Zeszyty Naukowe Wydziału Informatycznych Technik Zarządzania
Wyższej Szkoły Informatyki Stosowanej i Zarządzania
„Współczesne Problemy Zarządzania”
Nr 1/2015

IM WIĘCEJ, TYM LEPIEJ? O PEWNEJ ANALIZIE Z DZIEDZINY WYSZUKIWANIA INFORMACJI TEKSTOWEJ

Wojciech Ryczaj, Jan W. Owsiński

Wyższa Szkoła informatyki Stosowanej i Zarządzania,
01-447 Warszawa, ul. Newelska 6

Niniejszy artykuł jest oparty na pracy magisterskiej pierwszego z autorów (Ryczaj, 2014). Przedmiotem tej pracy była analiza zależności skuteczności pewnych funkcji, związanych z wyszukiwaniem informacji, w zależności od długości reprezentacji dokumentów, stanowiących potencjalny obiekt wyszukiwania. Chodziło przede wszystkim o sprawdzenie pozornie intuicyjnej tezy, że im dłuższa taka reprezentacja, tym bardziej efektywne wyszukiwanie. Jednym z wniosków pracy było wskazanie istnienia pewnego optimum w tym zakresie, tj. długości reprezentacji, poniżej której wyniki są gorsze, ale także powyżej której już się nie poprawiają, a nawet mogą się znów pogorszyć.

Słowa kluczowe: wyszukiwanie informacji, dokumenty, indeksy, długość indeksu

1. Wprowadzenie

Obecnie chyba już nikt nie jest w stanie sobie wyobrazić świata bez komputerów i Internetu. Często dzieci, które nie potrafią jeszcze dobrze mówić, są w stanie włączyć i w ograniczonym zakresie obsłużyć komputer, laptopa, tablet, czy smartfon. Internet i jego wykorzystanie wkroczyły już niemal w każdą dziedzinę życia, oferując dostęp do edukacji, kultury, nowych mediów, umożliwiając kontakty towarzyskie. Tak szerokie zastosowania powodują olbrzymi przyrost danych cyfrowych. Codziennie powstają tysiące blogów, digitalizowane są zasoby dziedzictwa kulturowego, coraz większą popularność zyskują e-booki, o filmach czy grach już nie wspominając. Z badań specjalistów z IDC wynika, że tylko w 2008 roku wygenerowano 487 trylionów bajtów (miliardów gigabajtów) nowych danych cyfrowych a wzrost w kolejnych latach ma wynieść 60 procent rocznie[1].

Brak racjonalnej polityki przechowywania danych w połączeniu z brakiem jednego zdefiniowanego modelu danych sprawia, że lokalizacja potrzebnych infor-

[1] http://www.chip.pl/news/wydarzenia/trendy/2010/03/czy-czeka-nas-potop-cyfrowych-danych[data dostępu: 07.07.2014r.]

macji jest trudna i czasochłonna. Dlatego coraz większego znaczenia nabierają umiejętności wyszukiwania, selekcji i zarządzania informacją. I nie dotyczy to tylko specjalistów IT, ale każdego z nas. Już szkoła podstawowa powinna być miejscem uczącym, kształtującym umiejętności wyszukiwania i zarządzania informacją.

Wyszukiwanie informacji polega na znalezieniu tych dokumentów, które poświęcone są wskazanemu w kwerendzie (zapytaniu) tematowi (Kłopotek, 2001). Aby umożliwić odnalezienie szukanej informacji stosuje się różnego rodzaju systemy klasyfikujące lub grupujące dokumenty tekstowe. Systemy te przyporządkowania do poszczególnych kategorii (klasyfikacja) czy podziału na odpowiednie grupy (grupowanie) dokonują na podstawie charakterystyk dokumentów. Charakterystyką taką jest ***indeks dokumentu*** lub zbiór słów kluczowych, sformułowany według określonych reguł, odzwierciedlający zasadniczy temat lub przedmiot dokumentu.

Powszechnie wiadomo, że im krótszy indeks tym większa szybkość wyszukiwania, lecz jednocześnie mniejsza dokładność i kompletność, i odwrotnie – wraz z wydłużaniem indeksu spada szybkość a wzrasta dokładność. Głównym celem pracy jest zbadanie powyższej zależności oraz dobór optymalnej długości indeksu dla wybranych procesów klasyfikacji i grupowania wiadomości tekstowych.

2. Trochę historii

Ludzkość niemal od początku swoich dziejów gromadziła informacje. Najpierw były to malunki ludzi pierwotnych na ścianach, następnie informacja zapisywana na tabliczkach glinianych, papirusach, pergaminie, czy papierze. Dokumenty te często gromadzone były w archiwach i bibliotekach. Dla sprawnego zarządzania zgromadzonymi zbiorami informacji, które z czasem stawały się coraz obszerniejsze niezbędne było wypracowanie metod organizowania, porządkowania i wyszukiwania informacji. Do czasu wynalezienia komputerów głównym kołem napędowym rozwoju systemów wyszukiwania informacji były systemy biblioteczne.

Jedną z pierwszych metod organizacji informacji była alfabetyzacja, czyli klasyfikowanie informacji zgodnie z porządkiem alfabetycznym. Po raz pierwszy zastosowali ją greccy bibliotekarze już w trzecim stuleciu p.n.e. w słynnej Bibliotece Aleksandryjskiej. Księgozbiór biblioteki szacowano na 400-750 tys. ksiąg. Uporządkowania tego ogromnego, jak na tamte czasy, zbioru podjął się Kallimach z Cyreny (ok. 310 - ok. 240 r. p.n.e.), tworząc chyba pierwszy w dziejach katalog, który dał podstawy działającym po dziś dzień systemom bibliotecznym.

Kallimach podzielił zasób ksiąg na dwa główne działy: poezję i prozę, a następnie na poddziały według rodzajów literackich. W obrębie każdej grupy autorzy ułożeni byli alfabetycznie. Dla każdego autora podawano jego biografię i spis utworów. Biografia zawierała nazwisko, wiadomości o ojczyźnie, pochodzeniu, szkołach i nauczycielach. Spis utworów zawierał tytuły dzieł, nadane w większości przez

Kallimacha oraz ich przynależność do odpowiedniego działu (nadawanie utworom literackim tytułów nie było jeszcze powszechne). Twórca katalogu, aby uniknąć możliwych nieporozumień, obok tytułów umieszczał początkowe słowa dzieł. Następnie podana była dokładna liczba wierszy, a niekiedy i streszczenie utworu. Dzieła poszczególnych pisarzy ułożone były w porządku alfabetycznym[2].

Za kolejną z metod organizacji wiedzy można uznać hierarchizację. Również w tym przypadku początków należy szukać w starożytności (podział tekstu na księgi, a ksiąg na rozdziały, stosowany był w literaturze antycznej). Przykładem jest „Historia naturalna" Pliniusza Starszego (23-79 r. n.e), rodzaj encyklopedii, w której pojęcia umieszczone były w różnych działach. Na dzieło składa się trzydzieści siedem ksiąg, zawierających około 20 000 informacji ze wszystkich dziedzin wiedzy[3].

W kolejnych wiekach systemów tych używano w mniejszym lub większym stopniu, nie było jednak bodźca, który wymusiłby ich rozwój. Tak było aż do wynalezienia druku przez Johannesa Gutenberga (ok. 1450 r.). Pojawiało się coraz więcej książek, ich ceny spadły, powstawało coraz więcej bibliotek i zwiększały się ich zasoby, w związku z czym pojawiła się potrzeba prostego opisu książek. Powstają podstawy używanych obecnie systemów klasyfikacyjnych, takich jak klasyfikacja Biblioteki Kongresu (LCC - The Library of Congress Classification) oraz System Klasyfikacji Dziesiętnej. Jako podstawę LCC wymienia się system wymyślony przez Tomasza Jeffersona na potrzeby jego biblioteki w Monticello, która stała się zalążkiem słynnej Biblioteki Kongresu Stanów Zjednoczonych. System ten był w użyciu do końca XIX w. Wobec rosnących potrzeb, na początku XX w. został rozbudowany przez pracowników Biblioteki, Jamesa Hansona i Charlesa Martela. W ciągu XX w. zaczęło stosować LCC blisko 90% bibliotek naukowych USA i Kanady. Klasyfikacja LCC należy do systemów alfanumerycznych (ciągi liter i cyfr)[4].

W Europie więcej zwolenników miał System Klasyfikacji Dziesiętnej, opracowany w drugiej połowie XIX w. przez Melvilla Deweya, również stanowiący do dziś podstawę klasyfikacji w wielu bibliotekach. Klasyfikacja Deweya stworzona została w układzie dziesiętnym, z cyfrowymi symbolami w postaci tak zwanego minimum trzycyfrowego, czyli od 000 do 999. Po trzech cyfrach używa się kropki, tworząc symbole rozbudowane[5].

Dziś, w epoce komputerów, powstają nowe, ulepszone systemy biblioteczne, jednak większość bibliotek nie rezygnuje całkowicie z opisanych systemów klasyfikacji, jako systemów zapasowych na wypadek awarii komputerów.

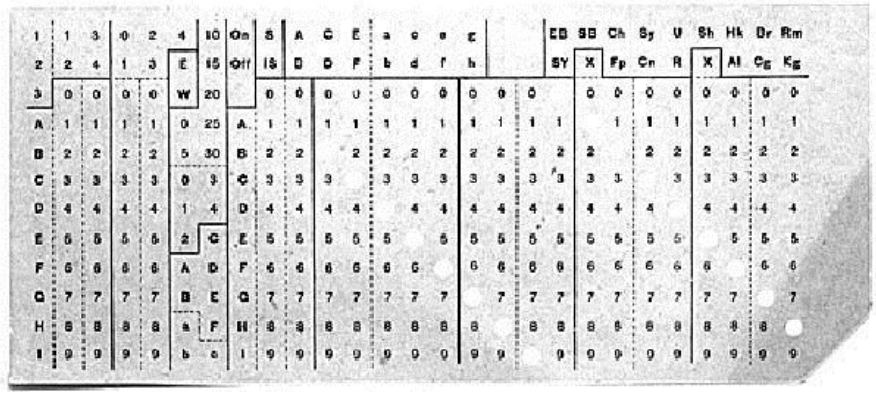
---

[2] http://www.wsp.krakow.pl/whk/biografie/spisk.html
[3] http://www.wsp.krakow.pl/whk/biografie/spisp.html
[4] http://pl.wikipedia.org/wiki/Klasyfikacja_Biblioteki_Kongresu
[5] http://pl.wikipedia.org/wiki/Klasyfikacja_Dziesiętna_Deweya

3. Era komputeryzacji

Za pierwszą próbę automatyzacji przetwarzania danych można uznać opracowanie przez Hermana Holleritha „tabulatora" do odczytywania danych demograficznych, zapisanych na kartach perforowanych, sumowania ich i drukowania raportów. Szacuje się, że wykorzystanie tabulatora w spisie ludności USA w roku 1890 pozwoliło skrócić czas pracy nad danymi z 8 lat (spis z roku 1880) do jednego roku. Urządzenia Holleritha przyjęły się szybko i funkcjonowały przez następne prawie 60 lat. Z kart perforowanych korzystały także systemy biblioteczne. Perforacja w nich odpowiadała wartości indeksu, i umożliwiała „pół-automatyczne" wydobywanie odpowiednich kart katalogowych.

Rys. 1. Tabulator Holleritha 1890 rok, źródło:
http://en.wikipedia.org/wiki/Tabulating_machine



Rys. 2. Karta perforowana Holleritha, źródło:
http://en.wikipedia.org/wiki/Tabulating_machine

W latach 50-tych XX wieku karty perforowane zastąpiono taśmą magnetyczną. Komputery, korzystające z nowej technologii, mimo sporych rozmiarów, zajmowały znacznie mniej miejsca niż te z poprzedniej generacji. Jedna taśma mieściła tyle danych, co dziesięć tysięcy kart. Kolejną zaletą nowych urządzeń w porównaniu do tabulatorów był brak potrzeby fizycznej zmiany obwodów urządzenia, by przystosować je do nowych zadań, wystarczyło je odpowiednio zaprogramować. Po taśmie magnetycznej przyszły magnetyczne dyski, które pozwalały na swobodny (a nie sekwencyjny – jak przy taśmie magnetycznej) dostęp do danych. Z dyskami magnetycznymi, ulepszonymi i unowocześnionymi, mamy do czynienia do dziś.

W kontekście współczesnych rozwiązań i zastosowań informatyki wspomnijmy, że systemy wyszukiwania informacji (**WI**) tekstowej różnią się istotnie od systemów zarządzania bazami danych – jakkolwiek, niewątpliwie, istnieją tutaj punkty styczności – ponieważ w przypadku tych pierwszych nie mamy określonych precyzyjnie „warunków spełniania kwerendy", czyli zapytania – nawet, jeśli używamy do tego, w celach praktycznych, pewnych miar, ostatecznym kryterium jest akceptacja użytkownika („chodziło mi o to, a nie o to").

4. Wyszukiwarki internetowe

Kolejnym kamieniem milowym w rozwoju systemów WI było powstanie i upowszechnienie Internetu. Wraz z kolejnymi milionami nowych stron i użytkowników, zapotrzebowanie na systemy WI gigantycznie wzrosło. Konieczne było opracowanie i wdrożenie systemów WI działających w Internecie. Założenia systemów pozostały takie same jak systemów „stacjonarnych", ale ich realizacja musiała być inna. Głównymi problemami, na jakie napotkały internetowe systemy WI były:

- brak narzuconych struktur oraz form stron i dokumentów znajdujących się w sieci (np. tekst, HTML – niestrukturalne, XML – podstrukturalne, BD - strukturalne, multimedia, dokumenty PDF, DOC - różnie);
- różne języki, zbiory znaków, brak zgodnego nazewnictwa;
- rozproszenie informacji na milionach serwerów – możliwa powtarzalność informacji;
- olbrzymi rozmiar zbiorów informacji;
- problemy w sformułowanie potrzeb informacyjnych przez użytkowników (nie zawsze użytkownik dokładnie wie czego szuka);
- brak możliwości weryfikacji prawdziwości informacji.

Za pierwszą wyszukiwarkę internetową uznaje się Archie, stworzoną w 1990 roku przez kanadyjskiego studenta Alana Emtage. Działanie Archie polegało na okresowym odpytaniu serwerów FTP i zapisaniu ich listy plików, bez przeszukiwania tych plików. Kolejnymi wyszukiwarkami były wykorzystujące protokół Gopher - Veronika i Jughead; warto wymienić jeszcze Wandex oraz Aliweb. Pierwszą z najbardziej znanych przeglądarek było Yahoo (od 1994 roku), które zapoczątkowało katalogowanie stron według linków dodawanych przez ludzi. Z uwagi na dobre wyniki wyszukiwania, Yahoo szybko zajął miejsce lidera w rankingu przeglądarek i pozostał na nim do czasu nadejścia Google.

