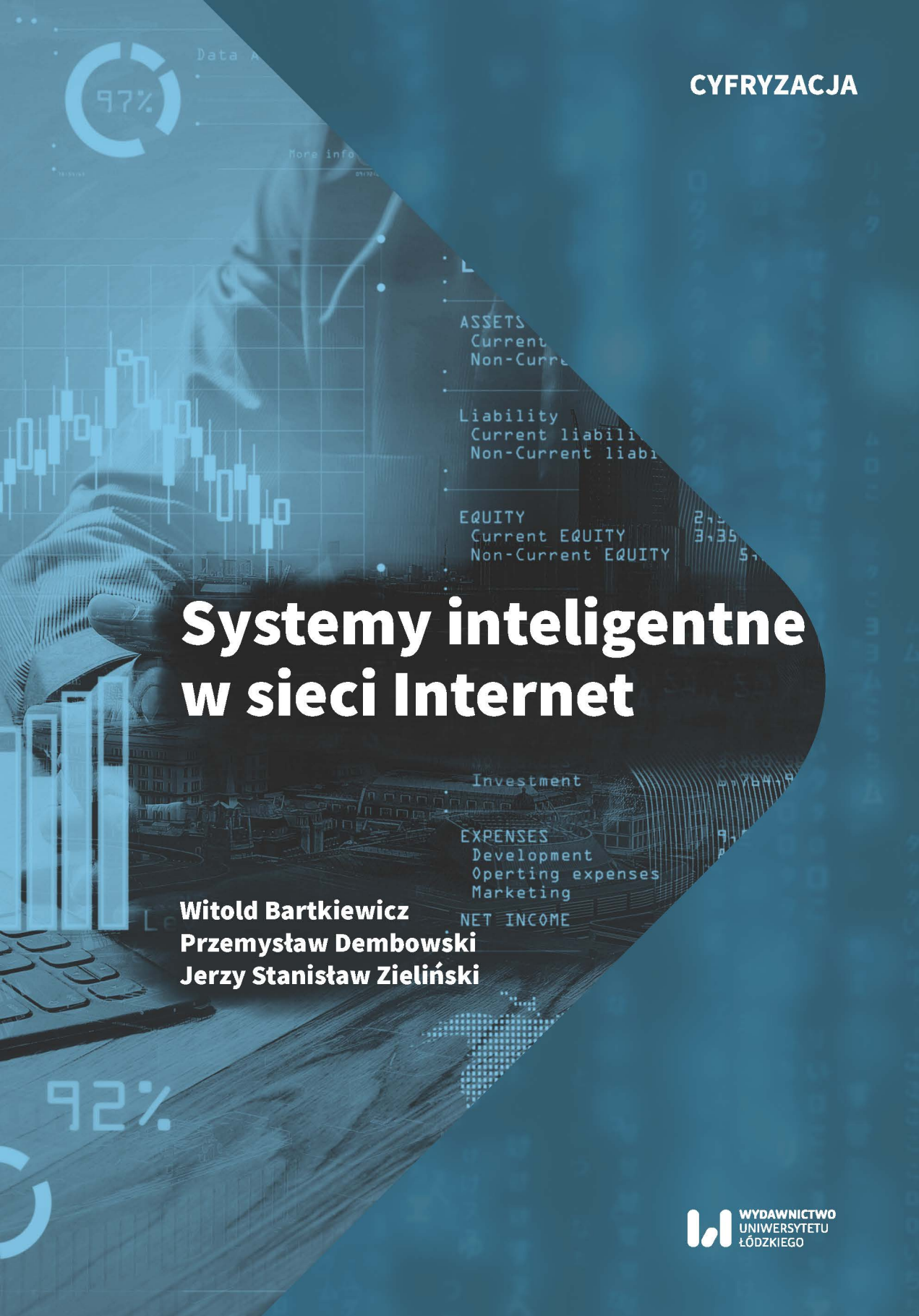
CYFRYZACJA



Systemy inteligentne w sieci Internet

Witold Bartkiewicz
Przemysław Dembowski
Jerzy Stanisław Zieliński

WYDAWNICTWO UNIWERSYTETU ŁÓDZKIEGO

Systemy inteligentne w sieci Internet

WYDAWNICTWO
UNIWERSYTETU
ŁÓDZKIEGO

CYFRYZACJA

Systemy inteligentne w sieci Internet

Witold Bartkiewicz
Przemysław Dembowski
Jerzy Stanisław Zieliński

WYDAWNICTWO
UNIWERSYTETU
ŁÓDZKIEGO
Łódź–Kraków 2020

Witold Bartkiewicz, Przemysław Dembowski, Jerzy Stanisław Zieliński
– Uniwersytet Łódzki, Wydział Zarządzania, Katedra Informatyki
90-237 Łódź, ul. J. Matejki 22/26

RECENZENCI
Ryszard Tadeusiewicz
Zdzisław Szyjewski

REDAKTOR INICJUJĄCY
Beata Koźniewska

OPRACOWANIE REDAKCYJNE
Mateusz Malicki

SKŁAD I ŁAMANIE
AGENT PR

PROJEKT OKŁADKI
AGENT PR
Anna Basista

Na okładce wykorzystano grafikę
© Depositphotos.com/grandfailure

Wydane przez Wydawnictwo Uniwersytetu Łódzkiego
Wydanie I. W.09588.19.0.K

Ark. wyd. 12,5; ark. druk. 13,625

ISBN 978-83-8220-353-0
e-ISBN 978-83-8220-354-7
ISBN AGENT PR 978-83-64462-67-2

https://doi.org/10.18778/8220-353-0

Wydawnictwo Uniwersytetu Łódzkiego
90-131 Łódź, ul. Lindleya 8
www.wydawnictwo.uni.lodz.pl
e-mail: ksiegarnia@uni.lodz.pl
tel. 42 665 58 63

Spis treści

Wprowadzenie

Internet jest jednym z największych i najważniejszych zasobów informacyjnych we współczesnym świecie. Dostarcza on informacji zarówno dla organizacji, oferując im olbrzymie możliwości budowania przewagi konkurencyjnej na rynku, jak również dla klientów i indywidualnych użytkowników. Należy jednak zwrócić uwagę, że wykorzystanie zasobów dostępnych za pośrednictwem Internetu nie jest zadaniem łatwym.

Źródła informacji w sieci rozproszone są na miliardach serwerów, co samo w sobie powoduje już ogromne problemy. Brak jest przy tym koordynacji i uporządkowania zarówno samej struktury sieci, jak i jej zawartości. Internet podlega nieustannemu wzrostowi i ma charakter bardzo dynamiczny, tempo przyrostu rozmiarów zasobów informacyjnych jest bardzo wysokie.

Informacje w Internecie mają różnorodny charakter. W większości są to jednak niestrukturalne dokumenty (tekst, strony HTML, dokumenty multimedialne). Nie posiadają one ściśle zdefiniowanej, struktury informacyjnej, określanej często jako schemat informacji, a przynajmniej tego rodzaju struktura zazwyczaj nie jest znana korzystającym z nich użytkownikom.

Niniejsza książka poświęcona jest zagadnieniom wspomagania wykorzystania zasobów informacyjnych dostępnych w sieci. Rozmiary Internetu oraz znajdujących się w nim źródeł informacji powodują, że dla efektywnego korzystania z nich, niezbędne jest zastosowanie metod automatycznego przetwarzania treści zawartych w sieci. Problemy, związane w znacznej mierze z niestrukturalnym charakterem dostępnych w Internecie informacji, powodują, że obecny poziom rozwoju tego typu oprogramowania jest daleki od satysfakcjonującego – wystarczy tu choćby wspomnieć kwestie funkcjonowania wyszukiwarek internetowych.

Typowe dane strukturalne (jak np. w relacyjnych bazach danych, plikach rekordów itp.) mają ściśle zdefiniowaną strukturę semantyczną. W tego rodzaju źródłach zasoby informacyjne podzielone są na określone elementy o znanym, jednorodnym znaczeniu. Semantyka każdego elementu danych jest określona, znany jest również schemat zależności znaczeniowych w całej strukturze. Wiadomo, czego dotyczy każdy jej element – przykładowo, wiemy, że w relacyjnej bazie w tabeli o nazwie Studenci w kolumnie Imię znajdują się imiona studentów, a nie, powiedzmy, nazwy tradycyjnych potraw regionalnych.

Oczywiście w Internecie znajdziemy również źródła strukturalne – głównie udostępniane bazy danych. W wielu jednak przypadkach zawarte w nich dane dostępne są za pośrednictwem niestrukturalnych stron HTML.

Dokument zaś to zwykle duży zasób różnorodnych informacji, pozbawiony uporządkowanego podziału na jednorodne semantycznie, proste elementy danych. Wykorzystanie informacji niestrukturalnej wymaga czegoś w rodzaju „zrozumienia" treści dokumentu, co jest zadaniem wykonalnym dla ludzi, lecz ogromnym wyzwaniem dla systemów informatycznych. Ponadto w przypadku najczęstszej formy informacji niestrukturalnej – tekstów – ich treść zapisana jest w języku naturalnym, który ma charakter nieprecyzyjny. Opis informacji na poziomie leksykalnym często jest wieloznaczny, znaczenie słów wymaga uwzględnienia ich kontekstu. Dla prawdziwego zrozumienia dokumentu w wielu przypadkach niezbędna jest również pewna wiedza z danego obszaru, którego dotyczy jego zawartość.

Wszystkie te kwestie powodują, że usprawnienia funkcjonowania systemów informatycznych wspomagających wykorzystanie zasobów internetowych, szuka się na gruncie sztucznej inteligencji, oczywiście w połączeniu z elementami analizy języka naturalnego, lingwistyki obliczeniowej i szeregu pokrewnych dziedzin. Zrozumienie zawartości dokumentu, niezbędne dla lepszego jego przetwarzania, wymaga od systemu informatycznego cech związanych z inteligentnym jego działaniem.

Tak ogromny zbiór problemów związanych z narastającą ilością informacji i metod przetwarzania jej wpływa na sposób przedstawienia problemów w rozdziale pierwszym, umożliwiający Czytelnikom zebranie ogólnej informacji o problemie, a w razie zainteresowania jakimś tematem, ułatwienie znalezienia odpowiedniej publikacji. Dla zobrazowania nakładu pracy potrzebnej dla rozwiązania rozważanych zagadnień drugi rozdział przedstawia obszernie inteligentne usługi informacyjne, zaś trzeci rozdział skrótowo zarysowuje nowe technologie w sieci Internet.

Pierwszy rozdział rozpoczyna się od opisu rozwoju hierarchicznych informacyjnych systemów zarządzania (ISZ) do systemów rozproszonych i do zintegrowanych systemów informacyjnych najnowszych generacji stosujących narzędzia sztucznej inteligencji (*Artificial Intelligence – AI*) takich jak na przykład Systemy Ekspertowe, Sztuczne Sieci Neuronowe, Algorytmy Ewolucyjne, Systemy Hybrydowe, Inteligencja Komputerowa i szereg innych. Przedstawiono nowe narzędzia informatyczne: Internet Rzeczy (*Internet of Things – IoT*) Internet Wszystkiego (*Internet of Everything – IoE*) Chmurę Obliczeniową (*Cloud Computing – CC*). Podano

przykłady badań w obszarze sztucznej inteligencji we Francji i w Polsce, omówiono poglądy na temat zagrożeń związanych z zastosowaniem sztucznej inteligencji.

Najobszerniejszą częścią niniejszej książki jest rozdział drugi poświęcony inteligentnym metodom przetwarzania informacji niestrukturalnej. Koncentrujemy się tutaj na metodach analizy tekstu, z wykorzystaniem słów kluczowych (tzw. termów indeksujących). Rozpoczynamy od przeglądu tradycyjnych podejść do wyszukiwania informacji tekstowej – modelu boolowskiego, wektorowego oraz probabilistycznego, co pozwoli nam się przyjrzeć odmiennym spojrzeniom na możliwe sposoby analizy tekstów.

Dalej przechodzimy do wykorzystania metod inteligentnych do ulepszenia działania samego modelu wyszukiwawczego, tj. przede wszystkim sposobu oceny dopasowania dokumentu do zapytania. Przyjrzymy się różnorodnym rozwiązaniom, wypływającym ze wszystkich trzech podstawowych paradygmatów wyszukiwania informacji, tj. logicznego, wektorowego (algebraicznego) oraz probabilistycznego.

Rozpoczynamy od zagadnień logiki rozmytej, oraz wykorzystania właściwości zbiorów rozmytych do wyrażania nieprecyzji właściwej naszemu aparatowi lingwistyczno-pojęciowemu, w celu łatwiejszego, bardziej precyzyjnego formułowania zapytań. Logika rozmyta stanowi w tym przypadku narzędzie pozwalające na rozszerzenie paradygmatu boolowskiego modelu wyszukiwania informacji. Następnie przechodzimy do skojarzeniowego wyszukiwania informacji, z zastosowaniem różnego rodzaju sieci pojęć: semantycznych, konekcjonistycznych (neuronowych o lokalnej reprezentacji), oraz wnioskujących (bayesowskich). Umożliwiają one polepszenie jakości etapu dopasowywania zapytań i dokumentów, w modelach wektorowym i probabilistycznym, poprzez wykorzystanie do tego celu wiedzy dziedzinowej, wbudowanej w struktury zastosowanej sieci.

Rozdział drugi zakończymy przeglądem wykorzystania metod uczenia maszynowego do nauki bardziej złożonych funkcji rankingujących dokumenty, wychodzących poza proste miary podobieństwa i uczenie polegające na zebraniu podstawowych statystyk wykorzystania słów kluczowych w kolekcji. Do tego zadania przede wszystkim stosowane są różnego rodzaju sieci neuronowe, o globalnej, subsymbolicznej reprezentacji wiedzy.

W naszej książce koncentrujemy się przede wszystkim na metodach inteligentnych wspomagających dostęp do informacji w Internecie. W rozdziale trzecim przedstawiamy jednak krótką charakterystykę wybranych

innych zastosowań systemów inteligentnych w sieci. Omówione zostały zagadnienia związane z inteligentnymi interfejsami użytkownika w Internecie, robotami softwareowymi, czy też podejściem opartym na Semantic Web.

Autorzy mają nadzieję, że książka odpowie na przynajmniej jedno z poniższych pytań:

- Co to jest ta informatyka?
- Czy informatyka oferuje społeczeństwu narzędzia radzenia sobie z ogromnym zalewem informacji we współczesnym świecie?
- Co to jest sztuczna inteligencja i czy jest ona niebezpieczna?
- Czy rozwój systemów sztucznej inteligencji jest niezbędny, nawet pomimo problemów i zagrożeń, jakie mogą wiązać się z nimi w przyszłości?
- Czy potrzebne są duże nakłady na badania i zastosowania informatyki?

Jeśli udało się udzielić takich odpowiedzi, to ta książka może i powinna zainteresować studentów uczelni humanistycznych, jak i technicznych z kierunków nieinformatycznych, i ogólnie ludzi ciekawych świata.

Rozdział 1

Wybrane problemy zarządzania w społeczeństwie informacyjnym

1.1. Zmiany w organizacji zarządzania

Rewolucja przemysłowa, która rozpoczęła się, w XVIII wieku w Anglii i Szkocji zainicjowała powstawanie nowych miejsc produkcji zlokalizowanych w pobliżu źródeł energii i wody, powodując konieczność zorganizowania ich pracy w sposób umożliwiający działanie i rozwój. Początkowo przedsiębiorstwa były zlokalizowane w jednym miejscu, ale w miarę rozwoju postępu technicznego i gospodarki, rozrastały się terytorialnie na sąsiednie rejony, kraje ,i kontynenty. Tak więc początkowo hierarchiczny proces zarządzania skupiony w jednym miejscu, musiał stać się systemem rozproszonym.

Proces ten można przedstawić na przykładzie rozwoju systemu elektroenergetycznego [Ciach, Czerwonka i in., 2019] którego podstawowe elementy (źródła prądu elektrycznego, linia przesyłowa, odbiornik) powstały w końcu XIX wieku, kiedy to w większych miastach zastosowano elektryczne oświetlenie, zasilane z małych prądnic. Dostarczenie coraz popularniejszej energii elektrycznej wymagało budowania elektrowni o odpowiedniej mocy, które powstawały w dużych miastach na początku XX wieku (w Łodzi w 1907 roku uruchomiono elektrownię EC1, w której źródłem energii pierwotnej był węgiel dowożony ze Śląska). W niektórych krajach posiadających odpowiednie warunki hydrologiczne (np. wodospady jak Norwegia i inne państwa) powstawały elektrownie wodne. Niezależnie od rodzaju źródła, droga przepływu prądu biegła od źródła przez linię elektroenergetyczną do odbiorcy, a więc była to droga charakterystyczna dla tak zwanego monopolu naturalnego zarządzanego w układzie hierarchicznym.

W drugiej połowie XX wieku zaczęły powstawać elektrownie wiatrowe w miejscach do których ze względów ekonomicznych lub technicznych nie opłacało się dostarczać energii elektrycznej z oddalonych źródeł (np. wyspy na morzu Śródziemnym). Proces ten rozwinął się w wyniku deficytu źródeł energii pierwotnej pochodzących z paliw kopalnych (węgiel, ropa naftowa) powodujących brak energii elektrycznej. Koniecznością stało się dołączanie lokalnych źródeł tak zwanej energii odnawialnej (OZE) powodujące rozproszenie źródeł energii, a zatem powstanie rozproszonego systemu zarządzania energią elektryczną związanego z nową organizacją i oprzyrządowaniem sieci elektroenergetycznej nazwanej Smart Grid [Ciach, Czerwonka i in., 2019].

Zarządzanie organizacją gospodarczą związane jest z przyjmowaniem zbiorów danych, przechowywaniem danych przetworzonych do postaci potrzebnej w przedsiębiorstwie, generowanie nowych danych emitowanie ich do kontrahentów. Wszystkie te działania związane z danymi, można określić ogólnym mianem: przetwarzanie [Bartkiewicz, Jabłoński, 2005]. Ze względu na wzrastającą ilość danych niezbędną staje się automatyzacja przetwarzania w stopniu wynikającym z potrzeb, i zasobności przedsiębiorstwa oraz dostępności odpowiednich urządzeń na rynku.

1.2. Wpływ komputeryzacji na zarządzanie Systemy informatyczne zarządzania

Pierwsza publiczna prezentacja elektronicznego (lampowego) komputera ENIAC (*Electronic Numerical Integrator And Calculator*) w 1946 roku wzbudziła ogromne nadzieje na zastąpienie człowieka w wykonywaniu skomplikowanych obliczeń, ale dopiero komputery tranzystorowe, a zwłaszcza na układach scalonych, umożliwiły praktyczne przetwarzanie danych dla potrzeb zarządzania. Możliwości obliczeniowe komputerów budowanych w latach czterdziestych ub. wieku były ograniczone do czterech działań arytmetycznych, co sprzyjało budowaniu systemów informatycznych zarządzania (SIZ), nazwanych systemami transakcyjnymi (*Transaction Processing System – TPS*) [Zieliński, 2018; Bartkiewicz, Bolek i in., 2008; Kisielnicki, 2013]. Znaczące postępy w stopniu integracji układów scalonych ułatwiały konstruowanie coraz wydajniejszych komputerów, a zatem i tworzenie nowych języków programowania [Bartkiewicz, Jabłoński, 2005] niezbędnych w budowie nowych systemów informatycznych zarządzania.

Systemy wspomagania decyzji SWD (*Decision Support System – DSS*) umożliwiają [Kisielnicki, 2013]:

- wykonywanie skomplikowanych analiz i porównań,
- przetwarzanie dużych ilości danych,
- pobieranie i przetwarzanie danych z różnych źródeł,
- pracę zarówno z tekstem, jak i grafiką.

Postępujący rozwój nowych systemów informatycznych zarządzania o różnych funkcjach i zakresach stosowania spowodował konieczność integracji istniejących rozwiązań prowadzącej do zintegrowanego systemu informatycznego [Kisielnicki, 2013] umożliwiającego:

- planowanie potrzeb materiałowych (*Material Resources Planning – MRP*),
- planowanie zasobów produkcyjnych (MRPII),
- planowanie zasobów przedsiębiorstwa (*Enterprise Resources Planning – ERP*).

Profesor Kisielnicki stosuje inne nazewnictwo spowodowane tym, że obecnie nie ma przedsiębiorstwa działającego w globalnym otoczeniu (rynku), które nie stosowało by komputerów wspomagających zarządzanie, zatem można opuścić przymiotnik „informatyczny" i mówić: informacyjny system zarządzania. W swojej książce [Kisielnicki, 2013] przedstawia on również nieco odmienną klasyfikację SIZ przedstawioną tam na rys. III.12, na którym wyraźnie widać, że system transakcyjny jest elementem każdego SIZ. To spostrzeżenie potwierdza literatura wskazująca, że systemy transakcyjne są wykorzystywane w 80–90% przypadkach zastosowania SIZ.

W XXI wieku pojawiło się i intensywnie rozwija się e-zarządzanie (zarządzanie elektroniczne) wykorzystujące istniejące i nowo rozwijane narzędzia elektroniczne jak Internet, przetwarzanie w chmurze (*Cloud Computing – CC*) Internet wszystkiego (*Internet of Everything – IoE*), nowe generacje systemów komunikacyjnych np. 5G (obecnie stosowane są druga, trzecia i czwarta generacje: 2G, 3G i 4G).

W przedsiębiorstwach można spotkać dwa różniące się systemy automatyzacji: procesów biznesowych oraz procesów technologicznych, które mogą być powiązane ze sobą w stopniu zależnym od rodzaju przedsiębiorstwa. W obu procesach stosowane są komputery i roboty z tym, że komputery stosowane są do sterowania całością lub znaczącą częścią procesu biznesowego lub technologicznego, a roboty tylko jego fragmentem (kasa fiskalna, podnośnik itp.). Bliżej o tych sprawach będzie mowa w rozdziale 3.

1.3. Sztuczna inteligencja

1.3.1. Wprowadzenie

Działanie ludzkiego mózgu fascynowało ludzi już w starożytności (Arystoteles) i zainteresowanie to utrzymywało się przez wieki, ale brak wiedzy i odpowiednich narzędzi badawczych uniemożliwił rozwiązanie zagadki. Dopiero rewolucja przemysłowa, rozwój badań nad ludzkim organizmem i teorią systemów [Zieliński, 1984] umożliwiły w pierwszej połowie XX wieku formułowanie pierwszych koncepcji w tej dziedzinie [Bartkiewicz, Bolek i in., 2008]. Uruchomienie elektronicznego komputera ENIAC wywołało nadzieję na rozwiązanie odwiecznej zagadki i zintensyfikowało prace nad tym zagadnieniem w USA.

Powszechnie przyjęto, że w 1956 roku narodziła się *sztuczna inteligencja (Artificial Intelligence – AI*) choć M. Flasiński przedstawia [Flasiński, 2011] uzasadnienie dla przyjęcia roku 1955 jako daty narodzin AI.

Nazwie *sztuczna inteligencja* M. Flasiński [Flasiński, 2011] przypisuje dwa znaczenia:

- jest to wspólna nazwa dwóch dziedzin: informatyki oraz *biotyki*,
- jest to cecha sztucznych systemów umożliwiająca wykonywanie czynności, które w przypadku człowieka wymagają inteligencji.
  (W tym drugim znaczeniu, tzn. jako cecha systemów [Sidigue, Hojjat, 2013] jest przedmiotem badań nie tyle informatyki i robotyki, ile dziedziny zwanej *kognitywistyką*).

