentu tj. informacje takie jak: lista nazwisk autorów i ich afiliacji, kody klasyfikacyjne pracy, streszczenie (inaczej: abstrakt), słowa kluczowe i treść, data publikacji oraz dane o jednostce publikującej (np. Springer, Elsevir).

Aby odróżnić osobę autora od nazwiska widniejącego na pracy wprowadza się pojęcie **kontrybutora**. Praca napisana przez kontrybutora to **kontrybucja**. Każdy kontrybutor łączy się z jedną kontrybucją. Ostatecznie pojęć tych można używać zamiennie, przy czym kiedy mówi się o **kontrybutorze** kładzie się nacisk na osobę tworzącą artykuł, kiedy zaś mowa o **kontrybucji** podkreśla się rolę dokumentu. Zadanie rozróżniania autorstwa prac polega na zgrupowaniu kontrybutorów, których uznaje się za jednego autora na podstawie podobieństwa kontrybucji. Dzięki temu dostarcza się użytkownikowi biblioteki wirtualnej informację o innych pracach autora powiązanego z daną kontrybucją. Dla przykładu weźmy artykuł o ID BibliotekaX:123. Na jego liście autorów widnieją: "A.Abacki", "B.Babacki", "C.Cabacki". Otrzymujemy zatem odpowiednio trzy kontrybucje (trzech kontrybutorów): "BibliotekaX:123#c1", "BibliotekaX:123#c2", "BibliotekaX:123#c3", gdzie np. kontrybucja "BibliotekaX:123#c3" jest powiązana z panem "C.Cabackim".

Niech znana też będzie **funkcja mieszająca**, która przyjmuje ciąg znaków, np. nazwisko, i dokonuje na nim przekształcenia. Tym przekształceniem może być dokonanie transliteracji do alfabetu łacińskiego i usunięcie znaków diakrytycznych oraz

zamiana liter na małe.

Określmy jako **zbiór roboczy** taki zbiór kontrybutorów, dla których wynik funkcji mieszającej jest taki sam. Jeżeli za funkcję mieszającą przyjmiemy taką, która usuwa znaki diakrytyczne, to w jednym zbiorze roboczym znajdą się zatem kontrybutorzy "L.Możdżer" i "L.Mozdzer".

Atrybutem kontrybutora, bądź **cechą** kontrybutora określane będą konkretne pole metadanych. Wartością atrybutu "rok wydania" może być zatem "2012", wartością atrybutu "kod klasyfikacji" – "MSC:03C15". W przypadku braku informacji o danym atrybucie przypisana mu będzie **wartość pusta**, tzw. **null**.

Narzędzie dokonujące sprawdzenia podobieństwa kontrybucji, tzw. **wskazówka**, dokonuje porównania każdej pary z zadanych kolejno zbiorów roboczych. Wskazówka zwraca wartość ze zbioru [-1,1], gdzie wartości 1 oznacza "na pewno tego samego autora", -1 "na pewno innego autora", zaś 0 "trudno powiedzieć". Miarą istotności danej wskazówki jest jej **waga**, która jest nieujemną wartością rzeczywistą. **Podobieństwem cząstkowym** jest iloczyn wartości wskazówki oraz jej wagi. **Podobieństwo całkowite** to suma podobieństw cząstkowych.

Kontrybucje włączone na podstawie podobieństwa całkowitego do jednego zbioru (klastra) łączy ta sama **persona**, inaczej **tożsamość**, czyli domniemany autor. Persona to obiekt sztucznie utworzony, opisany unikalnym ID np. "BibliotekaX:123#p1".

Niech będzie zdefiniowane również pojęcie **pary kontrybutorów**. Obiekt taki jest połączony z dwoma różnymi kontrybutorami. Jego atrybutami są **surowe cechy**, czyli liczba wspólnych wystąpień danej cechy dla dwóch kontrybucji. Dla cechy "wspólny kod klasyfikacyjny" wartość surowej cechy może być równa np. 3, tzn. że dwie kontrybucje współdzielą trzy kody klasyfikacyjne. Innym przykładem może być surowa cecha "współdzielone nazwiska innych autorów" równa 2 [27].

Surowe cechy, inaczej niż wskazówki, zwracają wartość całkowitą ze zbioru liczb całkowitych nie mniejszą niż -1. -1 jest w tym wypadku specjalną wartością mającą za zadanie wskazać, że jedna lub obie z kontrybucji nie posiadały informacji o badanej cesze w bazie danych. Ma to miejsce, np. wtedy kiedy tylko o jednej prac wiemy, kiedy została opublikowana.

Wartości **opracowanych cech** to wartości surowych cech, na których dokonano pewnego przekształcenia, np. liniowego przeskalowania do wartości ze zbioru [-1,1]. Takie przekształcenia zmienia opracowane cechy we wskazówki.

4.2 Etapy procesu rozróżniania autorów

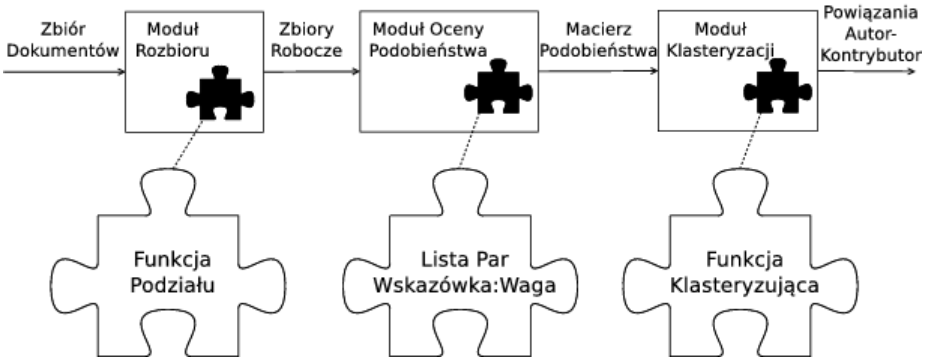
Proces rozróżniania autorstwa składa się z trzech zasadniczych etapów: wczytania metadanych, podzielenia wszystkich kontrybucji na zbiory robocze (gdzie kontrybutorzy dzielą wspólną cechę, jak nazwisko), zapisanie ID zbiorów powiązanych z ID ich elementów. Wyróżnia się zatem trzy części:

1. Podzielenie kontrybucji na zbiory robocze

2. Wyznaczenie podobieństwa całkowitego par w zbiorze roboczym

3. Klastrowanie par kontrybucji na podstawie macierzy podobieństwa całkowitego par

Opisane etapy prezentuje rysunek 4.1

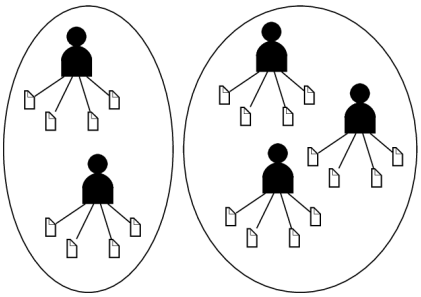


Rysunek 4.1: Etapy rozróżniania autorstwa. Puzle reprezentują modyfikowalne części szkieletu.

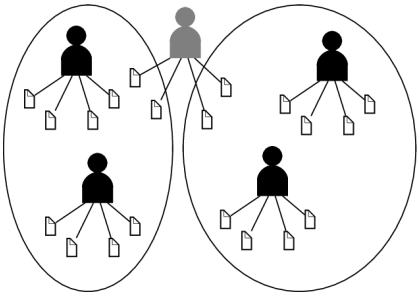
4.2.1 Podział na zbiory robocze

Porównanie każdej kontrybucji, a później ich klastrowanie, oznacza złożoność obliczeniową na poziomie $\mathcal{O}(N^2)$. Typowo w procesie rozróżniania autorstwa biorą udział miliony dokumentów. Aby zadanie dało się zrealizować w rozsądnym czasie, ważna jest jego dekompozycja, tj, podział wejściowego zbioru kontrybucji na zbiory robocze z wykorzystanie funkcji mieszającej. Odpowiednio zdefiniowana funkcja mieszająca sprawia, że kontrybucje pojawiające się w jednym zbiorze roboczym, nie pojawiają się w innych. Wzmacniając lub osłabiając warunki dla tej funkcji, można sterować rozmiarem największego zbioru roboczego, jak również poprawnością uzyskanego podziału na tożsamości. Przykładem na osłabienie funkcji mieszającej, prowadzącym do zwiększenia rozmiary zbiorów roboczych, jest zwracanie inicjałów pierwszego imienia i nazwiska kontrybutora.

Idealna funkcja ma za zadanie umieścić w jednym zbiorze wszystkie kontrybucje tego samego autora (możliwe, że z pewną nadwyżką innych kontrybucji) oraz w maksymalnym stopniu rozdzielić prace różnych autorów. Biorąc pod uwagę wcześniej wspominane ryzyka, jak zmiana nazwiska, zadanie tak postawione może być trudne do osiągnięcia. Na odpowiednich rysunkach zilustrowano rezultatu działania poprawnego (rysunek 4.2) i niepoprawnego (rysunek 4.3).



Rysunek 4.2: Kontrybucje (ikony dokumentów) tej samej osoby (ikona człowieka) powinny znaleźć się w tym samym zbiorze roboczym (elipsa).



Rysunek 4.3: Rezultatem działania źle przygotowanej funkcji mieszającej jest rozłożenie prac (ikona dokumentu) jednego autora (ikona człowieka) na więcej niż jeden zbiór roboczy (elipsa). Utwory wyszarzonego autora są rozdzielone w ten sposób.

Porównanie nazwisk kontrybutorów

Za przykład funkcję mieszającą weźmy taką, która porównuje nazwiska o usuniętych znakach diakrytycznych, a której użyjemy do przetworzenia zbioru Muzeum Historii Polski (MHP) oraz wydawnictwa Springer.

Zbiór kontrybutorów w bazie MHP zawiera około 100.000 kontrybutorów. Największy zbiór roboczy liczył sobie 250 kontrybucji, a 55% kontrybucji mieściło się w zbiorach nie większych niż 10 elementów. Podział taki jest w najwyższym stopniu zadowalający, a czas wykonania całego procesu rozróżniania autorów - krótki. Problem złożoności kwadratowej obliczeń na zadanych zbiorach roboczych znacznie stracił na pracochłonności w porównaniu z sytuacją, gdzie takiego podziału by nie przeprowadzono.

Zbiór kontrybutorów w bazie Springer liczy sobie 38 milionów kontrybucji. Zbiory robocze liczące mniej niż 100 kontrybutorów stanowią 60,27% wszystkich kontrybutorów i 99,23% wszystkich zbiorów roboczych. Zbiory mniejsze niż 560 to odpowiednio 77,5% oraz 99,9%, przy czym największy z nich ma rozmiar 35,5 tysiąca. Podsumowując: można przetworzyć niemal wszystkie zbiory w stosunkowo krótkim czasie, jednak duża liczba kontrybutorów zgromadzona jest bardzo licznych zbiorach.

W tym kontekście bardzo ważny staje się koszt poszczególnych porównań. O ile na zbiorze MHP można dokonywać obliczeń w bazie umieszczonej na dysku twardym (mała ilość zapytań oznacza akceptowalny narzut czasowy na komunikację), o tyle w zbiorze Springer kluczowe staje się przeniesienie obliczeń poszczególnych zbiorów roboczych do pamięci.

Przykłady innych funkcji mieszających

Torvik i Smalheiser [57] porównując kontrybutorów mających ten sam adres e-mail sprawdzili na zbiorze MEDLINE, że problem różnych reprezentacji nazwiska tyczy się przynajmniej 1,8% autorów. Oznacza to, że usunięcie znaków diakrytycznych może być rozwiązaniem niewystarczającym. W mniejszych zbiorach danych takich jak MHP, gdzie osłabienie funkcji mieszającej nie zwiększy drastycznie czasu obliczeń, można zatem pokusić się o inne rozwiązania. Jednym z nich jest porównanie fonetycznej reprezentacji nazwiska. Przykładem funkcji służącej do tego jest algorytm Soundex [37] . W szczególności powstał on na zapotrzebowanie nazwisk pochodzenia amerykańskiego. Przetwarza się w nim nazwisko w kilku krokach tj. wybiera się pierwszą literę, usuwa samogłoski, przypisuje się pozostałym literom określone liczby - jeżeli liczb jest więcej niż 3 - usuwa się nadwyżkę, jeżeli mniej dodaje się zera. Finalnie uzyskuje się kod składający się z jednej litery i trzech liczb w kolejności ich wystąpienia w nazwisku.