Google (pierwotnie BackRub), to przeglądarka stworzona w 1996 roku przez Amerykanina Larry'ego Page'a i Rosjanina Sergeya Brina w ramach studenckiego projektu http://www.historiawyszukiwarek.pl/ - p4 na Uniwersytecie Stanforda. Dzięki zastosowaniu nowatorskiego algorytmu PageRank oceny jakości stron, wyniki zwracane przez wyszukiwarkę są zazwyczaj bardziej trafne i adekwatne do zadanego pytania, niż dla rozwiązań konkurencyjnych. Jako niezależny portal Google zadebiutowało w 1998 roku, niemalże z miejsca deklasując pod względem popularności inne wyszukiwarki. Do dnia dzisiejszego pomimo prób odebrania pozycji lidera przez Bing Microsoftu (wcześniej MSN Search i Live Search), czy chińskiego Baidu, Google pozostaje liderem obsługując około 40 000 nowych zapytań w ciągu sekundy[6].

Nowością, która pozwoliła Google zdobyć tak olbrzymią popularność, było PageRank. PageRank (ranga czy ranking strony) to algorytm obliczający ważność stron internetowych. Algorytmu działa tak, że ważność strony jest tym większa, im więcej innych stron się do niej odnosi. Ponadto linki na stronach, które są oceniane wysoko, mają większą wagę niż linki ze stron mało znanych. Jako ciekawostkę przytoczmy przykład z 2004 roku, kiedy to polscy internauci, dzięki wykorzystaniu

[6] http://www.internetlivestats.com/google-search-statistics/

powyższego algorytmu, spowodowali umieszczenie na pierwszej pozycji wyników wyszukiwania słowa „kretyn", strony internetowej Andrzeja Leppera.

Mimo olbrzymiej i wciąż rosnącej popularności coraz częściej słychać opinię, że okres rozwoju wyszukiwarek ogólnego przeznaczenia już minął. Główną wadą tych systemów jest powierzchowność dokonywanego przez nie przetwarzania danych (wynikająca z ograniczeń na czas przetwarzania), oraz duża ilość nietrafionych wyników (w szczególności, jeśli szukamy czegoś z wąskiej dziedziny wiedzy). Obecnie coraz większą popularność zyskują wyszukiwarki specjalizujące się w określonych dziedzinach zastosowań (tzw. wyszukiwarki dedykowane). Czas wyszukiwania jest z reguły dłuższy, ale za to otrzymujemy pełniejsze i trafniejsze wyniki w porównaniu do wyszukiwarek ogólnych.

Niezależnie od wyszukiwarek, istnieje wiele mechanizmów, których przeznaczeniem jest zdobywanie informacji z Internetu i/lub ich dostarczanie. Przegląd tych różnych form zawiera książka Kłopotka (2001). Są to takie mechanizmy jak multiwyszukiwarki, wyszukiwarki (aplikacje) osobiste, webringi, portale, itp. Nie będziemy jednak się nimi tutaj zajmować.

5. Cechy wyszukiwarek

Każda wyszukiwarka zawiera moduł odpowiedzialny za pobieranie dokumentów z sieci (spider, crawler, bot) którego zadaniem jest dodawanie stron do wyszukiwarek, sprawdzanie kodu strony, zbieranie informacji o stronie, monitorowanie nowości i tworzenie kopii stron. Kolejne strony odwiedzane są zgodnie z odsyłaczami hipertekstowymi umieszczonym na aktualnie badanych stronach.

Inny wspólny moduł wyszukiwarek to tzw. Indekser, odpowiedzialny za tworzenie bazy danych wyszukiwarki. Indekser analizuje treści zawarte na stronie i określa kategorię tematyczną strony. Ponadto, na podstawie analizy położenia poszczególnych słów i liczby ich wystąpień, indekser tworzy listę słów kluczowych dla strony. Po wpisaniu tekstu zapytania, jest on, po odpowiedniej modyfikacji, porównywany z listą słów kluczowych i na podstawie oceny tego porównania wyświetlają nam się, uporządkowane według trafności, wyniki wyszukiwania.

Każda baza danych wyszukiwarki posiada pewne właściwe możliwości formułowania zapytań. Najczęściej powtarzające się formy przedstawiono w Tabeli 1.

6. Wstępne przetwarzanie tekstu

Człowiek ma naturalną zdolność przetwarzania języka naturalnego. W miarę łatwo potrafimy rozpoznać języki (nawet jeśli ich nie znamy), szybko potrafimy uchwyć sens słów lub zdań wieloznacznych oraz uchwycić powiązania między wyrazami. Nawet ton głosu czy styl pisania dostarczają nam informacji o tekście. Niestety, do dziś nie udało nam się tych umiejętności zaimplementować komputerowo.

Dlatego też, przed zastosowaniem metod, na których opierają się systemy WI, konieczne jest sprowadzenia tekstu do czytelnej struktury, która byłaby rozpoznawalna przez wszystkie systemy WI. Odbywa się to różnymi metodami, choć głównym celem jest podział tekstu na wyrazy. Niestety, zamieniając dokument na ciąg wyrazów tracimy większość zależności językowych. Pozwala to jednak na obiektywne porównywanie ze sobą poszczególnych dokumentów, jak również pozwala wyznaczyć indeks dokumentów, czyli zbiór zwykle najczęściej występujących i najbardziej znaczących wyrazów, określających tematykę dokumentu.

Należy zaznaczyć, że wyszukiwanie pełnotekstowe, czyli w pełnej treści dokumentu, ma sens jedynie w przypadku kolekcji o niewielkich rozmiarach. W przypadku większej liczby dokumentów czas takiego szukania staje się niemożliwy do zaakceptowania.

Część stosowanych technik musi być dostosowana do języka dokumentu. Z oczywistych względów najwięcej algorytmów przetwarzania wstępnego zostało stworzonych dla języka angielskiego, najmniej zaś dla mało popularnych języków z bogatą fleksją i dużą liczbą form gramatycznych. W przypadku dokumentów strukturalnych, informacje o języku często możemy pobrać z metadanych dokumentu (np. znaczniki META, atrybuty LANG). W pozostałych przypadkach język trzeba określić za pomocą samego tekstu dokumentu.

Najczęściej stosowane metody rozpoznawania języka to:

- Rozpoznawanie rodzaju czcionki (pozwala poznać konkretny język lub zawęzić poszukiwania do grupy języków które używają danej czcionki),
- Porównanie najczęściej występujących wyrazów tekstu ze stop-listami charakterystycznymi dla danych języków (definicję stop-lisy podajemy dalej),
- Sprawdzenie występowania znaków diakrytyczne charakterystycznych dla danego języka (znaków graficznych: kropek, kresek, daszków, ogonków, kółeczek, akcentów itp., oznaczających brzmienie odmienne od podstawowego, wyrażanego ową literą. Np.: á, ą, â, ă, ä; ś, ş, š; ď, itp.),
- Analiza częstości wystąpień n-gramów literowych, czyli ciągów wchodzących w skład dokumentu o długości n znaków (każdy z języków posiada charakterystyczny rozkład częstości występowania różnych n-gramów).

Analiza tekstu rozpoczyna się od przetworzenia dokumentu tekstowego na tzw. „worek wyrazów" (ang. bag of words – „BOW"). W worku wyrazów ich kolejność nie ma znaczenia, nie są też połączone żadnymi operatorami logicznymi. Uwzględniana jest jedynie liczba wystąpień poszczególnych słów w dokumencie.

Tabela 1. Rodzaje wyszukiwań w wyszukiwarkach internetowych, źródło: opracowanie własne

Rodzaje wyszukiwania	Opis - Zwracane rezultaty
Wyszukiwania wg słów kluczowych (wyszukiwanie proste, simple search)	**1. Wg pojedynczego słowa** – zwraca wszystkie strony które zawierały dane słowo **2. Kilka słów na raz:** a. Wszystkie słowa - zwróci strony zawierające wszystkie wpisane słowa b. Którekolwiek ze słów - zwróci stronę zawierającą przynajmniej jedno z wpisanych słów
Wyszukiwanie boolowskie [AND, OR, NOT],	**1. & - I (koniunkcja)** - zwróci strony zawierające oba słowa. **2. | - LUB (alternatywa)** zwróci strony zawierające przynajmniej jedno ze słów. **3. ~ - NIE (zaprzeczenie)** wyłącza słowo z rezultatów. Zazwyczaj używa się go z koniunkcją (&), bo wtedy powoduje pominięcie pewnych rezultatów. Na przykład „adam & ~ewa" zwróci adresy wszystkich stron które zawierają słowo„adam" ale nie zawierają słowa „ewa". Wyszukiwanie samego „~ewa" nie zwróci nic, **4. () - grupowanie** umożliwia tworzenie złożonych wyrażeń. Na przykład„(adam | marek) & ~ewa" zwróci adresy stron zawierających słowo „adam" lub „marek" ale nie zawierających słowa „ewa".
Wyszukiwanie koncepcyjne	Polega na odnajdywaniu dokumentów znaczeniowo związanych z podanym słowem, lecz niekoniecznie użytym w ich tekście
Szukanie frazy (ciągu wyrazów, pełnych zdań),	Zwraca strony zawierające ciąg wyrazów lub pełne zdanie; osiąga się przez zamknięcie poszukiwanego łańcucha słów w cudzysłowie
Szukanie z określeniem odległości słów	Polega na określeniu odległości, w jakiej powinny się znaleźć w dokumencie podane słowa, tzn. ile słów między nimi może się znaleźć
Tezaurus	Wyszukiwanie z użyciem specjalnego słownika - zbioru synonimów, które mogą być użyte dla podanych w zapytaniu słów, jeżeli nie pojawiają się one w dokumentach
Wyszukiwanie rozmyte	Zastosowanie masek, np. *- zastępuje kilkuliterową końcówkę wyrazu, ? – zastępuje jeden znak

Proces zaczyna się od podziału dokumentu na słowa. Jako znaków rozdzielających używa się spacji, kropki, przecinka, znaku końca wiersza, czasem średnika, itp. Ponieważ ten proces jest automatyczny, czasami tracona jest informacja zapisywana w niezwyczajny sposób, np. o adresach internetowych czy adresach email, w których występuje znak kropki. Jeżeli program nie obsługuje wyjątku, który taki

adres zapisałby jako jeden wyraz, zostanie on rozbity na mało znaczące ciągi liter (np. inną informację przedstawia „wyraz" „www.e-podróżnik.pl", a inną wyrazy „www", „e", „podróżnik", „pl", o których nie wiemy, że są ze sobą powiązane). Dodajmy, że takie informacje często są charakterystyczne dla dokumentu i mogłyby kandydować do uwzględnienia ich w indeksie dokumentu.

Kolejny problem stanowią daty, które mogą zawierać ważną i charakterystyczną informację, zwłaszcza w tekstach o tematyce historycznej. Tutaj również, jeżeli program nie obsługuje takich wyjątków i nie rozpozna daty, to zamiast istotnej informacji dostaniemy nic nie znaczący ciąg znaków. Dochodzi tutaj też problem zapisu daty w różnych formatach np. 11 listopada 1990, 11 XI 1990 i 11.11.90 mogą zostać zapisane jako trzy różne wyrażenia, choć ich znaczenie jest takie samo.

Następnie, wszystkie litery występujące w dokumencie zamieniane są na litery wielkie bądź (częściej) litery małe. Uznano, że zysk z zastosowania takiej operacji (zmniejszenie liczby analizowanych wyrażeń) znacznie przewyższa jej wady. Większość wielkich liter wynika z zasad pisowni i nie zmienia sensu danego wyrażenia. Należy jednak uważać przy obsłudze niektórych języków (np. język niemiecki), gdzie relacja pomiędzy małymi i wielkimi literami nie zawsze jest taka sama.

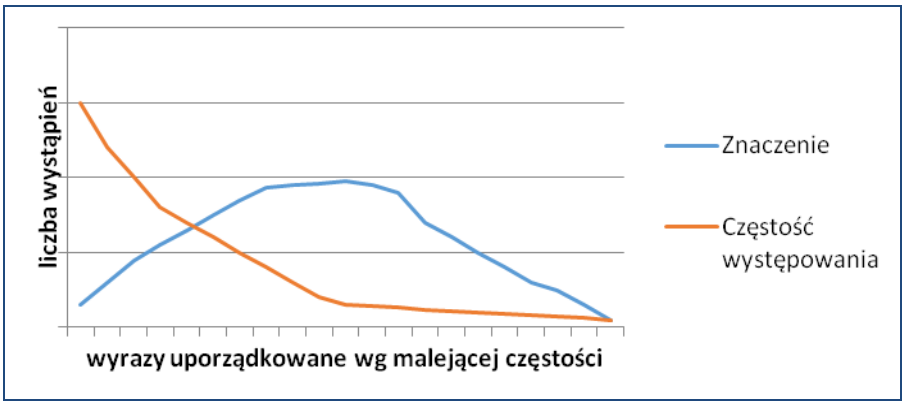
Gdybyśmy analizę częstości występowania słów przeprowadzali ręcznie lub przyjrzeli się analizie wykonanej przez komputer, zauważymy, że na szczycie listy przeważają wyrażenia mało znaczące dla treści dokumentów. W dodatku te słowa powtarzają się w każdym zbiorze, niezależnie od tematyki dokumentu. Są to głównie spójniki, rodzajniki, zaimki zwrotne, partykuły, czy też słowa pomocnicze służące do budowy czasów złożonych (np. „have" w języku angielskim). Zbiór takich słów nazywamy „stop-listą", a jego użycie polega na usunięciu z dalszej analizy wyrazów z tej listy. Często do stop-listy dorzuca się mało znaczące słowa, występujące sporadycznie lub charakterystyczne dla całej danej kategorii dokumentu.

Zastosowanie stop-listy ma wiele zalet. Ważna jest oszczędność czasu analizy i wykorzystywanej do niej pamięci, związana z zmniejszeniem zbioru do analizy. Szacuje się, że usunięcie wyrazów o zerowym znaczeniu automatycznie zmniejsza objętość dokumentu o około 40%.

Stosuje się trzy metody tworzenia stop-list:

- metodę słownikową – ustalona przez ekspertów lista słów, charakterystyczna dla danego języka i/lub danej kategorii dokumentu (przykłady stop-listy słownikowej dla języka angielskiego i polskiego zamieszczono w Tabeli 2),
- metodę statystyczną – wyrazy, których częstość występowania znajduje się w założonym przedziale lub np. słowa składające się z 1, 2 lub 3 liter,
- metodę hybrydową - czyli połączenie powyższych metod.