1.3.2. Systemy ekspertowe (systemy z bazą wiedzy)

Badania nad sztuczną inteligencją i jej praktycznymi zastosowaniami prowadzone były w amerykańskich uniwersytetach i placówkach naukowych, dopiero po kilkunastu latach zaczęły się ukazywać prace z W. Brytanii i Japonii. Celem początkowych badań było odwzorowanie funkcji ludzkiego mózgu, ale po paru latach osiągnięto poziom ludzkiego niemowlęcia i zrezygnowano z tego projektu implikując burzliwe dyskusje w środowiskach naukowych. Postanowiono sformułować skromniejszy cel badań, jakim stało się zbudowanie systemu odwzorowującego eksperta określonej dziedziny. Zawężenie celu przyniosło pozytywny efekt i w 1965 roku w trzech amerykańskich uniwersytetach

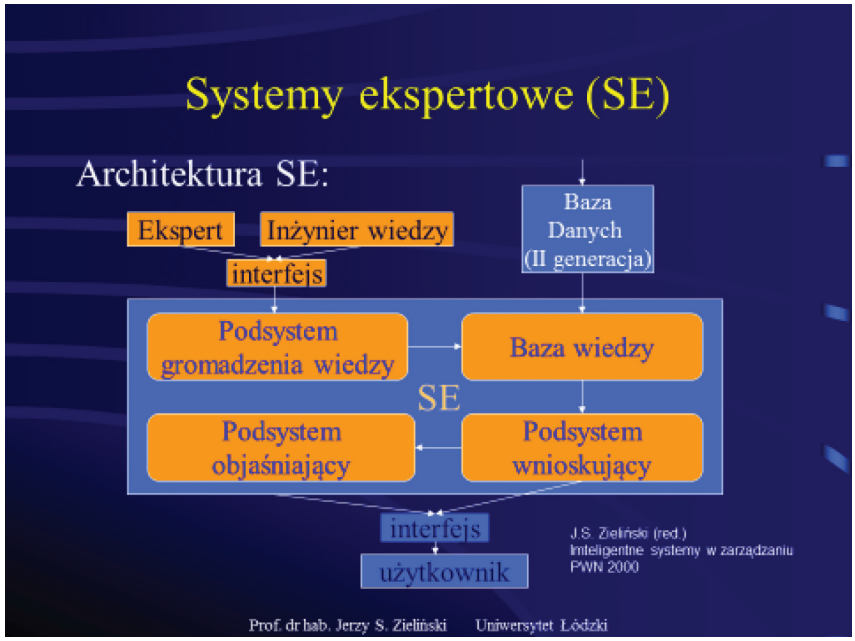
uruchomiono pierwsze trzy systemy ekspertowe [Bartkiewicz, Czajkowska i in., 1999]:

Dendral	Uniwersytet Stanford – informacje o strukturze chemicznej.
Macsyma	Uniwersytet MIT – złożona analiza matematyczna.
Hersay	Uniwersytet Canegie-Mellon – interpretacja naturalnego języka.

Ponieważ systemy ekspertowe (SE) były przez kilkanaście lat jedynymi nowoczesnymi, efektywnymi narzędziami przynoszącymi korzyści, szybko powstawały nowe systemy najpierw w USA, później w Wielkiej. Brytanii i Japonii. W literaturze można spotkać różne definicje SE [Bartkiewicz, Czajkowska i in., 1999], jedna z nich ma następującą postać: „System ekspertowy jest programem komputerowym, który wykonuje złożone zadania o dużych wymaganiach intelektualnych i robi to tak dobrze, jak człowiek będący ekspertem w tej dziedzinie" [Mulawka, 1996].

Podstawą do klasyfikacji SE jest liczba reguł zastosowanych do rozwiązania problemu (wydania ekspertyzy): 50, 100–299, 300–499, 500–999, ponad 1000.

Do działania systemu ekspertowego niezbędne są: baza wiedzy, podsystem wnioskujący i interfejs wyjściowy wydający wynik na dowolnym urządzeniu, którym może być na przykład układ sterujący procesem technologicznym. W fazie projektowania potrzebny jest także interfejs wejściowy umożliwiający *inżynierowi wiedzy* wprowadzanie do bazy wiedzy poprzez podsystem gromadzenia wiedzy niezbędnych o działania pojęć, reguł itd. Podsystem objaśniający ułatwia prześledzenie drogi utworzenia wydanej decyzji. Tak utworzony SE jest systemem statycznym, działającym na podstawie parametrów statycznych wprowadzonych w fazie projektowania. Dla usunięcia tego niedostatku, wprowadzamy dane z modelu matematycznego badanego problemu, dostarczając do bazy aktualniejsze dane. Architekturę tak utworzonego SE przedstawiono na podstawie [Bartkiewicz, Czajkowska i in., 1999] na rys. 1.3.1. Systemy ekspertowe znajdują zastosowanie w wielu dziedzinach, np. w elektroenergetyce [Bartkiewicz, Czajkowska i in., 1999; Tadeusiewicz, 1993; Sidigue, Hojjat, 2013; Zieliński, Jęczkowska i in., 1993] i wielu innych [McDonald, Burt i in., 1997].



Rys. 1.3.1. Struktura systemu ekspertowego

Źródło: [Bartkiewicz, Czajkowska i in., 1999].

1.3.3. Sztuczne sieci neuronowe

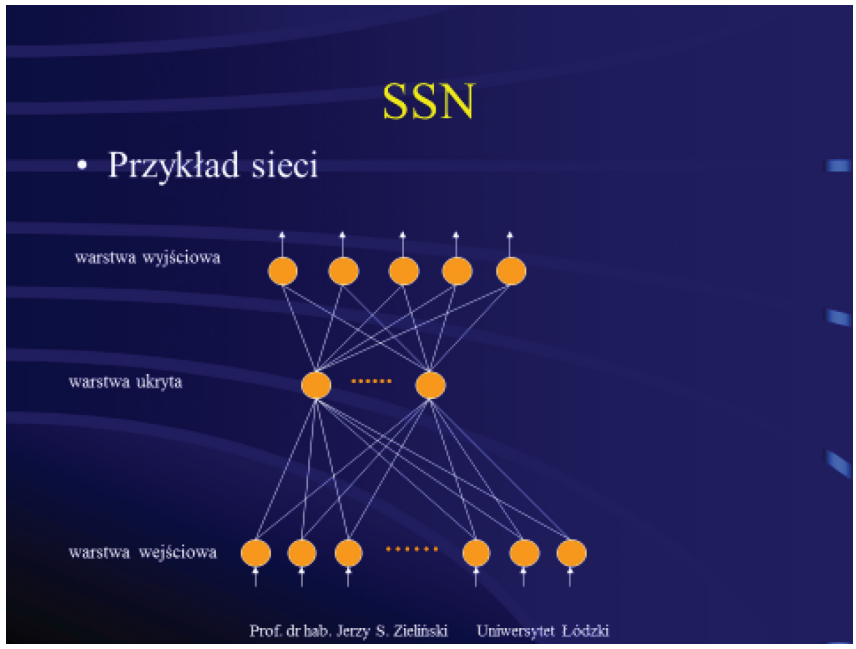
Inspiracją do badania sztucznych sieci neuronowych (SSN) jest ludzki mózg złożony z neuronów o specyficznych właściwościach:

1) człowiek w dniu narodzin ma 1,5–10^{10} neuronów i każdej doby traci ich ok.15 000 (neurony nie regenerują się),
2) liczba połączeń jednego neuronu wynosi 10^4,
3) pojemność ludzkiej pamięci wynosi 10^{18} neuronów,
4) liczba operacji/sekundę mózgu wynosi 10^{16},
5) energia użyta w jednej operacji to 10^{-16} dżula.

Wartości parametrów 3, 4, 5 są nieosiągalne dla komputerów zbudowanych według zasad fizyki Newtona [Bartkiewicz, Bolek i in., 2008].

Podstawowym elementem SSN jest model sztucznego neuronu opracowany w 1943 przez Warrena S., McCullocha i Waltera Pittsa przedstawiający komórkę z licznymi wejściami (*dendrytami*) i jednym wyjściem (*aksonem*) zakończonym wieloma *synapsami* [Tadeusiewicz, 1993]. Komórkę charakteryzują dwa parametry decydujące o możliwości przyjęcia

nadchodzących sygnałów i ich wysłaniu po przekształceniach [Bartkiewicz, Czajkowska i in., 1999]. Przez odpowiednie połączenie neuronów otrzymujemy sztuczną sieć neuronową, której architektura i zasada działania określa jej typ.

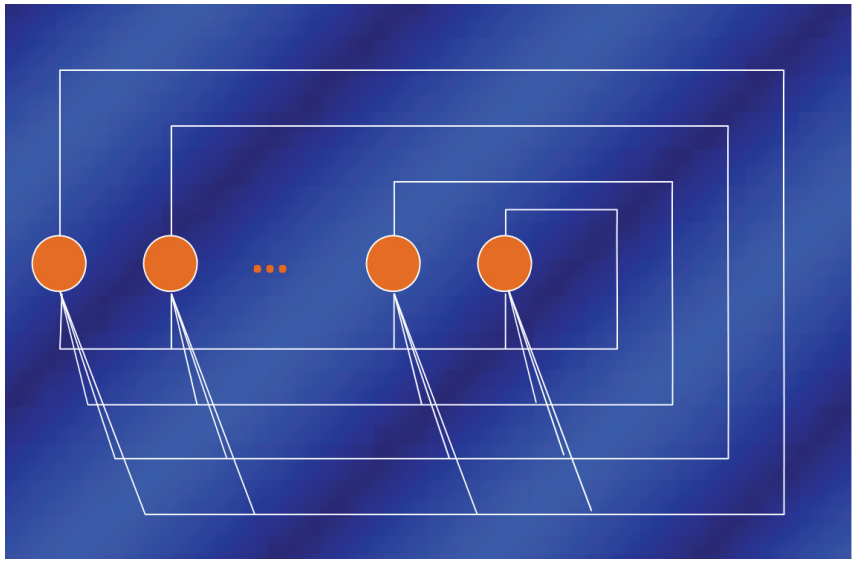


Rys. 1.3.2. Warstwowa sieć perceptronowa

Źródło: [Bartkiewicz, Czajkowska i in., 1999].

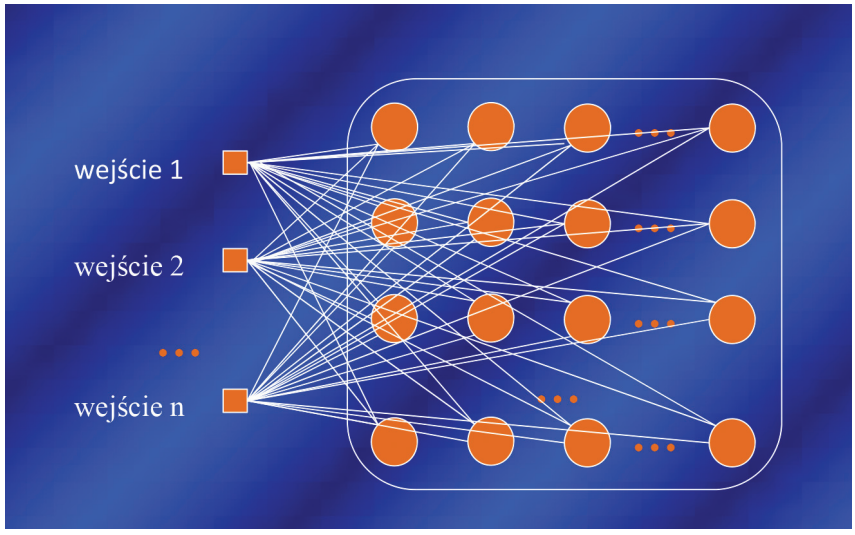
Warstwowa sieć perceptronowa przedstawiona na rys. 1.3.2 [Bartkiewicz, Czajkowska i in., 1999] składać musi się z co najmniej trzech warstw: *wejściowej* z liczbą neuronów równą liczbie znanych parametrów opisujących zjawisko, *ukrytej* (może być tych warstw więcej) i *wyjściowej* złożonej z tylu neuronów, ilu oczekujemy wyników. W sieci przedstawionej na rys.1.3.2 proces rozwiązania przebiega w jednym kierunku: od wejścia do wyjścia (*sieć warstwowa jednokierunkowa*) natomiast sieć Hopfielda przedstawiona na rys. 1.3.3 jest siecią rekurencyjną, w której sygnały wyjściowe są wprowadzone na wejścia neuronów

Odmienną strukturę natomiast mają sieci Self Organizing MAP, przykład której przedstawiony został na rys. 1.3.4 [Bartkiewicz, Czajkowska i in., 1999], Nazywane są one również *Sieciami Kohonena*. Składają się zwykle z jednej warstwy wejściowej neuronów i z jednej warstwy neuronów przetwarzających (warstwa rekurencyjna, warstwa Kohonena).



Rys. 1.3.3. Struktura w pełni połączonej sieci rekurencyjnej

Źródło: opracowanie własne.



Rys. 1.3.4. Mapa cech Kohonena

Źródło: opracowanie własne.

Istnieje szereg innych SSN, których zasady projektowania, użytkowania i zastosowania przedstawiono w [Bartkiewicz, Czajkowska i in., 1999; Tadeusiewicz, 1993].

1.3.4. Inne narzędzia sztucznej inteligencji

Istnieją liczne inne narzędzia sztucznej inteligencji, jak logika rozmyta, algorytmy ewolucyjne, teoria chaosu, teoria uczenia, inteligencja zbiorowości (*swarm intelligence*) i bardzo ważne – systemy hybrydowe. Systemy hybrydowe mogą zawierać (jako narzędzie sztucznej inteligencji) co najmniej jedno z narzędzi sztucznej inteligencji i dowolny model matematyczny lub fizyczny. Systemy hybrydowe ułatwiają efektywniejsze rozwiązania, a nawet są jedynym możliwym narzędziem rozwiązania (np. AlphaGO stosując hybrydę SSN i algorytmów ewolucyjnych, samodzielnie nauczył się starej chińskiej gry Go i pokonał mistrza świata w tej grze). Hybrydy zawierające m.in. SSN i algorytmy ewolucyjne są obecnie stosowane w poważnych, złożonych przedsięwzięciach pojawiających się w społeczeństwie informacyjnym.

Należy w tym miejscu zwrócić uwagę na pojawienie się nowego określenia: inteligencja obliczeniowa, do której nie zalicza się systemów ekspertowych z uwagi na to, że w fazie tworzenia systemu wybiera się jedną z metod rozwiązania (sieci semantyczne ramy, scenariusze, systemy produkcyjne i inne [Bartkiewicz, Czajkowska i in., 1999]), które tworzy operator. W związku z tym system hybrydowy zawierający system ekspertowy lub na przykład model fizyczny z operatorem nie należy do inteligencji obliczeniowej.

1.3.5. Zastosowania sztucznej inteligencji

Sztuczna inteligencja w końcu dwudziestego wieku znalazła zastosowanie w elektroenergetyce [McDonald J.R., Burt i in., 1997; Zieliński, Jęczkowska i in., 1993; Zieliński, 2019]; ciekawym przykładem zastosowań w tej dziedzinie okazała się potrzeba rejestrowania stanu pogody wzdłuż długiej, mocno obciążonej linii przesyłowej, gdzie w systemie prognozowania pogody wykorzystano laserowe sensory wiatru zapewniające większą dokładność od zwykle stosowanych sensorów i sztuczną inteligencję do samodzielnego uczenia się sensorów. Ten system prognozowania pogody sterował przesyłem energii elektrycznej w linii zapewniając od 15 do 30 procent efektywniejszy przesył energii elektrycznej.

Sztuczna inteligencja znajduje zastosowanie we wszystkich obszarach działalności człowieka, o zakresie oczekiwań może świadczyć pytanie: „czy sztuczna inteligencja może urealnić kwantową komputeryzację?" [Wiltz, 2019] na co odpowiedź brzmi: „Miną dziesięciolecia zanim komputer kwantowy trafi pod strzechy" [„Polityka", 2019].

1.3.6. Co dalej ze sztuczną inteligencją?

Pytanie zawarte w tytule paragrafu pojawia się coraz częściej różnych środowiskach, ale dla lepszego zrozumienia odpowiedzi warto poznać odpowiedzi na dwa następujące pytania: jak rozwijają się badania nad AI w świecie, a jak w Polsce? Rozmowa z profesorem Włodzisławem Duchem z Uniwersytetu Mikołaja Kopernika w Toruniu [Bendyk, 2019] wyjaśnia obydwie sprawy. Przodujące badania prowadzone są przez amerykańskie korporacje Google, Facebook, Amazon, Intel, IBM ponoszące gigantyczne nakłady, za nimi uniwersytety pracujące nad AI od połowy ubiegłego wieku (patrz punkt 1.3.2.).

Po to by sprostać konkurencji USA, Chin i innych krajów, nakłady UE winny wynosić 200 mld euro rocznie. W związku z tym Francja w listopadzie 2018 roku zamierza do 2022 roku wydać na badania nad Sztuczną Inteligencją 665 mln euro na powołanie 40 nowych katedr poświęconych badaniom nad AI, poszukując na całym świecie (również w Polsce,) kompetentnych wykładowców. W Polsce opracowano jeden dobry raport, ale brakuje konkretnego dokumentu rządowego o rozwoju SI, brak skoordynowanych działań. Powołana przez rząd Platforma Przemysłu Przyszłości w Radomiu (wspierana przez istniejący tam Uniwersytet Technologiczno-Humanistyczny) Akademia Innowacyjnych Zastosowań Cyfrowych ma rozpocząć działania w 2021 roku z budżetem 100 mln złotych.

Pytanie, co dalej z AI, pojawia się zarówno w środowiskach wybitnych naukowców, ludzi wykształconych różnych profesji, jak i zwykłych ludzi. Padające odpowiedzi należą do jednej z dwóch grup: optymistycznej lub pesymistycznej, przy czym odpowiedzi optymistyczne przeważają wśród ludzi wykształconych – specjalistów (informatyków, inżynierów, menadżerów wysokiego i średniego szczebla), przewidujących intensywny rozwój sztucznej inteligencji, przyczyniający się do wzrostu dochodów i rozwój filozofii, etyki i socjologii. Zdolność uczenia się z zastosowaniem narzędzi AI w najrozmaitszych dziedzinach jest ważna nie tylko dla odpowiedzi na pytanie: „Czy 5G zwycięży bez AI", lecz także dlatego, że być może AI spowoduje, że komputer kwantowy stanie się rzeczywistością.

Odpowiedzi pesymistyczne powstają w grupie wybitnych specjalistów (noblistów, informatyków światowego formatu), prognozujących zbudowanie takiego systemu sztucznej inteligencji, który wymknie się spod ludzkiej kontroli i zniszczy rodzaj ludzki. Pesymizm wśród zwyczajnych ludzi wynika z obawy zastępowania ludzi przez inteligentne urządzenia. Autor niniejszego tekstu należy do grupy pesymistów na podstawie obserwacji zachowań ludzi najwyższego szczebla krajowego i światowego w upartym

niszczenia środowiska, mimo przekraczania progu wytrzymałości ziemskiego ekosystemu.

Powyższe rozważania zostały napisane w końcu 2019 roku przed pojawieniem się na świecie pandemii koronawirusa, która spowoduje zmianę priorytetów w budżetach wielu krajów, zdecydowanie ograniczając nakłady na badania. Przewidywane konsekwencje tej pandemii spowodują konieczność odwrotu od dalszej dewastacji ziemskiego ekosystemu, wymagającej zmian priorytetów dalszego rozwoju ludzkości.

1.4. Nowe narzędzia informatyczne

W dalszych rozważaniach pod określeniem „nowe narzędzia informatyczne" rozumiemy metody wykorzystania nowoczesnych narzędzi komputerowych i oprogramowania do wspomagania rozwiązań nieosiągalnych kilka lat temu. Należy przy tym pamiętać że postęp w badaniach oraz zwiększające się potrzeby społeczeństwa informacyjnego spowodują powstanie nowych narzędzi informatycznych.

1.4.1. Internet rzeczy (*Internet of Things – IoT*) oraz Internet wszystkiego (*Internet of Everything – IoE*)

Są to nowe narzędzia informatyczne, które pojawiły się w końcu XX wieku i okazały się bardzo przydatne w rozwijającym się społeczeństwie informacyjnym z dynamiczną globalną gospodarką. Przewiduje się, że do roku 2020 od 50 do 100 bilionów rzeczy będzie połączonych elektronicznie, zatem technologia IoT umożliwia komunikację między tą ilością różnorodnych rzeczy. Definicja IoT jest obszerna, umożliwiająca określenie istoty tego pojęcia [Vermessan, Friess, 2014]:

> Internet of Things jest globalną infrastrukturą dla społeczeństwa informacyjnego, umożliwiającą zaawansowane usługi, wiążąc wzajemnie (fizyczne i wirtualne) rzeczy, wykorzystując istniejące i rozwijane technologie informacyjne i komunikacyjne.

To samo źródło podaje również drugą definicję: „IoT jest globalną sieciową infrastrukturą z możliwością samodzielnej rekonfiguracji opartych na obustronnie działających standardach i wzajemnie działających

protokołach komunikacyjnych, gdzie fizyczne i wirtualne „rzeczy" mają identyfikatory, fizyczne atrybuty i wirtualną osobowość i używają inteligentne interfejsy i są zintegrowane w informacyjnej sieci".

Na podstawie przytoczonych definicji można oczekiwać wielu zastosowań IoT w różnorodnych dziedzinach w [Sun, Ansan, 2016] wymieniono następujące:

1) inteligentne monitorowanie żywności /wody,
2) inteligentną opiekę zdrowotną,
3) inteligentne życie,
4) inteligentne monitorowanie środowiska,
5) inteligentną produkcję,
6) inteligentną elektroenergetykę,
7) inteligentne budownictwo,
8) inteligentny transport i podróżowanie,
9) inteligentny przemysł, Industrial IoT(IIoT),
10) inteligentne miasto.

Szczególnie ważne jest zastosowanie IoT w dziedzinach mocno powiązanych ze sobą, jak na przykład: 1–4, 2–3, 5–9 i inne.

Wyjaśnienia wymaga wielokrotnie użyte słowo „inteligentne". Jest to przyjęte w Polsce tłumaczenie angielskiego słowa „smart", które zgodnie z wydanym przez PWN w 2004 roku słownikiem oznacza: „inteligentny, wygląda elegancko, doskonale coś robi". Warto o tym pamiętać przeglądając na przykład powyższą listę.

Zastosowanie IoT wymaga utworzenia tzw. sieci sensorowej, która może być siecią przewodową lub bezprzewodową umożliwiającą przesyłanie pomierzonych przez sensory (czujniki) sygnałów (np. przepływu cieczy, temperatury, prędkości wiatru, stanu konta, alarmu i in.) do rejestratora danych. Duża liczba dziedzin obsługiwanych danych wymienionych na powyższej liście oznacza wielokrotnie większą liczebność różnych czujników niezbędnych do zbierania danych. W przypadku bezprzewodowej sieci czujniki muszą posiadać własne źródła energii elektrycznej (choć rozważa się również możliwość bezprzewodowego zaopatrywania czujników w tę energię). Projektując sieć sensorową trzeba pamiętać o kosztach, a zatem mierzyć tylko to, co jest niezbędne, planować tylko niezbędne funkcje dla uzyskania oczekiwanego efektu.

Nawiązując do rozważań zmianie systemu zarządzania w elektroenergetyce (punkt 1) w [Zieliński, 2018] wprowadzono IoE przyrównując przepływ danych w Internecie do przepływu energii elektrycznej. W literaturze można znaleźć wiele przykładów podobnych zabiegów, na podstawie których można sformułować pojęcie IoE – Internet wszystkiego (a nie Internet wszechrzeczy) bowiem IoE obejmuje również organizmy żywe (ludzi, zwierzęta).

1.4.2. Chmura obliczeniowa [Czerwonka 2016]

W literaturze można znaleźć wiele definicji chmury obliczeniowej (*Cloud Computing – CC*) z których najbardziej ogólną wydaje się podana przez NIST (National Institute of Standards and Technology) jako „sposób dostępu poprzez sieć komputerową do współdzielonych i łatwo konfigurowanych zasobów obliczeniowych (sieci, serwerów, magazynów danych, aplikacji i usług), które na żądanie, dynamicznie mogą być przydzielane i zwalniane, przy równoczesnym minimalnym zaangażowaniu serwisów technicznych".