Przy zachowaniu odpowiednich środków algorytm Soundex jest stosowany m.in. do wyszukiwania imion w zapisanej alfabetem łacińskim wersji Koranu. Niestety algorytm ten dobrze służy tylko nazwiskom pochodzącym z danego kręgu kulturowego. W szczególności nazwiska “Szulc”, “Schultz”, “Schulz” nie zwrócą tej samej wartości dla podstawowej implementacji algorytmu Soundex. Wybierając modyfikacje funkcji mieszającej należy mieć na uwadze jego ostateczne przeznaczenie i poprzedzić decyzję wyczerpującą analizą danych.

Jeżeli przypuszcza się, że w bazie znajdują się nazwiska pochodzące z różnych kręgów kulturowych, odradza się użycie algorytmu Soundex jako funkcji mieszającej.

Wyliczanie podobieństw par kontrybucji

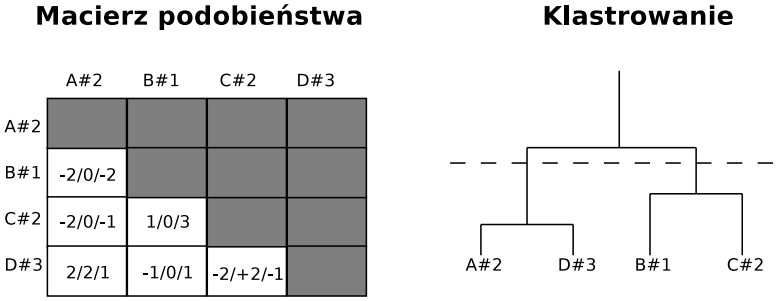
Po podzieleniu wszystkich kontrybutorów pomiędzy zbiory robocze, praca odbywa się osobno na każdym z nich. Ze zbioru roboczego wybiera się parę kontrybucji, przekazując ją do wskazówek zwracających wartość z przedziału [-1,1]. Następnie tak uzyskaną wartość mnoży się przez wagę wskazówki, uzyskując podobieństwo cząstkowe. Zsumowane podobieństwa cząstkowe stanowią podobieństwo całkowite. Końcowym produktem modułu wyliczania podobieństw par jest macierz podobieństw między poszczególnymi kontrybutorami ze zbioru roboczego.

4.2.2 Klastrowanie

Na podstawie macierzy podobieństw kontrybutorów należących do zbioru roboczego dokonywane jest klastrowanie hierarchiczne wstępujące. Jest kilka metod dokonania takowego, a najbardziej popularne to:

1. Klastrowanie proste (ang.Single Linkage Clustering, złożoność $\mathcal{O}(N^2)$)
2. Klastrowanie bazujące na średnich (ang.Average Linkage Clustering, złożoność $\mathcal{O}(log(N) \cdot N^2)$)
3. Klastrowanie pełne (ang. Complite Linkage Clustering, złożoność $\mathcal{O}(log(N) \cdot N^2)$)

Mając szczególnie na uwadze maksymalne możliwe obniżenie złożoności obliczeniowej w zaimplentowanym w tej pracy narzędziu, wykorzystano klastrowanie proste. Proces klastrowania ilustruje rysunek 4.4.



Rysunek 4.4: Dla każdej pary kontrybucji w obrębie zbioru roboczego wyliczane jest na podstawie wartości wybranych atrybutów podobieństwo całkowite. Następnie wykonywane jest klastrowanie hierarchiczne wstępujące, aby określić grupy kontrybutorów, do których przypisane zostaną tożsamości.

Rozdział 5

Opis eksperymentów

Celem nadrzędnym niniejszej pracy było dostarczenie na potrzeby EuDML modułu, który wczytuje zadaną kolekcję dokumentów, przeprowadza proces rozróżniania autorstwa i zwracając wynik do postaci plików CSV. W istocie chodziło zatem o przypisanie grupie podobnych kontrybutorów unikalnego ID persony. Problem taki w literaturze określany jest mianem klastrowania.

Aby proces ten był wykonany najszybciej i najskuteczniej postawiono sobie cztery dodatkowe cele:

1. zaproponowanie sposobu mierzenia istotności poszczególnych wskazówek
2. zaproponowanie przekształceń surowych wartości atrybutów
3. zbadanie jak przekształcenia wpływają na końcowy efekt klasyfikacji
4. zbadanie możliwości bazy Sesame oraz bigdata®, w kontekście efektywnego przeprowadzania na nich obliczeń

Niniejszy rozdział opisuje zabiegi zastosowane, aby osiągnąć wymienione cele, zaś rozdział opisuje wyniki związanych z nimi eksperymentów.

5.1 Problem klastrowania, a problem kategoryzacji

W trakcie procesu opisanego w rozdziale 4. zachodzi porównanie każdych dwóch kontrybutorów ze sobą, a następnie połączenie podobnych kontrybucji w persony (klastry). Jeżeli operuje się na zbiorze, gdzie dana jest informacja o oczekiwanym podziale, tak jak to miało miejsce w przypadku zbioru Zentralblatt MATH, można pokusić się o przejście od problemu klastrowania do problemu klasyfikacji. Zabieg taki zastosowali Wang *et* al. [58]. Operacja polega na przeprowadzeniu poniższych kroków:

1. utworzeniu obiektu pary
2. przypisaniu mu dwóch, różnych obiektów bazowych (kontrybucji)
3. sprawdzeniu, czy obiekty bazowe mają przypisane
    a) takie samo ID
    b) różne ID
4. przypisuje się obiektowi pary wartości atrybutu wynikowego, odpowiednio
    a) "prawda"/"ten sam autor"
    b) "fałsz"/"różni autorzy"

Dzięki tej prostej operacji można przejść od problemu klastrowania do problemu klasyfikacji, w szczególności klasyfikacji do dwóch kategorii.

5.2 Mierzenie istotności wskazówek

Jednym z istotnych tematów w dziedzinie rozróżnianiu autorstwa oraz ogólnie klastrowaniu jest z jednej strony dostarczenie najpełniejszej puli danych i uzupełnienie wszelkich braków w metadanych (np. dzięki użyciu technik ekstrakcji danych z pdf-ów [56][55]), z drugiej zaś odfiltrowanie atrybutów, uznanych po wyczerpującej analizie, za nie mające dostatecznie dużej istotności. Chodzi o to, aby stały wysiłek obliczeniowy poniesiony na wzięcie pod uwagę danego atrybutu (jak np. w przypadku klastrowania 38 milionów kontrybucji ze zbioru Springera) dla każdej pary obiektów przyniósł istotny wzrost dokładności podziału, np. o 1%. Aby móc odfiltrowywać atrybuty o niskim poziomie istotności, nie czyniąc tego w sposób arbitralny, bądź też nie przeprowadzając każdorazowo kosztownej czasowo analizy danych, należy stworzyć zautomatyzowany sposób przydzielania wag atrybutom.

Za pomocą opisanych w poprzedniej części obiektów par, możliwym stało się wykorzystanie do tego celu metody maszyn wektorów nośnych (SVM), jako dobrze przebadanej i sparametryzowanej metody uczenia z nadzorem. Mierzenie istotności wskazówki (inaczej: ważenie wskazówki) to proces, który ogólnie opisując przebiega w następujący sposób:

1. na wejściu modułu podane są obiekty par z wartościami wskazówek oraz wartością atrybutu decyzyjnego ("Ten sam autor","Różni autorzy")
2. zbiór dzielony jest na n równych części
3. na podstawie każdej części budowany jest liniowy klasyfikator SVM
4. skuteczność każdego z klasyfikatorów jest testowana względem połączonych części zbioru, na których klasyfikator nie był trenowany

- miarą skuteczności jest dokładność (suma poprawnie zaklasyfikowanych obiektów przez ilość wszystkich obiektów)

5. z najlepszego klasyfikatora odzyskiwany jest wzór, na podstawie którego dokonano klasyfikacji, zwracany jako wynik modułu.

Uważny czytelnik zauważy, że proces opisany w punktach 2-4 to n-stopniowa walidacja krzyżowa. Jeżeli za klasyfikator przyjmiemy SVM o liniowym jądrze przekształcenia, a wartości atrybutów mieścić się będą w przedziale [-1,1], to uzyskana hiperpłaszczyzna służąca do klasyfikacji również będzie się mieściła w tym zakresie. Funkcja opisująca tę hiperpłaszczyznę wygląda następująco:

$$f(x_1, x_2, \ldots, x_n) = sign(\sum_{i=1}^{n} a_i\, x_i + bias)$$

gdzie wartość $f(x_1, x_2, \ldots, x_n) \geq 0$ oznacza, że para kontrybutorów jest tą samą osobą, a w.p.p. że nią nie jest. Wektor $\mathbf{a} = [a_1, a_2, \ldots, a_n]$, można zaś interpretować jako wektor wag, tj. miar istotności atrybutów wejściowych. Tak otrzymane wagi zawierają się również w przedziale [-1,1]. Bias to zaś wyraz wolny równania, przesunięcie. W przypadku braku wartości atrybutów wskaże on najczęściej wybieraną decyzję. Ogromnym plusem tej metody jest łatwość z jaką można interpretować hiperpłaszczyznę oraz zależności między wskazówkami (wagi wskazówek). Minusem zaś jest założenie, że zależność między wartościami wskazówek, a atrybutem decyzyjnym jest liniowa. Przykładowo, nie można zaprzeczyć, że wystąpienie kilku wskazówek o niskich wagach może być istotniejszą informacją niż wystąpienie jednej wskazówki, przykładowo o większej wadze niż suma pozostałych. Uznanie takiego podejścia oznaczałoby konieczność wykorzystania nieliniowego, np. wielomianowego jądra przekształcenia.

Jak pokazują jednak doświadczenia, poszczególne atrybuty mają współczynniki istotności różne o rzędy wielkości, zatem klasyfikacja dokonana ze wszystkimi cechami oraz wyłącznie z cechami wysoko ocenionymi (o kilka rzędów wielkości wyżej niż reszta) dawały bardzo zbliżone rezultaty. Można zatem przyjąć, że ważenie atrybutów klasyfikatorem liniowym jest dość dobrym przybliżeniem w dziedzinie rozróżniania autorstwa.

Warto zauważyć, że uzyskane w ten sposób wagi, mogę nie spełnić ograniczenia, jakim jest konieczność należenia do zbioru wartości nieujemnych. Biorąc pod uwagę arbitralność przyjęcia tego ograniczenia można jednak z niego zrezygnować, bez istotnego uchybienia procesowi rozróżniania wag.

5.3 Badania wpływu przekształceń danych na wynik klasyfikacji

Na potrzeby determinacji, czy dwie kontrybucje należą do tej samej, czy innej osoby można, podobnie jak w przypadku ważenia istotności wskazówek, wykorzystać SVM o liniowym jądrze przekształcenia. Intuicyjnie wydaje się jednak, że wykorzystanie innego, bardziej elastycznego jądra przekształcenia, może dać lepsze rezultaty. W poprzedniej części założono, że dysponujemy zbiorem obiektów par, gdzie każdy atrybut obiektu jest wskazówką, tj. jego wartość zawiera się w przedziale [-1,1]. Aby tak się stało, należy dokonać pewnego przekształcenia z przestrzeni surowych cech (wartości całkowite, $x \geq -1$) do przedziału [-1,1].

W tym miejscu warto zadać dwa pytania:

1. jak przekształcić dane wejściowe
    a) jak potraktować wartości puste, odnotowywane w surowych cechach jako -1?
        i. jako wartość minimalną (połączyć z surową wartością 0)
        ii. jako wartość środkową ($\frac{1}{2}MAX$)
        iii. pozostawić wartością -1
    b) jak przeskalować dane do przedziału [-1,1]
        i. liniowo
        ii. przenieść wartości do zbioru -1,0,1
        iii. wykorzystać inne przekształcenie, np. sigmoidalne
2. jakie jądro przekształcenia jest najlepsze do celów rozróżniania autorstwa

Intuicyjnie wydaje się, że powyższe wybór przekształcenia może mieć wpływ na uzyskany rezultat, tj. odpowiednie zadane przekształcenie danych może premiować określone jądro przekształcenia (choćby w kwestii czasu budowy modelu).

W toku badań przebadano wymienione wyżej przekształcenia oraz następujące jądra przekształceń: liniowe, wielomianowe, radialne, sigmoidalne. Opisane są one następującymi równaniami:

- wielomianowe – $K(x_i, x_j) = (\gamma x_i^T x_j + c)^d$
- radialne – $K(x_i, x_j) = e^{-\gamma|x_i - x_j|^2}$
- sigmoidalne – $tanh(\gamma x_i^T x_j + c)$

gdzie wymienione zmienne oznaczają:

- d – stopień wielomianu
- γ – współczynnik gamma
- c – wyraz wolny

Eksperyment polegał na porównaniu wszystkich kombinacji z zadanych przedziałów powyższych parametrów, jąder przekształceń SVM oraz przekształceń danych pod kątem:

1. czasu budowy
2. maksymalnej dokładności (w procesie pięciostopniowe walidacji krzyżowej)
3. średniej dokładności
4. minimalnej dokładności

Wyniki eksperymentu przedstawiono w rodziale 6. Nie wymienionym powyżej parametrem jest koszt pomyłki za złą klasyfikację obiektu, jaki uwzględnia się budując model.