Dodajmy, że jako pierwszy, ogólną zależność pomiędzy częstością występowania a znaczeniem słów w dokumencie opisał w 1957 roku Hans Peter Luhn. Zależność tą w postaci tzw. „krzywej Luhna" przedstawiono na Rys. 3.



Rys. 3. Zależność między częstością a znaczeniem wyrazów, źródło: opracowanie własne

Jeśli zastosowaliśmy już wszystkie wymienione metody wstępnego przetwarzania tekstu, zajrzyjmy do naszego „worka wyrazów". Zauważymy, naturalnie, grupki podobnych wyrazów w różnych odmianach lub formach gramatycznych. Wiadomo, że oznaczają one to samo, a rozbite na odmiany nie odzwierciedlą ważności (uwzględniając liczbę wystąpień) trzonu głównego.

W celu uniknięcia tej sytuacji stosuje się metody redukcji form fleksyjnych, czyli wyłuskiwania rdzeni. To metody stemmingu i lematyzacji. Ich zastosowanie jest szczególnie ważne w językach bogatych w różne formy gramatyczne.

Operacja stemmingu polega na usunięciu końcówek i/lub przedrostków wyrazu pozostawiając jego rdzeń (ang.: stem), który niekoniecznie musi być poprawny gramatyczne. Lematyzacja uwzględnia kontekst słowa i budowę gramatyczną zdania, aby odtworzyć jego podstawową formę gramatyczną, czyli np. mianownik liczby pojedynczej dla rzeczownika lub bezokolicznik dla czasownika.

Można wyróżnić dwie główne metody realizacji powyższych operacji:

- podejście słownikowe, polegające na wykorzystaniu słownika zawierającego wszystkie znane formy gramatyczne słów i ich formy podstawowe;
- podejście algorytmiczne, polegające na wykorzystaniu zbioru reguł, pozwalających wykryć i usunąć różnice pomiędzy poszczególnymi formami danego słowa np. wykorzystując zbiór końcówek i przedrostków.

Tabela 2. Przykładowe stop-listy , źródło: http://pl.wikipedia.org/wiki/Wikipedia:Stopwords

język	Słowa wchodzące w skład stop-listy
ANGIELSKI	*a, able, about, across, after, all, almost, also, am, among, an, and, any, are, as, at, be, because, been, but, by, can, cannot, could, dear, did, do, does, either, else, ever, every, for, from, get, got, had, has, have, he, her, hers, him, his, how, however, i, if, in, into, is, it, its, just, least, let, like, likely, may, me, might, most, must, my, neither, no, nor, not, of, off, often, on, only, or, other, our, own, rather, said, say, says, she, should, since, so, some, than, that, the, their, them, then, there, these, they, this, tis, to, too, twas, us, wants, was, we, were, what, when, where, which, while, who, whom, why, will, with, would, yet, you, your*
POLSKI	*a, aby, ach, acz, aczkolwiek, aj, albo, ale, ależ, ani, aż, bardziej, bardzo, bo, bowiem, by, byli, bynajmniej, być, był, była, było, były, będzie, będą, cali, cała, cały, ci, cię, ciebie, co, cokolwiek, coś, czasami, czasem, czemu, czy, czyli, daleko, dla, dlaczego, dlatego, do, dobrze, dokąd, dość, dużo, dwa, dwaj, dwie, dwoje, dziś, dzisiaj, gdy, gdyby, gdyż, gdzie, gdziekolwiek, gdzieś, i, ich, ile, im, inna, inne, inny, innych, iż, ja, ją, jak, jakaś, jakby, jaki, jakichś, jakie, jakiś, jakiż, jakkolwiek, jako, jakoś, je, jeden, jedna, jedno, jednak, jednakże, jego, jej, jemu, jest, jestem, jeszcze, jeśli, jeżeli, już, ją, każdy, kiedy, kilka, kimś, kto, ktokolwiek, ktoś, która, które, którego, której, który, których, którym, którzy, ku, lat, lecz, lub, ma, mają, mało, mam, mi, mimo, między, mną, mnie, mogą, moi, moim, moja, moje, może, możliwe, można, mój, mu, musi, my, na, nad, nam, nami, nas, nasi, nasz, nasza, nasze, naszego, naszych, natomiast, natychmiast, nawet, nią, nic, nich, nie, niech, niego, niej, niemu, nigdy, nim, nimi, niż, no, o, obok, od, około, on, ona, one, oni, ono, oraz, oto, owszem, pan, pana, pani, po, pod, podczas, pomimo, ponad, ponieważ, powinien, powinna, powinni, powinno, poza, prawie, przecież, przed, przede, przedtem, przez, przy, roku, również, sam, sama, są, się, skąd, sobie, sobą, sposób, swoje, ta, tak, taka, taki, takie, także, tam, te, tego, tej, temu, ten, teraz, też, to, tobą, tobie, toteż, trzeba, tu, tutaj, twoi, twoim, twoja, twoje, twym, twój, ty, tych, tylko, tym, u, w, wam, wami, was, wasz, wasza, wasze, we, według, wiele, wielu, więc, więcej, wszyscy, wszystkich, wszystkie, wszystkim, wszystko, wtedy, wy, właśnie, z, za, zapewne, zawsze, ze, zł, znowu, znów, został, żaden, żadna, żadne, żadnych, że, żeby*

Jak zawsze, każde rozwiązanie ma swoje wady i zalety. Metoda słownikowa sprawdza się tylko wtedy, gdy mamy słownik uwzględniający wszystkie odmiany i przypadki. Wymaga też większych zasobów systemu, ponieważ program musi zarezerwować pamięć na słownik, oraz mieć algorytmy jego szybkiego przeszukiwania. Wspomnijmy, że im bogatszy w formy gramatyczne język, tym reguł jest więcej (szacuje się, że dla języka francuskiego liczba reguł stemmingowych wynosi około 60 000[7]). Wadą metody algorytmicznej jest z kolei mniejsza dokładność. Czasami dobrym wyjściem jest użycie metody mieszanej, czyli korzystanie z metody słownikowej, stosując metodę algorytmiczną w przypadku braku słowa w słowniku.

[7] http://search.blox.pl/2010/01/Myslec-po-polsku.html

Trudność stworzenia efektywnie działającego algorytmu stemmingu czy lematyzacji zależy od struktury języka, na którym algorytm ma operować. Najwięcej działających algorytmów zostało stworzonych dla języka angielskiego (m.in. algorytm Lovins, 1968, algorytm Portera, 1979, oraz algorytm Lancaster, 1990).

Innym problemem, związanym z przekształceniem BOW we właściwy zbiór do analizy i tworzenia indeksów jest polisemia i synonimiczność, charakterystyczne dla większości istniejących języków. Nie będziemy się jednak nimi tutaj zajmowali.

7. Tworzenie indeksów

Jeśli mamy już zbiór słów, tworzących dany dokument, w postaci listy wraz z częstościami wystąpień, możemy – mając podobne charakterystyki innych dokumentów – sporządzić indeks tego dokumentu. W tym celu wprowadzimy oznaczenia: i – numer słowa kluczowego, j – numer dokumentu, m_{ij} – liczbą wystąpień słowa i w dokumencie j. Oznaczmy także przez n liczbę wszystkich dokumentów, które bierzemy pod uwagę, a przez n_i – liczbę dokumentów, w których występuje słowo i. Aby określić wagę jakiegoś słowa i dla danego dokumentu j, najczęściej stosujemy następujące wyrażenie:

$$w_{ij} = (m_{ij}/\max_i m_{ij})\log_2(n/n_i), \tag{1}$$

gdzie w_{ij} jest właśnie ową szukaną wagą. Sens powyższego wyrażenia wydaje się dość oczywisty: wyższą wagę przypisujemy tym słowom, których względna częstość występowania w dokumencie jest większa (pierwszy czynnik) i tym, które pojawiają się w mniejszej liczbie dokumentów (drugi czynnik).

Wzór (1) można jeszcze przedstawić w nieco innych formach, lub w ogóle zastąpić nieco innym wyrażeniem, ale sens podstaw wyznaczania wagi słowa w dokumencie, w_{ij}, pozostanie taki sam. To jednakże oznacza, że musimy jeszcze wykonać jedną czynność, a mianowicie określić dolną granicę wartości wagi, od której można uznać, że dane słowo faktycznie reprezentuje dany dokument. Innym wyjściem mogłoby być ustanowienie stopnia „precyzji", tj. zaokrąglenia, co, wobec działania drugiego z czynników, prowadziłoby do wyeliminowania wielu słów. Tym niemniej, mając do czynienia z wyliczonymi w_{ij}, stajemy już wobec całkiem innego zagadnienia.

8. Modele wyszukiwania informacji tekstowej

Zajmiemy się obecnie modelami, służącymi do porównywania zapytań, zadawanych wyszukiwarkom, z indeksami dokumentów. Założymy zatem, że dysponujemy już indeksami, opartymi na wyznaczonych wartościach w_{ij}. Do porównywania wykorzystujemy trzy rodzaje „modeli".

Model klasyczny (boolowski)

Model klasyczny (boolowski) to najprostszy model, polegający na tym, że zarówno indeks dokumentu (cały dokument w wyszukiwaniu pełnotekstowym) jak i kwerenda to dwa ciągi słów, reprezentowanych przez zmienne binarne (1-wystąpienie słowa, 0-brak wystąpienia słowa). Model jest oparty na klasycznej logice binarnej i może być wyrażony w postaci zależności binarnych; uwzględnia on także podstawowe operacje algebry Boole'a (negację, alternatywę, koniunkcję).

Sprawdza się, czy słowa z zapytania (Z) występują w indeksie sprawdzanego dokumentu (I). Cząstkową funkcję podobieństwa $s_i(Z,I)$ między zapytaniem a indeksem dokumentu, określamy jako iloczyn zmiennych dla poszczególnych słów kluczowych. Zasady liczenia prawdopodobieństwa cząstkowego określa Tabela 3.

Tabela 3. Podobieństwo cząstkowe w modelu klasycznym, źródło: opracowanie własne

		Słowo nie występuje w zapytaniu	Słowo występuje w zapytaniu
		0	1
Słowo nie występuje w indeksie dokumentu	0	**1**	**0**
Słowo występuje w indeksie dokumentu	1	**1**	**1**

Dla całego zapytania mamy $s(Z,I) = s_1(Z,I_j) \times s_2(Z,Ij) \times ... \times s_m(Z,I_j)$. Tak więc funkcja podobieństwa przybiera tylko wartość 0 lub 1, czyli mamy do czynienia albo z odrzuceniem dokumentu (wartość 0) jeśli indeks dokumentu nie zawiera jakiegokolwiek, choćby jednego, ze słów kluczowych z kwerendy, albo z jej akceptacją (wartość 1), jeśli zawiera je wszystkie.

Główną zaletą omawianego modelu jest jego efektywność i prostota implementacji. Wady, to z jednej strony brak elastyczności w wyszukiwaniu (czyli przypadków częściowego spełniania warunków zapytania), a z drugiej - nadmiarowość w wyszukiwaniu, szczególnie wtedy, kiedy zapytanie zawiera niewielką ilość słów losowych (w skrajnych przypadkach jedno – co wcale nie jest takie rzadkie), lub są to słowa popularne dla dużej liczby dokumentów. Nadmienić należy, że model nie daje możliwości przetwarzania wyników wyszukiwania a każdy wyszukany dokument spełnia zapytania tak samo dobrze (dla każdego funkcja podobieństwa będzie mieć wartość 1). Pomimo swoich wad, model boolowski jest dość często stosowany np. w systemach informacji bibliotecznej czy edytorach tekstu takich jak MS Word.

Model klasyczny może być rozszerzany na dwa sposoby: (1) przez uwzględnienie innych funkcji podobieństwa cząstkowego, niż podstawowa, wyrażona w Tabeli 3; (2) przez uwzględnienie zarówno w zapytaniach, jak i sprawdzaniu do-

pasowania, bardziej skomplikowanych wyrażeń podobieństw, opartych na funkcjach logicznych.

Model zliczania wag binarnych (algebraiczny)

Aby wyeliminować podstawowe wady modeli boolowskich można użyć modeli algebraicznych. Najprostszy z nich, tzw. „model zliczania wag binarnych" daje możliwość akceptacji dokumentów spełniających warunki kwerendy tylko w pewnym stopniu. Model ten polega na zliczaniu jedynek występujących jednocześnie w wektorach reprezentujących zapytanie (z_i) oraz wektorach odpowiadających indeksowi dokumentu j (c_{ij}), a następnie podzieleniu otrzymanej liczby przez liczbę (niezerowych) elementów zapytania:

$$W_j(Z) = \sum_i c_{ij} z_i / \sum_i z_i \qquad (2)$$

gdzie: c_{ij} – wartości, składające się na wektor binarny charakteryzujący indeks dokumentu *j*-tego; z_i – wartości, składające się na wektor binarny charakteryzujący zapytanie; $W_j(Z)$ – waga (ocena) dokumentu *j*-tego na podstawie zapytania *Z*.

Podobnie jak w przypadku modelu boolowskiego, omawiany model można rozwijać poprzez zastosowanie różnego rodzaju wyrażeń i konstrukcji algebraicznych i logicznych.

Model wektorowy

Model wektorowy to w istocie tylko rozszerzenie modelu algebraicznego o wagi w_{ij}. Wagi przyjmują wartości z zakresu [0,1]. I tak, $w_{ij} = 0$ świadczy o nieobecności słowa kluczowego w indeksie dokumentu, zaś każda wartość >0 świadczyć będzie o tym, że słowo takie występuje.

Ocena dopasowania dokumentu do zapytania w tym modelu jest dana jako

$$W_j(Z) = \frac{\sum_i w_{ij} z_i}{\sqrt{\sum_i w_{ij}^2} \cdot \sqrt{\sum_i z_i^2}} \qquad (3)$$

Powyższą wartość można interpretować jako wartość kosinusa kąta między wektorami, odpowiadającymi opisom indeksu dokumentu i zadanego zapytania. Zaletą tego modelu jest niezależność od długości obu wektorów (zapytania i indeksu), czyli bezwzględnych wartości ich współrzędnych. Istotne są wyłącznie proporcje między wartościami dla poszczególnych współrzędnych.

Model probabilistyczny

Model probabilistyczny opiera się na prawdopodobieństwach wylosowania dokumentów, odpowiadających (lub nie) zapytaniu Z. Główną rolę w tym modelu odgrywa twierdzenie Bayesa i prawdopodobieństwa warunkowe. Przy założeniu, że występowanie poszczególnych słów kluczowych jest niezależne (w sensie probabilistycznym)[8], odpowiednie wzory można odpowiednio uprościć, a następnie doprowadzić je do takiej postaci, aby, mając odpowiednie statystyki – w istocie analogiczne do tych, jakie stoją u fundamentów modelu algebraicznego – móc otrzymywać coraz lepsze przybliżenia szukanych prawdopodobieństw „relewantności"[9] dokumentów do zapytań.