Wymagane funkcjonalności chmury obliczeniowej określone przez NIST zapewniają podstawowe właściwości, które mogą być atrakcyjne dla klienta, jak:

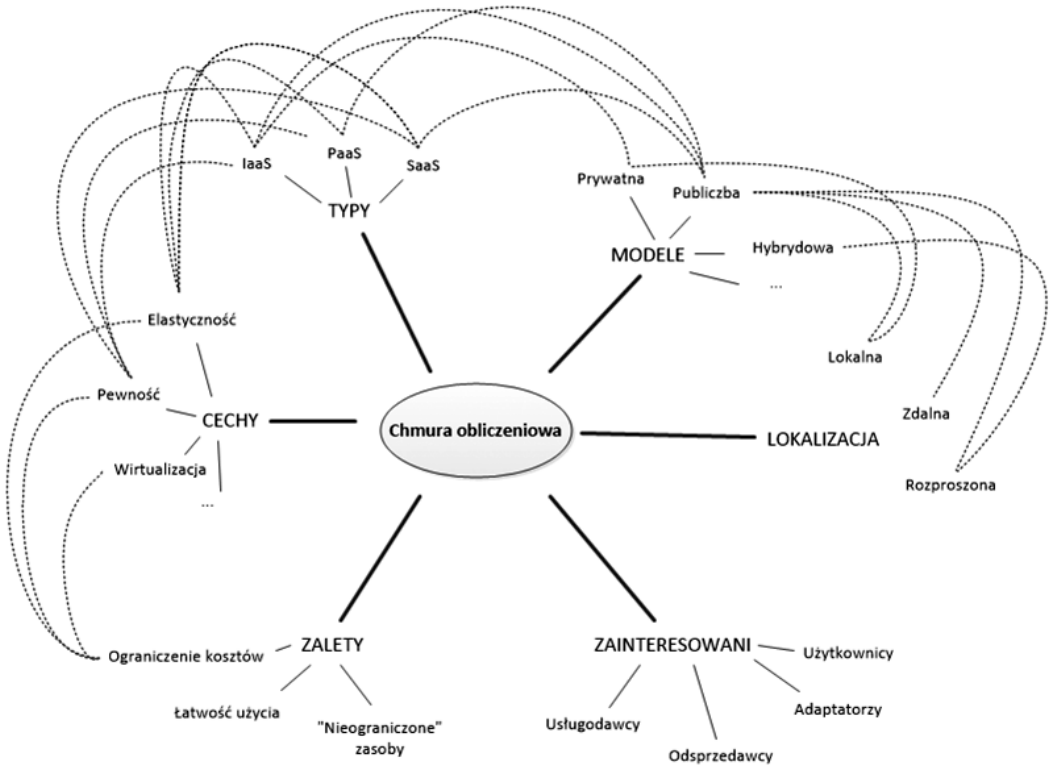
1) wysoka elastyczność zasobów,
2) bezobsługowość infrastruktury,
3) wysoka dostępność wynikająca ze skali,
4) model płacenia tylko za wykorzystane zasoby.

Działanie chmur obliczeniowych wspierane jest przez wielkie centra obliczeniowe wyposażone w tysiące serwerów, np. w 2013 roku Google posiadał 13 centrów danych zawierających 900 tysięcy serwerów pobierających moc 200 tysięcy watów, Amazon – 7 centrów z 450 tysiącami serwerów; kolejne firmy: Microsoft, Facebook, IBM posiadają centra z setkami tysięcy serwerów. Niewielu klientów potrzebuje takich mocy obliczeniowych mogąc wybrać jedną z poniższych usług:

1) chmura prywatna o strukturze konfigurowanej wyłącznie do użytkowania tylko przez jednego użytkownika,
2) chmura publiczna o strukturze dostępnej dla otwartego publicznego użytkownika,
3) chmura społecznościowa, której zasoby są przeznaczone do wyłącznego użytku przez konkretną społeczność klientów organizacji,
4) chmurę hybrydową o e-strukturze będącej połączeniem dwóch odmiennych, wymienionych wyżej struktur. (rys. 1–rys. 1.5. w [Czerwonka, 2016]).

Natomiast rodzaj i sposób udostępniania zasobów z chmury zależy od modelu realizacji; NIST definiuje trzy podstawowe:

1) oprogramowanie jako usługa – SaaS (Software as a *Service*),
2) platforma jako usługa – PaaS (*Platform as a Service*),
3) infrastruktura jako usługa – IaaS (*Infrastructure as a Service*).



Rys. 1.4.1. Widok głównych wymiarów formujących systemy chmurowe

Źródło: [Czerwonka, 2016].

W przypadku współpracy CC z systemem IoT, może wystąpić potrzeba przekazania do chmury ogromnej ilości danych *Edge computing* (przetwarzanie brzegowe), co wymagałoby dużej liczby kanałów łączności wymuszając duże nakłady finansowe. W celu ograniczenia kosztów można użyć metody *dew* (rozproszony) lub *fog* (mgły) *computing* [Bartkiewicz, Czajkowska i in., 1999]; obie te metody stosują podobne podejście: przetwarzanie wszystkich danych niezbędnych do wykorzystania w miejscu ich powstania, i przesyłanie do chmury tylko pozostałych danych

1.4.3. Integracja informatycznych systemów zarządzania i nowych narzędzi informatycznych

Zgodnie z poprzednimi rozważaniami, informatyczne systemy zarządzania rozwijają się ponad pół wieku, począwszy od systemów transakcyjnych poprzez kolejne generacje, zatem metody integracji istniejących i nowych systemów są znane. Czy zebrane dotychczas doświadczenia mogą być użyteczne przy wdrażaniu nowego narzędzia informatycznego? Po zapoznaniu się z sygnalną informacją [Miller, 2018], można stwierdzić, że integracja przemysłowego IoT (IIoT) z istniejącym systemem przebiega także zgodnie z zasadami inżynierii systemowej stosowanej w tworzeniu i eksploatacji systemów zarządzania ISZ.

Literatura

Bartkiewicz W., Bolek C., Kaczorowska A., Krygier N., Lech T., Łuczak M., Matusiak B., Pamuła A., Papińska-Kacperek J., Podgórski G., Zieliński J.S. (2008), *Społeczeństwo informacyjne*, Wydawnictwo Naukowe PWN, Warszawa.

Bartkiewicz W., Czajkowska R., Gontar Z., Jęczkowska B., Pamuła A., Zieliński J.S. (red.) (1999), *Inteligentne systemy w zarządzaniu*, Wydawnictwo Naukowe PWN, Warszawa.

Bartkiewicz W., Jabłoński W.J. (2005), *Systemy informatyczne zarządzania. Wprowadzenie do technologii przetwarzania danych*, Oficyna Wydawnicza Włocławskiego Towarzystwa Naukowego, Włocławek.

Bendyk E. (2019), *Po co nam sztuczna inteligencja (Rozmowa z profesorem Włodzisławem Duchem)*, „Polityka", nr 42, s. 52–54.

Czerwonka P. (2016), *Zastosowanie chmury obliczeniowej w polskich organizacjach*, Biblioteka, Łódź.

Flasiński M. (2011), *Wstęp do sztucznej inteligencji*, Wydawnictwo Naukowe PWN, Warszawa.

Gontar B. (red.) (2019), *Zarządzanie danymi w organizacji*, Wydawnictwo UŁ, Łódź.
Kisielnicki J. (2013), *Systemy informatyczne zarządzania*, Agencja Wydawnicza Placet, Warszawa.
McDonald J.R., Burt G.M., Zieliński J.S., McArthur S.D., Knight U.G., Masucco S., Moyes A., Weir B., Young D.J. (1997), *Intelligent knowledge based systems in electric power systems*, Chapman & Hall, London.
Miller J. (2018), *Integrating IIoT equipment into manufacturing systems*, „Control Engineering" 2018.08.01.
Miną dziesięciolecia zanim komputer kwantowy trafi pod strzechy (2019), „Polityka", nr 46, s. 65.
Mulawka J.J. (1996), *Systemy ekspertowe*, WNT, Warszawa.
Sidigue N., Hojjat A. (2013), *Computational Intelligence*, Wiley.
Sun Y., Ansan M. (2016), *Edge IoT: Mobile Edge Computing for the Internet of Things*, „Communication", vol. 54, No. 12, s. 22–29.
Tadeusiewicz R. (1993), *Sieci neuronowe*, Akademicka Oficyna Wydawnicza, Warszawa.
Vermessan O., Friess P. (2014), *Internet of Things – From Research and Innovation to Market Deployment*, River Publisher, Aalborg.
Wiltz C. (2019), *AI Could Make Quantum Computers a Reality*, „Design News", 2019.02.06.
Zieliński J.S. (1984), *Inżynieria systemowa*, Skrypt. Uniwersytet Łódzki.
Zieliński J.S. (2018), *Wpływ Smart Grid na informatyczny system zarządzania elektroenergetyką*, „Przegląd Organizacji", nr 10, s. 46–48.
Zieliński J.S. (2019), *Sztuczna inteligencja i nowe narzędzia w elektroenergetyce*, „Biuletyn Techniczno-Informacyjny SEP, Oddział Łódzki", nr 2, s. 18–20.
Zieliński J.S., Jęczkowska B., Górnicki W., Kopczyńska D., Kupras A. (1993), *Systemy ekspertowe wspomagające dyspozytorów w systemie elektroenergetycznym*, „Energetyka", nr 5, s. 156–160.

Spis rysunków

Rozdział 2

Inteligentne usługi informacyjne

2.1. Tradycyjne modele wyszukiwania informacji tekstowej

2.1.1. Model boolowski wyszukiwania informacji

Model boolowski był pierwszym z tekstowych, opartych na słowach kluczowych, podejść do wyszukiwania informacji. Stanowił on zresztą naturalne podejście do realizacji zadań wyszukiwawczych. Wskutek swojej prostoty i jasnego formalizmu model boolowski przyciągał w przeszłości wiele uwagi i do dziś stanowi podstawę bardzo wielu systemów wyszukiwawczych w Internecie.

Model boolowski jest prostym modelem wyszukiwania, opartym na działaniach na zbiorach i logice klasycznej (algebrze boolowskiej).

- Dokumenty indeksowane są poprzez wyodrębnienie w nich słów kluczowych (termów), określających ich zawartość.
- Zapytania specyfikowane są jako wyrażenia boolowskie, złożone ze słów kluczowych połączonych operatorami logicznymi (na ogół koniunkcja, alternatywa i negacja) np. „gwiazda AND kosmos AND NOT film".

Dopasowanie dokumentu do zapytania w modelu boolowskim określane jest poprzez sprawdzenie dla niego prawdziwości podanego w zapytaniu wyrażenia. Dla przykładowego zapytania „gwiazda AND kosmos AND NOT film" wyszukane zostaną dokumenty, które posiadają w swoim opisie słowa kluczowe „gwiazda" i „kosmos", natomiast nie posiadają słowa kluczowego „film".

Zwróćmy więc uwagę, że w usłudze wyszukiwawczej wykorzystującej model boolowski dla danego zapytania dokument może znaleźć się w zbiorze wynikowym lub nie i nie ma innych możliwości. Wyszukane zostaną dokumenty, które dokładnie odpowiadają zapytaniu i tylko

takie. Dokumenty, które choćby w minimalnym stopniu nie odpowiadają sformułowanemu w nim warunkowi, nie zostaną wyszukane. Wynikiem oceny dopasowania dokumentu do zapytania, jest tylko prawda lub fałsz. Ponadto model boolowski zakłada, że wszystkie dokumenty w zbiorze wynikowym są równoważne pod względem relewancji i w konsekwencji jednakowo istotne dla danego zapytania.

Jak więc widzimy, model boolowski jest bardzo naturalny, jego wyniki są przewidywalne i łatwe do objaśnienia dla użytkowników. Z implementacyjnego punktu widzenia jest on na ogół najbardziej efektywny z omawianych modeli wyszukiwania, ponieważ dokumenty są szybko eliminowane z dalszych rozważań, w przypadku gdy nie spełniają kolejnych sprawdzanych członów w zapytaniu [Croft, Metzler, Strohman, 2009; Jo, 2019].

Zagadnienia implementacyjne w naszej książce pozostawiamy nieco na boku – prezentujemy raczej koncepcje, metody, a nie finalne rozwiązania – jednakże kilka stron poświęcimy na zarys głównych idei związanych z realizacją procesu wyszukiwania przy pomocy modelu boolowskiego. Przedstawimy tylko wybrane główne elementy zagadnienia implementacji wyszukiwarki boolowskiej i to w formie przykładowej, tak aby czytelnik lepiej zrozumiał specyfikę działania tego rodzaju systemów. Natomiast oczywiście szczegółowa charakterystyka tej tematyki jest niemożliwa z powodu szczupłości miejsca i odsyłamy czytelnika do pozycji takich jak: [Frakes, Baeza-Yates, 1992; Manning, Raghavan, Shütze, 2007; Croft, Metzler, Strohman, 2009; Kowalski, 2011; Jo, 2019].

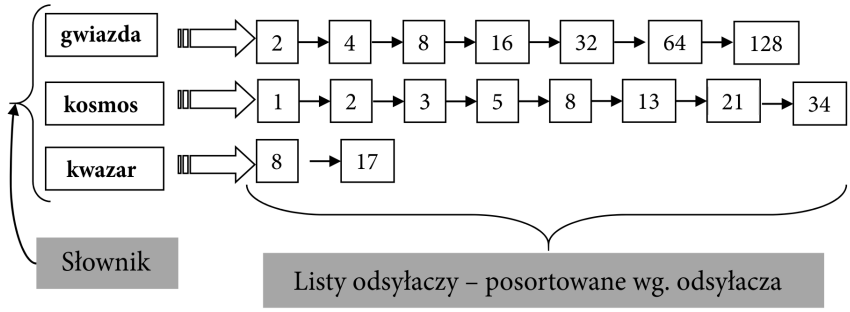
Problem wydaje się zresztą dosyć interesujący. Oczywistym jest bowiem, że bezpośrednie podejście polegające na sprawdzaniu określonego w zapytaniu warunku, po kolei dla wszystkich dokumentów w kolekcji, w teorii pewnie powinno działać, ale w praktyce raczej okaże się rozwiązaniem naiwnym, niemożliwym do zastosowania ze względów efektywnościowych. W przypadku wyszukiwarki indeksującej miliardy dokumentów webowych, sprawdzenie warunku zapytania dla każdego z nich trwałoby zdecydowanie zbyt długo.

Otóż podstawowy pomysł stojący za efektywną implementacją przetwarzania zapytania w wyszukiwarce boolowskiej, polega na zamianie operacji logicznych na działania na zbiorach dokumentów. Skądinąd jest to podejście naturalne, związek między logiką klasyczną i teorią mnogości jest przecież doskonale znany i powszechnie wykorzystywany w matematyce.

Do usprawnienia procesu przetwarzania zapytania, stosowana jest ponadto pomocnicza struktura danych, nazywana indeksem. Jak już powiedzieliśmy w punkcie 2.1.3, omawiając typową budowę wyszukiwarki

dokumentów, cechy dokumentów (w tym wypadku słowa kluczowe dla poszczególnych dokumentów) przechowywane są w indeksie. Indeks ten nie ma jednak postaci tzw. listy prostej, to znaczy listy uporządkowanych po kolei dokumentów, wraz z opisującymi je kluczowymi. W usługach wyszukiwawczych stosowane indeksy niemal zawsze mają charakter tzw. indeksu odwrotnego, lub listy inwersyjnej.

Indeks odwrotny (przykładowy indeks przedstawiony został na rysunku 2.1.1) składa się ze słownika, zawierającego wszystkie słowa kluczowe wykorzystywane do indeksowania dokumentów w całej kolekcji. Zazwyczaj słownik ma formę odpowiedniej struktury danych (np. drzewa B+), umożliwiającej szybkie wyszukanie danego słowa kluczowego. Do każdego słowa kluczowego przypisana jest lista odsyłaczy do wszystkich dokumentów, do których słowo to zostało przypisane. Odsyłacz do dokumentu, to identyfikator dokumentu (np. jego numer), lub każda inna informacja pozwalająca bezpośrednio uzyskać do niego dostęp (np. fizyczna informacja o jego lokalizacji). Listy odsyłaczy są uporządkowane według ich wartości.



Rys. 2.1.1. Przykładowy indeks odwrotny dla wyszukiwania boolowskiego

Źródło: opracowanie własne.

W przykładzie na rysunku 2.1.1, dokumenty zostały zaindeksowane przy pomocy trzech słów kluczowych (termów), przy czym np. term „kwazar" występował jako słowo kluczowe tylko w dokumentach 8 i 17.

Przy pomocy indeksu odwrotnego łatwo można wykonywać operacje boolowskie na termach zapytania, przy pomocy odpowiednich działań na listach dokumentów związanych z tymi termami. Najłatwiejsza sytuacja jest w przypadku gdy zapytanie składa się tylko z jednego słowa kluczowego. Musimy znaleźć je słowniku, a następnie jako wynik wyszukiwania przyjąć wszystkie dokumenty znajdujące się na przypisanej mu liście odsyłaczy. Na przykład, jeśli zapytanie składało się ze słowa

kluczowego „kosmos" – to znajdujemy je w słowniku i jako wynik wyszukiwania przyjmujemy zbiór dokumentów z przypisanej mu listy inwersyjnej, o numerach {1, 2, 3, 5, 8, 13, 21, 34}.

Jeśli w zapytaniu występuje koniunkcja lub alternatywa dwóch słów kluczowych musimy połączyć dwie listy dokumentów. Rozważmy dla przykładu zapytanie „gwiazda AND kosmos". Pierwszy krok polega na wyszukaniu list dokumentów związanych z każdym słowem kluczowym.

gwiazda	{2, 4, 8, 16, 32, 64, 128}
kosmos	{1, 2, 3, 5, 8, 13, 21, 34}

Następnie musimy połączyć te dwie listy, wybierając do listy wynikowej dokumenty powtarzające się w obydwu z nich. Ponieważ listy odsyłaczy w indeksie odwrotnym są uporządkowane według identyfikatorów dokumentów, można zastosować prosty i szybki algorytm łączenia dwu uporządkowanych zbiorów danych. Kilka jego kroków wyglądałoby następująco:

1. Porównujemy zawsze pierwsze elementy w listach. W tym przypadku będą to dokumenty 2 i 1. Ponieważ 1 < 2, więc dokument o numerze 1 nie może występować w pierwszej liście (musiałby być wcześniej, przed 2), w związku z tym wyłączmy go z wyniku koniunkcji – w drugiej liście (dla słowa „kosmos") ustawiamy się na kolejny element.
2. Porównujemy 2 i 2. Ponieważ 2 = 2, więc dokument 2 występuje w listach dla obu słów kluczowych. Zapisujemy go do wynikowej listy połączonej i w listach obu słów kluczowych przesuwamy się na kolejne elementy.
3. Porównujemy 4 i 3. Ponieważ 3 < 4, więc dokument 3 z pewnością w pierwszej liście nie występuje – w drugiej liście (dla słowa „kosmos") ustawiamy się na kolejny element.
4. Porównujemy 4 i 5. Tym razem 4 < 5, tak więc w liście drugiej z pewnością nie występuje dokument 4 – tym razem ustawiamy się na następny element w pierwszej liście, dla słowa „gwiazda".
5. Porównujemy 8 i 5. Ponieważ 5 < 8, przesuwamy się w liście drugiej.
6. Porównujemy 8 i 8. Tym razem 8 = 8, a więc dokument 8 występuje w obu listach. Zapisujemy go do zbioru wynikowego i w obu listach przesuwamy się na następne dokumenty itd.

Jak widzimy, algorytm wyławiania dokumentów powtarzających się w obu list jest prosty i szybki. Przy jego wykonaniu musimy wykonać

tylko tyle kroków, ile jest dokumentów w obu listach, czyli 7 + 8 = 15. Po przejrzeniu obu list, bez trudu znajdziemy wszystkie „duplikaty". W naszym przykładzie, oczywiście otrzymujemy:

gwiazda AND kosmos {2, 8}

W zbliżony sposób łatwo możemy wykonywać inne operacje logiczne na słowach kluczowych, stosując podobne algorytmy łączenia list dokumentów związanych z tymi słowami. Oczywiście zapisując dokumenty do zbioru do zbioru wynikowego, zgodnie z regułą dla danej operacji logicznej.

I tak np. dla alternatywy musimy w liście wynikowej umieścić dokumenty wszystkie występujące na obydwu listach wyjściowych:

gwiazda OR kosmos {1, 2, 3, 4, 5, 8, 13, 16, 21, 32, 34, 64, 128}

A dla wyrażenia „gwiazda AND NOT kosmos", wszystkie dokumenty z pierwszej listy, które w drugiej nie występują:

gwiazda AND NOT kosmos {4, 16, 32, 64, 128}

Co w przypadku jeśli zapytanie zawiera warunek logiczny złożony z większej liczby słów kluczowych? Na przykład koniunkcję trzech termów „gwiazda AND kosmos AND kwazar". W takim przypadku musimy połączyć aż trzy listy dokumentów:

gwiazda	{2, 4, 8, 16, 32, 64, 128}
kosmos	{1, 2, 3, 5, 8, 13, 21, 34}
kwazar	{8, 17}

Łatwo, oczywiście byłoby przerobić nasz algorytm łączenia list tak, aby działał on dla trzech list, a nie dwu. Komplikowałoby to jednak program naszej wyszukiwarki. Ponadto, co jeżeli użytkownik wykona zapytanie złożone z koniunkcji większej liczby słów kluczowych: czterech, pięciu, a może sześciu? Lepszą strategią będzie skorzystanie z prawa łączności i scalać nasze listy parami.

Pamiętajmy jednak, że kolejność scalania list może być dosyć dowolna:

gwiazda AND kosmos AND kwazar = (gwiazda AND kosmos) AND kwazar =

= gwiazda AND (kosmos AND kwazar) = (gwiazda AND kwazar) AND kosmos

Którą z tych możliwości powinniśmy wybrać? Wszystkie one dają takie same wyniki, czyli identyczną jakość wyszukiwania, należy więc wybrać ten wariant który daje najlepszą efektywność – po prostu jest najszybszy. Kiedy obliczenia będą najefektywniejsze? Wtedy, gdy jak najszybciej poskracamy nasze listy dokumentów. Porównajmy dwa podejścia.

Pierwsze podejście: „(gwiazda AND kosmos) AND kwazar".

Łączymy najpierw „gwiazda AND kosmos" (7+8 = 15 kroków), wyniku otrzymujemy listę {2, 8}. Łączymy tę listę wynikową z listą dokumentów dla termu „kwazar" (2 + 2 = 4 kroki), otrzymujemy wynik zapytania {8}. Wymagało to razem 15 + 4 = 19 kroków.

Drugie: „(gwiazda AND kwazar) AND kosmos"

Łączymy najpierw „gwiazda AND kwazar" (7 + 2 = 9 kroków), otrzymujemy listę wspólnych dokumentów {8}. Łączymy ją z listą dla słowa „kosmos" (1 + 8 = 9) kroków, otrzymując wynik {8}. W sumie 9 + 9 = 18 kroków.

Podejście drugie wymaga więc mniejszej liczby kroków. Zysk może nie jest duży, ale też i nasze przykładowe listy dokumentów są krótkie. W przypadku gdy słowo kluczowe może występować w setkach, a nawet tysiącach dokumentów, zysk będzie znacznie większy. Wniosek: Przy koniunkcji wielu słów kluczowych, listy dokumentów łączymy w kolejności coraz większych ich rozmiarów. Tak, by jak najszybciej się one poskracały.

Pozostał nam najtrudniejszy problem. Jak wykonywać zapytania w sytuacji, gdy kilka słów kluczowych połączonych jest różnymi działaniami logicznymi, powiedzmy w formie takiego wyrażenia, jak: „gwiazda AND (kosmos OR NOTkwazar)".