Intuicyjnie wydaje się, że powyższe wybór przekształcenia może mieć wpływ na uzyskany rezultat, tj. odpowiednie zadane przekształcenie danych może premiować określone jądro przekształcenia (choćby w kwestii czasu budowy modelu).

5.4 Dobór używanych baz danych

W niniejszej pracy metadane dokumentów są składowane w postaci trójek opisanych w rozdziale 3.3. Sprawdzono możliwość wykorzystania baz Sesame (rozdział 3.3.1) oraz bigdata® (rozdział 3.3.2). Wykorzystanie pierwszej z nich było istotne ze względu na małe wymagania sprzętowe (brak osobnych wątków zarządzających komunikacją z bazą danych) oraz szybkość składowania danych. Przewagą drugiej z baz jest możliwość przechowania ok.140 razy większej ilości trójek. Wadą zaś są duże narzuty czasowe na uzyskiwanie połączeń, nawet korzystając ze specjalnie przygotowanego typu połączeń tylko do odczytu.

W tej pracy sprawdzono również możliwość łączenia obydwu rozwiązań, tj. wykorzystania bazy bigdata® jako repozytorium głównego, a bazy Sesame jako wykorzystywanej lokalnie do składowania danych powiązanych z konkretnym zbiorem roboczym. Tak ustalona struktura pozwoliła na uzyskanie sporej poprawy czasu obliczeń. W rozdziale 6 podaje się szczegółowe wyliczenia na ten temat.

Rozdział 6

Wyniki

6.1 Ważenie wskazówek

Jak opisano w rozdziale 5.2., jednym z istotnych aspektów rozróżniania autorstwa jest pozyskanie jak największej ilości informacji odnośnie dokumentów, które należy przypisać autorowi. Kolejnym krokiem jest postawienie sobie pytania - które dane mają największe znaczenie. Intuicja podpowiada, że dla potwierdzenie zgodności autorstwa ważniejsze od tego samego tytułu czasopisma jest to jaki są tematy prac, czy też ich słowa kluczowe. Dużą wartością dodaną byłoby zamienienie intuicji na pewną miarę istotności.

W tym rozdziale opisana jest miara istotności danego atrybutu wytworzona w oparciu o SVM z liniowym jądrem przekształcenia. Dzięki przyjęciu liniowej i otrzymanemu wzorowi na hiperpłaszczyznę, otrzymujemy w przejrzysty sposób wagi poszczególnych atrybutów.

6.1.1 Opis danych

Dane użyte w tym eksperymencie stworzono przez połączenie zbiorów EuDML oraz Zentralblatt MATH. Na ich podstawie stworzono obiekty par opisane przez atrybuty. Poniżej podane są ich skróty z opisem cechy oraz odnośnikiem do wykresów opisujących zależność wartości atrybutu do ilości wskazań "ten sam autor"/"różni autorzy".

1. CS - Liczba wspólnych współautorów. Wartość ta powstaje przez zliczenie przystających do siebie nazwisk, po usunięciu z nich znaków diakrytycznych oraz zamiany liter na małe. (Rysunek 6.3.)

2. EM - Liczba wspólnych adresów e-mail. Do części kontrybutorów przypisane były dwa adresy e-mail. (Rysunek 6.5.)

3. EL - Liczba wspólnych prefiksów adresów e-mail. Dla adresu "j.kowalski@elka.pw.edu.pl" prefiksem jest część "j.kowalski". (Rysunek 6.4.)

4. CC - Liczba wspólnych kodów klasyfikacyjnych. Kody te w omawianym zbiorze należą do grupy kodów MSC. Porównaniu podlegają całe kody, bez analizy hierarchii kategorii, tj. poszczególnych członów kodu. (Rysunek 6.1.)

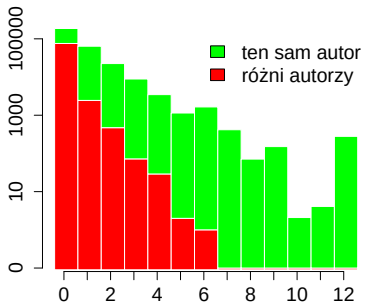
5. KP - Liczba wspólnych fraz kluczowych. Frazą kluczową jest np. "uczenie maszynowe". (Rysunek 6.6.)

6. KW - Liczba wspólnych słów kluczowych. Słowem kluczowym dla frazy "uczenie maszynowe" są "uczenie" oraz "maszynowe". (Rysunek 6.7.)

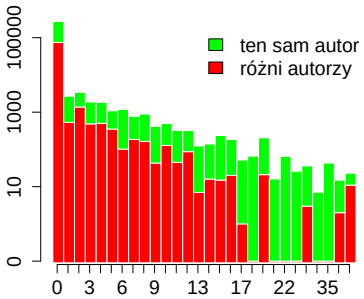
7. RF - Liczba wspólnych cytowań. W bazie EuDML dostępne są unikalne dla tej bazy identyfikatory cytowanych dokumentów. (Rysunek 6.8.)

8. YR - Różnica w roku publikacji prac przypisanych do kontrybutorów. (Rysunek 6.9.)

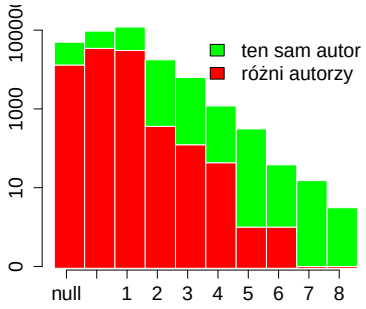
9. CI - Wystąpienie samocytowania. Zjawisko zachodzi jeżeli jedna z prac cytuje drugą z nich. (Rysunek 6.2.)



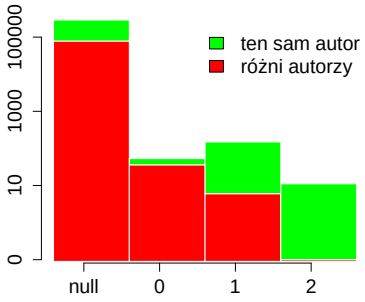
Rysunek 6.1: Liczba obiektów z wartością wskazówki CC, a klasyfikacja "ten sam autor"/"różni autorzy"



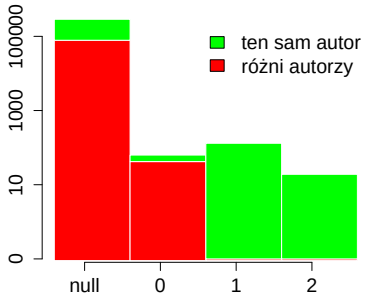
Rysunek 6.2: Liczba obiektów z wartością wskazówki CI, a klasyfikacja "ten sam autor"/"różni autorzy"



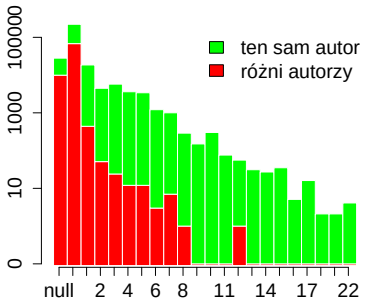
Rysunek 6.3: Liczba obiektów z wartością wskazówki CS, a klasyfikacja "ten sam autor"/"różni autorzy"



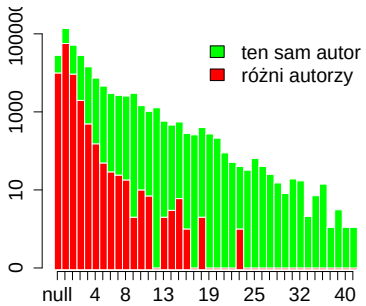
Rysunek 6.4: Liczba obiektów z wartością wskazówki EL, a klasyfikacja "ten sam autor"/"różni autorzy"



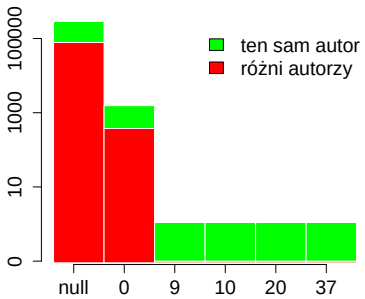
Rysunek 6.5: Liczba obiektów z wartością wskazówki EM, a klasyfikacja "ten sam autor"/"różni autorzy"



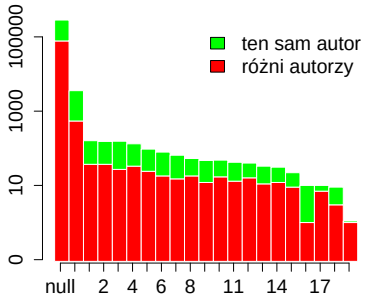
Rysunek 6.6: Liczba obiektów z wartością wskazówki KP, a klasyfikacja "ten sam autor"/"różni autorzy"



Rysunek 6.7: Liczba obiektów z wartością wskazówki KW, a klasyfikacja "ten sam autor"/"różni autorzy"



Rysunek 6.8: Liczba obiektów z wartością wskazówki RF, a klasyfikacja "ten sam autor"/"różni autorzy"



Rysunek 6.9: Liczba obiektów z wartością wskazówki YR, a klasyfikacja "ten sam autor"/"różni autorzy"

Warto pochylić się nad wymienionymi wykresami, gdyż same z siebie niosą bardzo istotne informacje. Otóż biorąc pod uwagę, że metoda SVM nie przyjmuje wartości pustych, a w tym eksperymencie przyjęto, że dane null-owe nie będą brane pod uwagę, należy wybrać taki podzbiór cech, który będzie zawierał jak najwięcej obiektów. Przyglądając się rysunkom 6.4., 6.5., 6.8., 6.9. łatwo zauważyć, że opisane na nich cechy w przeważającej większości przyjmują wartość null. Jest to bardzo niekorzystna właściwość zmuszająca do wyłączenia tych cech z dalszej analizy. Dokładne rozróżnienie liczby obiektów przyjmujących wartości puste, bądź liczbowe widać w tabeli 6.1.

Cecha	Wartość nie pusta		Wartość pusta	
	Liczba wystąpień	Procent wystąpień	Liczba	Procent wystąpień
EM	213	0.07	288796	99.93
EL	213	0.07	288796	99.93
CS	239753	82.95	49256	17.04
CC	289009	100.00	0	0.00
KP	260879	90.26	28130	9.73
KW	260879	90.26	28130	9.73
RF	1556	0.53	287453	99.46
YR	4744	1.64	284265	98.36
CI	289009	100.00	0	0.00
IS	265861	91.99	23148	8.01

Tablica 6.1: Częstość wystąpień wartości null w surowych cechach.

Podsumowując w eksperymencie wykorzystane zostaną obiekty, których wartości dla cech CS, CC, KP, KW jednocześnie nie są wartościami pustymi. Ważeniu podlegać będą tylko wymienione cechy.

6.1.2 Wyniki

Do eksperymentu wybrano obiekty, których wartości cech CS, CC, KP, KW były różne niż null. Następnie przekształcono wartości cech do zbioru dwuwartościowego zbioru {-1,1}. Biorąc pod uwagę przewagę wystąpień wskazań "ten sam autor" losowo usunięto nadmiarowy zbiór obiektów z tą wartością.

Następnie przeprowadzono badanie polegające na wyznaczeniu hiperpłaszczyzny z liniowym jądrem przekształcenia. Inaczej niż w przypadku eksperymentu opisanego w rozdziale 6.2. wykonano trzystopniową walidację krzyżową. Polega ona na wyznaczeniu klasyfikatora (utożsamianego w tym przypadku z hiperpłaszczyzną) dla każdego z pięciu podzbiorów zadanego zbioru obiektów, a następnie ocenie jego skuteczności na zbiorze pozostałych – nie użytych do trenowania klasyfikatora – obiektów. Przy średniej dokładności klasyfikacji wynoszącej 79,46%, uzyskane wagi wskazówek przedstawione w tabeli 6.2.