9. Klasyfikacja dokumentów tekstowych

Zadanie klasyfikacji polega na analizie zbioru dokumentów i przypisaniu każdego z nich, na podstawie zawartych w nim informacji, do jednej z wcześniej zadeklarowanych klas (kategorii). Kategorie (klasy decyzyjne), albo są już ustalone i system ma za zadanie jedynie przypisać nowe dokumenty do jednej z kategorii, albo są one ustalane na początku w procesie uczenia maszynowego. O zastosowaniu klasyfikacji wspomnieliśmy już, omawiając funkcje indeksera wyszukiwarki.

Klasyfikacja dokumentów tekstowych jest obecnie szeroko stosowana. Często nawet nie zdajemy sobie sprawy, że miała ona miejsce. Najczęstsze przypadki stosowania algorytmów klasyfikacji to:

- klasyfikacja dokumentów internetowych na użytek serwisu typu Yahoo;
- porządkowanie dynamicznych zbiorów dokumentów, np. przydział tekstów dla tłumaczy w biurze tłumaczeń lub podział tekstów w serwisie agencji prasowej;
- wybór informacji interesujących dla użytkownika, zgodnych z jego profilem; np. na rynku reklamowym – na podstawie informacji o użytkowniku, pozyskanych z różnych źródeł, agencje reklamowe starają się wysyłać tylko te reklamy, które mogły użytkownika zainteresować; analogicznie postępują systemy rekomendujące;

[8] Zauważmy, że w istocie takie założenie milcząco uczyniono także w pozostałych omawianych modelach WI. Model probabilistyczny ma tę nad nimi wyższość, że czyni to założenie jawnym. Dalsze postępy, zwłaszcza w modelu algebraicznym, opierają się praktycznie wyłącznie na uwzględnieniu i wykorzystaniu faktu zależności wzajemnej słów kluczowych (ich występowania i częstości, zwłaszcza w określonych zbiorach dokumentów).

[9] „Relewantność" jest podstawowym terminem systemów WI, oznaczającym, że dany dokument jest, w jakiś sposób, uznany za „wystarczająco" odpowiedni dla danego zapytania.

- automatyczne indeksowanie dokumentów z użyciem słownika kontrolowanego (słowa kluczowe jako kategorie);
- rozstrzyganie znaczenia homonimów w zależności od kontekstu.

Największym problemem, które napotykają klasyfikatory, jest problem wysokiej wymiarowości przestrzeni reprezentacji dokumentów. Wiadomo, że im więcej atrybutów jakiejś informacji, tym trudniej ustalić granice kategorii. Im więcej atrybutów, tym więcej możliwych zależności między nimi. W niniejszej pracy będę starał się badać właśnie powyższy problem. Atrybutami, o których tu mowa będą słowa kluczowe, wybrane dla analizowanych dokumentów.

Metody klasyfikacji tekstów możemy podzielić na następujące grupy:

- metody bayesowskie (zastosowanie reguły Bayesa);
- algorytm Rocchio (obliczenie dla każdej klasy jej centroidu);
- algorytm k-NN („k najbliższych sąsiadów") ;
- sieci neuronowe;
- drzewa decyzyjne;
- support vector machines (metoda wektorów podpierających).

Niektóre z powyższych metod zostaną kolejno opisane w skrócie.

Metody bayesowskie

Tak zwany naiwny klasyfikator Bayesa to jedna z najstarszych i najpopularniejszych metod uczenia maszynowego. Stosowany jest głównie do sortowania i klasyfikacji. Naiwny klasyfikator Bayesa jest klasyfikatorem statystycznym, zbiór klas decyzyjnych musi być skończony i zdefiniowany wcześniej.

Omawiany klasyfikator opiera się na tzw. regule Bayesa, która mówi, że przykład X klasyfikujemy do klasy decyzyjnej C_i, dla której wartość prawdopodobieństwa, że przykład X należy do tej klasy, $P(C_i|X)$ jest największa. Aby obliczyć prawdopodobieństwo $P(C_i|X)$. posługujemy się twierdzeniem Bayesa:

$$P(C_i|X) = P(X|C_i)P(C_i)/P(X)$$

gdzie:

$P(C_i)$: prawdopodobieństwo wystąpienia klasy C_i (tj. prawdopodobieństwo, że dowolny przykład należy do klasy C_i);

$P(X|C_i)$: prawdopodobieństwo warunkowe, że X wystąpi przy wylosowaniu klasy C_i;

$P(X)$: prawdopodobieństwo a-priori wystąpienia przykładu X.

Mianownik tej zależności, czyli $P(X)$, jest stały dla wszystkich klas i w dalszych obliczeniach można go pominąć. Pozostaje zatem pytanie: w jaki sposób obliczyć $P(X|C_i)$? Przy dużych zbiorach danych, o dużej liczbie atrybutów, obliczenia

takie będą bardzo skomplikowane i kosztowe (np. dla 30 zmiennych binarnych musielibyśmy oszacować liczbę prawdopodobieństw rzędu 2^{30}). Rozwiązaniem jest przyjęcie założenia o niezależności atrybutów, czyli, że wystąpienie w tekście dokumentu jednego słowa kluczowego nie dostarcza nam żadnych informacji o prawdopodobieństwie wystąpienia drugiego. Założenie o warunkowej niezależności zmiennych przy danych klasach nazywamy naiwnym założeniem Bayesa. Stąd bierze się też nazwa klasyfikatora. Jest to istotne uproszczenie, choć liczne badania dowiodły, że nie ma ono dużego negatywnego wpływu na dokładność klasyfikacji, a wyniki działania algorytmu są porównywalne z algorytmami znacznie bardziej skomplikowanymi.

Założenie o niezależności atrybutów prowadzi do poniższego wzoru, który można bezpośrednio implementować w systemach klasyfikujących:

$$P(X|C_i) = \prod_{j=1}^{n} P(X_j \mid C_i)$$

Podsumowując naiwny klasyfikator Bayesa, należy stwierdzić, że założenie o niezależności atrybutów znacznie redukuje koszt obliczeń. Jeżeli założenie jest spełnione, klasyfikator jest optymalny - zapewnia najlepszą dokładność klasyfikacji w porównaniu z innymi klasyfikatorami. Taka sytuacja występuje dość rzadko, jednak i w przypadkach niespełniania założenia, klasyfikator Bayesa, mimo swojej prostoty, daje wyniki porównywalne, a czasem lepsze niż inne metody klasyfikacji.

Algorytm Rocchio

Algorytm Rocchio klasyfikuje dokument do danej kategorii na podstawie jego podobieństwa do centroidu (profilu) tej kategorii. Profil (prototyp) każdej kategorii tworzony jest jako centroid (wektor średnich) z wektorów wszystkich egzemplarzy przykładowych należących do danej kategorii, jednakże zależność, zaproponowana przez Rocchio jest znacznie bardziej skomplikowana:

$$d_{k,i} = \alpha \sum_{v \in R_k} \frac{\langle v_i \rangle}{|R_k|} - \beta \sum_{v \in N_k} \frac{\langle v_i \rangle}{|N_k|}$$

gdzie:

$d_{k,i}$ - i-ta składowa wektora dla kategorii k;

v_i - i-ta składowa wektora dokumentu ze zbioru trenującego;

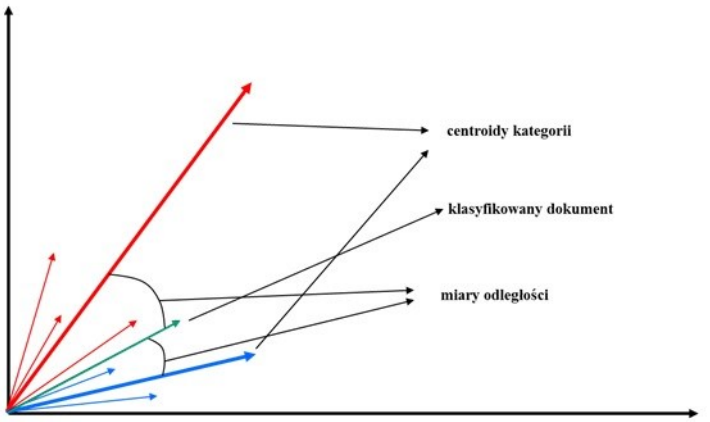
R_k - zbiory przykładów relewantnych (istotnych) względem kategorii k;

N_k - zbiory przykładów nierelewantnych (nieistotnych) względem kategorii k;

α,β - parametry określające istotność przykładów relewantnych i nie.

Tworząc wektor zgodnie z metodą Rocchio klasyfikator stara się wyłonić cechy charakterystyczne dokumentów dla danej kategorii i odrzucić te, które określają

dokumenty nieistotne względem kategorii. Można też stosować uproszczoną wersję algorytmu – podstawiając $\beta = 0$. Po wyznaczeniu centroidów, klasyfikator dokonuje porównania wektorów centroidów oraz wektora dokumentu klasyfikowanego (wg wybranej miary odległości). Dokument zostaje zakwalifikowany do tej kategorii, której odległość w przestrzeni wielowymiarowej do klasyfikowanego dokumentu jest najmniejsza. Przykład działania algorytmu można obejrzeć na Rys. 4. Klasyfikowany dokument zostanie przypisany do kategorii dolnej, ponieważ odległość między wektorami jest mniejsza niż odległość do kategorii górnej.



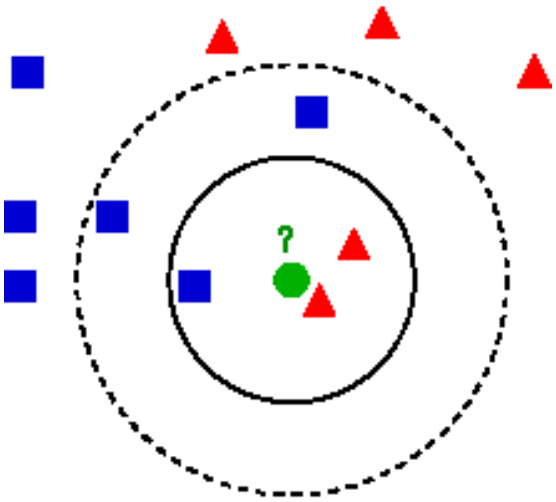
Rys. 4. Przykład klasyfikacji dokumentu wg algorytmu Rocchio, źródło: opracowanie własne

Algorytm k-NN („k najbliższych sąsiadów")

Zasada działania algorytmu kNN (ang. *k nearest neighbours*, *k najbliższych sąsiadów*) polega na wyszukaniu w zbiorze trenującym k najbardziej podobnych do klasyfikowanego (najbliższych) dokumentów i przypisanie mu kategorii takiej, jaką ma większość znalezionych. Wartość k dobierana jest eksperymentalnie, zazwyczaj $k = 1$, 3 lub 5, ale znane są też przypadki tworzenia klasyfikatora przy k=30.

Jeśli wśród k najbardziej podobnych są dokumenty różnych kategorii, to należy rozstrzygnąć, do której z nich klasyfikowany tekst przypisać. Pierwszy sposób to policzenie, ile dokumentów z każdej kategorii występuje w ustalonym sąsiedztwie i przypisanie dokumentu do kategorii najliczniejszej grupy. Jeżeli elementów z różnych kategorii jest tyle samo – należy zmniejszyć bądź zwiększyć liczbę sąsiadów. Drugi sposób to określenie dla każdej z kategorii średniego podobieństwa pomiędzy badanym dokumentem, a kategorią, do której należą dokumenty wybrane jako najbliższe. Możliwe jest też ważenie odległością, tj. założenie, że punkty bliższe są ważniejsze, więc należy nadać im większe znaczenie.

Przykład działania algorytmu możemy zaobserwować na Rys. 5. Rysunek przedstawia elementy już zakwalifikowane i podzielone na dwie kategorie (klasa 1 - trójkąty i klasa 2 - kwadraty) oraz nowy element (kółko), który chcemy sklasyfikować. Najpierw użyto metody 3-NN – rozważając trzy elementy najbliższe nowo klasyfikowanego (obiekty w ciągłym kole). W takim sąsiedztwie mamy dwa elementy z klasy pierwszej i jeden element z drugiej, i nowy obiekt zostanie przypisany do klasy trójkątów. Jeżeli zwiększymy sąsiedztwo do pięciu elementów (5-NN) sytuacja zmieni się diametralnie (obiekty w przerywanym kole). Mamy teraz ponownie z dwa elementy z klasy pierwszej, ale już trzy elementy z klasy drugiej. W tym przypadku nowy obiekt zostanie przypisany do klasy kwadratów.



Rys. 5. Przykład klasyfikacji dokumentu wg algorytmu k-NN, źródło: opracowanie własne

Sieci neuronowe

Sztuczne sieci neuronowe (SSN) to systemy, których struktura i funkcjonowanie wzorowane są na działaniu systemu nerwowego i mózgu. Bazowym elementem SSN jest neuron, jednostka przetwarzania informacji. Na wejścia neuronu trafiają sygnały wejściowe (parametry badanego obiektu czy zjawiska). Wejścia posiadają pewne wagi. W neuronie obliczana jest wypadkowa wag sygnałów wchodzących i na podstawie jej wartości neuron, za pomocą funkcji aktywacji, jest aktywowany (przekazuje sygnał dalej) lub nie. Neurony tworzą sieci, ułożone w „warstwy". W sieci zawsze występuje warstwa wejściowa - która dostarcza danych wejściowych, oraz wyjściowa - zwracająca wynik działania sieci. Dodatkowo może występować od jednej do kilku warstw pośrednich (ukrytych), w których dokonywane są

kolejne obliczenia, a które mogą np. reprezentować kolejne etapy przetwarzania informacji.

Sieci neuronowe mają zdolność do uczenia się, poprzez zdolność do dostosowywania współczynników wagowych. Zatem, uczenie sieci jest to wymuszenie na niej określonego zareagowania na sygnały wejściowe. Sieci neuronowe znajdują szczególnie ważne zastosowanie tam, gdzie nie ma prostych reguł klasyfikacji. Można je bowiem uczyć na przykładach, korygując własne błędy i wykorzystując wcześniej zdobyte doświadczenia do poprawnego wnioskowania.

Jeśli chodzi o zastosowanie w systemach wyszukiwana informacji to po odpowiednim dostosowaniu SSN może zapewnić obsługę takiego systemu. Warstwa wejściowa odpowiadałaby w takim wypadku zapytaniu, warstwa wyjściowa wskazywałaby na konkretne dokumenty lub kategorie. Neurony z warstwy pośredniej reprezentowałyby kolejne słowa kluczowe wraz z obliczonymi wagami. Proces propagacji sygnału poprzez sieć musiałby zostać zmodyfikowany tak, aby przebieg sieci nie kończył się już po pierwszym przejściu sieci. Modyfikacja mogłaby polegać na zaprojektowaniu przebiegów propagacji wstecznej, bazujących na drobnej modyfikacji zapytań po każdym przebiegu.