W przypadku takich wyrażeń, bezpośrednie ich wykonanie może być bardzo nieefektywne. Przypomnijmy sobie jak wyglądały listy wynikowe dla koniunkcji i alternatywy. Otóż w przypadku koniunkcji wynikowa lista dokumentów jest niedłuższa od każdej z list źródłowych, a zazwyczaj znacznie od nich krótsza. Koniunkcja redukuje więc liczbę dokumentów, które trzeba rozważyć w dalszych działaniach. Alternatywa – odwrotnie. Daje w wyniku listę nie krótszą od każdej z list źródłowych, a zazwyczaj znacznie od nich dłuższą. Alternatywa wydłuża więc listy dokumentów.

Tak więc realizując określone w zapytaniu wyrażenie logiczne, wyszukiwarka powinna tak je wykonywać, aby najpierw obliczyć wszystkie możliwe koniunkcje, a dopiero potem alternatywy. Dzięki temu będziemy operować przez cały czas na możliwie jak najkrótszych listach dokumentów. Przekształcanie wyrażeń logicznych tak, aby spełniały one powyższe warunki wymaga zastosowania metod redukcji wyrażeń algebry Boole'a.

Podsumowując zawartość niniejszego podrozdziału, widzimy, że przedstawiony w nim model boolowski stanowi niezbyt skomplikowane i naturalne podejście do zadania wyszukiwania informacji. Pokazaliśmy (w ogólnym zarysie oczywiście) sposób efektywnej implementacji usługi wyszukiwawczej opartej na tej koncepcji, wykorzystujący stosunkowo prostą strukturę danych, jaką jest indeks odwrotny, oraz szybko działające algorytmy pozwalające na określenie wynikowego zbioru dokumentów, odpowiadającego warunkom określonym w zapytaniu użytkownika.

Z punktu widzenia wykorzystania, model boolowski ma wiele zalet. Przede wszystkim wymienić tu należy prostotę i jasny formalizm konstrukcji zapytania. Zapytania boolowskie są precyzyjne. Uważnie zaprojektowane, pozwalają na zachowanie pełnej kontroli, transparentności i precyzyjności procesu wyszukiwania. Możliwe jest skonstruowanie zapytań, które bardzo dokładnie i ściśle ograniczą wynikowy zbiór dokumentów do niemal relewantnego. Wyniki działania wyszukiwarki boolowskiej są bardzo przewidywalne i łatwe do wyjaśnienia użytkownikom.

Powyższe zalety powodują, że zapytania boolowskie są często preferowane przez profesjonalistów zajmujących się wyszukiwaniem informacji z danej dziedziny. Model ten stanowi podstawę wielu komercyjnych systemów bibliograficznych. Ponadto łatwo do niego włączyć inne cechy dokumentów poza słowami kluczowymi, takie jak na przykład różnego rodzaju metadane dokumentu. Wymaga to jedynie rozszerzenia zapytania o kolejne człony logiczne, sprawdzające określane przez użytkownika warunki, nakładane na wartości dodatkowo wykorzystywanych cech.

Niestety model boolowski, oprócz szeregu zalet, niesie ze sobą również wiele niedogodności. Podstawowy problem wiąże się z faktem, że strategia wyszukiwawcza modelu boolowskiego bazuje na binarnym kryterium decyzyjnym – dokument albo jest relewantny, albo nie, bez żadnej pośredniej gradacji. Ponadto, wszystkie dokumenty w zbiorze wynikowym są tak samo relewantne i istotne dla danego zapytania. Nie ma możliwości określenia, że jedne z nich są lepiej, a inne gorzej dopasowane do zapytania. Może to pogarszać jakość działania usługi wyszukiwawczej, zwłaszcza biorąc pod uwagę nieprecyzyjny charakter języka naturalnego,

na którym przecież bazują wszystkie podejścia do wyszukiwania informacji, oparte na tekście i słowach kluczowych.

Powoduje to, że pomimo iż wyrażenia boolowskie mają precyzyjną semantykę, wcale nie jest łatwo przełożyć na nie potrzebę informacyjną. Tak jak powiedzieliśmy, możliwe jest skonstruowanie zapytania, które bardzo precyzyjnie ograniczy wynikowy zbiór dokumentów do niemal relewantnego, ale nawet w niezbyt złożonych sytuacjach takie zapytanie często musi być bardzo długie i skomplikowane.

Przyczyną tego faktu jest binarny charakter decyzji odnośnie relewancji dokumentu, który powoduje że warunki logiczne stosowane w konstrukcji zapytania działają w sposób bardzo ostry, silnie modyfikując wynikowy zbiór dokumentów. Wykorzystanie słowa kluczowego w połączeniu z alternatywą może bardzo rozszerzyć wynik zapytania, włączając do niego wiele dokumentów nierelewantnych. Z drugiej strony użycie słowa kluczowego w koniunkcji z innymi może bardzo mocno ten wynik ograniczyć. Aby zilustrować ten problem, zastanówmy się nad następującym przykładem.

Przyjmijmy że w usłudze wyszukiwawczej obejmującej kolekcję dokumentów na różnorodne tematy, użytkownika interesuje wyszukanie dokumentów na temat astronomii. Powiedzmy więc, że aby wyrazić swoją potrzebę informacyjną, korzysta ze słowa kluczowego:

kosmos

Jednak nie wszystkie dokumenty poświęcone astronomii muszą literalnie to słowo kluczowe zawierać. Aby uwzględnić inne możliwości z tym związane, użytkownik postanawia więc rozszerzyć katalog wykorzystywanych w zapytaniu słów kluczowych o kolejny term często występujący w tekstach astronomicznych:

kosmos OR gwiazda

Problem polega na tym, że dodanie słowa „gwiazda" może spowodować bardzo znaczne rozszerzenie wyniku zapytania, o wszystkie dokumenty zawierające ten term, niekoniecznie poświęcone astronomii. Na przykład, również dotyczące filmu i gwiazd filmowych. Aby zablokować tę możliwość trzeba wykluczyć pewne użycia termu „gwiazda". Na przykład:

kosmos OR (gwiazda AND NOT film)

Ale z kolei wykluczenie słowa film, powoduje odrzucenie wszystkich dokumentów zawierających ro słowo. Również tekstów poświęconych filmom dokumentalnym o astronomii, a co gorsza także tekstów na temat wykorzystania filmów w astronomii, np. do badania widma… gwiazd. Powiedzmy, że ten problem da się jakoś rozwiązać, dalej komplikując zapytanie. Filmy w astronomii prawdopodobnie rzadko korzystają z aktorów, a więc zapytanie w rodzaju:

kosmos OR (gwiazda AND NOT (film AND aktor))

powinno chyba dostatecznie precyzyjnie opisać sytuację. Ale słowo „gwiazda" występuje również często np. w tekstach o Dzikim Zachodzie, jako odznaka wyróżniająca stróżów prawa na tym terenie. A „kosmos" jest powszechną areną akcji tekstów z gatunku fantastyki naukowej…

Jak więc widzimy, nawet w takiej stosunkowo prostej sytuacji, wyrażenie potrzeby informacyjnej w formie zapytania boolowskiego, chociaż możliwe, może być dosyć złożonym zadaniem. Dlatego wyszukiwarki boolowskie najlepiej sprawdzają się przy obsłudze w miarę jednorodnych kolekcji dokumentów z dobrze określonej dziedziny. Przy czym preferowane są one przez profesjonalistów zajmujących się tą dziedziną, dobrze znających słownictwo w niej wykorzystywane i przyzwyczajonych do precyzyjnego i uporządkowanego wyrażania swoich myśli.

Natomiast większość zwłaszcza nieprofesjonalnych użytkowników uważa wyrażanie swoich zapytań w postaci wyrażeń boolowskich za trudne. Ograniczają się więc do najprostszych warunków, które nie pozwalają na właściwy opis potrzeb informacyjnych. Podobnie dotyczy to wykorzystania obszernych, bardzo niejednorodnych tematycznie systemów wyszukiwawczych w otwartej sieci.

2.1.2. Model wektorowy wyszukiwania informacji

Model wektorowy wyszukiwania opracowany został przez Geralda Saltona w szeregu prac prowadzonych w drugiej połowie lat sześćdziesiątych i pierwszej połowie lat siedemdziesiątych ubiegłego wieku [Salton, 1968, 1991; Salton, Wong, Yang, 1975; Salton, McGill, 1983; Salton, Buckley, 1987]. Celem jego wprowadzenia było rozwiązanie, a przynajmniej złagodzenie szeregu problemów wiążących się z wyszukiwaniem boolowskim. Model wektorowy (w formie podstawowej lub z rozmaitymi

modyfikacjami) do dziś stanowi dominujące rozwiązanie w wyszukiwarkach tekstowych wykorzystujących słowa kluczowe.

Przypomnijmy, że w modelu boolowskim obowiązywała binarna reguła decyzyjna: dokument albo odpowiadał zapytaniu, albo nie. Dawało to dobre wyniki w przypadku doświadczonych użytkowników, dobrze rozumiejących swoje potrzeby, dziedzinę wyszukiwania i bazę dokumentów. Niestety wyrażenie potrzeby informacyjnej w formie ścisłych wyrażeń logicznych zwykle jest dosyć złożone. Model boolowski znacznie gorzej spisywał się więc dla (większości) użytkowników, którzy nie potrafili tak precyzyjnie formułować swoich potrzeb.

Alternatywą jest ocena relewancji dokumentu dla zapytania w formie ciągłej. Dzięki temu dokument może odpowiadać zapytaniu w pewnym stopniu, co często pozwala uzyskać satysfakcjonujące użytkownika wyniki, nawet jeśli jego potrzeba informacyjna sformułowana jest nie do końca precyzyjnie.

Możliwe jest ponadto wskazanie, które z wyszukanych dokumentów są lepiej dopasowane do zapytania, a które gorzej. Wynik zapytania przyjmuje wtedy formę rankingu dokumentów, posortowanego malejąco (nierosnąco) według stopnia relewancji. Najbardziej relewantne dokumenty prezentowane są użytkownikowi na początku rankingu, najmniej relewantne – na końcu, co ułatwia przeglądanie wyników zapytania złożonych z wielu dokumentów.

W modelu wektorowym do określenia stopnia dopasowania dokumentu do zapytania, wykorzystuje się miarę podobieństwa. W tego rodzaju podejściach, możemy je ogólnie określić wyszukiwaniem topologicznym, zapytania formułowane są w postaci tzw. zapytania prostego lub „worka słów" (*bag of words*), czyli zestawu termów (słów kluczowych) nie połączonych żadnymi operatorami logicznymi. Użytkownik w zapytaniach po prostu określa słowa kluczowe, które opisują jego potrzebę informacyjną.

Reprezentacja dokumentu w systemie również ma charakter zestawu słów kluczowych, reprezentujących jego treść. Pod tym względem sytuacja jest więc zbliżona do tego, z czym mieliśmy do czynienia w wyszukiwaniu boolowskim. Zadaniem wyszukiwarki jest znalezienie przechowywanych w systemie dokumentów, które opisane są zestawem słów kluczowych „podobnym" do podanych przez użytkownika w zapytaniu. Podobieństwo to określane jest z wykorzystaniem miar podobieństwa, opartych na liczbie słów kluczowych występujących jednocześnie (współwystępujących) w opisie dokumentu i w zapytaniu.

Oznaczmy sobie przez Q zbiór termów określonych przez użytkownika w zapytaniu, zaś przez D, zbiór termów wykorzystanych w opisie

dokumentu. Naturalnym kandydatem na miarę dopasowania dokumentu do zapytania (podobieństwa) wydaje się w tym przypadku tzw. współczynnik dopasowania prostego, równy po prostu liczbie termów współwystępujących w dokumencie i zapytaniu:

$$|Q \cap D| \tag{2.1.1}$$

Przez $|\cdot|$ rozumiemy moc zbioru, czyli w tym przypadku liczbę jego elementów.

Współczynnik dopasowania prostego jest jasną, prostą i intuicyjną miarą podobieństwa, ale ma jedną podstawową wadę. Otóż miara ta nie bierze pod uwagę długości opisu dokumentu, która przecież w danej kolekcji dokumentów może być bardzo różna. Dokumenty mogą być zaindeksowane różną liczbą termów indeksujących, pod tym względem nie stosuje się żadnego ujednolicenia. A przecież, jeśli na przykład, dwa termy z zapytania występują w opisie dokumentu złożonego z, powiedzmy, trzech słów kluczowych, to znacznie silnie wskazuje to na ten dokument, niż gdyby takie dopasowanie wystąpiło w przypadku opisu złożonego ze słów dwudziestu.

Aby rozwiązać ten problem, jako miary podobieństwa dokumentu i zapytania, stosuje się raczej miary znormalizowane, niewrażliwe na kwestię długości opisów. Przykładem tego rodzaju miernika może być np. współczynnik Dice'a:

$$2\frac{|Q \cap D|}{|D|+|Q|} \tag{2.1.2}$$

Inną czasami stosowaną miarą podobieństwa jest również współczynnik Jaccarda:

$$\frac{|Q \cap D|}{|D \cup Q|} \tag{2.1.3}$$

Jednak zdecydowanie najczęściej stosowaną miarą podobieństwa dokumentu do zapytania jest tzw. współczynnik cosinusoidalny (albo współczynnik cosinusów):

$$\frac{|Q \cap D|}{\sqrt{|D|}\sqrt{|Q|}} \tag{2.1.4}$$

Współczynnik cosinusów (2.1.4) lub jego różne (na ogół proste) modyfikacje, stanowi powszechnie stosowaną miarę podobieństwa dokumentu do zapytania, w usługach wyszukiwawczych opartych na modelu wektorowym wyszukiwania. Zarówno z powodów implementacyjnych, jak dla jaśniejszego pokazania kolejnych elementów tego modelu, wygodniej nam będzie sformułować go w formie algebraicznej, nie zaś jak we wzorze (2.1.4) w oparciu o operacje teorii mnogości.

Przyjmijmy, że w kolekcji wyszukiwarki znajduje się m dokumentów. Jak wcześniej wspomnieliśmy, dokumenty w modelu wektorowym, opisywane są przy pomocy zbiorów słów kluczowych. Załóżmy więc dalej, że do opisu dokumentów wykorzystujemy słownik, złożony z n słów kluczowych. Wówczas opis każdego j-tego dokumentu $\mathbf{d}_j$, $j = 1, \ldots, m$ możemy przedstawić jako pewien wektor $\mathbf{d}_j = [w_{1j}, w_{2j},..., w_{nj}]^T$ w n wymiarowej przestrzeni wszystkich słów kluczowych R^n, gdzie:

$$w_{ij} = \begin{cases} 1 \text{ gdy } i-\text{ty term występuje w dokumencie } \mathbf{d}_j \\ 0 \text{ w przeciwnym wypadku} \end{cases} \tag{2.1.5}$$

Zbiór termów reprezentujących każdy z dokumentów, przedstawiamy więc jako binarny wektor w wielowymiarowej przestrzeni (o bardzo dużym wymiarze – liczba wszystkich słów kluczowych zazwyczaj jest bardzo duża, często n jest równe kilka, nawet kilkadziesiąt tysięcy), który ma wartości 1 na pozycjach odpowiadających słowom kluczowym występującym w opisie tego dokumentu, 0 na pozycjach pozostałych słów kluczowych.

Analogicznie, również zapytanie użytkownika $\mathbf{q}$ możemy reprezentować jako wektor w tej samej przestrzeni wszystkich możliwych słów kluczowych występujących w słowniku wyszukiwarki $\mathbf{q} = [q_1, q_2,..., q_n]^T$, gdzie

$$q_i = \begin{cases} 1 \text{ gdy } i-\text{ty term występuje w zapytaniu } q \\ 0 \text{ w przeciwnym wypadku} \end{cases} \tag{2.1.6}$$

Podobieństwo dokumentu $\mathbf{d}_j$ do zapytania $\mathbf{q}$, $sim(\mathbf{d}_j, \mathbf{q})$ określamy przy pomocy współczynnika cosinusów (2.1.4), przy czym dla reprezentacji w formie wektorów, możemy go zapisać następująco:

$$sim\left(\boldsymbol{q},\boldsymbol{d}_j\right) = \frac{\left|\boldsymbol{q} \cap \boldsymbol{d}_j\right|}{\sqrt{|\boldsymbol{q}|}\sqrt{\left|\boldsymbol{d}_j\right|}} = \frac{\sum_{i=1}^{n} q_i w_{ij}}{\sqrt{\sum_{i=1}^{n} q_i^2 \sum_{i=1}^{n} w_{ij}^2}} \tag{2.1.7}$$

Zauważmy, że we wzorze (2.1.7) zapisaliśmy współczynnik cosinusów (2.1.4) przy pomocy prostych formuł algebraicznych. Występujące w liczniku wyrażenia są równe 1 jeśli *i*-te słowo kluczowe występuje zarówno w dokumencie (wówczas $q_i = 1$), jak i w zapytaniu (wówczas $q_i w_{i,j} = 1$). W przeciwnym przypadku wyrażenia $q_i w_{i,j}$ są równe 0 (ponieważ wówczas albo $w_{i,j} = 0$, albo $q_i = 0$). A więc po ich zsumowaniu, licznik współczynnika we wzorze (2.1.7) jest równy liczbie słów kluczowych współwystępujących w dokumencie i w zapytaniu. Analogicznie w mianowniku wzoru (2.1.7) wyrażenie $\sum_{i=1}^{n} q_i^2$ jest po prostu równe liczbie słów kluczowych w zapytaniu, zaś $\sum_{i=1}^{n} w_{ij}^2$ liczbie słów kluczowych w opisie dokumentu.

Zasada działania wyszukiwarki wektorowej jest dosyć oczywista. Dla danego zapytania **q**, obliczamy wartość $sim(\mathbf{d}_j, \mathbf{q})$ czyli podobieństwo każdego z dokumentów $\mathbf{d}_j$, $j = 1, \ldots, m$ do tego zapytania, korzystając z wzoru (2.1.7). Następnie porządkujemy dokumenty według wyznaczanych podobieństw i zwracamy jako wynik zapytania utworzony ranking. W praktyce zwracanie dokumentów, których podobieństwo jest równe lub niemal równe zero nie miałoby sensu, dlatego wyszukiwarki zwracają na ogół tylko ranking *k* dokumentów najbardziej podobnych do zapytania. Powyższy opis, oczywiście, dotyczy jedynie zasady działania wyszukiwarki wektorowej. Kwestią charakterystyki jej praktycznego sposobu implementacji, zajmiemy się w dalszej części niniejszego punktu.

Spróbujmy teraz przyjrzeć się zasadzie działania modelu wektorowego na przykładzie. Pozwoli nam to lepiej zrozumieć ideę jego funkcjonowania oraz stworzy podstawy do omówienia dalszych jego elementów.

Przyjmijmy, że wyszukiwarka obsługuję kolekcję 12 dokumentów ($m = 12$). Zostały one zaindeksowane w sumie 13 słowami kluczowymi ($n = 13$). Sposób przypisania słów kluczowych poszczególnym dokumentom przedstawiony został w macierzy termów-dokumentów, przedstawionej na rysunku 2.1.2. Poszczególne jej kolumny stanowią wektory reprezentacji dokumentów. Na przykład, pierwszy dokument zaindeksowany został czterema słowami kluczowymi: {Blaster, Gwiazda, Nadprzestrzeń, Podróż}. Reprezentujący go wektor ma więc postać $\mathbf{d}_1 = [1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0]^T$. Jedynki występują na pozycji 1 (ponieważ słowo kluczowe „Blaster" jest pierwsze w naszym słowniku), pozycji 4 (słowo kluczowe „Gwiazda" jest czwarte), pozycji 8 (słowo „Nadprzestrzeń" jest na ósmym miejscu), oraz na pozycji 11 (to z kolei jest numer w słowniku termu „Podróż"). Na pozostałych pozycjach

występują wartości zerowe. Podobnie drugi dokument zaindeksowany został termami {Gwiazda, Kowboj, Podróż, Rewolwerowiec}. Reprezentujący go wektor ma więc postać $\mathbf{d}_2 = [0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0]^T$. Podobnie pozostałe dokumenty. Wektory reprezentujące poszczególne dokumenty znajdują się na rysunku 2.1.2 w kolumnach macierzy, opatrzonych tytułami odpowiadającymi ich numerom.

Term\Dokument	1	2	3	4	5	6	7	8	9	10	11	12	
Blaster	1	0	0	0	1	0	0	0	0	0	0	1	1
Czarna dziura	0	0	0	1	1	0	1	0	0	1	0	0	0
Grawitacja	0	0	1	1	0	1	1	0	0	1	0	0	0
Gwiazda	1	1	0	0	1	1	0	1	0	1	1	1	1
Indianie	0	0	1	0	0	0	0	0	1	0	1	0	0
Kosmos	0	0	0	1	1	1	1	1	0	1	1	1	1
Kowboj	0	1	1	0	0	0	0	0	1	0	0	1	1
Nadprzestrzeń	1	0	0	0	1	0	1	0	0	0	0	1	1
Obserwacja	0	0	1	1	0	1	0	1	0	0	0	0	0
Planeta	0	0	0	1	0	0	0	1	1	1	0	0	0
Podróż	1	1	0	0	0	1	0	1	1	0	1	0	0
Rewolwerowiec	0	1	1	0	0	0	0	0	1	0	1	0	0
Teleskop	0	0	0	1	0	1	1	0	0	1	1	1	1

3	2	1	3	4	4	4	3	1	4	3	3
0,61	0,41	0,18	0,50	0,73	0,67	0,73	0,55	0,18	0,67	0,50	0,50

Rys. 2.1.2. Działanie przykładowej wyszukiwarki wektorowej

Źródło: opracowanie własne.

Przyjmujmy, że do systemu wykonane zostało zapytanie, złożone z następujących słów kluczowych: {Blaster, Gwiazda, Kosmos, Kowboj, Nadprzestrzeń, Teleskop}, co odpowiada wektorowi zapytania $\mathbf{q} = [1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1]^T$. Na rysunku 2.1.2, wektor zapytania $\mathbf{q}$ przedstawiony został w kolumnie po prawej stronie.