Kod cechy	Przyznane wagi			
	1-stopień	2-stopień	3-stopień	Uśredniona wartość
CS	$3.41\dot{1}0^{-13}$	$-4.99\dot{1}0^{-05}$	$1.13\dot{1}0^{-13}$	$-1.66\dot{1}0^{-05}$
CC	0.99	0.99	0.99	0.99
KP	1.00	0.99	0.99	0.99
KW	$3.55\dot{1}0^{-14}$	$4.29\dot{1}0^{-05}$	$6.78\dot{1}0^{-05}$	$3.69\dot{1}0^{-05}$

Tablica 6.2: Wagi wskazówek

Na podstawie wyników można wywnioskować, że z puli wybranych cech najistotniejsze to dzielenie wspólnych kodów klasyfikacyjnych oraz dzielenie wspólnych fraz kluczowych.

Co zaskakujące, dzielenie wspólnych nazwisk autorów nie jest cechą bardzo istotną. Może to wynikać z faktu, że inaczej niż w przypadku cech KP i CC, liczba wskazań "ten sam autor" oraz "różni autorzy" jest zbliżona.

Inną istotną okolicznością jest fakt, że wskazane wagi sugerują łączenie kontrybutorów w tożsamości li tylko na podstawie fraz kluczowych i kodów klasyfikacyjnych. Zamykałoby to drogę do połączenia kontrybucji osoby publikującej w dwóch różnych polach zainteresowań o ile osoba taka nie używa tych samych metod (wymienionych we frazach kluczowych). Zakładając, że zainteresowanie autora może zmieniać się stopniowo, np. przejście w obrębie "uczenia maszynowego" z "rozróżniania autorstwa dokumentów" na "usuwanie duplikatów", celowym działaniem byłaby analiza hierarchii kodu klasyfikacyjnego. Za przykład mogą posłużyć kody MSC, dzielące się na trzy człony, gdzie każdy kolejny z nich służy do zawężenia tematu. Gdyby sprawdzać zbieżność nie tyle całego kodu, ale poszczególnych jego zagłębień umożliwiłoby to wzięcie pod uwagę stopniowych zmian tematu w obrębie dziedziny. Tak uzyskane wskazówki należałoby ponownie poddać procesowi ważenia.

6.2 Dobór przekształceń danych i jądra SVM

6.2.1 Analiza danych

Zanim rozważy się jak traktować wskazania puste, warto zadać sobie pytanie o celowość takiego działania, tj. o skalę problemu.

Ogólnie zbiór wejściowy liczy sobie 463837 przykładów wejściowych. Przynajmniej jedno wskazanie puste zawiera połowa z nich (229706 obiektów). Dokładne zestawienie liczby wierszy z określoną liczbą wartości pustych zawiera tabela 6.3. Biorąc pod uwagę, że tylko 50% danych nie zawiera wartości null przyjęcie odpowiedniej strategii postępowania nabiera dużego znaczenia. Każdy przykład opisany jest pięcioma

Dokładna liczba null-i	Liczba przykładów	Procentowa zawartość
0	234131	50%
1	158764	34%
2	67180	15%
3	3762	1%
4	0	0%

Tablica 6.3: Liczba wierszy zawierających wartość pustą

Nazwa atrybutu	Kod atrybutu	Wartości atrybutu
Liczba wspólnych nazwisk współautorów	CS	$\{null\} \cup [0, 6]$
Wspólne kody klasyfikacyjne	CC	$[0, 12]$
Wspólne frazy kluczowe	KP	$\{null\} \cup [0, 22]$
Różnica w latach publikacji	YR	$\{null\} \cup [0, 20]$

Tablica 6.4: Lista atrybutów wraz z kodami

atrybutami, w tym atrybutem decyzyjnym. Poszczególne z nich opisano w tabeli 6.4. Sprawdzając liczbę null-i w poszczególnych kolumnach widać, że jedyną kolumną nie zawierającą tego wskazania jest informacja o kodach klasyfikacyjnych (CC). Wynika to z obowiązkowej obecności tego kodu we wzbogacanej bibliotece bazowej. W kontekście dalszych porównań warto przypomnieć, że w zbiorze wejściowym 71% par ma pozytywne wskazanie "ten sam autor". Wartości puste wystąpiły dla

- YR – 42% obserwacji – w tym 73.15% ze wskazaniem "ten sam autor"
- CS – 13% obserwacji – w tym 72.16% ze wskazaniem "ten sam autor"
- KP – 9.2% obserwacji – w tym 65.43% ze wskazaniem "ten sam autor"

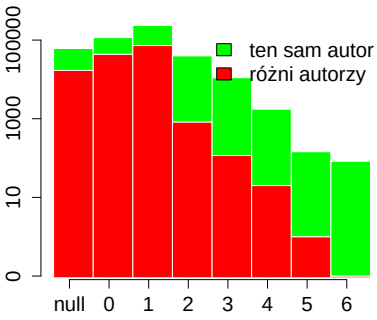
Brak informacji o wspólnych frazach (KP) o ok. 7% częściej niż w innych przypadkach (YR,CS) tyczy się wskazań "różni autorzy". Pełne zestawienie wartości jakie przyjmują

wskazówki, porównane z liczbą obserwacji null-owych widać na rysunkach 6.10., 6.11., 6.12., 6.13.

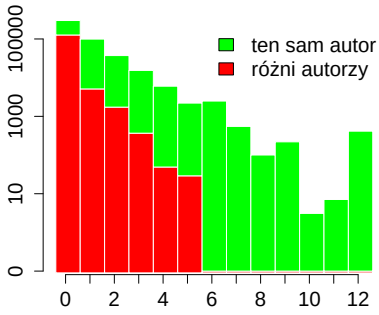
6.2.2 Wpływ przekształceń danych na ich rozróżnialność

Dzięki zabiegowi zastosowanemu w części 5.1 otrzymano obiekty par opisane surowymi cechami oraz atrybut decyzyjny {"ten sam autor","różni autorzy"}. W części 5.3 zaproponowano następujące przekształcenia wartości null reprezentowanej przez -1:

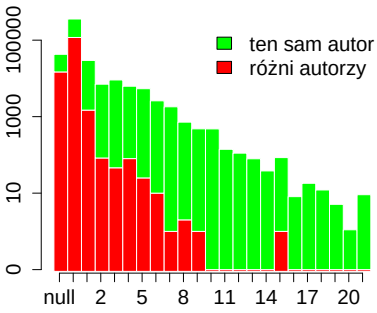
1. brak zmian
2. zamiana obserwacji z wartością -1 na wartość 0 i wartości 0 na -1 (Określane dalej jako rodzina Bool. Wartość -1 przenosi informację "fałsz"/"nie jest tym samym autorem", wartość 1 przenosi informację "prawda"/"jest tym samym autorem", zaś 0 "nie wiadomo")
3. przypisanie wartości null wartości 0 (Określane dalej jako rodzina Bay. Wartość 0 przenosi informację "cecha nie wystąpiła", wartość 1 przenosi informację "cecha wystąpiła")



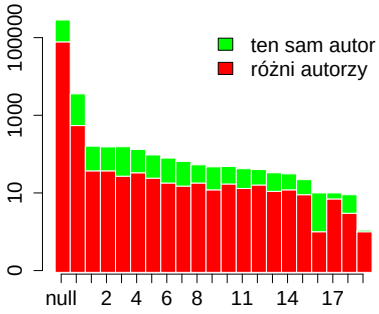
Rysunek 6.10: Liczba obserwacji (nie) będących tą samą osobą, a wartość wskazówki CS



Rysunek 6.11: Liczba obserwacji (nie) będących tą samą osobą, a wartość wskazówki CC



Rysunek 6.12: Liczba obserwacji (nie) będących tą samą osobą, a wartość wskazówki KP



Rysunek 6.13: Liczba obserwacji (nie) będących tą samą osobą, a wartość wskazówki YR

Zaproponowano również następujące przeskalowanie danych:

1. brak zmian
2. przekształcenie do zbioru {-1,0,1}, bądź w przypadku braku wartości -1 do zbioru {0,1}
3. przeskalowanie do przedziału [-1,1] (odpowiednio [0,1]
4. przekształcenie sigmoidalne, tj. zwracające wartości z przedziału [0,1]

Znane z literatury jest pojęcie skalowania wielowymiarowego (ang. Multidimensional Scaling, MDS, [44]). Pozwala ono dzięki przekształceniom matematycznym przejść z przestrzeni wielowymiarowej do przestrzeni trójwymiarowej, gdzie dwa wymiary są odwzorowane w przestrzeni euklidesowej, zaś trzeci wymiar, odzwierciedlający kategorię, opisany jest kolorem. Powyższe przekształcenia dają wymienione niżej kombinacje wynikowe. Do każdej z nich załączono wykresem przekształcenia oraz wizualizację algorytmu MDS danych wykorzystywanych dalej w eksperymencie. Ze względu na poglądowy charakter tego drugiego, pominięto na nim wartości osi X oraz Y.

1. Brak zmian (Rysunek 6.14)
2. Brak zmian w wartości null i zero, rodzina Bool, przekształcenie do zbioru wartości (Rysunki 6.15., 6.16.)
3. Brak zmian w wartości null i zero, rodzina Bool, przekształcenie liniowe (Rysunki 6.17., 6.18.)
4. Brak zmian w wartości null i zero, rodzina Bool, przekształcenie sigmoidalne (Rysunki 6.19., 6.20.)
5. Zmiana w wartości null i zero, rodzina Bool, przekształcenie do zbioru wartości (Rysunki 6.21., 6.22.)

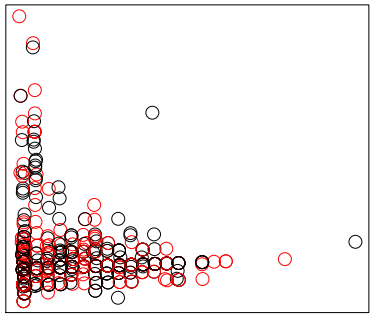
6. Zmiana w wartości null i zero, rodzina Bool, przekształcenie liniowe (Rysunki 6.23., 6.24.)

7. Zmiana w wartości null i zero, rodzina Bool, przekształcenie sigmoidalne (Rysunki 6.25., 6.26.)

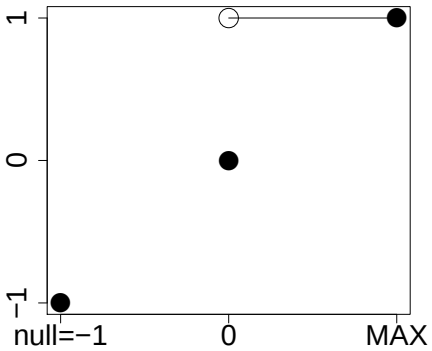
8. (Zamiana wartości null z zerem bez znaczenia) rodzina Bay, przekształcenie do zbioru wartości (Rysunki 6.27., 6.28.)

9. (Zamiana wartości null z zerem bez znaczenia) przekształcenie liniowe (Rysunki 6.29., 6.30.)

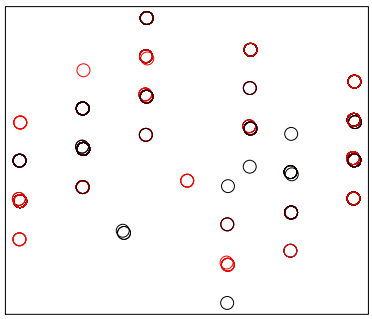
10. (Zamiana wartości null z zerem bez znaczenia) przekształcenie sigmoidalne (Rysunki 6.31., 6.32.)