Największą zaletą takiej sieci jest możliwości „douczania" jej w trakcie bieżącego wykorzystywania.

Drzewa decyzyjne

Klasyfikacja tą metodą polega na zbudowaniu odpowiedniego drzewa decyzyjnego, które jest skierowanym acyklicznym grafem o strukturze drzewiastej. Wierzchołek (węzeł) w drzewie reprezentuje test na atrybucie, łuk reprezentuje wynik testu, a liście to już pojedyncza klasa lub rozkład wartości klas. Pierwszy (najwyższy) wierzchołek drzewa nazywany jest korzeniem drzewa (ang. root). Proces klasyfikacji sprowadza się do umieszczenia dokumentów w korzeniu drzewa i w oparciu o odpowiedzi, udzielane w kolejnych węzłach, przesuwania go coraz niżej w hierarchii, aż do liścia, który odpowiada właściwej kategorii.

Algorytm, według którego tworzone jest drzewo decyzyjne, wybierany jest ze względu na przyjęte kryterium podziału, czyli sposobu, w jaki tworzone są nowe węzły wewnętrzne. Każdy tworzony węzeł powinien „jak najlepiej" dzielić zbiór danych treningowych należących do tego wierzchołka. Najprostszym i najpopularniejszym algorytmem jest metoda zachłanna (dziel i rządź). Z innych znanych algorytmów można by wymienić algorytmy ID3 oraz C4.5, czy też metodę CART.

Główną zaletą drzew decyzyjnych jest przejrzystość i łatwość interpretacji modelu. Wadą jest natomiast ryzyko zbytniego przystosowaniu się systemu do zbioru testowego. Oznacza to, że klasyfikacja dokumentów należących do zbioru testowego daje bardzo dobre rezultaty, natomiast wyniki względem nieznanych doku-

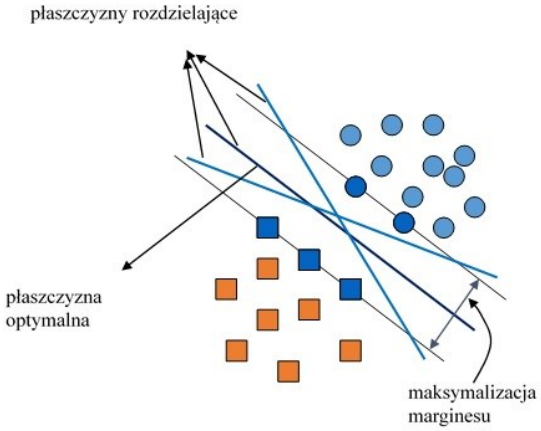
mentów są znacznie gorsze. Drzewa nie sprawdzają się również przy dużej liczbie atrybutów decyzyjnych – szczególnie wtedy, kiedy są one w miarę „równoważne", czyli mają podobny wpływ na zakwalifikowanie dokumentu do danej kategorii.

Support vector machines

Klasyfikatory SVM (ang. Support Vector Machines) zostały opracowane przez Vapnika (1963, 1992, 1995). Służą one do klasyfikacji binarnej, a więc do przypadku, w którym mamy dokładnie dwie klasy obiektów. W dużym skrócie można powiedzieć, że głównym zadaniem klasyfikatora jest określenie możliwie najlepszej płaszczyzny separacji pomiędzy klasami obiektów.

Intuicyjnie, za najlepsze moglibyśmy uznać płaszczyzny, przebiegające możliwie daleko od obu klas obiektów, czyli „pośrodku" i z dala od skupień. Dla takiej granicy oczekujemy, że popełnimy niewiele błędów klasyfikacji nowych obiektów. Taką płaszczyzny określamy przez maksymalizację marginesu wokół hiperpłaszczyzny rozdzielającej (Rys. 6) za pomocą metod programowania kwadratowego.

W przypadku liniowo separowalnym (gdy istnieje przynajmniej jedna płaszczyzna separacji oddzielająca klasy), metoda gwarantuje znalezienie płaszczyzny o maksymalnym marginesie separacji. W przypadku nieseparowalnym, stosując tzw. podniesienie wymiarowości, można za pomocą metody SVM znaleźć krzywoliniową granicę klasyfikacji o dużym marginesie separacji, która klasyfikuje obiekty na tyle poprawnie, na ile jest to możliwe i jednocześnie przebiega możliwie daleko od typowych skupień dla każdej z klas.



Rys. 6. Dobór płaszczyzny separacji wg metody SVM, źródło: opracowanie własne

10. Grupowanie dokumentów tekstowych

Grupowanie, nazywane również analizą skupień, albo klasteryzacją (ang. clustering) jest działaniem, prowadzącym do otrzymania podziału zbioru elementów na grupy. W przeciwieństwie do klasyfikacji, grupy nie są tutaj z góry narzucone, i nie mamy zbioru przykładów ze wstępnie określonymi kategoriami. Podział tworzony jest podczas analizy zależności między obiektami zbioru dokumentów w taki sposób, by elementy przypisane do jednej grupy były do siebie jak najbardziej podobne, podczas, gdy elementy z różnych grup jak najbardziej różne. Grupowanie można określić jako uczenie klasyfikacji bez nauczyciela. Liczba grup i sposób ich tworzenia jest zależny od przyjętych na początku założeń i użytego algorytmu.

Grupowanie może być wykorzystane do:

- wspomagania nawigacji w bazie dokumentów (umożliwia tworzenie hierarchii grup lub map tematycznych dokumentów, odpowiadających hierarchiom tematycznym - użytkownik wybiera odpowiadające mu kategorie, aby rozwinąć następny bardziej szczegółowy poziom);
- poprawy kompletności procesu wyszukiwania (dokumenty o podobnej zawartości są ze sobą związane tematycznie, tak więc są zwracane jako wyniki dla tych samych zapytań – np. zapytanie zawierające słowo kluczowe „auto” powinno zwrócić dokumenty zawierające słowo kluczowe „auto” jak również dokumenty zawierające słowo kluczowe „samochód”);
- wspomagania nawigacji w zbiorze wynikowym wyszukiwania (np. gdy zapytanie ma niejednoznaczny sens, grupy wprowadzają dodatkowy porządek w wynikach i ułatwiają wybór bardziej szczegółowej kategorii dokumentów do przeszukania;
- przyśpieszenia procesu wyszukiwania (zamiast obliczać podobieństwo zapytania do wszystkich dokumentów w kolekcji, może być ono wyznaczane jedynie dla środków poszczególnych grup dokumentów).

Algorytmy grupowania można podzielić na trzy zasadnicze grupy (Owsiński, 2014):

- agregacji hierarchicznej i podziału hierarchicznego;
- centrowania i reallokacji;
- podziału przestrzeni i identyfikacji obszarów podwyższonej gęstości.

Algorytmy agregacji hierarchicznej i podziału hierarchicznego

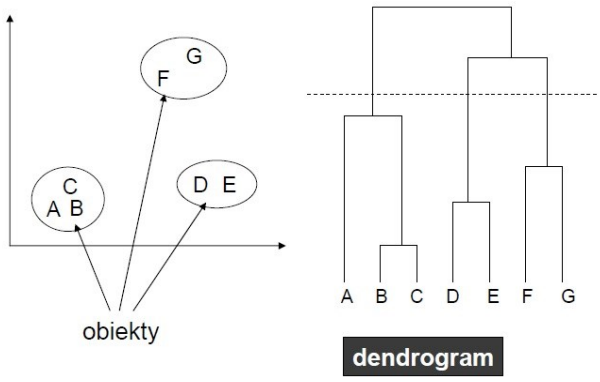
Algorytmy grupowania hierarchicznego tworzą ze zbioru dokumentów drzewo binarne, w którym węzłami są połączenia dokumentów. Drzewo takie nosi nazwę dendrogramu. Korzeniem dendrogramu jest grupa zawierająca wszystkie dokumenty, kolejne węzły odpowiadają klastrom na kolejnych poziomach szczegółowości, a liście to najczęściej grupy jednoelementowe.

Algorytmy hierarchiczne to algorytmy agregacji i podziału. Te pierwsze zaczynają od podziału, w którym każdy dokument stanowi odrębną jednoelementowa grupę i w każdym kolejnym kroku dwie najbardziej podobne grupy są łączone. Te drugie działają na odwrót - na początku tworzą grupę zawierającą wszystkie dokumenty z kolekcji i w kolejnych krokach dzielą jedną z grup na dwie podgrupy.

Przebieg algorytmu agregacji przedstawia się następująco:

1. liczymy odległości dla wszystkich par obiektów w zbiorze;
2. znajdujemy parę obiektów (***x*** i ***y***), których odległość jest najmniejsza;
3. łączymy te dwa obiekty w jeden klaster;
4. modyfikujemy zbiór obiektów usuwając z niego obiekty ***x*** i ***y*** oraz tworząc nowy obiekt ***z*** będący klastrem składającym się z połączonych ***x*** i ***y***;
5. wracamy do pkt 1 (modyfikujemy odległości dotyczące ***x*** i ***y***; modyfikacja obejmuje obliczenie nowej odległości ***z*** do pozostałych).

Przykład dendrogramu powstałego w wyniku grupowania siedmiu obiektów przedstawia Rys. 7. Na początku (patrząc od dołu) mamy zbiór nie pogrupowanych obiektów A, B, …,G. W kolejnych krokach, obiekty są łączone w klastry (B i C, D i E, F i G). Następnie łączymy obiekt A z klastrem zawierającym B i C. Potem łączymy klastry zawierające D i E oraz F i G w jeden. Na końcu mamy już tylko jeden klaster, zawierający wszystkie obiekty.



Rys. 7. Grupowanie hierarchiczne - dendrogram, źródło: http://wazniak.mimuw.edu.pl/

Odpowiedni algorytm podziału intuicyjnie powinien dać wynik zbliżony do algorytmu agregacji, lub nawet identyczny. Jednakże konstrukcja algorytmów podziałowych powoduje, że ich popularność jest nieporównywalnie mniejsza niż agregacyjnych, a sprawność także mniejsza. Niewiele jest prac, zmierzających do ustalenia „dualności" między tymi dwoma rodzajami algorytmów.

Podobieństwo pomiędzy klastrami maleje przy przechodzeniu w górę dendrogramu. Nie interesuje nas ostatni klaster, zawierający wszystkie dokumenty, bo wtedy grupowanie nie miałoby sensu. Agregację trzeba więc przerwać w pewnym momencie lub przeciąć dendrogram na interesującym nas poziomie. Algorytm można zatrzymać, gdy stworzona zostanie uprzednio zdefiniowana liczba klastrów, lub gdy podobieństwo pomiędzy klastrami spadnie poniżej określonego poziomu.

Zasadniczą kwestią pozostaje liczenie odległości dla klastrów. Najczęściej stosuje się następujące podejścia:

– metoda odległości minimalnej (metoda najbliższego sąsiada, single-linkage): podobieństwo pomiędzy grupami jest równe wartości podobieństwa dwóch najbliższych sobie elementów z tych grup:

$$D(C_1,C_2) = \min_{i\in C_1, j\in C_2} d(i,j) \tag{4}$$

gdzie $D(.,.)$ jest funkcją odległości między skupieniami, $d(.,.)$ – funkcją odległości między obiektami (tutaj: i,j), zaś C_l oznacza tworzone skupienia.

– metoda odległości maksymalnej (metoda najdalszego sąsiada, complete-linkage): podobieństwo pomiędzy grupami jest równe wartości podobieństwa dwóch najbardziej od siebie odległych elementów tych grup:

$$D(C_1,C_2) = \max_{i\in C_1, j\in C_2} d(i,j) \tag{5}$$

– metoda średniej odległości (average, arithmetic-mean): podobieństwo miedzy grupami równe jest średniej arytmetycznej wartości podobieństwa pomiędzy każdym elementem z pierwszej grupy a każdym elementem z drugiej grupy:

$$D(C_1,C_2) = \frac{1}{|C_1||C_2|\sum_{i\in C_1}\sum_{j\in C_2} d(i,j)} \tag{6}$$

– metoda centroidów – podobieństwo miedzy grupami równe jest wartości podobieństwa ich centroidów – punktów (rzeczywistych lub sztucznych) reprezentujących geometryczny środek klastra (wzór 2.15 i 2.16)

$$D(C_1,C_2) = d(c_1,c_2) \tag{7}$$

$$c_k = \frac{1}{|C_k|}\sum_{i\in C_k} i \tag{8}$$

Należy podkreślić, że wyniki otrzymywane dla tych samych zbiorów danych z użyciem różnych metod liczenia odległości klastrów mogą się istotnie różnić. Metoda najdalszego sąsiada jest skuteczniejsza w przypadku, gdy zbiór obiektów jest wyraźnie i regularnie podzielony, zaś metoda najbliższego sąsiada dobrze radzi sobie w przypadku, gdy grupy mają nieregularne, łańcuchowe kształty.

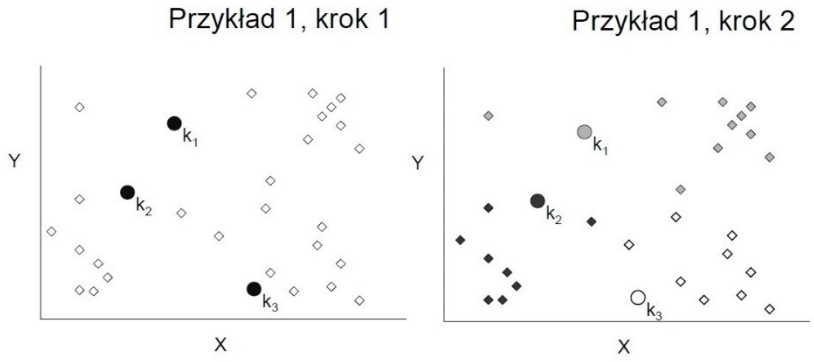
Algorytmy centrowania i realokacji.

W tych metodach liczbę klastrów musimy ustalić jeszcze przed rozpoczęciem procesu grupowania. Najbardziej znanym algorytmem z tej grupy jest algorytm k-średnich (ang. k-means). Jest on jednym z najstarszych algorytmów grupowania danych. Ze względu na swoją prostotę oraz niską złożoność obliczeniową zyskał bardzo dużą popularność. Przebieg algorytmu przedstawia się następująco:

1. wybierz *k* obiektów, które będą pierwszymi centroidami klastrów;
2. każdy obiekt przydziel do tego klastra, dla którego odległość obiektu od środka klastra jest najmniejsza;
3. dla każdego klastra uaktualnij (według wzoru (8)) jego środek (centroid);
4. wróć do pkt 2.