Korzystając ze wzoru (2.1.7) obliczmy teraz podobieństwo każdego z dokumentów kolekcji $\mathbf{d}_j$, $j = 1, \ldots, 12$ do zadanego przez użytkownika zapytania $\mathbf{q}$. I tak, dla pierwszego dokumentu $\mathbf{d}_1$, wyrażenie w liczniku służy do wyznaczenia liczby słów kluczowych współwystępujących w zapytaniu i tym dokumencie:

$$1\cdot1 + 0\cdot0 + 0\cdot0 + 1\cdot1 + 0\cdot0 + 1\cdot0 + 1\cdot0 + 1\cdot1 + 0\cdot0 + 0\cdot0 + 0\cdot1 + 0\cdot0 + 1\cdot0 = 3$$

Tak więc w pierwszym dokumencie występują 3 słowa kluczowe z podanych w zapytaniu. W naszym prostym przykładzie, łatwo możemy sprawdzić na rysunku 2.1.2, iż rzeczywiście tak właśnie jest. Musimy jeszcze obliczyć wyrażenie w mianowniku wzoru (2.1.7), wyznaczające pierwiastki z liczby słów kluczowych występujących w zapytaniu i dokumencie:

$$\sqrt{(1^2+0^2+0^2+1^2+0^2+1^2+1^2+1^2+0^2+0^2+0^2+0^2+1^2)}\cdot$$
$$\sqrt{1^2+0^2+0^2+1^2+0^2+0^2+0^2+1^2+0^2+0^2+1^2+0^2+0^2} =$$
$$=\sqrt{6}\cdot\sqrt{4}=2\sqrt{6}$$

Czyli ostatecznie podobieństwo pierwszego dokumentu z kolekcji, do wykonywanego zapytania, wynosi:

$$sim(\boldsymbol{q},\boldsymbol{d}_1)=\frac{\sum_{i=1}^{n} q_i w_{i1}}{\sqrt{\sum_{i=1}^{n} q_i^2 \sum_{i=1}^{n} w_{i1}^2}}=\frac{3}{2\sqrt{6}}\approx 0{,}61$$

Analogicznie wyliczmy korzystając ze wzoru (2.1.7), podobieństwo drugiego dokumentu $\mathbf{d}_2$ do zapytania **q**. Obliczając liczbę współwystępujących termów w liczniku, otrzymamy tym razem:

$$1\cdot0 + 0\cdot0 + 0\cdot0 + 1\cdot1 + 0\cdot0 + 1\cdot0 + 1\cdot1 + 1\cdot0 + 0\cdot0 + 0\cdot0 + 0\cdot1 + 0\cdot1 + 1\cdot0 = 2$$

W opisie drugiego dokumentu występują więc, jak widzimy, tylko dwa słowa wykorzystane w zadanym przez użytkownika zapytaniu. Podobnie obliczmy człon normalizacyjny w mianowniku:

$$\sqrt{1^2+0^2+0^2+1^2+0^2+1^2+1^2+1^2+0^2+0^2+0^2+0^2+1^2}\cdot$$
$$\sqrt{0^2+0^2+0^2+1^2+0^2+0^2+1^2+0^2+0^2+0^2+1^2+1^2+0^2} =$$
$$=\sqrt{6}\cdot\sqrt{4}=2\sqrt{6}$$

Co ostatecznie pozwala nam wyznaczyć podobieństwo drugiego dokumentu do wykonanego przez użytkownika zapytania **q**:

$$sim(\boldsymbol{q},\boldsymbol{d}_2)=\frac{\sum_{i=1}^{n} q_i w_{i2}}{\sqrt{\sum_{i=1}^{n} q_i^2 \sum_{i=1}^{n} w_{i2}^2}}=\frac{2}{2\sqrt{6}}\approx 0{,}41$$

Jak widać, podobieństwo drugiego dokumentu do zapytania jest nieco mniejsze niż w przypadku pierwszego. Wynika to z faktu, że występują w nim jedynie dwa słowa kluczowe użyte w zapytaniu, a nie trzy (jak w dokumencie pierwszym).

Obliczenie podobieństw pozostałych dokumentów pozostawiamy już czytelnikowi. W naszym przykładzie na rysunku 2.1.2 otrzymane podobieństwa zostały przedstawione w dolnej części kolumny każdego dokumentu. Dla łatwiejszej kontroli wyników, przedstawione zostały również liczby słów kluczowych współwystępujących w każdym z dokumentów i zapytaniu, czyli wartości w liczniku wzoru (2.1.7).

Porównajmy jeszcze szybko podobieństwo dokumentu pierwszego oraz czwartego. Widzimy, że liczba słów kluczowych współwystępujących w tych dokumentach i w zapytaniu jest jednakowa – w obydwu przypadkach są to trzy termy. A jednak podobieństwo dokumentu pierwszego jest nieco wyższe, ponieważ wynosi 0,61, podczas gdy czwartego tylko 0,50. Wynika to z faktu, że opis dokumentu czwartego jest znacznie dłuższy niż pierwszego. I tak, w dokumencie pierwszym mamy 3 współwystąpienia słów kluczowych na 4, zaś z dokumencie czwartym 3 na 6. Dokument pierwszy jest więc lepiej dopasowany do zapytania.

Aby zakończyć przykład pozostaje nam już tylko sporządzenie wyniku wyszukiwania. Powiedzmy, że wyszukiwarka prezentuje ranking wynikowy 5 najbardziej dopasowanych dokumentów. Analizując obliczone na rysunku 2.1.2 podobieństwa, widzimy że najbardziej dopasowane do zapytania są dokumenty piąty i siódmy (z takim samym podobieństwem 0,73). Dalej w kolejności występują dokumenty sześć i dziesięć (podobieństwo 0,67), a za nimi jest dokument pierwszy (0,61). Oczywiście przy takim samym podobieństwie kolejność dokumentów jest dowolna (zazwyczaj wynikać będzie z kolejności obliczeń), ale ranking wynikowy pięciu dokumentów najlepszych dla naszego zapytania, mógłby wyglądać następująco:

5, 7, 6, 10, 1

Przedstawiony przykład prezentuje istotę działania modelu wektorowego wyszukiwania informacji. Jak widzimy, pozwala on zamienić ze swej istoty logiczną procedurę wyszukiwania na operacje algebraiczne

w pewnej wielowymiarowej przestrzeni wektorowej, w której reprezentujemy cechy przechowywanych w kolekcji dokumentów oraz wykonywanego przez system zapytania użytkownika.

Jak do tej pory zakładaliśmy jednak, że zarówno wektor zapytania $\mathbf{q} = [q_1, q_2,..., q_n]$ jak i wektory opisujące dokumenty kolekcji $\mathbf{d}_j = [w_{1j}, w_{2j},..., w_{nj}]$ mają charakter binarny, i zawierają wyłącznie współrzędne 0 lub 1. Dokładniej rzecz biorąc, oznacza to, że dany term może wystąpić w dokumencie (zapytaniu) lub nie. Wszystkie słowa kluczowe w opisie dokumentu (albo zapytaniu) traktowane są więc równoważnie i nie mamy żadnej możliwości wyspecyfikowania informacji, że pewien term jest bardziej lub mniej istotny dla treści dokumentu niż inny.

Kwestia ta ma może mniejsze znaczenie dla specyfikacji zapytań, w większości usług wyszukiwawczych użytkownicy i tak nie mają możliwości definiowania ważności poszczególnych używanych w nich słów kluczowych (chociaż w dalszej części książki będziemy mówili o systemach wyszukiwawczych, które tego rodzaju działania podejmują automatycznie). Podają jedynie słowa kluczowe, opisujące ich potrzebę informacyjną. Nietrudno jednak chyba sobie wyobrazić, że możliwość określania istotności termów byłaby istotną właściwością z punktu widzenia modelowania nieprecyzji opisu zawartości dokumentów. Opis treści przy użyciu słów kluczowych jest nieprecyzyjny z natury. Wydaje się dosyć naturalne, że pewne słowa kluczowe mogą być ważniejsze niż inne, które także oddają zawartość dokumentu, ale nie w tak istotny sposób.

Binarny charakter przypisania dokumentom termów był niezwykle istotny w sytuacji gdy spoglądaliśmy na proces wyszukiwania jak na procedurę opartą na logice klasycznej. Ponieważ model wektorowy ma (w przeciwieństwie do boolowskiego) charakter algebraiczny, a nie logiczny, możemy bez trudu zrezygnować z tego ograniczającego założenia. Dla wyznaczenia cosinusoidalnej miary podobieństwa dokumentu do zapytania, przy pomocy zależności (2.1.7) (podobnie zresztą jak i dla innych stosowanych miar) nie ma znaczenia czy analizowane wektory mają współrzędne o wartościach binarnych, czy też ogólnie – rzeczywistych.

Tak więc, w modelu wektorowym wyszukiwania informacji zarówno wektory zapytania $\mathbf{q} = [q_1, q_2,..., q_n]$, jak i opisu dokumentu $\mathbf{d}_j = [w_{1j}, w_{2j},..., w_{nj}]$ mogą być dowolnymi wektorami w przestrzeni R^n. Term i w opisie zapytania lub dokumentu j, reprezentowany jest przez pewną nieujemną liczbę rzeczywistą q_i lub w_{ij}, nazywaną „wagą". Im większa wartość wagi termu, tym bardziej jest on istotny dla opisu treści dokumentu (zapytania). Zerowa wartość wagi oznacza, że dane słowo kluczowe w opisie dokumentu lub zapytania nie występuje.

Nieco dłużej musimy się zatrzymać nad kwestią określania wartości wag w_{ij}, termów w opisie dokumentów kolekcji systemu wyszukiwawczego. Do realizacji tego zadania opracowanych zostało wiele podejść, nadal jednak najczęściej wykorzystywana jest klasyczna reguła, stworzona przez twórcę modelu wektorowego, Geralda Saltona, określana często jako tzw. reguła tf*idf. Zgodnie z nią, do oceny istotności słowa kluczowego nie wystarczy wziąć pod uwagę jedynie znaczenia termu dla samego dokumentu, ale również jego siłę dyskryminacyjną, pozwalającą na rozróżnianie dokumentów w obrębie całej obsługiwanej przez system kolekcji.

Zgodnie z regułą tf*idf, wagi termów tworzone są więc jako iloczyn współczynnika *tf* oraz *idf*, gdzie:

- Częstość termu *tf* (ang. *term frequency*) jest miarą oceniającą znaczenie słowa kluczowego dla danego określonego dokumentu. Przy najczęściej stosowanym automatycznym indeksowaniu pełnotekstowym, kiedy jako termy indeksujące przypisywane są słowa wchodzące w skład tekstu dokumentu, na ogół jako współczynnik *tf* przyjmowana jest po prostu (zgodnie z nazwą) liczba (lub odsetek) wystąpień danego słowa w dokumencie. Możliwe jednak są inne podejścia. Na przykład przy ręcznym indeksowaniu, wartość *tf* może być określana przez eksperta, decydującego o wyborze termów indeksujących.
- Odwrotność częstości dokumentu *idf* (ang. *inverse document frequency*), jest miarą wartości informacyjnej termu, czyli jego przydatności dla rozróżniania treści różnych dokumentów w kolekcji. Opiera się ona zazwyczaj na liczbie dokumentów, w których występuje dane słowo kluczowe, określanej jako tzw. częstość dokumentu *df* (ang. *document frequency*). Jeśli występuje ono w wielu dokumentach, to jego wartość dyskryminacyjna jest relatywnie niższa. Jeśli w niewielu, to podanie go przez użytkownika w zapytaniu, bardzo wyraźnie wskazuje które dokumenty są dla niego relewantne. Do oszacowania odwrotnej częstości dokumentu *idf* stosowane są różne podejścia, począwszy od zwykłej odwrotności częstości dokumentu 1/*df*, zazwyczaj jednak nieco „spłaszcza" się wartości *df*, poprzez operację logarytmowania.

Klasyczna, wprowadzona przez Saltona, nadal jednak często wykorzystywana, metoda wyznaczania wartości wagi w_{ij}, *i*-tego słowa kluczowego dla *j*-tego dokumentu, określona jest przy pomocy zależności:

$$w_{ij} = tf_{ij} \cdot idf_i = tf_{ij} \cdot \log\left(\frac{m}{df_i}\right) \tag{2.1.8}$$

gdzie tf_{ij} jest częstością termu i w dokumencie j (np. liczbą lub odsetkiem wystąpień w tekście dokumentu), idf_i – odwrotnością częstości dokumentów dla i-tego słowa kluczowego, df_i – liczbą dokumentów zawierających i-te słowo kluczowe, zaś m – liczbą wszystkich dokumentów znajdujących się w kolekcji.

Nie jest to jednak jedyna metoda ważenia termów, na przykład inna obecnie dosyć często stosowana formuła korzysta z zależności [Croft, Metzler, Strohman, 2009]:

$$w_{ij} = \frac{\left(\log\left(tf_{ij}\right)+1\right)\cdot\log\left(m/df_i\right)}{\sqrt{\sum_{i=1}^{n}\left(\left(\log\left(tf_{ij}\right)+1\right)\cdot\log\left(m/df_i\right)\right)^2}} \tag{2.1.9}$$

Wartość 1 dodawana jest do komponentu częstości termu ze względów technicznych, by termom z częstością 1 nie została przydzielona waga o zerowej wartości.

Na koniec, aby uzupełnić zarys podstaw modelu wektorowego wyszukiwania informacji, krótkie wyjaśnienie, dlaczego miarę podobieństwa dokumentu $\mathbf{d}_j$ do zapytania $\mathbf{q}$, daną przez zależność (2.1.7), określa się jako współczynnik cosinusoidalny. Łatwo zauważyć, że w wyrażenie w liczniku wzoru (2.1.7) jest iloczynem skalarnym wektorów $\mathbf{d}_j$ oraz $\mathbf{q}$, zaś w mianowniku znajduje się iloczyn długości tych wektorów. Przypomnijmy dalej, że iloczyn skalarny wektorów równy jest iloczynowi długości tych wektorów, pomnożonemu przez cosinus kąta między nimi. Tak więc przekształcając (2.1.7), otrzymujemy:

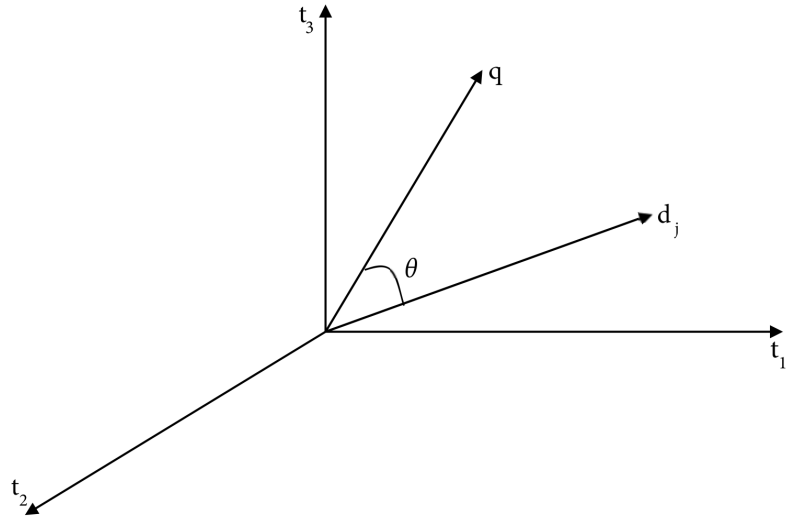
$$sim\left(\boldsymbol{q},\boldsymbol{d}_j\right) = \frac{\sum_{i=1}^{n} q_i w_{ij}}{\sqrt{\sum_{i=1}^{n} q_i^2 \sum_{i=1}^{n} w_{ij}^2}} = \frac{\boldsymbol{q},\mathbf{d}_j}{\boldsymbol{q}\boldsymbol{d}_j} = \cos\left(\boldsymbol{q},\boldsymbol{d}_j\right) \tag{2.1.10}$$

Jak więc widzimy z (2.1.10), wykorzystywana w modelu wektorowym wyszukiwania informacji miara podobieństwa między wektorem dokumentu i zapytaniem, określona przy pomocy współczynnika (2.1.7), równa jest cosinusowi kąta między tymi wektorami. Tak więc o podobieństwie tym nie decydują dokładne wartości wag (wektory równoległe są takie same, czyli mają podobieństwo 1), ale różnice w wartości kąta między wektorami w przestrzeni termów. Przykładowa ilustracja dla trójwymiarowej przestrzeni termów, pokazana została na rysunku 2.1.3.

Aby zakończyć naszą prezentację modelu wektorowego wyszukiwania informacji, pozostało nam jeszcze omówienie podstawowych kwestii

implementacyjnych usługi wyszukiwawczej, wykorzystującej ten model. Oczywiście, podobnie jak w przypadku modelu boolowskiego, nie będziemy zagłębiać się w szczegóły konstrukcji tego rodzaju systemów, poprzestaniemy jedynie na ogólnym zarysie tej tematyki.

Oczywiście nasze poprzednie rozważania odnośnie działania modelu wektorowego miały charakter przedstawienia modelu tego działania, a nie praktycznego sposobu jego realizacji. Podejście oparte na wyznaczaniu podobieństwa wszystkich dokumentów w kolekcji, byłoby zupełnie niepraktyczne, z powodu efektywności działania usługi wyszukiwawczej. Określenie rankingu zwracanych dokumentów w czasie akceptowalnym przez użytkownika byłoby po prostu niemożliwe.



Rys. 2.1.3. Podobieństwo w modelu wektorowym jako miara kąta między wektorami dokumentu i zapytania

Źródło: opracowanie własne.

Zwróćmy przy tym uwagę, że wymiar wektorów opisów dokumentów i zapytania w modelu wektorowym, jest bardzo wysoki (równy liczbie wszystkich słów kluczowych w słowniku systemu wyszukiwawczego). Może on więc sięgać nawet dziesiątków tysięcy. Bezpośrednie stosowanie wzoru (2.1.7) do obliczania podobieństw, wymagałoby więc również ogromnych ilości obliczeń, a przecież należałoby je powtórzyć dla wszystkich dokumentów.

W dodatku wektory te mają charakter bardzo rzadki. Ze wszystkich dziesiątków tysięcy dostępnych w słowniku termów, w opisie

pojedynczego dokumentu występować będzie najprawdopodobniej niewielki ich odsetek – na przykład tekst typowej strony webowej obejmuje kilkadziesiąt–kilkaset różnych słów. Zapytania użytkowników zazwyczaj są jeszcze krótsze. W ich skład wchodzi zwykle jedno, dwa słowa kluczowe, a maksymalna ich liczba bardzo rzadko przekracza poziom kilku.

Oznacza to, że w typowym wektorze dokumentu, zaledwie kilkadziesiąt, może kilkaset, współrzędnych na kilkadziesiąt tysięcy będzie miało wartości niezerowe. Pozostałe będą równe zero. W wektorze zapytania odsetek współrzędnych niezerowych będzie jeszcze niższy, i to znacznie. Bezkrytyczne bezpośrednie stosowanie wzoru (2.1.7) nie ma więc po prostu sensu. Wykonywalibyśmy mnóstwo obliczeń na wartościach zerowych, nie mających kompletnie żadnego wpływu na wynik końcowego podobieństwa.

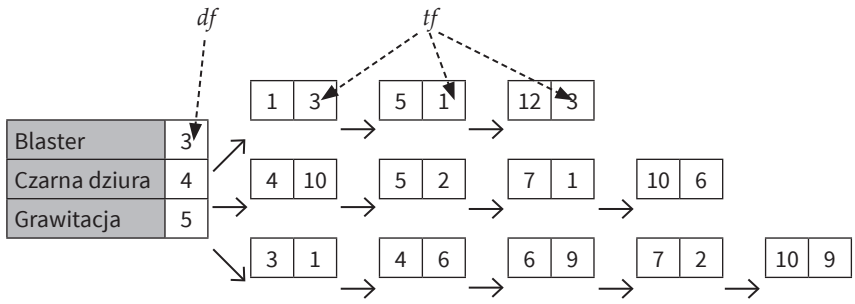
(a) Przykładowa tablica term-dokument z wagami $w_{ij} = tf_{ij}*idf_i$ termów indeksujących

tf

Term\Dokument	1	2	3	4	5	6	7	8	9	10	11	12
Blaster	3	0	0	0	1	0	0	0	0	0	0	2
Czarna dziura	0	0	0	10	2	0	1	0	0	6	0	0
Grawitacja	0	0	1	6	0	9	2	0	0	9	0	0

df	*idf*
3	0,60
4	0,48
5	0,38

(b) Typowa jej reprezentacja w formie indeksu odwrotnego



Rys. 2.1.4. Indeks odwrotny dla wyszukiwania wektorowego

Źródło: opracowanie własne.

Dlatego implementacja wyszukiwarki w takim wypadku opiera się oczywiście na podstawach teoretycznych modelu wektorowego, stosuje jednak bardziej pragmatyczne algorytmy, wykorzystujące w większości przypadków pomocniczą strukturę danych w formie indeksu odwrotnego.

Przykład indeksu odwrotnego typowego dla implementacji wyszukiwarki wektorowej przedstawiony został na rysunku 2.1.4. W części (a) rysunku znajduje się przykładowa tabela term-dokument ze współczynnikami tf_{ij} (liczby wystąpień) poszczególnych termów w każdym z dokumentów. Jak widzimy, jest to tablica rzadka, większość jej elementów ma wartości zerowe, wskazujące że dany term nie występuje w opisie danego dokumentu. W części (b) rysunku przedstawiona została możliwa reprezentacja tabeli z części (a) w postaci indeksu odwrotnego.

Indeks odwrotny dla wyszukiwarki wektorowej ma, jak widzimy, ogólną strukturę zbliżoną do tych stosowanych przy podejściu boolowskim (patrz rysunek 2.1.1 w poprzednim punkcie), chociaż oczywiście istnieją pewne różnice w szczegółach organizacji przechowywanej informacji. W każdym razie, generalnie również i w przypadku wektorowym, indeks składa się ze słownika termów (na ogół w formie jakiejś struktury danych, pozwalającej na bardzo szybkie wyszukiwanie danego słowa kluczowego) zaś do każdego termu przypisana jest lista odsyłaczy do dokumentów, zaindeksowanych przez to słowo kluczowe.

Jak widzimy na rysunku 2.1.4, odsyłacze dokumentów w listach przypisanych poszczególnych słowom kluczowym, zawierają dwie informacje: identyfikator dokumentu (bądź innego rodzaju odsyłacz do jego zawartości), oraz informację niezbędną do określenia wagi termu w tym dokumencie w_{ij}. Indeks ma charakter gęsty, tzn. w listach odsyłaczy znajdują się już tylko dokumenty, dla których waga danego termu jest większa od zera. Na przykład słowo kluczowe „Blaster" w pierwszym wierszu tablicy termów-dokumentów w części (a) rysunku, ma niezerowe wagi w_{1j} tylko dla 1, 5 i 12 dokumentu. Dlatego w części (b) rysunku, w liście odsyłaczy dla tego termu występują tylko te trzy dokumenty.

Zauważmy dalej, że w listach odsyłaczy do dokumentów w indeksie, nie są przechowywane pełne współczynniki wagowe w_{ij} słów kluczowych, a jedynie ich część związana ze znaczeniem termu dla tego dokumentu tf_{ij}. Część związana z liczbą dokumentów, w których dane słowo kluczowe występuje df_i, przechowywana jest razem z samym słowem kluczowym w słowniku. Jest to celowy zabieg. Gdyby w listach odsyłaczy przechowywano finalne wagi, przy dodawaniu nowego dokumentu zaindeksowanego konkretnym słowem kluczowym, należałoby przejrzeć całą listę związanych z tym termem odsyłaczy, przeliczyć idf_i i zaktualizować wszystkie wagi w_{ij}. Przy takiej organizacji indeksu, jak na rysunku 2.1.4, wystarczy zaktualizować df_i w jednym miejscu – w słowniku termów. Oznacza to jednak, że ostateczne wagi muszą być obliczane „w locie", to znaczy w trakcie przetwarzania zapytania.

Ogólny schemat realizacji zapytania w usłudze wyszukiwawczej opartej na modelu wektorowym zazwyczaj wygląda następująco. Rozpoczynamy od znalezienia w słowniku indeksu odwrotnego wszystkich termów występujących w zapytaniu. Po zlokalizowaniu ich w słowniku, dla wszystkich dokumentów występujących w listach odsyłaczy tych termów obliczamy współczynniki cosinusów określające ich podobieństwo do zapytania, a następnie tworzymy ranking zawierający k najlepszych dokumentów.

W ten sposób, po pierwsze, nie musimy przetwarzać wszystkich dokumentów w kolekcji wyszukiwarki – tylko te występujące w listach odsyłaczy, czyli dla których wagi interesujących nas termów wpływających na podobieństwo dokumentu do zapytania są większe od zera. Po drugie, licząc podobieństwa dokumentów wykorzystujemy tylko termy występujące w zapytaniu. Inne będą miały wagi zerowe, a więc nie będą wpływać na podobieństwa. Dzięki tym dwom elementom, bardzo znacznie ograniczamy liczbę obliczeń niezbędnych do wykonania zapytania, redukując czas jego przetwarzania do możliwego do akceptacji poziomu.