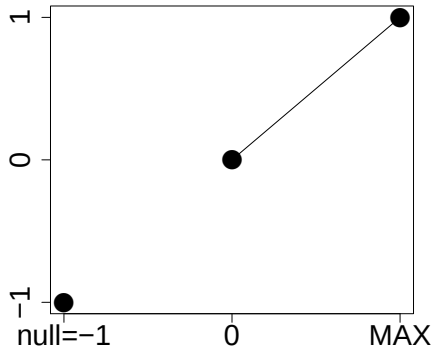
Rysunek 6.14: Brak przekształcenia



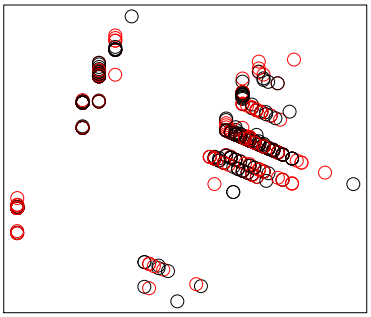
Rysunek 6.15: Brak zamiany wartości null x=-1 z wartością zero x=0, rodzina Bool, rzutowanie do zbioru {-1,0,1}



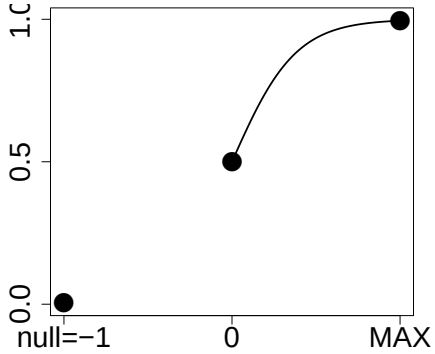
Rysunek 6.16: Brak zamiany wartości null x=-1 z wartością zero x=0, rodzina Bool, rzutowanie do zbioru {-1,0,1}



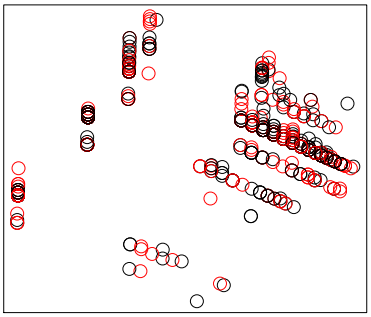
Rysunek 6.17: Brak zamiany wartości null x=-1 z wartością zero x=0, rodzina Bool, skalowanie liniowe



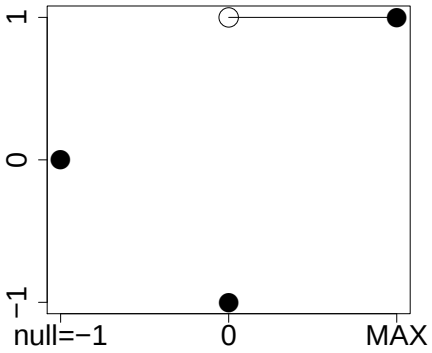
Rysunek 6.18: Brak zamiany wartości null x=-1 z wartością zero x=0, rodzina Bool, skalowanie liniowe



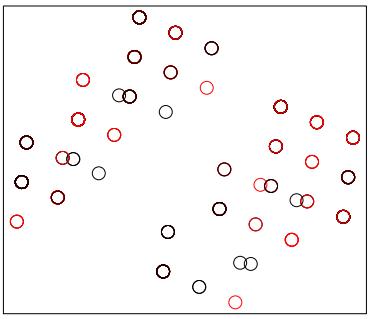
Rysunek 6.19: Brak zamiany wartości null x=-1 z wartością zero x=0, rodzina Bool, skalowanie sigmoidalne



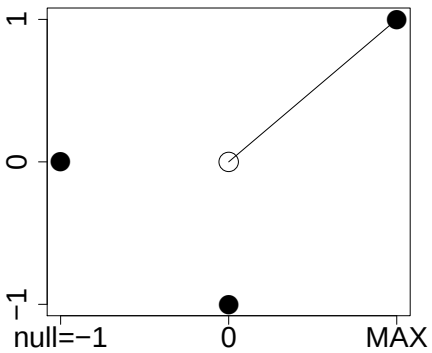
Rysunek 6.20: Brak zamiany wartości null x=-1 z wartością zero x=0, rodzina Bool, skalowanie sigmoidalne



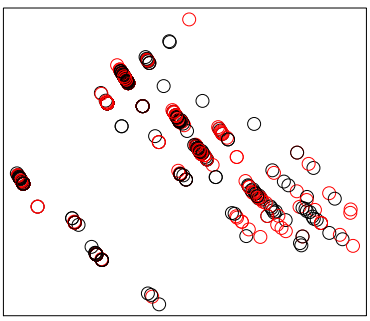
Rysunek 6.21: Zamiana wartości null x=-1 z wartością zero x=0, rodzina Bool, rzutowanie do zbioru {-1,0,1}



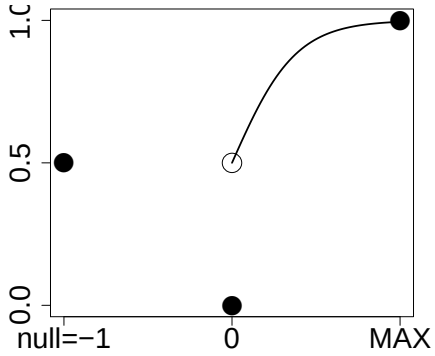
Rysunek 6.22: rodzina Bay, rzutowanie do zbioru {0,1}



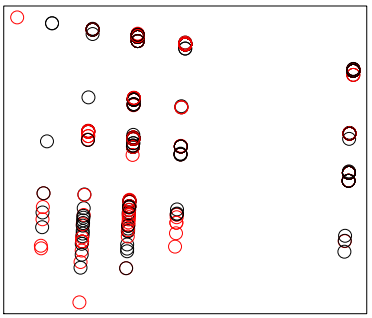
Rysunek 6.23: Zamiana wartości null x=-1 z wartością zero x=0, rodzina Bool, skalowanie liniowe



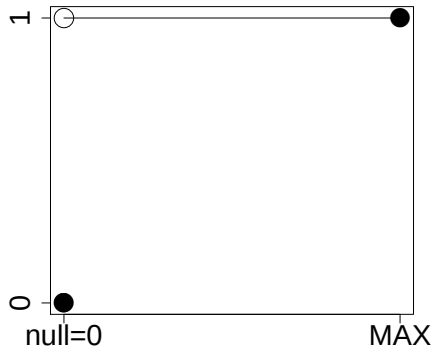
Rysunek 6.24: rodzina Bay, skalowanie liniowe



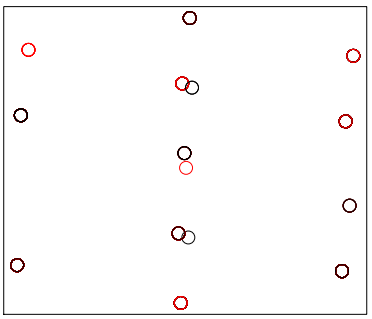
Rysunek 6.25: Zamiana wartości null x=-1 z wartością zero x=0, rodzina Bool, skalowanie sigmoidalne



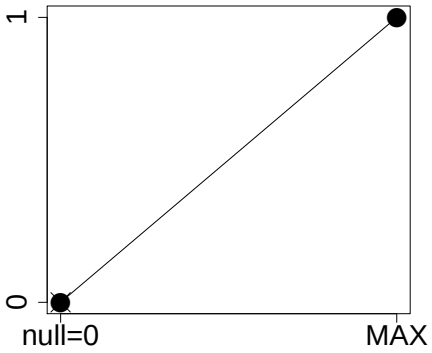
Rysunek 6.26: rodzina Bay, skalowanie sigmoidalne



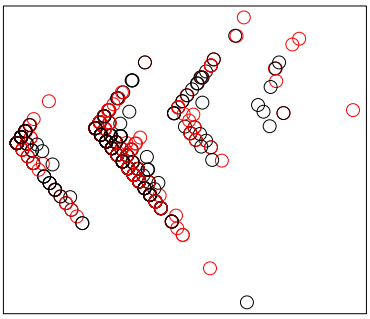
Rysunek 6.27: rodzina Bay, rzutowanie do zbioru {0,1}



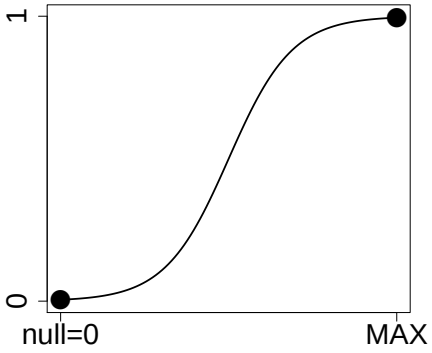
Rysunek 6.28: Zamiana wartości null x=-1 z wartością zero x=0, rodzina Bool, rzutowanie do zbioru {-1,0,1}



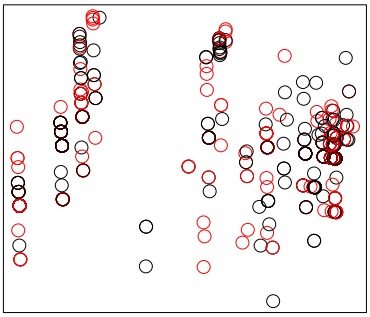
Rysunek 6.29: rodzina Bay, skalowanie liniowe



Rysunek 6.30: Zamiana wartości null x=-1 z wartością zero x=0, rodzina Bool, skalowanie liniowe



Rysunek 6.31: rodzina Bay, skalowanie sigmoidalne



Rysunek 6.32: Zamiana wartości null x=-1 z wartością zero x=0, rodzina Bool, skalowanie sigmoidalne

6.2.3 Wykonanie zadania

Eksperyment polegał na iteracyjnym sprawdzeniu wyników dla parametrów (tj. przekształcenia danych, jądra przekształcenia SVM, stopnia wielomiany, współczynnika gamma, wyrazu wolnego i kosztu złej klasyfikacji) z zadanych przedziałów jakie te parametry mogły przyjmować. Dla jądra liniowego współczynnik gamma i stopień wielomianu nie mają zastosowania.

Innym, przyjętym na sztywno parametrem jest część zbioru wejściowego, która ma zostać użyta do weryfikacji modelu. Do budowy każdego modelu brano ten sam podzbiór obserwacji liczący tysiąc egzemplarzy. Warto zwrócić tu uwagę, że model jest budowany na podstawie jednej piątej zadanego ułamka (zadanego podzbioru), biorąc pod uwagę pięciostopniową walidację krzyżową.

6.2.4 Rezultaty

Rezultaty składają się na 6600 obserwacji wyjściowych. Każdą opisują parametry wejściowe oraz minimalna, średnia i maksymalna dokładność (określana też mianem poprawności) modelów zbudowanych w trakcie walidacji krzyżowej. Ostatnim elementem opisu jest średni czas budowy modelu. Wyniki podzielono na następujące kategorie:

- Najlepsze
  - 10 minimalnych poprawności (Tabela 6.7.)
  - 10 średnich poprawności (Tabela 6.8.)
  - 10 maksymalnych poprawności (Tabela 6.9.)
  - 30 uśrednionych czasów budowy modelu (Tabela 6.10.)

- Najgorsze
  - 10 minimalnych poprawności (Tabela 6.11.)
  - 10 średnich poprawności (Tabela 6.12.)
  - 10 maksymalnych poprawności (Tabela 6.13.)
  - 30 uśrednionych czasów budowy modelu (Tabela 6.14.)

Przy tym kody przypisane przekształceniom danych oraz jąder przekształceń SVM obecne są odpowiednio w tabelach 6.5 i 6.6.

6.2.5 Omówienie wyników

Model przekształcenia danych

Analizując tabele wynikowe 6.3.-6.10. łatwo zauważyć, że najlepszym przekształceniem w świetle dokładności okazały się przekształcenia z rodziny Bay: liniowe oraz do zbioru. Tylko w kwestii najlepszych czasów wykonania wszystkie obserwacje wypadły w bardzo podobny sposób, tj. każde przekształcenie zostało wymienione. Wynika z tego, że najlepszymi przekształceniami są przekształcenie Bay liniowe oraz do zbioru. Jest to jednocześnie odpowiedzią na pytanie o najlepszą strategię traktowania nulli.

Jądro przekształcenia

Za najbardziej zobiektywizowaną miarę oceny modelu można uznać jego średnią dokładność. W kategorii najlepszych uśrednionych dokładności bezkonkurencyjne okazało się radialne jądro przekształcenia (RBF). Drugie najlepsze jądro przekształcenia – liniowe – uzyskało wynik o około 0,95% gorszy. Liczne wystąpienia jądra RBF w tabeli 6.8. świadczą na niekorzyść pozostałych jąder nieliniowych (wariantów liniowych z racji nieistotności współczynników gamma, stopnia wielomianu i wyrazu wolnego było wielokrotnie mniej). Dla odmiany do zestawień minimalnych dostało się również jądro wielomianowe.

Podsumowując – na zadanym zbiorze najlepiej zachowywało się jądro RBF przewyższając jądro liniowe o 0,95%.

Stopień wielomianu

Najlepszymi stopniami wielomianu okazały się wartości 1 i 3. Na liście najgorszych wyników znajdują się głównie stopnie większe niż 3. Często też pojawia się współczynnik 1, co sugeruje, że lepszym wyborem jest wartość 3.

Gamma

Najlepszą wartością współczynnika gamma jest 0.25. Drugą propozycją jest wartość 0.5. Inne wartości (np. ósme części jedynki) wypadły bardzo źle i tylko spowalniały budowę modelu.

Wyraz wolny

Najczęściej w zestawieniu najlepszych wartości występowały wartości 0.5 oraz 1. W zestawieniach najgorszych parametrów najczęściej widać 0 i 1. Stąd sugestia, że najlepszą wartością jest wartość 0.5.