Powyższy proces powtarzany jest tak długo, jak długo występują zmiany przydziału obiektów do klastrów, lub został osiągnięty inny, zdefiniowany wcześniej warunek stopu (np. zadana liczba iteracji, warunek czasu).

Aby lepiej zrozumieć przebieg algorytmu przyjrzyjmy się przykładom pokazanym na Rys. 8-10. Zakładamy, że zaprezentowany zbiór obiektów będziemy chcieli podzielić na trzy klastry. W kroku 1 losujemy trzy punkty (k1, k2, i k3), stanowiące początkowe centroidy trzech klastrów. W kroku 2 – poszczególne obiekty przydzielane są do klastrów. Przydział dokonywany jest na podstawie minimum odległości każdego z obiektów do punktów k1, k2, i k3.

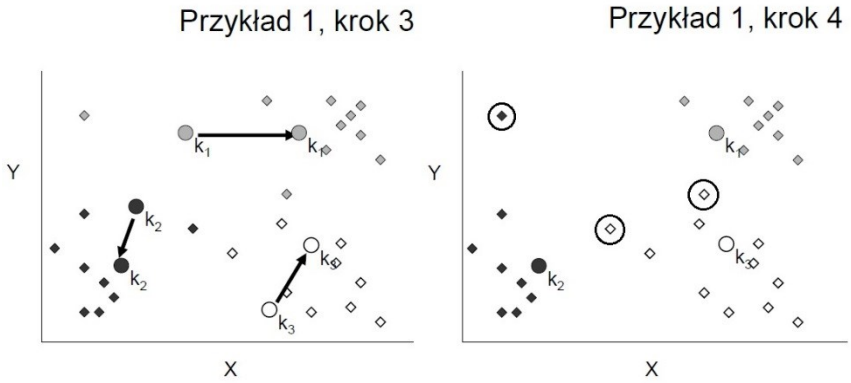


Rys. 8. Algorytm k-means – krok 1,2, źródło: http://wazniak.mimuw.edu.pl/images/8/86/ED-4.2-m11-1.01.pdf

W kroku 3 algorytmu centroidy wszystkich klastrów są aktualizowane i ulegają przesunięciu (na rysunku to przesunięcie zaznaczono strzałkami). Po realokacji centroidów ponownie wyznaczamy odległości obiektów od nich (krok 4). Zauważmy, że duża część obiektów nie zmieni swojego przypisania do klastrów. Realokacja

dotyczy trzech obiektów (zaznaczone kółkiem). Po zmianie środków klastrów i ponownym obliczeniu odległości okazało się, że zaznaczone obiekty teraz są bliższe innym klastrom. Realokujemy więc obiekty do nowych klastrów i - ponieważ nastąpiła zmiana - przechodzimy ponownie do kolejnego kroku aktualizacji środków klastrów (krok 5). Po przesunięciu środków okazuje się, że nie występują już obiekty, które trzeba by realokować. Można więc zakończyć działanie algorytmu.

Zaprezentowany przykład pozwala nam prześledzić przebieg algorytmu, ale zwraca też uwagę na zasadniczą jego wadę. Algorytm jest bardzo czuły na dane zaszumione lub zawierające punkty osobliwe, ponieważ one w istotny sposób wpływają na położenie centroidu klastra, powodując jego zniekształcenie. Łatwo to zauważyć na przykładzie klastra nr 2 w omawianym wyżej przykładzie. Punkt k2 (środek klastra) jest przesunięty w stosunku do środka „masy" tego klastra ze względu na jeden z punktów, który istotnie odbiega od pozostałych. Kolejną wadą jest fakt, że ostateczny podział obiektów pomiędzy klastrami jest silnie uzależniony od początkowego podziału obiektów (czyli wyboru początkowych środków klastrów). Innymi słowy – końcowy wynik działania algorytmu dla tych samych danych wejściowych może się znacząco różnić w zależności od tego, które punkty zostały wybrane w kroku 1 jako początkowe środki grup.

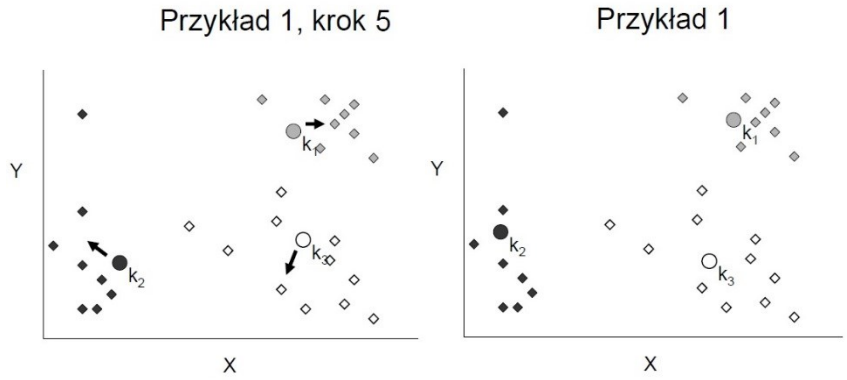


Rys. 9. Algorytm k-means – krok 3,4, źródło: http://wazniak.mimuw.edu.pl/images/8/86/ED-4.2-m11-1.01.pdf

Dla poradzenia sobie z wrażliwością na punkty odległe powstał algorytm k-medoidów, w którym zamiast środka, w roli reprezentanta występuje obiekt (dokument) najbliższy środka. Drugą wadę można niwelować przez wielokrotne wykonanie algorytmu z różnymi centroidami początkowymi, a następnie wybranie rozwiązania najlepszego, tj. minimalizującego faktyczną funkcję kryterium tych metod:

$$\sum_{i=1}^{k} \sum_{j \in C_i} d(c_i, j) \tag{11}$$

gdzie i jest numeracją klastrów (od 1 do k), a c_i jest reprezentantem i-tego klastra, tj. C_i.



Rys. 10. Algorytm k-means – krok 5, źródło: http://wazniak.mimuw.edu.pl/images/8/86/ED-4.2-m11-1.01.pdf

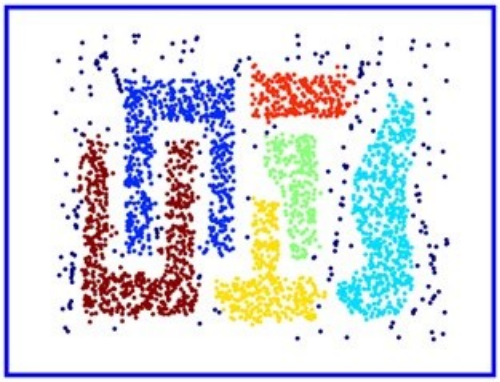
Algorytmy podziału przestrzeni i identyfikacji obszarów podwyższonej gęstości

Algorytmy opisane poprzednio, w szczególności algorytmy centrowania i realokacji znajdowały na ogół klastry, będące zbiorami wypukłymi, głównie o kształcie sferycznym[10]. Metody bazujące na gęstości mogą rozpoznawać skupienia mające różne kształty (przykład – Rys. 11). Ich idea polega na tym, że dany klaster jest rozszerzany o obiekty należące do jego sąsiedztwa, pod warunkiem, że gęstość obiektów w jego sąsiedztwie przekracza ustaloną wartość progową. Najbardziej znanym reprezentantem tej grupy algorytmów jest algorytm DBSCAN.

Zaletą omawianych metod jest niska złożoność obliczeniowa i możliwość znajdywania dowolnych kształtów. Dodatkowo, algorytm potrafi wskazać punkty odstające – nie są one klasyfikowane do żadnej z powstałych grup. Główną wadą jest trudność określenia prawidłowej wartości gęstości dla różnych danych wejściowych. Ponadto, jeśli klastry będą charakteryzować się różną gęstością, mogą nie zostać rozpoznane poprawnie.

Algorytmy grupowania oparte na gęstości są najczęściej stosowane w sferze rozpoznawania obrazów i dość rzadko stosuje się je w systemach wyszukiwania informacji tekstowych.

[10] Nie dotyczy to metody najbliższego sąsiada, a także pewnych zaawansowanych algorytmów z rodziny k-średnich.



Rys. 11. Algorytm DBSCAN – przykładowa klasyfikacja, źródło: http://www.hypertextbookshop.com/ datamin-ingbook/public_version/contents/chapters/chapter004/section004/blue/page003.html

11. Miary klasyfikacji

Zajmiemy się teraz wprowadzeniem do miar klasyfikacji, ponieważ będą one używane w prezentowanych eksperymentach. Podstawą miar jakości klasyfikacji jest tzw. macierz poprawności (ang. confusion matrix), pokazana jako Tabela 4.

Tabela 4. Schemat *confusion matrix*, źródło: opracowanie własne

		prawidłowa klasa	
		pozytywna	**negatywna**
klasa przypisana przez klasyfikator	**pozytywna**	TP (true positive)	FP (false positive)
	negatywna	FN (false negative)	TN (true negative)

Poszczególne pola tabeli oznaczają (przy założeniu, że klasyfikator przypisuje do klasy C_i dokumenty których indeks zawiera słowo t):

TP – Prawdziwe klasyfikacje pozytywne – liczba dokumentów ze słowem t należących do klasy C_i;

FP – Fałszywe klasyfikacje pozytywne – liczba dokumentów ze słowem t należących do innej klasy niż C_i;

FN - Fałszywe klasyfikacje negatywne - liczba dokumentów bez słowa t należących do klasy C_i;

TN - Prawdziwe klasyfikacje negatywne - liczba dokumentów bez słowa t należących do innej klasy niż C_i.

Najczęściej wykorzystywane miary klasyfikacji w systemach WI:

- **Accuracy** – dokładność klasyfikatora, tj. zdolność poprawnej predykcji klasy dla nowego przykładu: stosunek liczby dobrze rozpoznanych obiektów do liczby wszystkich obiektów:

$$acc = \frac{TP + TN}{TP + FP + TN + FN}. \tag{12}$$

- **Błąd (error)** – dopełnienie dokładności klasyfikatora: stosunek liczby źle rozpoznanych obiektów do liczby wszystkich obiektów:

$$error = \frac{FP + FN}{TP + FP + TN + FN}. \tag{13}$$

- **Precision (miara P, positive predictive value)** – dokładność klasyfikacji w obrębie rozpoznanej klasy lub ogólnie poprawność przypisania kategorii:

$$P = \frac{TP}{TP + FP} \tag{14}$$

- **Recall (miara R, true positive rate, sensitivity, czułość, kompletność),** mówi o tym na ile nasz klasyfikator potrafi rozpoznać obiekty z danej klasy, lub ogólnie, czy dana kategoria jest poprawna dla dokumentu:

$$R = \frac{TP}{TP + FN} \tag{15}$$

- **miara F1 (F1-measure, F1-score),** oparta na dokładności i kompletności:

$$F1 = \frac{2PR}{P + R} \tag{16}$$

Wszystkie przedstawione miary przyjmują wartości z przedziału [0,1], najlepszy wynik to 1, a najgorszy to 0 (z wyjątkiem błędu, gdzie jest na odwrót).

12. Miary odległości

Jednym z podstawowych sposobów postrzegania relacji (podobieństwa) między dokumentami, a także między zapytaniem a dokumentem, jest „odległość" między nimi. Istnieje wiele sposobów liczenia odległości, czyli miar odległości, a ich wybór zależy, w ogólności, od rodzaju danych, czyli sposobu charakteryzowania dokumentów i zapytań, a ponadto od typu i charakterystyki przyjętych metod grupowania obiektów. Najbardziej znane miary odległości to:

- **odległość euklidesowa (norma L2)** bardzo popularna, intuicyjna, wyznaczana z twierdzenia Pitagorasa, reprezentuje odległość jako długość linii prostej między punktami $\boldsymbol{x}$ i $\boldsymbol{y}$, obiektami w n-wymiarowej przestrzeni:

$$d(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{17}$$

gdzie x_i oraz y_i to poszczególne współrzędne obiektu (np. wagi słów kluczowych w indeksie dokumentu). Zazwyczaj, aby uniknąć problemów, związanych z odmiennością skal różnych współrzędnych, dane się normalizuje.

- **odległość Minkowskiego,** która jest uogólnieniem odległości euklidesowej:

$$d(\boldsymbol{x},\boldsymbol{y}) = \sqrt[q]{\sum_{i=1}^{n}(x_i - y_i)^q} \tag{18}$$

gdzie q określa wariant tej odległości (dla q=2 mamy odległość Euklidesową, dla q=1 tzw. odległość Manhattan, miejską lub taksówkowa, zaś dla $q = \infty$ tzw. odległość Czebyszewa (maksimum po współrzędnych).

Dodajmy, że w systemach WI używa się także innego rodzaju odległości, tzw. odległości edytorskich, np. odległości Levenshteina, w których obiektami są teksty jako ciągi znaków. Wykorzystuje się je na poziomie porównywania tekstów, nawet niekiedy dłuższych, ale nie ich szerokich zbiorów. Dlatego tez nie będziemy się nimi tutaj zajmowali.

13. Eksperymenty

Aplikacja i jej środowisko

W ramach pracy została wykonana aplikacja, posługująca się pakietem statystycznym R. R to nazwa języka programowania oraz platformy programistycznej wyposażonej w interpretator tego języka a także nazwa projektu, w ramach którego rozwijany jest zarówno język jak i środowisko. To dostępne bezpłatnie, na zasadach powszechnej licencji publicznej GNU, środowisko do analiz statystycznych. Wyposażone w elastyczny język programowania jest coraz chętniej wykorzystywane zarówno w sferze dydaktycznej jako narzędzie do nauki analizy danych, jak i przez naukowców z wielu dziedzin. O projekcie można poczytać na stronie http://www.r-project.org/ skąd można również pobrać instalator programu.

Jako główne zalety R można wymienić następujące cechy:

– ogromna liczba (tysiące) gotowych do pobrania pakietów realizujących wyspecjalizowane zadania z różnych dziedzin (nie tylko statystyki); lista pakietów jest na bieżąco uaktualniana, a ich instalacja jest prosta i automatyczna (np. poleceniem *install.packages(nazwa_pakietu)* w linii poleceń konsoli tekstowej lub przez wybór w odpowiedniej zakładce), wraz z wymaganymi pakietami zależnymi; każdy może stworzyć swój własny pakiet, który po weryfikacji zostanie dołączony do bazy pakietów programu;
– możliwość wykonywania funkcji z bibliotek w innych językach (C, C++, Fortran) i wykonywania funkcji dostępnych w R z poziomu innych języków (Java, C, C++ itp.); możemy więc część programu napisać np. w C++, a R wykorzystywać jako zewnętrzną bibliotekę funkcji statystycznych;

– wspomniana darmowość oraz dostępność pakietu;
– możliwość wykonywania wykresów opisujących analizowane dane.