Możliwy jest cały szereg strategii i algorytmów realizacji rankingu. Generalnie jednak wymagają one wykonania następujących operacji:

- Dla każdego dokumentu tworzymy i inicjujemy wartością 0 akumulator – zmienną w której następnie będziemy sumować (kumulować) miarę podobieństwa dokumentu do przetwarzanego zapytania (chodzi zwłaszcza o licznik we wzorze (2.1.7)). W zależności od implementacji akumulatory mogą być przechowywane w rozmaitych strukturach danych.
- Dla wszystkich odsyłaczy dokumentów, przypisanych każdemu ze słów kluczowych występujących w zapytaniu, dodajemy do akumulatora odpowiedniego dokumentu kolejne wyrazy $q_i^* w_{ij}$.
- Po przeliczeniu wszystkich odsyłaczy, wybieramy akumulatory o wartościach większych od 0.
- Porównując wartości wybranych akumulatorów, wyszukujemy k najwyższych wyników.
- Sortujemy otrzymane k akumulatorów w porządku malejącym i zwracamy odpowiadające im dokumenty jako wynik zapytania.

Tak jak to już powiedzieliśmy, sposób implementacji powyższych operacji może być bardzo różny. W przypadku pierwszej fazy procesu realizacji zapytania, związanej z wyznaczaniem podobieństw dokumentów do zapytania – obliczaniem wartości akumulatorów dokumentów, wyróżnia się jednak, zresztą w dosyć naturalny sposób, dwa podstawowe sposoby prowadzenia obliczeń:

- Według termów – w tym podejściu listy przetwarzane są po kolei, jedna po drugiej. Wadą tego rozwiązania jest fakt, że jeśli któryś z dokumentów występuje na listach odsyłaczy związanych z kilkoma termami użytymi w zapytaniu, przy przetwarzaniu każdej z nich musimy kilkukrotnie powracać do aktualizacji związanego z tym dokumentem akumulatora, co w niektórych rozwiązaniach implementacyjnych (ale nie zawsze) powoduje powstanie dodatkowych nakładów obliczeniowych (np. na jego wyszukanie).
- Według dokumentów – w tym podejściu listy odsyłaczy poszczególnych termów zapytania przetwarzane są jednocześnie, niejako „równolegle". Dzięki temu można wszystkie obliczenia związane z jednym dokumentem zamknąć w jedną całość, tak aby nie było już dalszej potrzeby wracania do aktualizacji akumulatora tego dokumentu. Aby to było możliwe, niezbędne jest jednak uporządkowanie list odsyłaczy do dokumentów, przypisanych słowom kluczowym indeksu, według identyfikatora dokumentu. Uniemożliwia to jednak zastosowanie niektórych zaawansowanych metod przyśpieszania obliczeń. Poza tym, przetwarzanie kilku list odsyłaczy jednocześnie wymaga pewnych dodatkowych obliczeń związanych z koordynacją tego procesu.

Wybór strategii obliczeń przy określaniu podobieństw dokumentów nie jest zresztą jedynym, z jakim mamy do czynienia podczas implementacji procesu wyszukiwania wektorowego. Kolejnym, na przykład jest kwestia doboru odpowiedniej struktury danych dla akumulatorów dokumentów. Jeszcze innym, na przykład kwestia sporządzenia finalnego rankingu – wykorzystać procedury sortowania danych, czy użyć do jego tworzenia jakiejś „samoporządkującej" się struktury danych, takiej jak np. kopiec.

Niestety, nie istnieje jakiś jeden, najlepszy sposób realizacji obliczeń dla usługi wyszukiwawczej, wykorzystującej model wektorowy. Jak wskazują badania empiryczne różnych algorytmów [Cambazoglu, Aykanat, 2006], ich sprawność działania zależy od szeregu czynników do których między innymi należą: liczba dokumentów w obsługiwanej przez wyszukiwarkę kolekcji, liczba termów indeksujących w słowniku, rozmiar zapytań, jakich się spodziewamy (liczba słów kluczowych w zapytaniach). W zależności od nich opłacalne może być zastosowanie w przeglądarce różnych podejść i rozwiązań implementacyjnych.

Podsumowując uwagi poczynione w niniejszym punkcie, możemy powiedzieć, że wektorowy model wyszukiwania informacji pozwolił rozwiązać wiele problemów, jakie wiązały się z modelem boolowskim.

Przybliżony, topologiczny sposób oceny dopasowania dokumentów do zapytania użytkownika, umożliwił uwzględnienie w procesie wyszukiwania nieodłącznej nieprecyzji, wiążącej się z opisem dokumentu poprzez wybrany zestaw cech, takich jak słowa kluczowe. W poważny sposób uprościł on rygorystyczny, boolowski sposób przekładania potrzeby informacyjnej użytkownika na zapytanie, modelując ją w postaci znacznie bardziej intuicyjnego w zastosowaniu, zapytania prostego. Odejście od binarnej reguły decyzyjnej w określeniu relewancji dokumentu dla zapytania umożliwia także tworzenie zbioru wynikowego wyszukiwania w formie rankingu ułatwiającego użytkownikowi korzystanie z systemu i interpretację otrzymanych wyników.

Podstawową wadą modelu wektorowego jest oparcie jego konstrukcji na zasadach niemalże wyłącznie empirycznych. Nie ma żadnych teoretycznych przesłanek, wskazujących dlaczego na przykład wykorzystywana w modelu cosinusoidalna miara podobieństwa jest lepsza od innych podobnych wskaźników. Po prostu znaczna liczba eksperymentów empirycznych z zakresu oceny jakości działania systemów wyszukiwawczych, wydaje się wskazywać, że daje ona najlepsze wyniki. Podobnie jest w przypadku szeregu innych elementów modelu, np. metody budowy wag termów. W zasadzie cała konstrukcja modelu wektorowego, poza samymi podstawami funkcjonowania, wynikającymi z algebry liniowej, stworzona została w oderwaniu od jakichkolwiek podstaw teoretycznych, opierając się na czystym eksperymencie.

Z punktu widzenia praktyki wyszukiwania informacji, kwestia ta nie wydaje się aż tak istotna. Cóż za różnica, że nie wiemy dlaczego coś działa, skoro działa. Dlatego model wektorowy jest nadal jedną z podstawowych metod tworzenia usług wyszukiwawczych. Ponadto jest on przez cały czas rozwijany, zwłaszcza że, jak zobaczymy w dalszej części bieżącego rozdziału, łatwo połączyć go z różnymi rozwiązaniami, podnoszącymi jakość wyszukiwania.

Tym niemniej luki w podstawach teoretycznych zawsze powinny doskwierać. Jeśli nie wiemy dlaczego jakaś metoda działa poprawnie, i przy jakich dokładnie założeniach będzie tak działać, zawsze możemy natknąć się na obszary, w których niestety działać nie będzie. Ponadto, jeśli mamy dobre teoretyczne podstawy funkcjonowania danego podejścia, łatwiej nam jest wskazać, skąd się biorą ewentualne luki w jego pracy i stworzyć rozwiązania, które je wyeliminują. Takiego komfortu przy zastosowaniach modelu wektorowego wyszukiwania, niestety nie mamy.

2.1.3. Model probabilistyczny wyszukiwania informacji

Omawiany w poprzednim punkcie model wektorowy w ogólnych zasadach swojego działania opierał się na algebrze liniowej. Korzystając z pewnych koncepcji pochodzących z przestrzeni liniowych starał się opisać nieprecyzję związaną z procesem wyszukiwania informacji oraz wynikającą z niej niepewność co do otrzymywanych wyników. Niestety szczegółowe rozwiązania wykorzystywane w modelu wektorowym, tak jak to wskazywaliśmy w poprzednim punkcie, takich podstaw teoretycznych już nie posiadają, opierając się niemal wyłącznie na wynikach praktycznych eksperymentów. Tymczasem istnieje przecież dobrze ugruntowania teoria modelowania niepewności, oparta na rachunku prawdopodobieństwa.

Od razu rozwiejmy pewien mit. Nie istnieje coś takiego jak jeden konkretny model probabilistyczny wyszukiwania informacji. Tą nazwą określa się szereg różnych podejść, opierających się na sformułowanej w latach siedemdziesiątych ubiegłego wieku, przez Stephena Robinsona tzw. zasada rankingowania probabilistycznego (ang. *Probability Ranking Principle*). Zgodnie z nią [Robertson, 1997], optymalne działanie systemu wyszukiwawczego może zostać osiągnięte poprzez rankingowanie dokumentów na podstawie z prawdopodobieństwa ich oceny jako relewantnych dla zapytania. Istotne jest przy tym, że prawdopodobieństwa te powinny zostać oszacowane tak dokładnie, jak to jest możliwe na podstawie dostępnych do tego celu danych.

Pomimo wszystko podejście probabilistyczne może więc wydawać się podobne do wykorzystanego w modelu wektorowym: nadal mamy kolekcję dokumentów, użytkownik wykonuje do niej zapytanie (w formie zapytania prostego), zwracana jest lista dokumentów, uporządkowanych odpowiednio dla potrzeb informacyjnych użytkownika. I w pewnym sensie tak jest, model, czy modele probabilistyczne możemy widzieć jako pewną specyficzną formę podejścia wektorowego.

Zwróćmy jednak uwagę na pewną fundamentalną różnicę. Otóż w modelu wektorowym dokument może być relewantny dla zapytania w pewnym stopniu (określonym przez jego podobieństwo). Natomiast w podejściu probabilistycznym dokument jest relewantny lub nie – binarna reguła, podobnie jak przy wyszukiwaniu boolowskim. Natomiast na podstawie dostępnych informacji i zapytania dokument jest klasyfikowany do jednej z tych dwu kategorii: relewantny, nierelewantny. Podstawą tworzenia rankingu jest prawdopodobieństwo tej klasyfikacji. Tak więc model probabilistyczny jest przykładem podejścia do zagadnienia wyszukiwania informacji od nieco innej strony – od strony zadania klasyfikacyjnego.

Musimy pamiętać o jeszcze jednym założeniu wynikającym z zasady rankingowania probabilistycznego. Otóż w modelu probabilistycznym zakładamy, że możemy rozpatrywać poszczególne dokumenty po kolei, dla każdego z nich niezależnie oceniając prawdopodobieństwo ich relewantności. Przyjmujemy więc, że to, czy jeden dokument jest relewantny (lub nie), nie ma wpływu na relewantność innych.

Przetwarzając zapytanie użytkownika q, rozpatrujemy więc po kolei każdy dokument w kolekcji. Dla danego dokumentu d możemy mieć do czynienia z dwoma zdarzeniami: R – zdarzenie polegające na tym, że dokument jest relewantny, NR – zdarzenie polegające na tym, że jest on nierelewantny. Aby zaklasyfikować dokument szacujemy prawdopodobieństwa warunkowe: $P(R \mid d, q)$, tj. że dokument ten jest relewantny pod warunkiem danego opisu dokumentu d i zapytania q, oraz $P(NR \mid d, q)$ – prawdopodobieństwo warunkowe jego nierelewantności pod warunkiem danego opisu dokumentu i zapytania. Ponieważ interesuje nas ranking dokumentów, przy ustalonym w danej chwili zapytaniu q (takim samym dla wszystkich ocenianych dokumentów), zależność od q będziemy dla uproszczenia pomijać, zapisując prawdopodobieństwa relewancji i nie relewancji dokumentu jako $P(R \mid d)$, $P(NR \mid d)$. Natomiast pamiętajmy, że cała analiza probabilistyczna odbywa się także pod warunkiem zapytania q.

Dokument będziemy uznawać za relewantny, przy wykorzystaniu Bayesowskiej reguły decyzyjnej, jeśli $P(R \mid d) > P(NR \mid d)$, albo $P(R \mid d) / P(NR \mid d) > 1$. Stosunek prawdopodobieństwa zdarzenia do prawdopodobieństwa zdarzenia przeciwnego nazywamy szansą. Z pewnych powodów wygodniej będzie nam zastosować do oceny relewantności dokumentu tę miarę niż bezpośrednio prawdopodobieństwo. Ponadto interesuje nas raczej stworzenie rankingu niż pojęcie decyzji. Widzimy, że generalnie im wyższa szansa relewantności dokumentu, tym pewniejsza nasza klasyfikacja. Ranking dokumentów, spełniający zasadę rankingowania probabilistycznego, może więc również opierać się na szansie. Ponadto do stworzenia takiego rankingu na ogół nie musimy korzystać z pełnego oszacowania szansy, wystarcz nam wyznaczenie tzw. RSV (Retrieval Status Value) – dowolnej miary proporcjonalnej do prawdopodobieństwa relewancji (RSV ~ $P(R \mid d)$).

W jaki sposób można ocenić prawdopodobieństwa $P(R \mid d)$ oraz $P(NR \mid d)$?

Chcielibyśmy dokonać tego na podstawie statystyk otrzymanych z dostępnej bazy dokumentów. Na przykład wybrać wszystkie dokumenty

o opisie *d*, a następnie policzyć, jaka część z nich jest relewantna dla danego zapytania, a jaka nie. Niestety, pamiętajmy że opis dokumentu *d*, składa się z zestawu słów kluczowych. Zazwyczaj wielu. Powiedzmy, dla przykładu, że opis dokumentu *d* składa się ze słów kluczowych: „gwiazda", „kosmos" i „kwazar". Załóżmy dalej, że słowo „gwiazda" występuje w 1% dokumentów kolekcji, „kosmos" w 5%, zaś „kwazar" w 0,1%. Zakładając nawet niezależny charakter występowania tych termów (ryzykowne, oczywiście, ale to najlepszy przypadek), to prawdopodobieństwo wystąpienia całego opisu *d*, wynosi: 0,01·0,05·0,001 = 0,0000005, czyli jeden na pięćdziesiąt milionów. Nawet w olbrzymich kolekcjach dokumentów, znaleźlibyśmy więc najwyżej parę dokumentów o takim samym opisie. Zbyt mało do oszacowania prawdopodobieństwa relewantności.

Często takie „trudne" prawdopodobieństwa, niemożliwe w zasadzie do bezpośredniego oszacowania, da się obliczyć nie wprost, korzystając z twierdzenia Bayesa:

$$P(R|d)=\frac{P(d|R)P(R)}{P(d)},\quad P(NR|d)=\frac{P(d|NR)P(NR)}{P(d)} \qquad (2.1.11)$$

gdzie (jeszcze raz to dokładnie wyjaśnijmy):

- $P(R \mid d)$, $P(NR \mid d)$ – prawdopodobieństwo, że dokument *d* jest relewantny (nierelewantny),
- $P(R)$, $P(NR)$ – prawdopodobieństwo *a priori* wyszukania relewantnego (nierelewantnego) dokumentu (czyli ogólnie, jaka część dokumentów w kolekcji jest relewantna lub nierelewantna),
- $P(d \mid R)$, $P(d \mid NR)$ – prawdopodobieństwo (tzw. wiarygodność – ang. *likelihood*) wystąpienia opisu dokumentu *d* (jak często występuje dany zestaw słów kluczowych), w zbiorze dokumentów relewantnych (nierelewantnych),
- $P(d)$ – prawdopodobieństwo wystąpienia opisu dokumentu *d* (jak często występuje dany zestaw słów kluczowych) w całej kolekcji.

Ktoś mógłby spytać, jaki zysk daje nam twierdzenie Bayesa – zależność (2.1.11), skoro zamiast prawdopodobieństw $P(R \mid d)$, $P(NR \mid d)$, musimy oszacować wiarygodności $P(d \mid R)$, $P(d \mid NR)$. Otóż przy obliczaniu tych drugich prawdopodobieństw możliwe jest zastosowanie założeń upraszczających odnośnie do niezależności zmiennych.

Przyjmijmy, że opis dokumentu d stanowi wektor wartości określających występowanie w nim poszczególnych termów indeksujących $d = (d_1, \ldots, d_n)$. Jeśli założymy niezależność występowania słów kluczowych w opisach dokumentów, to prawdopodobieństwo wystąpienia całego ich zestawu $P(d \mid R)$ możemy obliczyć jako iloczyn prawdopodobieństw występowania każdego z nich z osobna $\prod_{i=1}^{n} P(d_i \mid R)$, co jest znacznie łatwiejsze. Podobnie w przypadku $P(d \mid NR)$.

Wykorzystanie założenia niezależności, określane jest jako naiwna metoda Bayesowska. Oczywiście jest to założenie upraszczające, ponieważ współwystępowanie słów w tekstach nie jest niezależne. Jeśli np. tekst zawiera słowo „kosmos", to wystąpienie w nim słowa „gwiazda" jest znacznie bardziej prawdopodobne niż, powiedzmy, „marmolada". W przypadku słowa „lodówka", byłoby zupełnie odwrotnie. Należy więc spodziewać się pewnego błędu w działaniu systemu wyszukiwawczego związanego z zastosowaniem tego założenia.

Prowadzi nas to do prostego modelu wyszukiwania probabilistycznego, od którego zaczniemy rozważania w bieżącym rozdziale, modelu binarnej niezależności – BIR (ang. *Binary Independence Model*). Wykorzystuje on w swoim działaniu dwa podstawowe założenia.

Po pierwsze, binarność – dokumenty reprezentowane są przez binarne wektory występowania termów $d = (d_1, ..., d_n)$, gdzie n jest równe liczbie wszystkich termów w słowniku, przy czym $d_i = 1$, jeśli i-ty term występuje w opisie dokumentu i, $d_i = 0$, gdy nie. Podobnie przez binarne wektory $q = (q_1, ..., q_n)$ reprezentowane są również zapytania.

Po drugie, niezależność – występowanie termów w opisach dokumentów jest niezależne.

Zgodnie z rankingowaniem probabilistycznym, w modelu BIR do oceny prawdopodobieństwa relewantności dokumentu d przy danym zapytaniu q, stosujemy pewną miarę RSV(q, x) proporcjonalną do prawdopodobieństwa $P(R \mid q, d)$, którą wyznaczamy dla każdego dokumentu, i która posłuży nam następnie do sporządzenia ich rankingu.

Miarę tę zbudujemy w oparciu o szansę relewantności dokumentu $S(R \mid q, d)$ pod warunkiem danego opisu dokumentu d oraz zapytania q. Tak jak już wspomnieliśmy wcześniej, do wyznaczenia występujących w szansie prawdopodobieństw zastosujemy twierdzenie Bayesa, pamiętając jednak iż zakładaliśmy, że wszystkie rozpatrywane zdarzenia mają charakter warunkowy również względem zapytania q:

$$S(R|q,d)=\frac{P(R|q,d)}{P(NR|q,d)}=\frac{\frac{P(d|R,q)P(R|q)}{P(d|q)}}{\frac{P(d|NR,q)P(NR|q)}{P(d|q)}}= \\ =\frac{P(d|R,q)P(R|q)}{P(d|NR,q)P(NR|q)}=\frac{P(d|R,q)}{P(d|NR,q)}\cdot\frac{P(R|q)}{P(NR|q)} \tag{2.1.12}$$

Zauważmy, że w zależności (2.1.12) drugi czynnik, czyli jak łatwo zauważyć ogólna szansa relewantności dokumentu pod warunkiem zapytania q, $S(R \mid q)$, nie zależy od opisu dokumentu, jest więc stały dla wszystkich dokumentów. Ponadto jako iloraz prawdopodobieństw musi mieć on wartość dodatnią. Byłby on istotny gdybyśmy chcieli oszacować pełną szansę $S(R \mid q, d)$, natomiast do stworzenia miary RSV(d, q), której zadaniem jest rankingowanie dokumentów w sposób do niej proporcjonalny, możemy go pominąć. Pomnożenie (lub nie) przez dodatnią stałą, nie zmienia względnego uporządkowania wartości RSV dla różnych dokumentów.

Tak więc, do stworzenia RSV tak naprawdę musimy obliczyć tylko pierwszy czynnik (2.1.12). Pamiętając, że $d = (d_1, ..., d_n)$, wykorzystamy w tym celu założenie o niezależności występowania słów kluczowych w modelu BIR:

$$\frac{P(d|R,q)}{P(d|NR,q)}=\prod_{i=1}^{n}\frac{P(d_i|R,q)}{P(d_i|NR,q)} \tag{2.1.13}$$

Podstawiając (2.1.13) do (2.1.12), otrzymujemy:

$$S(R|q,d)=S(R|q)\prod_{i=1}^{n}\frac{P(d_i|R,q)}{P(d_i|NR,q)} \tag{2.1.14}$$

Pamiętajmy, że wartości d_i odpowiadają poszczególnym słowom kluczowym i są równe 1 dla słów kluczowych występujących w opisie badanego dokumentu lub 0 dla tych, które w opisie tym nie występują. W związku z tym, korzystając z łączności i przemienności mnożenia, możemy pogrupować czynniki w (2.1.14), rozdzielając je na obie wyżej wspomniane grupy. Tak więc:

$$S(R|q,d) = S(R|q) \prod_{d_i=1} \frac{P(d_i = 1|R,q)}{P(d_i = 1|NR,q)} \cdot \prod_{d_i=0} \frac{P(d_i = 0|R,q)}{P(d_i = 0|NR,q)} \quad (2.1.15)$$

Oznaczmy prawdopodobieństwo występowania danego słowa kluczowego, przy zadanym zapytaniu, w opisie dokumentu relewantnego, przez p_i, zaś prawdopodobieństwo występowania danego słowa przy tym zapytaniu, w opisie dokumentu nierelewantnego, przez r_i. Czyli:

$$\begin{aligned} p_i &= P(d_i = 1|R,q) \\ r_i &= P(d_i = 1|NR,q) \end{aligned} \quad (2.1.16)$$

Oczywiście jeśli prawdopodobieństwo występowania danego słowa kluczowego, przy zadanym zapytaniu, w opisie dokumentu relewantnego $P(d_i = 1|R,q)$ jest równe p_i, to prawdopodobieństwo jego niewystępowania $P(d_i = 0|R,q)$ wynosi 1 – p_i. Podobnie prawdopodobieństwo niewystępowania słowa kluczowego w opisach dokumentów nierelewantnych, $P(d_i = 0|NR,q)$ będzie równe 1 – r_i.

Tak więc przy przyjętych oznaczeniach (2.1.15), możemy naszą formułę na szansę (2.1.16) zapisać następująco:

$$S(R|q,d) = S(R|q) \prod_{d_i=1} \frac{p_i}{r_i} \prod_{d_i=0} \frac{1-p_i}{1-r_i} \quad (2.1.17)$$

Zastanówmy się teraz nad sytuacją, kiedy rozważany *i*-ty term nie występuje w zapytaniu, tzn. $q_i = 0$. Zapytanie w większości przypadków jest jedynym źródłem informacji na temat zbioru dokumentów relewantnych. W związku z tym, jeśli nie mamy żadnej dodatkowej informacji, słowa kluczowe niewystępujące w zapytaniu będą miały takie samo prawdopodobieństwo wystąpienia w dokumentach relewantnych i nierelewantnych – jako wyraz naszej niewiedzy. Tak więc przyjmiemy założenie, że jeśli $q_i = 0$, to wówczas $p_i = r_i$, lub też inaczej $p_i / r_i = 1$.

W związku z tym w (2.1.17) wszystkie czynniki odpowiadające termom niewystępującym w zapytaniu ($q_i = 0$) są równe 1 i możemy je pominąć.

$$S(R|q,d) = S(R|q) \prod_{\substack{d_i=q_i=1 \\ q_i=1}} \frac{p_i}{r_i} \prod_{\substack{d_i=0 \\ q_i=1}} \frac{1-p_i}{1-r_i} \quad (2.1.18)$$

Przyjrzyjmy się strukturze otrzymanej zależności (2.1.18) na szansę relewantności dokumentu *d*, przy zapytaniu *q*. Jak widać, obejmuje ona trzy główne czynniki. Pierwszy z nich, szansa znalezienia w naszej kolekcji dokumentu relewantnego dla zapytania *q*, $S(R \mid q)$, jak już mówiliśmy, jest stała i taka sama dla wszystkich dokumentów, a w związku z tym nieistotna z punktu widzenia ich rankingowania. Drugi czynnik, jak widzimy, obliczany jest w sytuacji, gdy $d_i = q_i = 1$, czyli tłumacząc to na bardziej zrozumiały język, dla wszystkich słów kluczowych zapytania, które mają swoje odpowiedniki w opisie dokumentu, lub krócej – dla wszystkich słów kluczowych dopasowanych. I pozostał nam jeszcze trzeci duży czynnik. Ten z kolei obliczany będzie, kiedy $d_i = 0$, $q_i = 1$, czyli tym razem dla wszystkich słów kluczowych zapytania, które nie mają swoich odpowiedników w opisie dokumentu.