Koszt pomyłki

W kategorii najniższych czasów najczęściej występującym kosztem jest koszt jednostkowy. Stanowi on większość w zestawieniach najlepszych wyników podanych w tabeli. Jest to zarazem domyślny koszt jaki jest przyjmowany w procesie budowy. Warto nadmienić, że poza zestawieniem znajdują się obserwacje dla kosztu 1000. Koszt taki sprawiał, że model budował się 100-1000 razy dłużej.

Podsumowanie

Najlepsze parametry wejściowe do budowy modelu to

1. **przekształcenia** z rodziny **Bay**, **liniowe** i **do zbioru**
2. **jądra przekształceń radialne** i (o 0,95% gorsze) **liniowe**
3. **stopień wielomianu** równy **3**
4. **gamma** równa **0.25**
5. **wyraz wolny** równy **0.5**
6. **koszt pomyłki** równy **1**

Zgodnie z oczekiwaniami opisane powyżej modele (liniowy i nieliniowy) znajdują się na szczycie listy najlepszych uśrednionych dokładności.

Architektura rozwiązania W eksperymentalnej części projektu wykorzystano kod napisany w języku R [51] oraz bibliotekę LIBSVM [20].

kod	opis
1	połączenie wartości null i 0, liniowe przeskalowanie danych do przedziału [0,1]
2	połączenie wartości null i 0, sigmoidalne przeskalowanie danych do przedziału [0,1]
3	połączenie wartości null i 0, przeskalowanie danych do zbioru {0,1}
4	null → -1, liniowe przeskalowanie danych do przedziału [-1,1]
5	null → -1, sigmoidalne przeskalowanie danych do przedziału [0,1]
6	null → -1, przeskalowanie danych do zbioru {-1,0,1}
7	zamiana wartości 0 → -1 i null → 0, liniowe przeskalowanie danych do przedziału [-1,1]
8	zamiana wartości 0 → -1 i null → 0, sigmoidalne przeskalowanie danych do przedziału [-1,1]
9	zamiana wartości 0 → -1 i null → 0, przeskalowanie danych do zbioru {-1,0,1}
10	brak zmian w danych

Tablica 6.5: Kody opisujące przekształcenia danych

kod	opis
Line	Jądro liniowe
Poly	Jądro wielomianowe
RBF	Jądro radialne/gaussowskie
Sigm	Jądro sigmoidane

Tablica 6.6: Kody opisujące jądra przekształceń SVM

przekształcenie	jądro	koszt	d	γ	c	min. popr. [%]	śr. popr. [%]	max popr. [%]	czas [s]
1	RBF	1	1	0.50	0.5	75.34884	75.86113	76.57005	0.099
3	RBF	10	3	0.25	0.5	74.93976	75.86197	76.42487	0.100
1	RBF	1	1	0.50	0.0	74.88479	75.24784	76.35468	0.099
1	RBF	1	3	0.25	0.5	74.81297	76.65604	78.49741	0.096
3	Poly	1	3	0.25	0.0	74.67700	75.59555	76.21483	0.065
1	RBF	1	3	0.25	1.0	74.61929	76.11157	77.41935	0.096
3	Line	100	NA	NA	NA	74.61140	75.44232	77.03704	0.065
3	Line	10	NA	NA	NA	74.50000	75.69206	76.55502	0.063
1	Line	100	NA	NA	NA	74.43038	75.42094	77.00258	0.069
1	Line	1	NA	NA	NA	74.32763	75.40376	76.67494	0.073

Tablica 6.7: Najlepsze **minimalne** poprawności

przekształcenie	jądro	koszt	d	γ	c	śr. popr. [%]	min. popr. [%]	max popr. [%]	czas [s]
1	RBF	1	3	0.25	0.5	76.65604	74.81297	78.49741	0.096
1	RBF	1	3	0.25	1.0	76.11157	74.61929	77.41935	0.096
3	RBF	10	3	0.25	0.5	75.86197	74.93976	76.42487	0.100
1	RBF	1	1	0.50	0.5	75.86113	75.34884	76.57005	0.099
1	RBF	1	1	0.50	1.0	75.82178	73.84259	76.81159	0.106
3	RBF	100	3	0.25	0.5	75.81767	73.73494	77.23971	0.091
3	RBF	100	3	0.25	1.0	75.78316	73.90244	78.46154	0.101
3	Line	1	NA	NA	NA	75.75639	73.86935	77.69231	0.072
3	Line	10	NA	NA	NA	75.69206	74.50000	76.55502	0.063
1	RBF	1	3	0.25	0.0	75.66602	71.46341	79.25000	0.105

Tablica 6.8: Najlepsze **średnie** poprawności

przekształcenie	jądro	koszt	d	γ	c	max. popr. [%]	min. popr. [%]	śr popr. [%]	czas [s]
1	RBF	1	3	0.25	0.0	79.25000	71.46341	75.66602	0.105
3	RBF	1	3	0.25	0.5	79.02813	69.77330	74.50854	0.092
1	Line	1	NA	NA	NA	78.55422	70.04608	73.59408	0.074
1	RBF	1	3	0.25	0.5	78.49741	74.81297	76.65604	0.096
3	RBF	100	3	0.25	1.0	78.46154	73.90244	75.78316	0.101
3	Sigm	1	3	0.25	0.5	78.08564	71.46341	74.01009	0.089
1	RBF	10	3	0.25	1.0	78.05486	71.21588	75.61372	0.097
3	Poly	10	3	0.25	0.0	77.91563	73.90244	75.04604	0.077
3	RBF	1	3	0.25	1.0	77.91262	73.79135	75.32863	0.098
1	Poly	1	1	0.50	1.0	77.85548	72.62181	74.91555	0.068

Tablica 6.9: Najlepsze **najwyższe** poprawności

przekształcenie	jądro	koszt	d	γ	c	czas [s]	min. popr. [%]	śr. popr. [%]	max popr. [%]
6	Line	1	NA	NA	NA	0.063	73.86935	74.79330	75.60386
3	Line	10	NA	NA	NA	0.063	74.50000	75.69206	76.55502
10	Line	1	NA	NA	NA	0.064	70.70218	71.98310	73.12661
6	Line	1	NA	NA	NA	0.064	73.75887	74.57914	75.60386
6	Line	10	NA	NA	NA	0.064	74.18605	74.71770	75.60386
4	Line	1	NA	NA	NA	0.064	72.55370	74.72320	76.58537
3	Line	1	NA	NA	NA	0.064	71.80723	73.06104	75.00000
3	Line	10	NA	NA	NA	0.064	71.46172	72.94290	74.75248
3	Line	10	NA	NA	NA	0.064	70.23499	71.84789	73.54497
9	Line	1	NA	NA	NA	0.064	64.44444	70.42610	72.72727
4	Line	1	NA	NA	NA	0.064	74.16880	74.79354	76.13065
9	Line	10	NA	NA	NA	0.064	72.75000	74.20638	75.13228
3	Poly	10	1	0.25	0.5	0.064	71.53110	72.87530	75.00000
4	Line	1	NA	NA	NA	0.064	64.69136	69.04842	71.46402
3	Poly	100	1	0.25	0.5	0.065	60.55980	67.78753	72.26277
9	Poly	10	1	0.25	0.0	0.065	70.41565	71.36836	72.46377
7	Poly	1	1	0.25	0.5	0.065	55.44554	67.80821	71.21212
6	Poly	1	1	0.50	0.5	0.065	73.33333	74.33871	76.22549
5	Line	1	NA	NA	NA	0.065	74.12587	74.72204	75.17900
3	Line	1	NA	NA	NA	0.065	70.85308	71.69793	73.05825
6	Line	1	NA	NA	NA	0.065	68.06283	70.25295	72.75000
4	Line	1	NA	NA	NA	0.065	66.93122	69.81270	72.00000
5	Line	10	NA	NA	NA	0.065	68.60759	70.99055	72.30769
3	Poly	1	5	0.25	0.5	0.065	64.81013	69.54144	73.50649
2	Line	1	NA	NA	NA	0.065	67.77494	70.77758	72.43108
8	Line	1	NA	NA	NA	0.065	69.15167	71.29659	73.58974
3	Poly	1	3	0.25	0.0	0.065	74.67700	75.59555	76.21483
3	Poly	1	3	0.25	1.0	0.065	69.92665	73.27524	75.90674
3	Poly	10	3	0.25	0.5	0.065	73.28431	75.66412	77.63819
3	Line	100	NA	NA	NA	0.065	74.61140	75.44232	77.03704

Tablica 6.10: Najlepsze **czasy**

przekształcenie	jądro	koszt	d	γ	c	min. popr. [%]	śr. popr. [%]	max popr. [%]	czas [s]
8	Sigm	1	1	0.875	1.0	42.04276	56.62591	65.72770	0.095
3	Sigm	1	1	0.875	1.0	43.17073	57.07457	64.06619	0.094
9	Sigm	100	1	0.500	1.0	43.63208	55.20168	65.78313	0.089
4	Sigm	10	7	0.500	1.0	43.80734	51.71852	56.52174	0.100
10	Sigm	1	5	0.250	1.0	43.84615	60.14019	68.51385	0.100
4	Sigm	10	5	0.250	0.5	44.32990	62.48444	71.42857	0.084
6	Sigm	100	4	0.250	1.0	44.97608	59.20266	64.66513	0.106
10	Sigm	10	5	0.250	1.0	44.98715	55.07894	60.44776	0.085
3	Sigm	10	1	0.500	1.0	45.51724	62.81768	71.05882	0.094
6	Sigm	10	7	0.250	1.0	45.54455	55.89822	65.43210	0.094

Tablica 6.11: Najgorsze **minimalne** poprawności

przekształcenie	jądro	koszt	d	γ	c	śr. popr. [%]	min. popr. [%]	max popr. [%]	czas [s]
4	Poly	1	6	0.250	0	51.22590	45.82339	53.69928	0.086
4	Sigm	10	7	0.500	1	51.71852	43.80734	56.52174	0.100
10	Sigm	1	7	0.500	1	52.28215	50.00000	55.09709	0.099
5	Poly	1	6	0.250	0	52.36772	48.16626	57.64706	0.129
4	Poly	10	6	0.250	0	52.37740	50.00000	56.94118	0.075
4	Sigm	100	6	0.500	1	52.85482	48.69565	56.30841	0.094
10	Sigm	10	7	0.500	1	53.02502	49.88558	55.83524	0.103
1	Poly	1	4	0.125	0	53.16409	49.58333	54.70852	0.093
7	Poly	1	4	0.125	0	53.29417	49.78261	54.74614	0.082
4	Sigm	10	1	0.875	1	53.36487	49.76077	56.19048	0.089

Tablica 6.12: Najgorsze **średnie** poprawności

przekształcenie	jądro	koszt	d	γ	c	max popr. [%]	min. popr. [%]	śr popr. [%]	czas [s]
4	Poly	1	6	0.250	0	53.69928	45.82339	51.22590	0.086
1	Poly	1	4	0.125	0	54.70852	49.58333	53.16409	0.093
7	Poly	1	4	0.125	0	54.74614	49.78261	53.29417	0.082
10	Poly	1	6	0.250	0	54.90654	51.97044	53.67795	0.087
10	Sigm	1	7	0.500	1	55.09709	50.00000	52.28215	0.099
5	Poly	10	6	0.250	0	55.26316	53.03738	53.79080	0.081
8	Poly	1	4	0.125	0	55.60345	52.91577	54.11910	0.092
5	Poly	1	4	0.125	0	55.72034	51.28205	53.74753	0.092
10	Sigm	10	7	0.500	1	55.83524	49.88558	53.02502	0.103
6	Poly	1	4	0.125	0	55.98291	52.81385	54.78047	0.084