Wadą języka R jest długi czas wykonywania operacji, szczególnie w porównaniu do analogicznych programów napisanych w językach kompilowanych. Jednak wraz ze wzrostem mocy obliczeniowej komputerów ma to coraz mniejsze znaczenie.

Zbiór testowy

Jako testowego zbioru tekstów użyto zbioru REUTERS-21578 (dostępny pod http://www.daviddlewis.com/resources/testcollections/reuters21578/). Składa się on z 21 578 wiadomości, które pojawiły się w znanej agencji informacyjnej Reuters w roku 1987. Dokumentów, które mają przypisane kategorie jest około 13 000. Kategorii jest ponad 600, jednakże wiele z nich występuje sporadycznie. Do badań wybrano dokumenty, które przypisano do 15 najczęstszych kategorii (około 8 500 dokumentów) oraz do 8 najczęstszych kategorii (około 7 750 dokumentów).

Przetwarzanie zbioru testowego

Za pomocą wbudowanych funkcji pakietu tm programu R, zbiór testowy został przetworzony zgodnie z zasadami już tutaj opisanymi. W szczególności, wykonano takie czynności jak: stwierdzenie, że dokumenty są w języku angielskim, usunięcie nadmiarowych białych znaków (spacje, tabulatury, nowe linie), usunięcie znaków interpunkcyjnych, zamiana wszystkich liter na małe, usunięcie liczb, usunięcie słów ze stop-listy (użyto standardowej stop-listy z pakietu tm, obejmującej 174 wyrazy), stemming (domyślnie: algorytmem Portera).

Początkowy zbiór testowy liczy 10 772 dokumenty. Po usunięciu znaków interpunkcyjnych i liczb, ale przed zastosowaniem stop-listy i stemmingu kolekcja liczyła 1 074 386 słów („termów"), z czego 33 203 różnych. Standardowa angielska stop-lista pozwoliła zredukować liczbę różnych słów zaledwie o 102 (do 33 101), ale to pozwoliło zmniejszyć liczność wszystkich słów o ponad 230 000 (do 840 530). Na tym przykładzie widać jak ważny jest odpowiedni dobór stoplisty. Zastosowanie stemmera zredukowało liczbę różnych termów do 24 578 rdzeni. Liczba wszystkich termów, zgodnie z przewidywaniami pozostała bez zmian.

Wyznaczanie wag

Pakiet pozwala na stosunkowo łatwe, automatyczne wyznaczenie macierzy częstości występowania słów, w tym przypadku zawierającej 10 772 wiersze i 24 578 kolumn. Na jej podstawie można wyznaczyć inne charakterystyki, w tym, na przykład, charakterystyki kolekcji na poziomie dokumentów (Tabela 5), czy też proste statystyki dla poszczególnych słów (Tabela 6 pokazuje je dla 25 najczęściej występujących w kolekcji).

Tabela 5. Charakterystyka dokumentów kolekcji, źródło: opracowanie własne

Charakterystyka	**Wartość charakterystyki**
Liczba dokumentów	10 772
Liczba wszystkich rdzeni	840 530
Minimalna liczba rdzeni w dokumencie	5
Mediana liczby rdzeni w dokumencie	53
Średnia liczba rdzeni w dokumencie	78
Maksymalna liczba rdzeni w dokumencie	812

Wspomniany pakiet pozwala także, na podstawie macierzy częstości wystąpień, wyznaczyć w sposób automatyczny wagi słów dla dokumentów, w_{ij}, przy czym jest także możliwe ustawienie automatycznej normalizacji przy tej czynności.

Mając te dane, można było przystąpić do właściwego eksperymentu, przy czym, ponieważ jego pierwszym etapem była klasyfikacja, dobrze jest obejrzeć pewne cechy rozpatrywanych kategorii. Pokazano je na Rys. 12.

Tabela 5. Charakterystyka 25 najczęstszych rdzeni, źródło: opracowanie własne

Lp.	*rdzeń*	*ilość wystąpień w kolekcji*	*w ilu dokumentach kolekcji występuje*	*średnia ilość wystąpień (w dokumentach w których występuje)*	*średnia ilość wystąpień (we wszystkich dokumentach kolekcji)*	*maksymalna ilość wystąpień w jednym dokumencie*
1	said	25343	6791	3,7	2,4	24
2	mln	18415	4850	3,8	1,7	45
3	dlrs	12175	4204	2,9	1,1	47
4	reuter	10314	9755	1,1	1,0	7
5	pct	9567	3220	3,0	0,9	33
6	cts	8181	3075	2,7	0,8	14
7	year	7624	3685	2,1	0,7	20
8	net	6958	2883	2,4	0,6	10
9	will	6017	2860	2,1	0,6	14
10	share	5733	2320	2,5	0,5	24
11	billion	5715	1686	3,4	0,5	35
12	compani	5504	2819	2,0	0,5	18
13	loss	5499	1534	3,6	0,5	17

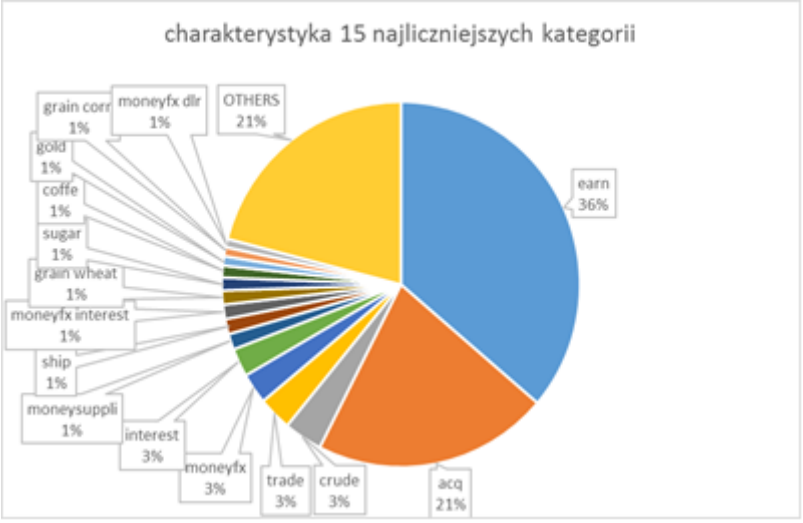
Lp.	*rdzeń*	*ilość wystąpień w kolekcji*	*w ilu dokumentach kolekcji występuje*	*średnia ilość wystąpień (w dokumentach w których występuje)*	*średnia ilość wystąpień (we wszystkich dokumentach kolekcji)*	*maksymalna ilość wystąpień w jednym dokumencie*
14	earn	5255	4162	1,3	0,5	10
15	bank	4904	1571	3,1	0,5	38
16	trade	4452	1539	2,9	0,4	29
17	price	4100	1663	2,5	0,4	21
18	shr	4054	2347	1,7	0,4	7
19	inc	4050	3104	1,3	0,4	7
20	market	3951	1844	2,1	0,4	16
21	profit	3510	1271	2,8	0,3	10
22	oil	3337	1064	3,1	0,3	24
23	corp	3324	2562	1,3	0,3	9
24	sale	3249	1790	1,8	0,3	24
25	oper	3246	1479	2,2	0,3	12

Wybrane metody klasyfikacji i grupowania oraz ich założenia

Do przeprowadzenie badań klasyfikacyjnych wybrano dwie metody: algorytm k-NN oraz naiwny klasyfikator Bayesa.

Ponieważ w przypadku algorytmów klasyfikacji mamy z góry narzucone kategorie dokumentów przyjrzyjmy się ich charakterystyce, pokazanej na Rys. 12.

W związku z takim podziałem kategorii, a tym samym praktycznym brakiem możliwości nauczenia klasyfikatora na pojedynczych przykładach, badania przeprowadzono dla 15 i dla 8 najliczniejszych kategorii. Ze zbioru wejściowego zostały usunięte wszystkie wiersze odpowiadające dokumentom z którejś spoza 15 lub 8 najliczniejszych kategorii. Liczność nowych zbiorów wejściowych przedstawia Tabela 7. Ten zbiór kategorii nie jest jednak idealny – najlepiej byłoby, aby wszystkie klasy były równoliczne, a tu dwie kategorie są znacznie liczniejsze od pozostałych razem wziętych. Podobnie, ograniczono liczbę rozpatrywanych słów (rdzeni) do 500 najliczniej występujących.



Rys. 12. Charakterystyka 15 najliczniejszych kategorii, źródło: opracowanie własne

Jeśli chodzi o podział zbioru danych na zbiór treningowy i testowy, to został on również wykonany przy pomocy poleceń języka R. Liczebności obu zbiorów pokazano także w Tabeli 7.

Tabela 7. Liczność zbiorów wejściowych do klasyfikacji, źródło: opracowanie własne

	dla 15 kategorii
zbiór treningowy	6 167
zbiór testowy	2 346
RAZEM	8 513

Do przeprowadzenia badań dotyczących grupowania, a więc bez z góry zadanych kategorii, zostały także wybrane dwie metody (algorytm agregacji hierarchicznej, faktycznie w kilku wariantach, oraz algorytm k-średnich, także w kilku wariantach).

Przeliczenia dla klasyfikacji; metoda k-NN

Podobnie jak i dla innych metod i eksperymentów, przytoczymy tylko wyrywkowe wyniki, ponieważ ich ogólna objętość wynosi kilkadziesiąt stron. Ponieważ chodziło głównie o zależność od długości indeksu (tutaj: n), więc pokażemy kilka wyników, odnoszących się do tej kwestii (badane były także zależności od

innych parametrów, w tym przypadku np. liczby uwzględnianych sąsiadów). Tabela 8 zawiera pierwsza ilustrację, z której wynika, że im dłuższy indeks, tym mniejszy błąd klasyfikacji.

Tabela 8. Wyniki klasyfikacji 5-NN: poprawność rozpoznawania klas, źródło: opracowanie własne

	klasy rozpoznane poprawnie	**klasy rozpoznane błędnie**	**% błędnie rozpoznanych**
n=5	1581	767	33%
n=20	1883	465	20%
n=50	2053	295	13%
n=100	2108	240	10%
n=500	2179	169	7%

Trzeba jednak zauważyć, że otrzymane wyniki nie są bynajmniej nadzwyczajne, jakkolwiek zapewne nawet dla n = 20 zadowoliłyby przeciętnego twórcę wyszukiwarki – nakład obliczeniowy wzrasta bowiem bardzo szybko wraz z n. Wyniki wyglądają jednak nieco inaczej, jeśli wydłużymy indeks jeszcze bardziej, co widać w Tabeli 9. Teraz widać, że istnieje (hipotetyczne) optimum powyżej 200 i poniżej 500. Natomiast inna ciekawa zależność otrzymana w tym badaniu jest pokazana w Tabeli 10.

Tabela 9. Wyniki klasyfikacji k-NN, poprawność rozpoznawania klas, źródło: opracowanie własne

	klasy rozpoznane poprawnie	**klasy rozpoznane błędnie**	**% błędnie rozpoznanych**
n=100	2100	248	10,6%
n=200	2157	191	8,1%
n=250	2191	157	6,7%
n=300	2189	159	6,8%
n=500	2177	171	7,3%
n=1000	2176	172	7,3%
n=2000	2150	198	8,4%
n=4000	2150	198	8,4%

Tabela 10. Wyniki klasyfikacji k-NN, n=500-poprawność rozpoznawania klas, źródło: opracowanie własne

	klasy rozpoznane poprawnie	**klasy rozpoznane błędnie**	**% błędnie rozpoznanych**
k=1	2181	167	7,1%
k=3	2177	171	7,3%
k=5	2179	169	7,2%
k=7	2172	176	7,5%
k=15	2143	205	8,7%

Tak więc – im więcej sąsiadów bierzemy pod uwagę, tym gorsze wyniki uzyskujemy. Wynika to zapewne z faktu, ze wiele kategorii zawiera mało obiektów i przy większej liczbie sąsiadów bierzemy już często pod uwagę obiekty (dokumenty) z wielu różnych kategorii.

Przeliczenia dla klasyfikacji; metoda naiwna Bayesa

Jak widać z Tabeli 11, wyniki dla naiwnego klasyfikatora Bayesa były jeszcze (znacznie) gorsze niż dla metody k-NN. Wynikało to, znów, z bardzo zróżnicowanych liczności kategorii i słabo uwarunkowanych prawdopodobieństw dla wielu z nich. Tym niemniej, równie łatwo można zauważyć, że najlepsze wyniki osiągnięto dla długości indeksu wyraźnie poniżej maksymalnego $n = 500$.

Tabela 11. Wyniki klasyfikacji NB - poprawność rozpoznawania klas, źródło: opracowanie własne

	klasy rozpoznane poprawnie	**klasy rozpoznane błędnie**	**% błędnie rozpoznanych**
n=5	993	1355	57,7%
n=20	1263	1085	46,2%
n=50	1397	951	40,5%
n=100	1513	835	35,6%
n=500	842	1506	64,1%

Z kolei w Tabeli 12 pokazano wyniki próby dokładniejszego określenia położenia hipotetycznego optimum dla wartości n. Jak widać, położenie takiego optimum, nawet dla konkretnego zbioru danych i metody, nie jest jednoznaczne (przy określonej liczbie przeliczeń i określonych założeniach), a poza tym jest zależne od wielu czynników. W tym przypadku wydaje się, że leży ono między $n = 80$ a 90.

Komentując wyniki dla metod klasyfikacji, dodajmy, że co prawda błędy dla naiwnego klasyfikatora Bayesa były większe (średnio), ale sprawdzał się on dobrze dla klas największych, a poza tym jest znacznie sprawniejszy obliczeniowo od klasyfikatora k-NN.

Tabela 12. Wyniki klasyfikacji NB - poprawność rozpoznawania klas 2, źródło: opracowanie własne

	klasy rozpoznane poprawnie	**klasy rozpoznane błędnie**	**% błędnie rozpoznanych**
n=50	1397	951	40,5%
n=75	1576	772	32,9%
n=80	1570	778	33,1%
n=85	1604	744	31,7%
n=90	1572	776	33,0%
n=100	1572	776	33,0%
n=125	1405	943	40,2%
n=150	1310	1038	44,2%

Wyniki dla metod grupowania: agregacja hierarchiczna

W odniesieniu do algorytmów agregacji hierarchicznej należy podkreślić, że w istocie nie dostarczają one wprost grup, lecz tylko podstawę (dendrogram) do ich wyznaczenia. Poza tym, istnieje wiele algorytmów agregacji hierarchicznej, różniących się mniej lub bardziej od siebie. Konieczne więc było uczynienie szeregu założeń przy prowadzeniu eksperymentu.

Po pierwsze, przebadano następujące warianty agregacji hierarchicznej: najbliższego sąsiedztwa (single link – SL), najdalszego sąsiedztwa (complete link – CL), mediany (M), średniego wiązania (S), centroidu (C), Warda (W) oraz McQuitty (MC). W przypadku wszystkich tych wariantów stosowano odległość euklidesową.