Jak więc widzimy, w dosyć znaczny sposób udało nam się ograniczyć ilość obliczeń niezbędnych do oszacowania $S(R \mid q, d)$. W zależności (2.1.18) mamy jednak jeszcze trochę informacji niezależnej od dokumentu, którą postaramy się jeszcze wyizolować, aby w ostateczności pozbyć się jej.

W tym celu zastosujemy pewną sztuczkę matematyczną. Otóż pomnożymy naszą formułę (2.1.18) przez pewne wyrażenie, które w sumie ma wartość 1. Możemy to oczywiście zrobić, ponieważ mnożenie przez 1 nie zmienia wyniku tego działania.

$$S(R|q,d) = S(R|q) \prod_{d_i=q_i=1} \frac{p_i}{r_i} \prod_{\substack{d_i=0 \\ q_i=1}} \frac{1-p_i}{1-r_i} \prod_{d_i=q_i=1} \left(\frac{1-r_i}{1-p_i} \cdot \frac{1-p_i}{1-r_i} \right) \quad (2.1.19)$$

Jak widać, tak naprawdę pomnożyliśmy (2.1.18) przez kilka jedynek, ponieważ ułamki w dodanych czynnikach redukują się nawzajem. Uporządkujmy jeszcze człony otrzymanego wyrażenia:

$$S(R|q,d) = S(R|q) \prod_{d_i=q_i=1} \frac{p_i}{r_i} \prod_{d_i=q_i=1} \frac{1-r_i}{1-p_i} \prod_{\substack{d_i=0 \\ q_i=1}} \frac{1-p_i}{1-r_i} \prod_{d_i=q_i=1} \frac{1-p_i}{1-r_i} \quad (2.1.20)$$

Jeśli spojrzymy na pierwsze dwa człony naszego wyrażenia (2.1.20), to widzimy, że wykonują się one dla tych samych słów kluczowych, $d_i = q_i = 1$, dopasowanych w zapytaniu i dokumencie. Możemy więc zgrupować ich czynniki razem, korzystając ze zwykłej przemienności mnożenia.

Z kolei w ostatnich dwu członach (2.1.20), mnożymy dokładnie takie same informacje, tylko dla $d_i = 0$, $q_i = 1$ (termy zapytania nie dopasowane w dokumencie) oraz $d_i = q_i = 1$, czyli $d_i = 1$, $q_i = 1$ (termy zapytania dopasowane w dokumencie). Razem więc, scalając te dwa czynniki, wykonujemy to mnożenie dla wszystkich termów zapytania, niezależnie od tego czy współwystępują one w dokumencie, czy nie ($q_i = 1$).

Możemy więc zależność (2.1.20) zapisać jako:

$$S(R|q,d) = S(R|q) \prod_{d_i=q_i=1} \frac{p_i(1-r_i)}{r_i(1-p_i)} \prod_{q_i=1} \frac{1-p_i}{1-r_i} \qquad (2.1.21)$$

Zwróćmy teraz uwagę, że ostatni czynnik w (2.1.21) obliczany jest dla wszystkich słów kluczowych występujących w zapytaniu. Nie zależy więc on zupełnie od zawartości informacyjnej opisu dokumentu. Podobnie więc jak w przypadku pierwszego czynnika $S(R \mid q)$, jego wartość jest taka sama dla wszystkich dokumentów. W związku z tym jedynym członem, który decyduje o zróżnicowaniu szansy (czyli także prawdopodobieństwa) relewantności poszczególnych dokumentów, przy danym zapytaniu, jest środkowy czynnik zależności (2.1.21).

Ostatecznie więc, naszą funkcję RSV, oceniającą prawdopodobieństwo relewantności dokumentu dla zapytania, możemy oprzeć na środkowym czynniku zależności (2.1.21). Ponadto zastosujemy jeszcze do niego operację logarytmowania, co pozwoli nam zapisać RSV w wygodniejszej obliczeniowo postaci. Logarytm, jako funkcja monotoniczna nie zmienia nam rankingu.

Tak więc, otrzymujemy ostatecznie następującą funkcję oceniającą dokument:

$$RSV(q,d) = \log \prod_{d_i=q_i=1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{d_i=q_i=1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)} \qquad (2.1.22)$$

Zwróćmy uwagę, że funkcję oceniającą dokument (2.1.22), możemy zapisać alternatywnej formie:

$$RSV(q,d) = \sum_{d_i=q_i=1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_i \log \frac{p_i(1-r_i)}{r_i(1-p_i)} d_i q_i \qquad (2.1.23)$$

Wynika to z faktu, że jeśli $d_i = 1$, $q_i = 1$ to iloczyn $d_i q_i$ także jest równy 1. W innych przypadkach iloczyn $d_i q_i$ ma wartość 0.

Tak więc, widzimy że funkcja rankingująca dokumenty w modelu BIR ma charakter funkcji liniowej:

$$RSV(q,d)=\sum_i c_i d_i q_i \tag{2.1.24}$$

gdzie współczynniki funkcji liniowej (wagi termów w dokumencie) dane są przez:

$$c_i=\log\frac{p_i(1-r_i)}{r_i(1-p_i)} \tag{2.1.25}$$

Jak widzimy, pomimo, że formuła wyszukiwania probabilistycznego wygląda w sposób dosyć złożony, w zasadzie można ją uznać za pewien wariant modelu wektorowego, wykorzystujący nieco inną formułę obliczania wag termów niż tf*idf.

Problemem podstawowym jest oszacowanie współczynników c_i, niezbędnych do wyznaczenia miary RSV dokumentu. W tym celu niezbędne jest, oczywiście, oszacowanie dla zadanego zapytania, prawdopodobieństwa występowania danego słowa kluczowego w opisie dokumentu relewantnego – p_i, oraz jego występowania w opisie dokumentu nierelewantnego – r_i. Modele probabilistyczne, oprócz statystyk termów w kolekcji dokumentów, umożliwiają włączenie do procesu rankingowania informacji zwrotnej od użytkownika, o pożądanych charakterystykach relewantnego zbioru dokumentów dla danego zapytania.

Załóżmy, że posiadamy informację o relewancji dokumentów, czyli które dokumenty powinny zostać wyszukane w odpowiedzi na dane zapytanie. Wówczas wagi c_i dla poszczególnych termów moglibyśmy wyznaczyć korzystając z prostej tabeli rozkładu liczności dokumentów pokazanej w tabeli 2.1.1.

Tabela 2.1.1. Tabela liczności dokumentów w poszczególnych grupach dla danego termu

	Relewantne	**Nierelewantne**	**Razem**
$d_i = 1$	s_i	$n_i - s_i$	n_i
$d_i = 0$	$S - s_i$	$N - n_i - S + s_i$	$N - n_i$
Razem	S	$N - S$	N

Źródło: opracowanie własne.

Korzystając bezpośrednio z danych w tabeli 2.1.1, możemy oszacować wartości szukanych prawdopodobieństw:

$$p_i = \frac{s_i}{S}$$
$$r_i = \frac{n_i - s_i}{N - S} \tag{2.1.26}$$

Wówczas, korzystając ze wzoru (2.1.25), możemy wyliczyć wagę danego słowa kluczowego c_i, co pozostawiamy już jako ćwiczenie czytelnikowi.

Wykorzystywanie takich oszacowań mogłoby jednak sprawiać problemy, gdyby niektóre z liczebności w tabeli 2.1.1 okazały się równe 0. Na przykład gdyby s_i było równe zero, to przy obliczaniu wagi termu c_i, pojawiłoby się wyrażenie log 0. Oczywiście można wychwytywać tego rodzaju przypadki szczególne, zastępując je zdefiniowanymi z góry wartościami standardowymi. Często jednak stosuje się rozwiązanie, polegające na zastosowaniu w obliczeniach niewielkich wartości korygujących, pozwalających w ogóle na uniknięcie wystąpienia 0.

Na przykład często przyjmuje się jako skorygowane oszacowania prawdopodobieństw p_i oraz r_i, następujące zależności:

$$p_i = \frac{s_i + 0{,}5}{S + 1} \tag{2.1.27}$$

Wykorzystanie tych oszacowań daje wzór na wagę termu c_i, w postaci [Croft, Metzler, Strohman, 2009]:

$$r_i = \frac{n_i - s_i + 0{,}5}{N - S + 1} \tag{2.1.28}$$

Do wyznaczenia dla danego zapytania i dokumentu wartości RSV, przy zastosowaniu funkcji rankingującej (2.1.24), ze współczynnikami (2.1.28) potrzebujemy oczywiście informacji z tabeli 2.1.1, na temat trafności wyników wyszukiwania, pochodzących od użytkownika. Co, jeśli takiej informacji nie mamy? Powinniśmy przyjąć wartości S oraz s_i, jako równe 0. Dlaczego? Ponieważ przy takich wartościach tych parametrów, prawdopodobieństwo występowania danego słowa kluczowego w opisie dokumentu relewantnego p_i, obliczone na podstawie wzoru (2.1.27), będzie równe 0,5. Przy braku informacji o relewantności dokumentów, w sytuacji niewiedzy, prawdopodobieństwo wystąpienia lub niewystąpienia termu w dokumencie relewantnym musimy przyjąć jako takie samo, czyli właśnie równe 0,5.

Tak więc w przypadku braku informacji od użytkownika na temat zbioru dokumentów relewantnych, powinniśmy przyjąć S = s_i = 0. Wówczas jednak nasze wagi, c_i, termów w funkcji rankingującej, będą wynosiły:

$$c_i = \log \frac{\left(s_i + 0{,}5\right)/\left(S - s_i + 0{,}5\right)}{\dfrac{n_i - s_i + 0{,}5}{N - n_i - S + s_i + 0{,}5}} = \\ = \log \frac{0{,}5/0{,}5}{\left(n_i + 0{,}5\right)/\left(N - n_i + 0{,}5\right)} = \log \frac{N - n_i + 0{,}5}{n_i + 0{,}5} \tag{2.1.29}$$

A to oznacza, że wagi termów w funkcji rankingującej, sprowadzają się w zasadzie do zwykłego elementu *idf* – oszacowania generalnej zdolności dyskryminacyjnej termu. Brak jest w nich członu *tf*, ponieważ opisy dokumentów mają charakter binarny.

Model binarnej niezależności, jako taki nie jest może najlepszym podejściem do wyszukiwania informacji. Połączenie dwu założeń – binarnego charakteru termów w opisie dokumentu i w zapytaniu, oraz niezależności występowania słów, powoduje więc że model ten staje się zbyt prosty i zazwyczaj nie uzyskuje satysfakcjonujących wyników jakości wyszukiwania.

Stanowi on jednak podstawę do wielu znacznie lepszych rozwiązań w tym zakresie, również o charakterze zastosowań metod inteligentnych. Model BIR ma również istotne znaczenie dla budowy rozwiązań wykorzystujące iteracyjne podejście, polegające na stopniowej kilkukrotnej poprawie zapytania na podstawie wiedzy od użytkownika o relewancji wyszukanych dokumentów (tzw. sprzężenie relewancji), o których będziemy jeszcze mówili dalej.

W punkcie bieżącym przestawimy jeszcze krótko jedno z klasycznych rozszerzeń modelu niezależności binarnej, pozwalające na rezygnację z jednego z dwu jego podstawowych założeń. Mianowicie umożliwia ono wprowadzenie rzeczywistych wag termów w dokumentach. Rozszerzenie powyższe oparte jest raczej na ogólnych argumentach probabilistycznych i weryfikacji praktycznej działania otrzymanego modelu, niż na formalnym jego wyprowadzeniu.

Algorytm rankingowania BM25, gdyż tak brzmi jego nazwa, stanowi znacznie lepsze podejście do wyszukiwania informacji, w porównaniu z modelem niezależności binarnej, i wykazuje wyraźnie wyższe od niego wyniki w badaniach jakości procesu wyszukiwania. Powoduje to, że obok modelu wektorowego należy on do najbardziej popularnych

metod rankingowania dokumentów, często stosowanych w komercjalnych usługach wyszukiwawczych w Internecie [Croft, Metzler, Strohman, 2009].

W praktyce stosowanych jest kilka wariantów funkcji oceniających dla algorytmu BM25, różniących się od siebie drobnymi szczegółami – zazwyczaj technicznymi. Najczęściej wykorzystywaną postacią jest [Croft, Metzler, Strohman, 2009]:

$$\sum_{i \in q} \log \frac{(s_i + 0{,}5)/(S - s_i + 0{,}5)}{(n_i - s_i + 0{,}5)/(N - n_i - S + s_i + 0{,}5)} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) qf_i}{k + qf_i} \qquad (2.1.30)$$

gdzie sumowanie wykonywane jest dla wszystkich termów zapytania, wartości S, s_i, N, n_i, pochodzą z tabeli 2.1.1 (jak w przypadku modelu BIR), przy czym jeżeli nie dysponujemy informacją o relewancji, przyjmujemy $S = s_i = 0$, f_i oznacza częstość termu w danym dokumencie, qf_i częstość termu w zapytaniu, zaś k_1, k_2, i K parametry, których wartości dobierane są empirycznie.

Parametr k_1 określa sposób zmiany członu *tf* w wadze termu, w miarę wzrostu f_i. Jeśli wartość $k_1 = 0$, to częstość termu w dokumencie będzie ignorowana i istotne będzie tylko to, czy term występuje w nim, czy nie. Jeśli k_1 jest duże, komponent *tf* w wadze termu będzie wzrastać niemal liniowo ze wzrostem f_i. Typowa wartość k_1 wynosi 1,2.

Parametr k_2, pełni bardzo podobną rolę, dla wagi termu w zapytaniu. Typowo przyjmuje się wartości tego parametru między 0 i 1 (a w przypadkach, gdy nie stosujemy wag termów w zapytaniu – po prostu 0).

K jest parametrem normalizującym czynnik *tf*, długością dokumentu. Do jego wyznaczenia korzysta się z formuły:

$$K = k_1 \left((1 - b) + b \frac{dl}{avgdl} \right) \qquad (2.1.31)$$

gdzie b jest parametrem, *dl* oznacza długość dokumentu, zaś *avgdl* jest średnią długością dokumentu w kolekcji. Stała b reguluje siłę procesu normalizacji, od $b = 0$, co oznacza brak normalizacji, do $b = 1$, pełna normalizacja.

2.2. Rozszerzenia modeli rankingujących z wykorzystaniem metod inteligentnych

2.2.1. Model rozmyty wyszukiwania informacji

Jedne z najstarszych prób rozszerzenia klasycznych paradygmatów wyszukiwania informacji wiązały się z wykorzystaniem zbiorów rozmytych i logiki rozmytej. Pierwsze prace w tej dziedzinie rozpoczęto w latach siedemdziesiątych ubiegłego wieku i prowadzone są one do dnia dzisiejszego [Radecki, 1979; Bookstein, 1980; Salton, McGill, 1983; Miyamoto, 1990; Bordogna, Pasi, 1993, 2000; Crestani, Pasi, 1999; Yager, 2000; Dominich, 2001, 2008; Zadrożny, Kacprzyk, 2005].

Zbiory rozmyte i logika rozmyta stosowane mogą być w usługach wyszukiwawczych na wielu różnych etapach i do realizacji rozmaitych zadań. W obecnym punkcie skupimy się tylko na stworzeniu przy pomocy logiki rozmytej efektywnego rozszerzenia klasycznego podejścia boolowskiego do wyszukiwania dokumentów, przy zachowaniu (a nawet rozwinięciu) logicznego paradygmatu wyszukiwania.

Najpoważniejsze problemy z wykorzystaniem modelu boolowskiego, wiązały się z binarnym charakterem reprezentacji termów w dokumentach oraz binarnym modelem relewancji dokumentu. Obie te kwestie słabo oddawały nieodłączną nieprecyzję wyszukiwania w źródłach niestrukturalnych. Ponadto, jak wskazywaliśmy w punkcie 2.1.1, skutkowały one problemami z przekładaniem potrzeby informacyjnej użytkownika na zapytanie – dokładne jej specyfikowanie niemal zawsze skutkuje bardzo rozbudowanymi i skomplikowanymi wyrażeniami logicznymi. Binarny model relewancji, w dodatku, w poważnym stopniu wpływa na trudności z interpretacją zbioru wynikowego przez użytkownika.

Zbiory rozmyte stanowią więc naturalną drogę poprawy jakości procesu wyszukiwania. Odchodzą one, jak wiadomo, od zasady binarnej przynależności do zbioru, zastępując ją płynnym, stopniowym przejściem od pełnej przynależności, do nie przynależności. Bezpośrednio więc oferują drogę przezwyciężenia problemów związanych z klasyczną definicją binarnej przynależności.

Rozpoczniemy od kwestii reprezentacji dokumentów w modelu wyszukiwania, opartym na logice rozmytej. W tej sprawie bowiem nie ma specjalnych kontrowersji. W zasadzie wszystkie warianty modelu rozmytego rozwiązują ją w dosyć jednolity sposób.

Przypomnijmy że w logicznym, boolowskim modelu wyszukiwania zakładaliśmy, że występowanie słów w opisie dokumentu ma charakter binarny, tzn. słowo kluczowe albo występuje w opisie dokumentu, albo nie. Reprezentacja dokumentu jako zbioru rozmytego słów kluczowych, polegać więc będzie na tym, że odejdziemy od tej zasady i dopuścimy przynależność termu do opisu dokumentu, w pewnym stopniu. Wystarczy, że dla każdego z dokumentów $d \in D$, zbudujemy funkcję przynależności poszczególnych termów:

$$\mu_d : T \rightarrow [0,1] \tag{2.2.1}$$

gdzie D oznacza zbiór wszystkich dokumentów, T zbiór wszystkich termów.

Uważnego czytelnika powinna w tym miejscu zastanowić pewna istotna kwestia. Przecież coś podobnego już zostało zrobione! Jeśli przypomnimy sobie opisany w punkcie 2.1.2 model wektorowy wyszukiwania informacji, to dopuściliśmy w nim występowanie rzeczywistych wag termów w dokumentach, *dw*(*d, t*), określanych na ogół przez różnego rodzaju szczegółowe warianty formuły *tf*idf* (patrz na przykład wzór (2.1.8) lub (2.1.9)). Oczywiście fakt ten został równie szybko dostrzeżony przez badaczy zajmujących się wyszukiwaniem informacji, i już wkrótce po sformułowaniu modelu wektorowego w latach siedemdziesiątych XX wieku pojawiły się prace, które proponowały wykorzystanie wag termów do rozmytego opisu treści dokumentów [Radecki, 1979; Salton, McGill, 1983].

Przy okazji, semantyka wag termów w modelu wektorowym, zgadzała się z semantyką funkcji przynależności zbioru rozmytego. Waga *dw*(*t, d*) określa przecież, w jakim stopniu dany term opisuje zawartość dokumentu. Ponadto rozkłady wag termów w dokumencie odpowiadają raczej charakterystyce rozkładów możliwości, niż prawdopodobieństwa. Nie spełniają probabilistycznych warunków normalizacyjnych (suma wartości wag wszystkich słów kluczowych w dokumencie nie jest równa 1), w dokumencie może być wiele termów o maksymalnej wadze – wszystkie one w najpełniejszym stopniu określają treść dokumentu. Cechy te pasują doskonale do semantyki funkcji przynależności zbiorów rozmytych.

Problemem może być jedynie fakt, że wartości funkcji przynależności zbioru rozmytego powinny pochodzić z przedziału [0, 1], natomiast przy wyznaczaniu wartości wag termów nie zawsze tego rodzaju charakterystyka jest istotna. Dlatego w tym przypadku, od wartości *dw*(*d, t*) wymagać będziemy znormalizowanej wartości. Na przykład [Bordogna, Pasi, 2000] może być to standardowa formuła *df*idf*:

$$dw(t,d) = tf(t,d) \cdot idf(t) \tag{2.2.2}$$

gdzie $tf(t,d)$ oznacza znormalizowaną częstość termu w dokumencie, np. liczbę wystąpień tego słowa kluczowego w dokumencie, podzieloną przez maksymalną liczbę wystąpień dowolnego termu w tym dokumencie. Odwrotność częstości dokumentu $idf(t)$ może mieć standardową wartość, jak w (2.1.8), tzn. log $(M / n(t))$, gdzie M oznacza liczbę dokumentów w kolekcji, $n(t)$ liczbę dokumentów zawierających w swoim opisie dany term (z wagą większą od 0).

Przy takiej definicji wagi termu, możemy więc zdefiniować funkcję przynależności dla zbioru rozmytego opisu dowolnego dokumentu $d \in D$, jako:

$$\mu_d(t) = dw(t,d) \tag{2.2.3}$$

dla każdego $t \in T$.

Oczywiście, jeśli będzie nam to wygodne, to oprócz rozmytych reprezentacji dokumentów, możemy również stosować rozmyte reprezentacje termów, poprzez ich rozkłady możliwości przynależności do dokumentów:

$$\mu_t(d) = dw(t,d) \tag{2.2.4}$$

dla każdego termu $t \in T$ i dokumentu $d \in D$.

Jak więc widzimy, w zakresie reprezentacji dokumentów model rozmyty wyszukiwania informacji sprowadza się w zasadzie do modelu wektorowego. Zbiory rozmyte opisu dokumentów definiowane są przez wagi termów w tych dokumentach, wyznaczane przy pomocy standardowej procedury *tf*idf*. Problem polega jednak na sposobie realizacji zapytania.

W modelu wektorowym zapytanie ma formę zestawu słów kluczowych, łatwego do reprezentacji w formie wektora, zaś stworzenie rankingu wynikowego na wyznaczeniu podobieństwa między wektorem dokumentu i zapytania, na ogół przy pomocy miary cosinusoidalnej. W modelu boolowskim, którego rozszerzenie przecież chcemy stworzyć, zapytanie ma formę wyrażenia logicznego, występujące w nim termy połączone są operatorami logicznymi: koniunkcji („AND"), alternatywy („OR") i negacji („NOT").

Dlatego w modelu rozmytym zapytanie będzie realizowane po prostu poprzez obliczenie dla każdego dokumentu $d \in D$, prawdziwości wyrażenia zdefiniowanego przez użytkownika, w sensie operacji logiki rozmytej. Zazwyczaj wykorzystuje się w tym celu klasyczne min-maxowe definicje działań logicznych.

Jeśli w wykonywanym zapytaniu q występują działania logiczne na termach t_1, $t_2 \in T$, to wówczas:

$$\mu_{t_1 \text{ and } t_2}(d) = \min\left(\mu_{t_1}(d), \mu_{t_2}(d)\right)$$

$$\mu_{t_1 or\, t_2}(d) = \max\left(\mu_{t_1}(d), \mu_{t_2}(d)\right) \tag{2.2.5}$$

$$\mu_{\text{not } t_1}(d) = 1 - \mu_{t_1}(d)$$

Jeśli oznaczymy przez $\mu_q(d)$, finalną prawdziwość pełnego wyrażenia logicznego w zapytaniu q, wyznaczoną dla dokumentu d, to oczywiście nie jest to wartość binarna – prawda, lub fałsz w sensie logiki binarnej, tylko stopień prawdziwości zapytania. Można ją interpretować jako wartość podobieństwa, czy RSV (*Retrieval Status Value*) dokumentu dla zapytania i wykorzystać do stworzenia wynikowego rankingu dla użytkownika.

Całą procedurę realizacji zapytania, możemy zarysować następująco:

1. Znaleźć wszystkie termy występujące w zapytaniu, $t \in Q$.
2. Dla każdego wyszukanego termu pobrać listę odsyłaczy do dokumentów, którą można traktować jako definicję funkcji przynależności termu $\mu_t(d)$.
3. Przy pomocy funkcji przynależności poszczególnych termów, obliczyć funkcję przynależności zbioru rozmytego prawdziwości wyrażenia logicznego w zapytaniu q, $\mu_q(d)$.
4. Na podstawie niezerowych wartości $\mu_q(d)$ sporządzić uporządkowany ranking dokumentów. Często, aby skrócić ranking, tworzymy go dla $\mu_q(d)$ przekraczających założony próg, albo dla k najwyższych wartości $\mu_q(d)$.