Tablica 6.13: Najgorsze **najwyższe** poprawności

przekształcenie	jądro	koszt	d	γ	c	czas [s]	min. popr. [%]	śr. popr. [%]	max popr. [%]
1	Poly	100	7	0.50	1.0	28.762	59.56938	66.25324	70.68558
10	Poly	100	6	0.50	1.0	11.322	55.50459	64.35074	70.86247
10	Poly	100	7	0.50	1.0	7.521	61.41176	66.26467	68.06527
10	Poly	100	6	0.50	0.0	6.046	58.49057	62.30940	65.03497
10	Poly	100	5	0.50	1.0	5.655	61.79775	65.40958	70.74830
5	Poly	100	7	0.50	1.0	5.383	64.25000	67.25495	68.46847
1	Poly	100	8	0.25	1.0	4.303	67.77778	71.05254	74.88584
4	Poly	100	9	0.25	1.0	3.130	56.50224	63.38009	69.51501
10	Poly	10	7	0.50	0.5	3.027	61.81384	65.53591	69.60784
4	Poly	100	7	0.50	1.0	2.963	63.21839	65.77568	68.76513
10	Poly	10	6	0.50	1.0	2.640	60.74766	62.86437	66.66667
5	Poly	100	7	0.50	0.5	2.300	66.35294	67.48079	69.64706
4	Poly	100	6	0.50	1.0	2.286	59.68109	63.63474	65.70156
1	Poly	100	6	0.50	0.0	2.136	47.12389	59.91604	64.43914
1	Poly	100	7	0.50	0.0	1.952	48.29268	61.90473	69.57547
7	Poly	100	7	0.50	1.0	1.942	63.17016	67.28541	70.76167
1	Poly	10	7	0.50	0.0	1.808	59.38242	65.33961	67.84870
4	Poly	10	7	0.50	1.0	1.388	64.06619	67.07271	72.18391
4	Poly	100	5	0.50	1.0	1.383	63.63636	66.59921	68.76457
10	Poly	100	7	0.50	0.0	1.354	55.50351	62.74097	66.50718
5	Poly	10	7	0.50	1.0	1.169	58.25472	64.06756	69.41176
10	Poly	100	7	0.25	1.0	1.114	60.96386	65.18970	71.07232
7	Poly	100	9	0.25	1.0	1.101	59.45330	64.94130	67.49436
4	Poly	100	7	0.50	0.5	1.084	64.57831	67.30552	70.48055
8	Poly	100	9	0.25	1.0	1.049	60.50228	64.43468	67.44731
1	Poly	100	9	0.25	1.0	1.034	63.53468	67.57583	72.33010
5	Poly	100	5	0.50	0.5	1.034	61.43791	64.83179	69.74596
5	Poly	100	6	0.50	1.0	1.001	63.38673	67.53941	71.39588
5	Poly	100	5	0.50	1.0	0.845	57.37327	63.70844	67.53813
1	Poly	100	6	0.25	0.5	0.690	60.04515	66.17953	69.61722

Tablica 6.14: Najgorsze **czasy**

6.3 Badanie możliwych rozwiązań architektonicznych

6.3.1 Wykonanie zadanie

W eksperymencie zbadano czas wykonywania rozróżniania autorstwa w przypadku wykorzystania jako repozytorium tylko bazy bigdata® porównany z użyciem tejże jako repozytorium główne wspomagane wykorzystaniem bazy Sesame.

6.3.2 Różnice między bigdata®, a Sesame

Nie wnikając w szczegóły implementacyjne baz bigdata® oraz Sesame warto zwrócić uwagę na ich dwie cechy, a mianowicie szybkość działania i sposób przechowywania. Bazę Sesame można umieścić w pamięci RAM, natomiast do korzystania z bazy bigdata® konieczne jest wykorzystanie dysku twardego. Wiąże się to z tą niedogodnością, że w przypadku wyliczania autorstwa dokumentów, podejmuje się wielokrotne łączenie z bazą danych. Jest rzeczą oczywistą, że korzystanie z pamięci jest rozwiązaniem szybszym niż odnoszenie się do dysku twardego. Nie można przy tym zapomnieć o przewadze bazy bigdata® nad bazą Sesame, polegającej na dużo większej pojemności tej pierwszej.

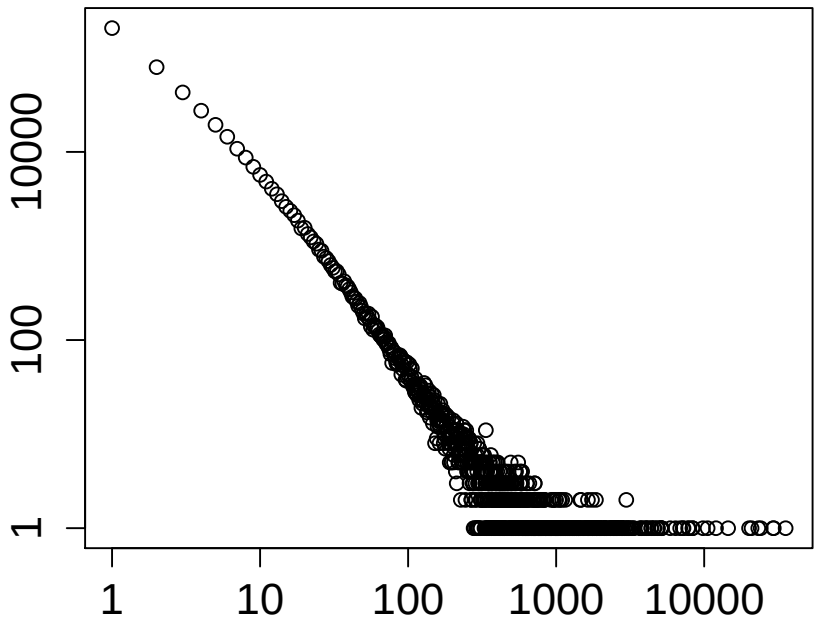
6.3.3 Idea repozytorium pomocniczego

Biorąc pod uwagę przewagę w czasie dostępu do bazy Sesame, warto jest wykorzystać ją jako repozytorium na potrzeby obliczeń poszczególnych zbiorów roboczych. Aby to zrobić należy ją zainicjalizaować, czyli przekazać wszystkie potrzebne do późniejszych obliczeń dane z podstawowego repozytorium bigdata®. W tym celu należy przenieść grafy wykorzystywane przez wskazówki, a powiązane z kontrybutorami należącymi do danego zbioru roboczego do bazy Sesame. Tak uzyskane repozytorium pomocnicze można przekazać do dalszych obliczeń.

6.3.4 Opis zbioru Springer

W eksperymencie do mierzenia wydajności poszczególnych rozwiązań wykorzystano zbiór biblioteki Springer liczącego blisko 500000 zbirów roboczych, dzielących ok. 3.8 miliona kontrybutorów, przy czym najliczniejsze są zbiory o rozmiarze 1 (207313 wystąpień), dalej zbiory o rozmiarze $2 \leq x \leq 7$ (liczące dziesiątki tysięcy wystąpień). Zbiory o rozmiarze $8 \leq x \leq 24$ mają tysiące wystąpień, zaś mniejsze niż 70 - setki. Większe zbiory pojawiają się nie częściej niż w kilkunastu wystąpieniach, przy czym zbiory większe niż 350 obiektów notują już tylko pojedyncze przypadki. Zależność tę odzwierciedla rysunek 6.33. Powiązanie rozmiaru zbioru od liczby wystąpień kontrybutorów i zbiorów roboczych w zbiorach mniejszych bądź równych zadanemu prezentuje rysunek . Tę samą zależność w perspektywie procentowej prezentuje rysunek 6.35. Z ostatniego rysunku można wysnuć następujące wnioski: zbiory robocze mniejsze lub równe sto stanowią ponad 99% całej ich populacji, ale znajduje się w

nich tylko 60% kontrybutorów. Nasuwa to przypuszczenie, że w stosunkowo tani sposób można przeliczyć większą część zbiorów roboczych. Można zadać sobie jednak pytanie nad istotnością takich wyliczeń, bo najpewniej ręczna weryfikacja zbioru kilku kontrybutorów nie stanowi dużego wyzwania dla człowieka. Rozróżnianie autorstwa jest najważniejsze właśnie dla zbiorów dużych (większych niż 50, czy 100 elementów), gdzie przy tym znajduje się 40% kontrybucji.



Rysunek 6.33: Rozmiar zbioru (oś X), a liczba wystąpień (oś Y).

6.3.5 Wyniki eksperymentalne

W trakcie wyliczania autorstwa dla zasobów biblioteki Springer z lipca 2010 roku pierwotnie wykorzystywano jedynie bazę bigdata®, jednak okazało się, że wyliczenie takie prowadzone sekwencyjnie na jednym rdzeniu trwałoby blisko 7 lat. Czas (w sekundach), który zajmuje przetworzenie pojedynczego zbioru w zależności od jego wielkości, przedstawia poniższe równanie

$$T_{bigdata}(x) = 0,0333722627 \cdot x^2$$

Dlatego wprowadzono wykorzystanie bazy Sesame jako repozytorium podręczne danego zbioru roboczego. Dzięki temu łączny czas obliczeń prowadzonych na jednej jednostce liczącej spadł do 51 dni, dając blisko 50 krotny zysk względem pierwotnego

rozwiązania. Zależność czasu wykonania dla rozwiązania drugiego opisuje wzór

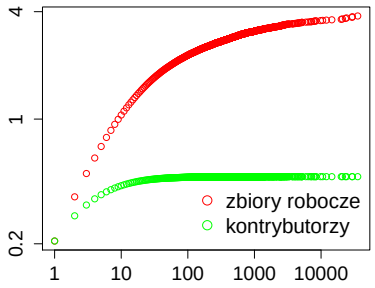
$$T_{bigdata \oplus Sesame}(x) = 0,0608099 \cdot x + 0,0005811 \cdot x^2$$

przy czym składnik $b \cdot x^2$ wynika z użycia klastrowania prostego (ang.single-linkage clustering) o złożoności $\mathcal{O}(N^2)$, zaś składnik $a \cdot x$ z konieczności zainicjalizowania repozytorium pomocniczego.

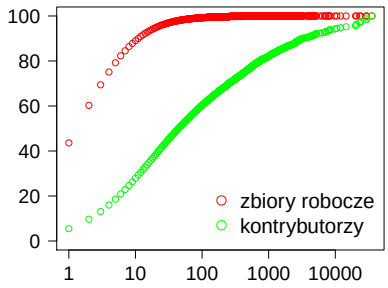
Wzory wyliczono na podstawie danych zaprezentowanych w 6.36, 6.37.

Warto pamiętać, że powyższe wzory, a w szczególności współczynniki a,b wynikają z wykorzystania danej jednostki obliczeniowej. Korzystając z innej jednostki należy przemnożyć uzyskane wzory przez iloraz szybkości nowej jednostki do szybkości wykorzystanej w eksperymencie. Cechą stałą jednak pozostanie iloraz współczynnika b dla zależności $T_{bigdata}$ oraz $T_{bigdata \oplus Sesame}$ wynoszący ok.57.

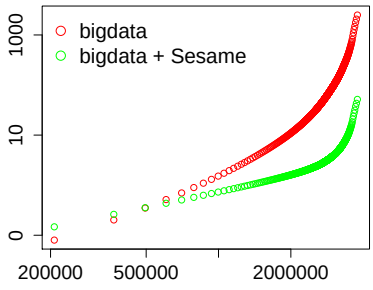
Należy dodać, że biorąc pod uwagę wielkość największego zbioru roboczego dla biblioteki Springer liczącego 35498 kontrybutorów i czas potrzebny na wyliczenia autorstwa tj. 8,5 dni, można skrócić faktyczny całkowity czas wykonania zadania do 8,5 dnia dodając więcej jednostek liczących. W wymienionym zbiorze osiągnie się ten cel korzystając z 6 jednostek obliczeniowych.



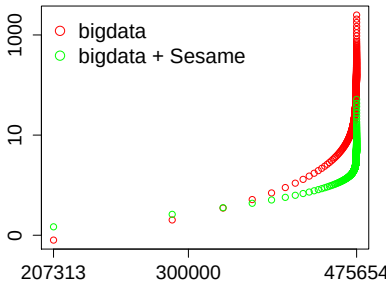
Rysunek 6.34: Maksymalny rozmiar zbioru roboczego (oś X), a ilość obecnych kontrybucji/zbiorów roboczych w zbiorach mniejszych od zadanego (oś Y) liczona w milionach



Rysunek 6.35: Maksymalny rozmiar zbioru roboczego (oś X), a procentowa obecnych kontrybucji/zbiorów roboczych w zbiorach mniejszych od zadanego



Rysunek 6.36: Liczba kontrybucji (oś X) przetworzona w czasie dni (oś Y)



Rysunek 6.37: Liczba zbiorów roboczych (oś X) przetworzona w czasie dni (oś Y)

Rozdział 7

Podsumowanie

Niniejsza praca jest kontynuacją prac podjętych w [17] i [27]. Opisuje wyzwania pojawiające się w wirtualnych bibliotekach, a w szczególności zagadnienie rozróżniania autorstwa prac. Wskazuje dotychczas wykorzystywane rozwiązania tego problemu.

Zaproponowano elastyczne narzędzie do rozróżniania autorstwa prac. Z uwagi na praktyczny wymiar omówionego narzędzia, szczególny nacisk położono na efektywność obliczeniową. Uzyskano rozwiązanie, które bez problemu może podjąć rywalizację z dostępnymi na rynku propozycjami.

Uzyskane narzędzie biblioteczne może służyć nie tylko do rozróżniania autorstwa prac, ale też do wiązaniu referencji z dokumentami oraz usuwaniu duplikatów w zbiorze, np. przy łączeniu zbiorów kilku bibliotek wirtualnych.