Po drugie, do oceny otrzymanych podziałów zbioru dokumentów użyto kilku kryteriów oceny podziałów, a mianowicie:

– *index dunn*, tj. czyli minimum odległości między dokumentami z różnych grup dzielone przez maksimum odległości między dokumentami

w grupie; im wyższa wartość indeksu tym grupowanie powinno być „lepsze";

- *miara entropii dla rozkładu grupowania* – im wyższa wartość indeksu tym grupowanie powinno być „lepsze";
- *wb.ratio*, średnia średnich odległości między dokumentami w tej samej grupie podzielona przez średnią średnich odległości między dokumentami z różnych grup; im niższa wartość indeksu tym grupowanie „lepsze";
- *ch.index* (Calinski and Harabasz index), kryterium bazujące na wariancjach - im wyższa wartość indeksu tym grupowanie powinno być „lepsze".

Istotnym parametrem przeliczeń, poza, naturalnie, długością indeksu, była także liczba skupień. Przedstawiamy tutaj wyniki dla 15 skupień, jako odpowiadające liczbie rozpatrywanych kategorii (Tabela 13).

Na podstawie zawartości Tabeli 13 można wysnuć szereg istotnych wniosków, w tym zwłaszcza o charakterze znacznie przekraczającym zakres tematyczny pracy.

Przede wszystkim widać, że wykorzystane – znane z literatury – kryteria nie są spójne (gdyby były, nie byłoby powodu projektować nowych!). Nie dają ona, poza tym, jednoznacznych wskazań – otrzymane ciągi wartości z reguły nie są monotoniczne. Należy się zatem kierować także oceną z poziomu „meta", czyli – biorąc pod uwagę sens poszczególnych kryteriów. Po drugie, co już sygnowaliśmy, różne algorytmy generują wyraźnie różne wyniki, w tym i silnie różniące się ogólnym charakterem (np. liczności grup), co zresztą było także analizowane w pracy. Tabela 14 pokazuje wartości i zakresy długości indeksu, *n*, hipotetycznie optymalne dla poszczególnych algorytmów i kryteriów, zgodnie z zawartością Tabeli 13.

Tabela 14, z kolei, stanowi próbę konstruktywnego odczytania wyników z Tabeli 13 w znanej sytuacji, gdy zarówno kryteria, jak i rozpatrywane metody są obciążone, tj. mają określone tendencje, jeśli chodzi o ocenę i tworzenie podziałów. Widać to wyraźnie w Tabeli 14. Jeśli zatem zmniejszymy wagi wskazań, wynikających z przypuszczalnego obciążenia (w kierunku maksimum długości indeksu dla indeksu Dunna, w kierunku minimum dla pozostałych), to staje się oczywistym, że istnieje pewne optimum – zapewne różne w różnych warunkach – długości indeksu ze względu na jakość grupowania przy pomocy algorytmów agregacji hierarchicznej.

Tabela 13. Grupowanie algorytmami agregacji hierarchicznej; porównanie wg długości indeksu, źródło: opracowanie własne

metoda	długość indeksu								
	5	**20**	**50**	**75**	**100**	**125**	**150**	**200**	**500**
DUNN INDEX									
CL	0,019	0,031	0,100	0,123	0,094	0,101	0,118	0,142	0,155
SL	0,088	0,180	0,248	0,316	0,314	0,310	0,335	0,326	0,420
S	0,025	0,119	0,117	0,172	0,175	0,172	0,211	0,290	0,349
M	0,084	0,030	0,149	0,225	0,229	0,225	0,187	0,194	0,356
C	0,015	0,133	0,145	0,188	0,232	0,281	0,251	0,319	0,393
W	0,002	0,006	0,032	0,059	0,041	0,057	0,061	0,074	0,046
MC	0,007	0,058	0,040	0,058	0,114	0,044	0,124	0,126	0,223
ENTROPIA									
CL	1,22	1,38	1,29	0,86	0,89	0,81	0,50	0,32	0,26
SL	0,09	0,05	0,05	0,05	0,06	0,05	0,05	0,06	0,04
S	0,99	0,43	0,40	0,12	0,17	0,15	0,12	0,10	0,07
M	0,32	0,37	0,06	0,05	0,05	0,05	0,05	0,06	0,04
C	0,79	0,15	0,05	0,06	0,06	0,06	0,07	0,07	0,04
W	2,00	2,26	2,23	2,19	2,15	2,07	2,13	2,14	2,12
MC	1,13	1,04	0,99	0,94	0,40	0,37	0,23	0,21	0,15
WB.RATIO									
CL	0,14	0,33	0,55	0,59	0,62	0,66	0,68	0,62	0,61
SL	0,33	0,50	0,55	0,55	0,43	0,45	0,47	0,46	0,48
S	0,15	0,42	0,55	0,53	0,49	0,49	0,48	0,45	0,45
M	0,29	0,53	0,51	0,56	0,44	0,46	0,48	0,46	0,45
C	0,19	0,39	0,52	0,51	0,41	0,43	0,44	0,43	0,44
W	0,06	0,35	0,56	0,61	0,62	0,68	0,68	0,70	0,80
MC	0,15	0,35	0,68	0,66	0,60	0,65	0,59	0,59	0,55
CH.INDEX									
CL	1116	340	163	88	80	74	40	30	22
SL	25	5	5	6	13	10	10	11	8
S	894	107	56	18	31	27	22	20	13
M	156	44	9	6	10	10	9	12	8

metoda	długość indeksu								
	5	**20**	**50**	**75**	**100**	**125**	**150**	**200**	**500**
DUNN INDEX									
C	430	41	7	9	16	15	14	17	9
W	1658	420	210	170	140	121	102	82	47
MC	839	290	92	70	44	33	27	24	16

Tabela 14. Długości indeksu, optymalizujące wartości kryteriów, dla poszczególnych algorytmów agregacji hierarchicznej. W nawiasach optima lokalne; źródło: opracowanie własne

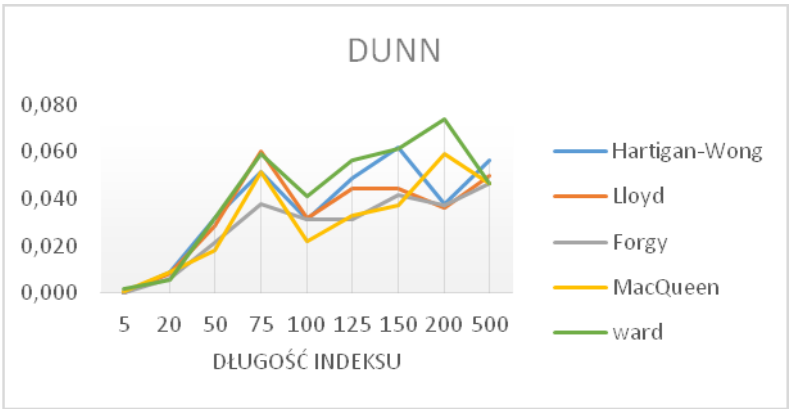
Metoda	**Kryterium**			
	Dunn	**Entropia**	**WB Ratio**	**CH Index**
CL	500 (75)	20 (100)	5 (500)	5
SL	500 (75,150)	5 (100, 200)	5 (100, 200)	5 (100, 200)
S	500 (20, 100)	5 (100)	5 (200, 500)	5 (100)
M	500 (5, 100)	20	5 (50, 100, 500)	5 (100-125, 200)
C	500 (125)	5 (150, 200)	5 (100, 200)	5 (100, 200)
W	200 (75)	20 (200)	5	5
MC	500 (20, 100)	5	5 (100, 500)	5

Wyniki dla metod grupowania: k średnich

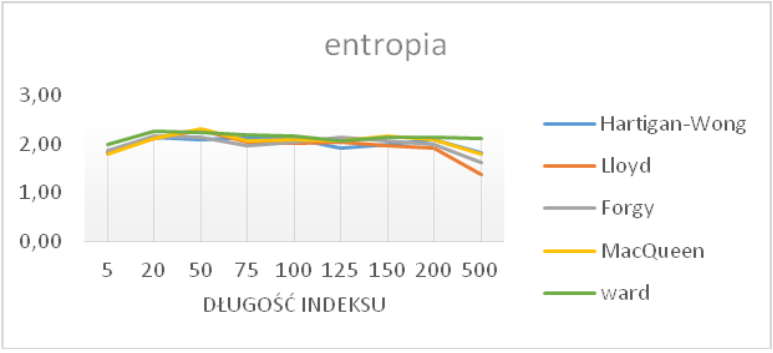
Podobnie, jak w przypadku algorytmów agregacji hierarchicznej, także w przypadku metody k-średnich testowano kilka wariantów tej metody, a mianowicie algorytmy: Hartigana-Wonga, Lloyda, Forgy'ego i MacQueena (tj. wersje oryginalną). Dodatkowo, na pokazanych diagramach uwzględniono, dla porównania, wyniki algorytmu Warda agregacji hierarchicznej, którego wyniki były wśród najbardziej akceptowalnych.

Na Rys. 13a, b, c, d widzimy znów zróżnicowanie ocen według kryteriów i brak ich monotoniczności, a także wyraźne obciążenie tendencjami.

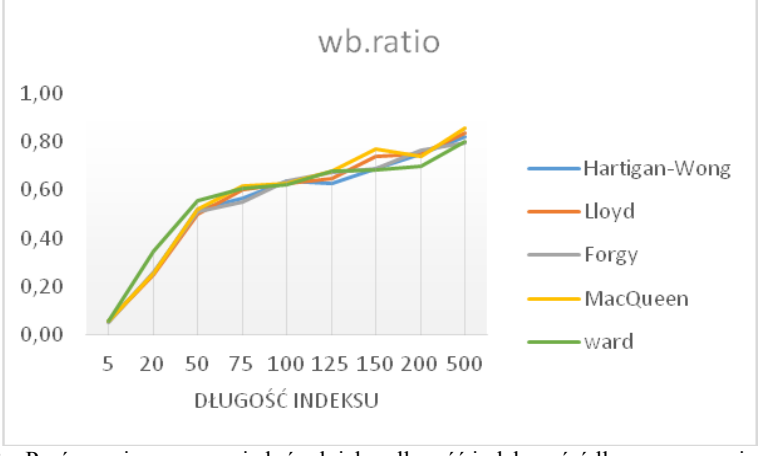
Tym niemniej, podobnie, jak w poprzednim przypadku, możliwe jest wyciągnięcie konstruktywnych wniosków. Po pierwsze, metody z grupy k-średnich, zastosowane tutaj, dają znacznie bardziej spójne wyniki niż algorytmy agregacji hierarchicznej. Widać to zwłaszcza dla kryterium Calińskiego-Harabasza, ale także dla entropii i współczynnika rb. Po drugie, i to jest tutaj najważniejsze, widać wyraźnie, że hipotetyczne optimum występuje z pewnością znacznie poniżej najwyższej tutaj rozpatrywanej długości indeksu, tj. $n = 500$. Już długość $n = 20$ jest w tej mierze całkiem dopuszczalną hipotezą, a jest dość pewne, że nie przekracza ona $n = 100$-150.



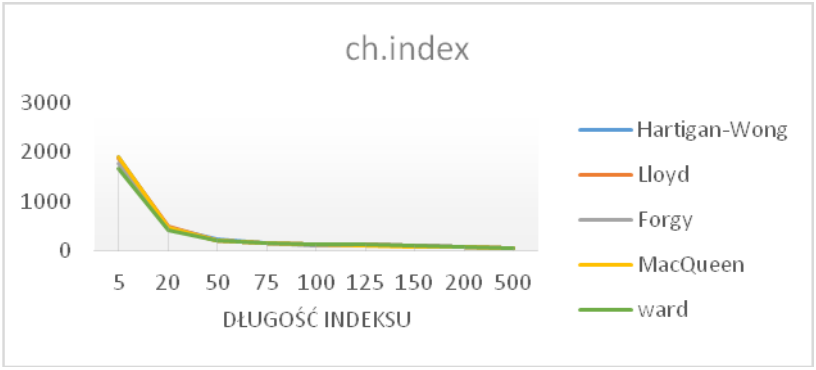
Rys. 13a. Porównanie grupowania k-średnich – długość indeksu; źródło: opracowanie własne



Rys. 13b. Porównanie grupowania k-średnich – długość indeksu; źródło: opracowanie własne



Rys. 13c. Porównanie grupowania k-średnich – długość indeksu; źródło: opracowanie własne



Rys. 13d. Porównanie grupowania k-średnich – długość indeksu; źródło: opracowanie własne

14. Podsumowanie

Zrelacjonowane tutaj, na tle zarysu całej dziedziny wyszukiwania informacji tekstowej, wstępne badanie, poświęcone zagadnieniu istnienia optymalnej długości indeksu dokumentu, dało niewątpliwie obiecujące wyniki. Po pierwsze, stworzony został, przez pierwszego z autorów, instrument do prowadzenia odpowiednich badań. Po drugie, potwierdzona została teza o możliwości istnienia takiej hipotetycznej optymalnej długości indeksu, i wreszcie – zostały wykonane przeliczenia, które mogą stanowić podstawę do dalszych prac, zmierzających do bardziej precyzyjnego wyznaczenia optymalnej długości indeksu, lub przynajmniej procedur jej wyznaczania dla konkretnych warunków (zbiorów dokumentów, metod i parametrów wyszukiwania).

Literatura[11]

Andrzejewski W., Królikowski Z., Morzy T. (bez daty) Bazy danych i systemy informatyczne oraz ich wpływ na rozwój informatyki w Polsce. WWW.fundacjarozwojunauki.pl/res/Tom2/10_Morzy.pdf

Kłopotek M.A. (2001) *Inteligentne wyszukiwarki internetowe*. Akademicka Oficyna Wydawnicza Exit, Warszawa.

Owsiński J. W. (2014) *Wprowadzenie do wyszukiwania informacji tekstowych: modele, techniki, zasadnicze zagadnienia*. WIT, Warszawa.

Ryczaj W. (2014) Badanie parametrów klasyfikacji i grupowania wiadomości tekstowych ze względu na długość tworzonego indeksu. Praca magisterska, WIT, Warszawa.

[11] Odwołania do stron internetowych zamieszczone są w przypisach dolnych w tekście.

THE MORE THE BETTER? AN ANALYSIS FROM THE INFORMATION RETRIEVAL DOMAIN

Abstract: The paper is based on the master thesis of the first author, prepared under the supervision of the second. It deals with the question of the potential optimum document index length. Against the background of the information retrieval field, the experiment is presented serving to verify the hypothetical existence of such an optimum. Even though the study is a very preliminary attempt, it appears that the hypothesis was positively verified and a number of constructive conclusions could be drawn, indicating also the future directions of investigations.