Przedstawiono również obiektywny sposób ważenia wskazówek z wykorzystaniem aparatu SVM. Uzyskano przy tym rezultat, który przy niskim nakładzie obliczeniowym, uzyskuje wyniki porównywalne z innymi podejściami (np. bazującymi na programowaniu genetycznym), bądź lepsze.

Porównano jądra przekształceń metody SVM względem dokładności i szybkości otrzymanej klasyfikacji. Skupiono się przy tym na sprawdzeniu wyników pod kątem parametrów wejściowych metody.

Ostatecznie wszystkie stawiane w pracy cele zostały osiągnięte.

Istnieje wiele kierunków dalszych badań, w których można wykorzystać wyniki niniejszej pracy. Usunięcie duplikatów prac i połączanie cytowań z dokumentami pozwala na wyszukiwanie prac najistotniejszych dla danego tematu. Jest to szczególnie ważne dla szybkiego wdrażania się w zagadnienie, aby z ogromnego zbioru pozycji wybrać najważniejsze. Powiązanie kontrybucji z tożsamościami pozwala na ocenę aktywności naukowej osób, bądź instytucji oraz poszukiwania autorów aktywnych w danej dziedzinie. Przede wszystkim daje to też narzędzie do łatwego zapoznania się ze wszystkimi dziełami danego autora, bez pracochłonnego procesu ręcznej selekcji.

Pojawia się też możliwość śledzenia w czasie aktywności danego autora.

Oczywiście część z przyjętych w pracy rozwiązań można udoskonalić. Warto zastanowić się nad możliwością zautomatyzowania doboru atrybutów obiektów (jeszcze przed ich ważeniem) i parametrów klasyfikatora, np. przez wykorzystanie metody symulowanego wyżarzania lasu.

Ciekawym kierunkiem wydaje się wykorzystanie innych niż SVM algorytmów, np. metody Analiza Głównych Składowych (ang. Principal Component Analysis, PCA), czy też Liniowej Analizy Dyskryminacyjnej (ang. Linear discriminant analysis, LDA)

W pracy przyjmowano, że obiekty par opisane są przez atrybuty reprezentujące wartości surowych wskazówek. Kolejnym ważnym przyrostem byłoby stworzenie metody automatycznej ekstrakcji wskazówek. Jest to istotne nie tylko ze względu na problem wydobywania tożsamości, ale też z punktu widzenia eksperta badającego zjawiska występujące na zadanym zbiorze danych. Uzyskaną w ten sposób wiedzę, może podlegać dalszej interpretacji wpływając na sposób organizacji instytucji naukowych.

Bibliografia

[1] Bigdata(r) Homepage. [online], grudzień 2011. [dostęp: 2011-12-20 20:00Z]. [cytowanie na str. 23]

[2] Comparison of Large Triple Stores. [online], grudzień 2011. [dostęp: 2011-12-20 20:00Z]. [cytowanie na str. 22, 23]

[3] Resource Description Framework (RDF). [online], grudzień 2011. [dostęp: 2011-12-20 20:00Z]. [cytowanie na str. 22]

[4] Sesame Documentation Webpage. [online], grudzień 2011. [dostęp: 2011-12-20 20:00Z]. [cytowanie na str. 22]

[5] SPARQL Query Language for RDF. [online], grudzień 2011. [dostęp: 2011-12-20 20:00Z]. [cytowanie na str. 22]

[6] The SeRQL query language (revision 1.2). [online], grudzień 2011. [dostęp: 2011-12-20 20:00Z]. [cytowanie na str. 22]

[7] Uniform Resource Identifiers (URI): Generic Syntax. [online], grudzień 2011. [dostęp: 2011-12-20 20:00Z]. [cytowanie na str. 22]

[8] Academic Researcher Social Network Search. [online], styczeń 2012. [dostęp: 2012-02-16 20:00Z]. [cytowanie na str. 10]

[9] Apache Hadoop Website. [online], styczeń 2012. [dostęp: 2012-02-16 20:00Z]. [cytowanie na str. 24]

[10] Apache License 2.0. [online], styczeń 2012. [dostęp: 2012-02-16 20:00Z]. [cytowanie na str. 24]

[11] GNU General Public License, version 2. [online], styczeń 2012. [dostęp: 2012-02-16 20:00Z]. [cytowanie na str. 23]

[12] Mathematics Subject Classification 2010. [online], styczeń 2012. [dostęp: 2012-02-16 20:00Z]. [cytowanie na str. 6]

[13] The DOI(r) System. [online], styczeń 2012. [dostęp: 2012-02-16 20:00Z]. [cytowanie na str. 8]

[14] Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 586–597. VLDB Endowment, 2002. [cytowanie na str. 8]

[15] Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001. [cytowanie na str. 11]

[16] Mikhail Bilenko, Raymond J. Mooney, William W. Cohen, Pradeep D. Ravikumar, and Stephen E. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003. [cytowanie na str. 8]

[17] Lukasz Bolikowski and Piotr Jan Dendek. Towards a Flexible Author Name Disambiguation Framework. In Petr Sojka and Thierry Bouche, editors, *Towards a Digital Mathematics Library*, pages 27–37. Masaryk University Press, 2011. [cytowanie na str. 4, 63]

[18] Kurt D. Bollacker, Steve Lawrence, and C. Lee Giles. Citeseer: an autonous web agent for automatic retrieval and identification of interesting publications. In *Proceedings of the second international conference on Autonomous agents*, AGENTS '98, pages 116–123, New York, NY, USA, 1998. ACM. [cytowanie na str. 10]

[19] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, pages 144–152, 1992. [cytowanie na str. 11]

[20] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. [cytowanie na str. 51]

[21] Olivier Chapelle and Vladimir Vapnik. Model selection for support vector machines. In *NIPS*, pages 230–236, 1999. [cytowanie na str. 11]

[22] Paweł Cichosz. *Systemy uczące się*. Wydawnictwo Naukowo-Techniczne, Warszawa, 2000. [cytowanie na str. 11]

[23] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. [cytowanie na str. 9]

[24] Pádraig Cunningham and Sarah Jane Delany. k-nearest neighbour classifiers, 2007. [cytowanie na str. 11]

[25] Philip M Davis and Michael J Fromerth. Does the arxiv lead to higher citations and reduced publisher downloads for mathematics articles? *Scientometrics*, 71(2):203–215, 2006. [cytowanie na str. 1]

[26] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association. [cytowanie na str. 23]

[27] Piotr Jan Dendek, Lukasz Bolikowski, and Michal Lukasik. Evaluation of Features for Author Name Disambiguation Using Linear Support Vector Machines. 2012. [cytowanie na str. 4, 8, 26, 63]

[28] Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alex J. Smola, and Vladimir Vapnik. Support vector regression machines. In *NIPS*, pages 155–161, 1996. [cytowanie na str. 11]

[29] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969. [cytowanie na str. 8]

[30] P. Galison and B. Hevly. *Big science: The growth of large-scale research*. Stanford University, Palo Alto, CA, 1992. [cytowanie na str. 1]

[31] Carmen Galvez and Félix Moya-Anegón. Approximate personal name-matching through finite-state graphs. *Journal of the American Society for Information Science and Technology*, 58(13):1960–1976, November 2007. [cytowanie na str. 3]

[32] Aldo Geuna. Allocation of funds and research output : the case of uk universities. Technical report, 1997. [cytowanie na str. 1]

[33] Hui Han, Lee Giles, Hongyuan Zha, Cheng Li, and Kostas Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. *Proceedings of the 2004 joint ACM/IEEE conference on Digital libraries - JCDL '04*, page 296, 2004. [cytowanie na str. 3, 8, 9]

[34] Hui Han, Hongyuan Zha, and C Lee Giles. Name disambiguation in author citations using a K-way spectral clustering method. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 334–343, New York, NY, USA, 2005. ACM. [cytowanie na str. 3]

[35] David Heckerman. A tutorial on learning with bayesian networks. Technical report, Learning in Graphical Models, 1996. [cytowanie na str. 11]

[36] J E Hirsch. An index to quantify an individualÊ1/4s scientific research output. *Proceedings of the National Academy of Sciences of the United States of America*, 102(46), 2005. [cytowanie na str. 10]

[37] J.R. Jacobs. Finding words that sound alike. the soundex algorithm. *Byte 7*, pages 473–474, 1982. [cytowanie na str. 29]

[38] Thorsten Joachims. A statistical learning model of text classification for support vector machines. In *In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 128–136. ACM Press, 2001. [cytowanie na str. 9]

[39] Koichi Kitazawa and Yongdi Zhou. The phd factory. *Labour*, 472(7343):0–3, 2009. [cytowanie na str. 1]

[40] Peder Olesen Larsen and Markus von Ins. The rate of growth in scientific publication and the decline in coverage provided by science citation index. *Scientometrics*, 84(3):575–603, 2010. [cytowanie na str. 1]

[41] F.H. Levin and C.A. Heuser. Using Genetic Programming to Evaluate the Impact of Social Network Analysis in Author Name Disambiguation. In *Proceedings of the Alberto Mendelzon Workshop on Foundations of Data Management. Buenos Aires, Argentina*. Citeseer, 2010. [cytowanie na str. 3, 8, 10]

[42] Paolo Manghi and Marko Mikulicic. Pace: A general-purpose tool for authority control. In Elena García-Barriocanal, Zeynel Cebeci, Mehmet C. Okur, and Aydın Öztürk, editors, *Metadata and Semantic Research*, volume 240 of *Communications in Computer and Information Science*, pages 80–92. Springer Berlin Heidelberg, 2011. [cytowanie na str. 8]

[43] Gideon S. Mann and David Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003* -, pages 33–40, Morristown, NJ, USA, 2003. Association for Computational Linguistics. [cytowanie na str. 3]

[44] Tomasz Maszczyk and Wlodzislaw Duch. Support vector machines for visualization and dimensionality reduction. In *ICANN (1)*, pages 346–356, 2008. [cytowanie na str. 43]

[45] Philipp Mayr and Anne-Kathrin Walter. An exploratory study of google scholar. [cytowanie na str. 10]

[46] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 169–178, New York, NY, USA, 2000. ACM. [cytowanie na str. 8]

[47] Hanna Pasula, Bhaskara Marthi, Brian Milch, Stuart Russell, and Ilya Shpitser. Identity uncertainty and citation matching. In *In NIPS*. MIT Press, 2003. [cytowanie na str. 8]

[48] D. Pavelec, L. S. Oliveira, E. Justino, F. D. Nobre Neto, and L. V. Batista. Compression and stylometry for author identification. *2009 International Joint Conference on Neural Networks*, pages 2445–2450, June 2009. [cytowanie na str. 3]

[49] Ted Pedersen, Anagha Kulkarni, Roxana Angheluta, Zornitsa Kozareva, and Thamar Solorio. An unsupervised language independent method of name discrimination using second order co-occurrence features. In *Computational Linguistics and Intelligent Text Processing*, pages 208–222. 2006. [cytowanie na str. 3]

[50] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1:81–106, March 1986. [cytowanie na str. 11]

[51] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0. [cytowanie na str. 51]

[52] Irina Rish. *An empirical study of the naive Bayes classifier*, pages 41–46. 2001. [cytowanie na str. 11]

[53] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 269–278, New York, NY, USA, 2002. ACM. [cytowanie na str. 8]

[54] Jie Tang, Jing Zhang, Duo Zhang, Limin Yao, Chunlin Zhu, and Juanzi Li. Arnetminer: An expertise oriented search system for web community. [cytowanie na str. 10]

[55] Dominika Tkaczyk and Lukasz Bolikowski. Workflow of Metadata Extraction from Retro-Born Digital Documents. In Petr Sojka and Thierry Bouche, editors, *Towards a Digital Mathematics Library*, pages 39–44. Masaryk University Press, 2011. [cytowanie na str. 32]

[56] Dominika Tkaczyk, Lukasz Bolikowski, Artur Czeczko, and Krzysztof Rusek. A modular metadata extraction system for born-digital articles. 2012. [cytowanie na str. 7, 32]

[57] Vetle I. Torvik and Neil R. Smalheiser. Author name disambiguation in MEDLINE. *ACM Transactions on Knowledge Discovery from Data*, 3(3):1–29, July 2009. [cytowanie na str. 29]

[58] Jinlong Wang, Shunyao Wu, H.Q. Vu, and Gang Li. Text document clustering with metric learning. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 783–784. ACM, 2010. [cytowanie na str. 31]

[59] Byung won On, Dongwon Lee, Jaewoo Kang, and Prasenjit Mitra. Comparative study of name disambiguation problem using a scalable blocking-based framework. In *INTERNATIONAL CONFERENCE ON DIGITAL LIBRARIES*, pages 344–353. ACM, 2005. [cytowanie na str. 8, 9]