Zilustrujmy metodę na poniższym przykładzie. Przyjmijmy, że kolekcja składa się z pięciu dokumentów, opisanych czterema termami indeksującymi. Macierz wag term/dokument $dw(t, d)$ dana jest następująco:

0	1/3	0	1/2	1/4
1/3	0	1/2	0	3/4
0	1/3	1/4	0	1
1	0	1/4	3/4	0

Niech zapytanie wykonane przez użytkownika ma postać: t_2 AND t_3.

Oczywiście termom zapytania odpowiadają wiersze 2 i 3 w macierzy $dw(t, d)$. Termy zapytania mają więc następujące funkcje przynależności: $t_2 = \{(d_1, 1/3), (d_2, 0), (d_3, 1/2), (d_4, 0), (d_5, 3/4)\}$, oraz $t_3 = \{(d_1, 0), (d_2, 1/3), (d_3, 1/4), (d_4, 0), (d_5, 1)\}$. Stosując operację minimum, obliczamy koniunkcję termów t_2 oraz t_3. W jej wyniku otrzymujemy zbiór rozmyty prawdziwości zapytania: $q = \{(d_1, 0), (d_2, 0), (d_3, 1/4), (d_4, 0), (d_5, 3/4)\}$. Ranking wynikowy składać się więc będzie z dwu dokumentów: d_5, d_3.

Widzimy więc, że model rozmyty pozwala w dosyć prosty sposób połączyć symboliczno-logiczny charakter modelu boolowskiego z algebraicznym podejściem stosowanym w modelu wektorowym, w znacznie lepszym stopniu pozwalającym na modelowanie niepewności opisu dokumentów. Model rozmyty stoi więc jakby pośrodku między tymi możliwościami, ze wszystkimi dodatnimi i ujemnymi skutkami tego stanu rzeczy.

Z jednej bowiem strony model rozmyty oferuje coś więcej niż model boolowski, z drugiej jednak coś mniej w porównaniu z modelem wektorowym. Mianowicie w tym ostatnim, nie ma problemu z przypisywaniem wag określających istotność również dla termów użytych w zapytaniu. W przedstawionej formie modelu rozmytego termy w zapytaniu mają charakter binarny – występują, albo nie występują. A przecież logika rozmyta oferuje tutaj spore i ciekawe możliwości. Dlatego większość z późniejszych prac nad modelem rozmytym koncentruje się na tworzeniu systemów realizacji bardziej złożonych zapytań.

Prace te można podzielić na trzy podstawowe grupy, w coraz większym zakresie rozszerzające wykorzystanie możliwości logiki rozmytej w języku zapytań systemu wyszukiwania informacji [Bordogna, Pasi, 2000; Crestani, Pasi, 1999; Zadrożny, Kacprzyk, 2005]:

- Zapytania z liczbowymi wagami istotności termów,
- Zapytania z lingwistycznymi wagami istotności termów,
- Zapytania z kwantyfikatorami lingwistycznymi.

Zapytania z liczbowymi wagami istotności termów stanowiły historycznie pierwsze podejście do rozszerzania możliwości języka zapytań. Ważone zapytania w tej postaci przyjmują formę wyrażenia logicznego, przy czym każdemu termowi w zapytaniu przypisana jest liczbowa waga z przedziału [0, 1], określająca istotność tego termu dla wyrażenia potrzeby informacyjnej użytkownika, a co za tym idzie dla wyniku zapytania. W przypadku braku wagi, tak jak w omawianych wyżej zapytaniach boolowskich, przyjmujemy jej wartość równą 1. Przykładem tego rodzaju zapytania, może być wyrażenie:

$$<t_1, qw_1> \text{ AND } (<t_2, qw_2> \text{ OR } <t_3, qw_3>) \qquad (2.2.6)$$

Istnieje kilka podejść do realizacji tego rodzaju zapytań, jednak zgadzają się one między sobą co do konieczności spełnienia tzw. właściwości separowalności, tj. że oszacowanie wpływu wagi na term w zapytaniu <*t*, *qw*> musi być operacją niezależną od oceny innych takich komponentów. Wymaganie spełnienia tej własności w zasadzie implikuje dwuetapowy sposób realizacji zapytania.

Oceniając prawdziwość zapytania dla danego dokumentu, musimy skonfrontować istotność termu dla zapytania (wyrażoną przez wagę *qw*) oraz istotność termu dla dokumentu (wyrażoną przez wagę *dw*) i na tej podstawie ocenić łączny stopień istotności termu. Po przeprowadzeniu tej oceny dla wszystkich termów zapytania przechodzimy do etapu drugiego, czyli określenia stopnia prawdziwości całego wyrażenia, polegającej na zastosowaniu do rezultatów etapu pierwszego operatorów logiki rozmytej, np. w formie ich implementacji (2.2.5).

Etap drugi przebiega więc w zasadzie tak, jak to przedstawiliśmy wyżej, w związku z tym zajmijmy się sposobem, a w zasadzie sposobami, realizacji etapu pierwszego. Wymaga on wprowadzenia pewnej funkcji $[0, 1] \times [0,1] \rightarrow [0,1]$, która wagom *qw* oraz *dw* przypisuje ostateczną istotność danego termu. Ponieważ jednak wagi termów w dokumencie *dw* mają charakter określony wcześniej, w czasie indeksowania dokumentu, wygodniej nam będzie spojrzeć na to, że to waga określona przez użytkownika w zapytaniu, *qw*, nakłada dodatkowe rozmyte (ponieważ spełnione w pewnym stopniu, z przedziału [0, 1]) ograniczenie na term, modyfikujące jego istotność.

Musimy więc zdefiniować funkcję przynależności tego ograniczenia:

$$\mu_{qw}\left(dw(t,d)\right) \tag{2.2.7}$$

Do określenia funkcji przynależności (2.2.7) zaproponowano kilka różnych rozwiązań, wynikających z nieco odmiennego spojrzenia na semantykę wagi termu w zapytaniu *qw*, a co za tym idzie definiowanego przez nią ograniczenia. Wymienić można trzy główne podejścia do rozumienia roli wagi zapytania:

- jako określenie względnego znaczenia termu w odniesieniu do innych,
- jako pewien próg, który musi przekroczyć dokument, aby znaleźć się w rankingu wynikowym, gwarantujący, że dokumenty będą dostatecznie zbliżone do potrzeby informacyjnej użytkownika,
- jako definiującą zbiór idealnych dokumentów, w tym sensie że ograniczenie μ_{qw} będzie mierzyło, jak blisko wagi *qw* jest waga *dw*.

Jak więc widzimy, w zależności od celu, w jakim wprowadzamy wagi zapytania i tego, jak będziemy rozumieć ich wykorzystanie, powinniśmy nieco inaczej zdefiniować sposób wykonania zapytania.

W przypadku pierwszym zadaniem wagi jest zróżnicowanie istotności słów kluczowych zapytania w taki sposób, by warunki nakładane na termy z różnymi wagami różnie wpływały na ocenę wyniku zapytania. Semantyka ta nie jest jednak taka prosta, jak by się mogło wydawać i musimy ją precyzyjniej omówić. Dokładniej mówiąc, waga równa 1 oznacza że term powinien działać ze swą normalną siłą, tak jak w omówionym wyżej modelu rozmytym z zapytaniem boolowskim. Spadająca wartość wagi, powinna stopniowo obniżać siłę oddziaływania termu, aż do zerowej [Bookstein 1980; Yager, 1987; Bordogna, Pasi, 2000].

Przy takiej interpretacji wag w wyrażeniu logicznym pojawia się jednak problem różnicy w semantyce dwu podstawowych operacji logicznych, tj. koniunkcji i alternatywy.

Wynik koniunkcji jest zdeterminowany właściwie przez jeden operand o najniższej wartości logicznej. Jeśli w wyrażeniu występuje koniunkcja nawet wielu warunków, i któryś z nich jest fałszywy, to prawdziwość (lub nie) pozostałych nie ma już znaczenia. Całe wyrażenie i tak jest fałszywe. Reguła ta w znacznym stopniu przenosi się na operacje logiki rozmytej – muszą one przecież wspierać klasyczną semantykę. Dokładniej rzecz biorąc, zależy to od normy trójkątnej wykorzystanej do modelowania koniunkcji rozmytej. Na przykład, przy zastosowaniu operacji minimum, przenosi się w pełni: min(0,3; 0,1; 0,8) = 0,1, niezależnie od tego, czy np. pierwszy warunek ma prawdziwość taką, jak obecnie 0,3, czy wynosiłaby ona, powiedzmy, 1.

W przypadku wykorzystania jako koniunkcji rozmytej operacji iloczynu algebraicznego reguła ta obowiązuje tylko przy występowaniu w wyrażeniu warunku w pełni fałszywego (czyli prawdziwości 0). W innych sytuacjach o stopniu prawdziwości wyrażenia decyduje prawdziwość wszystkich warunków. Tym niemniej jednak i tak niska prawdziwość któregoś z warunków zazwyczaj bardzo silnie obniża prawdziwość całej koniunkcji.

Z zupełnie odwrotną sytuacją mamy do czynienia w przypadku alternatywy. Prawdziwość wyrażenia alternatywnego, jest z kolei zdominowana przez operand o najwyższej prawdziwości. Jeśli w alternatywie jeden z warunków składowych jest prawdziwy, to prawdziwość lub nie pozostałych, nie ma już znaczenia. Podobnie w przypadku alternatywy rozmytej i użycia do jej modelowania operacji maksimum. Na przykład max(0,3; 0,1; 0,8) = 0,8 i jest ona zdeterminowana przez ten jeden maksymalny element, niezależnie o ile niższe prawdziwości mają pozostałe operandy.

W związku z tym, jak widzimy, interpretacja wagi termu w zapytaniu, w sensie jego istotności, czy też „siły" oddziaływania na wynikowy ranking, powoduje że waga ta musi inaczej funkcjonować w wyrażeniach koniunkcyjnych, a inaczej alternatywnych. W przypadku koniunkcji osłabianie wpływu termu musi powodować podnoszenie jego istotności. Waga $qw = 1$ oznacza istotność termu na normalnym poziomie dw. Waga równa 0, powinna oznaczać wzrost wartości do maksymalnej, czyli 1, ponieważ wtedy z pewnością nie będzie ona wpływać w żadnym stopniu na operację minimum (czy też iloczyn), implementującą koniunkcję rozmytą.

W przypadku alternatywy, oczywiście, mamy sytuację odwrotną. Istotność termu w operacjach logicznych powinna spadać od normalnej, dw (przy wadze $qw = 1$), do minimalnej zerowej (przy wadze $qw = 0$). Dopiero ta ostatnia wartość 0, z pewnością nie będzie wpływać na wynik alternatywy rozmytej (np. w formie maksimum).

Widzimy, że w tym przypadku nie udało się stworzyć implementacji wagi termu w zapytaniu, zachowującej wspomniany wcześniej warunek separowalności. Niezbędne są odmienne podejścia dla wyrażeń koniunkcyjnych i alternatywnych.

W literaturze proponowane są dwie implementacje działania tego typu wag dla wyrażeń koniunkcyjnych i alternatywnych.

Pierwsza z nich, sformułowana przez Booksteina [Bookstein, 1980]:

$$\mu_{qw}\left(dw(t,d)\right)=\min\left(1,dw(t,d)/qw(t)\right) \text{ – dla zapytań koniunkcyjnych} \tag{2.2.8}$$

$$\mu_{qw}\left(dw(t,d)\right)=qw(t)\cdot dw(t,d) \text{ – dla zapytań alternatywnych} \tag{2.2.9}$$

Druga, przez Yagera [Yager, 1987]:

$$\mu_{qw}\left(dw(t,d)\right)=\max\left(1-qw(t),dw(t,d)\right) \text{ – dla zapytań koniunkcyjnych} \tag{2.2.10}$$

$$\mu_{qw}\left(dw(t,d)\right)=\min\left(qw(t),dw(t,d)\right) \text{ – dla zapytań alternatywnych} \tag{2.2.11}$$

Jak widzimy, w obu przypadkach do modelowania oddziaływania wagi qw, wybrana została operacja implikacji rozmytej dla zapytań koniunkcyjnych oraz koniunkcji dla zapytań alternatywnych.

Działanie zapytań z wagami określającymi istotność wykorzystywanych w nim termów zilustrujemy przy pomocy następującego przykładu.

Rozważmy następującą macierz termów-dokumentów.

	d1	d2	d3	d4	d5
t1	0	0,8	0,2	0	0,6
t2	0,5	0,1	0	1	0
t3	1	0	0,5	0	1

Spróbujmy pokazać przy jej użyciu sposób wykonania zapytania: <t1, 1> AND <t2, 0,7> AND <t3, 0>.

Określone wyżej zapytanie rozumiemy w następujący sposób: użytkownika interesują dokumenty dotyczące tematyki określonej przez każde z tych trzech słów kluczowych. Przy czym najważniejszy jest term t1. Nieco mniej ważny ale dosyć istotny (w stopniu 0,7) jest term t2. Najmniej istotny dla potrzeby informacyjnej użytkownika, a w tej sytuacji w zasadzie nieistotny, jest term t3.

W pierwszym kroku stosujemy do poszczególnych termów, określone w zapytaniu wagi. Jak widzimy wyrażenie użyte w zapytaniu ma charakter koniunkcyjny. Do określenia ich wpływu, użyjemy więc zależności (2.2.10).

Obliczamy najpierw jak w wyniku zastosowania wagi zmieni się funkcja przynależności termu t1.

<t1, 1> = <{(d1, 0); (d2, 0,8); (d3, 0,2); (d4, 0); (d5, 0,6)}, 1> = {(d1, max(1–1, 0)); (d2, max(1–1, 0,8)); (d3, max(1–1, 0,2)); (d4, max(1–1, 0)); (d5, max(1–1, 0,6))} = {(d1, 0); (d2, 0,8); (d3, 0,2); (d4, 0); (d5, 0,6)}

Jak widzimy, ponieważ waga termu t1 wynosiła 1, jego funkcja przynależności nie uległa żadnej zmianie. Przeliczmy teraz wartości funkcji przynależności dla termu t2:

<t2, 0,7> = <{(d1, 0,5); (d2, 0,1); (d3, 0); (d4, 1); (d5, 0)}; 0,7> = {(d1, max((1–0,7), 0,5), (d2, max((1–0,7), 0,1), (d3, max((1–0,7), 0), (d4, max((1–0,7), 1), (d5, max((1–0,7), 0)} = {(d1, 0,5); (d2, 0,3); (d3, 0,3); (d4, 1); (d5, 0,3)}

I dla ostatniego termu t3:

<t3, 0> = <{(d1, 1); (d2, 0); (d3, 0,5); (d4, 0); (d5, 1)}; 0> = {(d1, max((1–0), 1); (d2, max((1–0), 0); (d3, max((1–0), 0,5); (d4, max((1–0), 0); (d5, max((1–0), 1)} = {(d1, 1); (d2, 1); (d3, 1); (d4, 1); (d5, 1)};

Zwróćmy uwagę, że dla termu 3 wszystkie wartości funkcji przynależności zostały ustawione na wartość maksymalną 1. Wynika to z faktu, że znaczenie tego termu ma być zerowe, czyli nie powinien w ogóle wpływać na wynik zapytania. I z pewnością tak będzie, ponieważ term ten na pewno dla żadnego dokumentu nie zmieni wartości operacji maksimum modelującej koniunkcję.

Następnie dla przeliczonych przy pomocy wag termów, obliczamy wyrażenie logiczne zastosowane w zapytaniu, w tym przypadku koniunkcję:

<t1, 1> AND <t2, 0,7> AND <t3, 0> = {(d1, min(0; 0,5; 1)); (d2, min(0,8; 0,3; 1)); (d3, min(0,2; 0,3; 1)); (d4, min(0; 1; 1)); (d5, min(0,6; 0,3; 1))} = {(d1, 0); (d2, 0,3); (d3, 0,2); (d4, 0); (d5, 0,3)}

Na koniec, pozostaje już tylko sporządzenie rankingu: d2, d5, d3.

Aby rozwiać wątpliwości, które mogłyby powstać u czytelnika, przeanalizujmy jeszcze kwestię dokumentu d5, który stanowi chyba najciekawszy przypadek, mogący wzbudzić kontrowersje. Otóż, gdyby zapytanie miało charakter boolowski, dokument ten uzyskałby ocenę RSV równą 0 i nie znalazłby się w rankingu. Natomiast dzięki zastosowaniu w zapytaniu wag, został on w rankingu uwzględniony i to dzieląc pozycję lidera, z oceną RSV = 0,3. Musimy jednak pamiętać, że niska ocena dokumentu d5 w zapytaniu boolowskim wynikałaby z zerowej wagi dla tego dokumentu występującego w zapytaniu termu t2. W zapytaniu ważonym istotność tego termu została oceniona tylko na 0,7, z tego powodu nie może on decydować o wyrzuceniu z rankingu dokumentów, które nie zostały nim zaindeksowane. Ponieważ term ten jest istotny w stopniu 0,7, może być nieistotny w stopniu 1 – 0,7 = 0,3, co podnosi na ten poziom ocenę dokumentu d5.

Druga z semantyk wag termów w zapytaniu, dyskutowana w literaturze, związana jest z rozumieniem ich jako pewnych progów (minimalnych akceptowalnych poziomów) dla wyszukanych przy pomocy zapytania dokumentów. Podając dane słowo kluczowe w zapytaniu, użytkownik określa pewien temat, którym jest zainteresowany, specyfikując próg chce zapewnić sobie wyszukanie dokumentów, które są w „dostatecznym stopniu" na ten temat. Podwyższając próg, użytkownik zmniejsza liczbę wyszukanych dokumentów. Aby dokument znalazł się w rankingu wynikowym, musi ten próg przekroczyć [Crestani, Pasi, 1999].

Najprostszy sposób modelowania tego rodzaju wag progowych zdefiniował Radecki [Radecki, 1979; Bordogna, Pasi, 2000]:

$$\mu_{qw}\left(dw(t,d)\right)=\begin{cases}0 & \text{dla } dw(t,d)<qw(t)\\ dw(t,d) & \text{dla } dw(t,d)\geq qw(t)\end{cases} \qquad (2.2.12)$$

Zauważmy jednak, że tak zdefiniowany próg ma charakter ostry, zaś nieciągłość w funkcji przynależności μ_{qw} może powodować duże zmiany liczby wyszukanych dokumentów, nawet przy niewielkiej zmianie wartości wag w zapytaniu. Buell i Kraft [Bordogna, Pasi, 2000] [Crestani, Pasi, 1999] zaproponowali bardziej elastyczną definicję progu, pozwalającą na płynne przejście między akceptacją i odrzuceniem dokumentu:

$$\mu_{qw}\left(dw(t,d)\right)= \\ =\begin{cases}P(qw)\cdot\dfrac{dw(t,d)}{qw} & \text{dla } dw(t,d)<qw(t)\\ P(qw)+Q(qw)\cdot\dfrac{dw(t,d)-qw}{1-qw} & \text{dla } dw(t,d)\geq qw(t)\end{cases} \qquad (2.2.13)$$

gdzie $P(qw)=(1+qw)/2$, zaś $Q(qw)=(1-qw^2)/4$.

Człon w ścieżce dla $dw(t, d) < qw(t)$ (2.2.13) mierzy „bliskość" wagi termu w dokumencie $dw(t, d)$ do określonego w zapytaniu progu $qw(t)$. Natomiast człon w ścieżce dla $dw(t, d) \geq qw(t)$ określa stopień dodatkowej satysfakcji z przekroczenia qw i zapewnia nieprzekroczenie wartości 1.

Jako ilustracja działania wag progowych, posłuży nam przykład oparty na tej samej macierzy termów-dokumentów, którą wykorzystaliśmy w poprzednim przypadku:

	d1	d2	d3	d4	d5
t1	0	0,8	0,2	0	0,6
t2	0,5	0,1	0	1	0
t3	1	0	0,5	0	1

Przyjmijmy, że w tej chwili zapytanie ma postać: <t1, 0,5> AND <t2, 0> AND <t3, 1>. Oczywiście obecnie interpretujemy określone w zapytaniu wagi jako minimalne progi nakładane na poszczególne słowa kluczowe. Waga 1 nakładana na trzeci term oznacza nałożenie na niego bardzo wysokich wymagań: wyszukane zostaną tylko dokumenty zaindeksowane tym słowem kluczowym z wagą (stopniem przynależności) 1.

Natomiast waga 0 nakładana na drugi term oznacza rezygnację z jakichkolwiek dodatkowych ograniczeń nakładanych na niego w zapytaniu.

Nałożone progi interpretować będziemy jako ostre, zgodnie semantyką Radeckiego określoną przez wzór (2.2.12). Obliczamy najpierw, jak w wyniku zastosowania wag zmienią się funkcję przynależności poszczególnych termów.

<t1, 0,5> = <{(d1, 0); (d2, 0,8); (d3, 0,2); (d4, 0); (d5, 0,6)}, 0,5> = {(d1, 0); (d2, 0,8); (d3, 0); (d4, 0); (d5, 0,6)}

<t2, 0> = <{(d1, 0,5); (d2, 0,1); (d3, 0); (d4, 1); (d5, 0)}; 0> = {(d1, 0,5); (d2, 0,1); (d3, 0); (d4, 1); (d5, 0)};

<t3, 1> = <{(d1, 1); (d2, 0); (d3, 0,5); (d4, 0); (d5, 1)}; 1> = {(d1, 1); (d2, 0); (d3, 0); (d4, 0); (d5, 1)};

Następnie dla przeliczonych przy pomocy wag termów, obliczamy wyrażenie logiczne zastosowane w zapytaniu:

<t1, 0,5> AND <t2, 0> AND <t3, 1> = {(d1, min(0; 0,5; 1)); (d2, min(0,8; 0,1; 0)); (d3, min(0; 0; 0)); (d4, min(0; 1; 0)); (d5, min(0,6; 0; 1))} = {(d1, 0); (d2, 0); (d3, 0); (d4, 0); (d5, 0)}

Widzimy więc, że tym razem żaden z dokumentów nie odpowiada zadanemu zapytaniu i ranking wynikowy będzie pusty.

Pozostała nam jeszcze do omówienia ostatnia interpretacja wag zapytania, jako pewnych wartości idealnych, dla których chcemy wyszukać dokumenty dopasowane w jak największym stopniu. To znaczy, jeśli w zapytaniu określimy dla jakiegoś termu wagę, powiedzmy 0,7, to najlepiej dopasowane będą dokumenty, w których ten term ma wagę jak najbliższą 0,7.

W takim przypadku, oczywiście, rozmyte ograniczenie generowane przez wagę powinno być określone w kategoriach pewnej miary podobieństwa do wzorca definiowanego przez zapytanie. Bordogna i Pasi [Bordogna, Pasi, 2000] zaproponowały wykorzystanie do tego zadania funkcji gaussowskiej:

$$\mu_{qw}\left(dw(t,d)\right) = e^{\ln k \cdot \left(dw(t,q) - qw(t)\right)^2} \tag{2.2.14}$$

Wartość $k \in [0, 1]$ jest parametrem określającym szybkość opadania grzbietów krzywej gaussowskiej, przy czym im wyższa wartość k, tym

większa rozpiętość krzywej i słabsze ograniczenie nakładane na bliskość do wzorca idealnego. Aby zachować odpowiednią zdolność dyskryminacyjną możliwości określania różnych wartości funkcji przynależności ograniczenia, Bordogna i Pasi zalecają wartość współczynnika *k* na poziomie wynoszącym 0,01 [Bordogna, Pasi, 1993].

I znów, jako przykład, posłużymy się tą samą macierzą termów-dokumentów, co w poprzednich przypadkach:

	d1	d2	d3	d4	d5
t1	0	0,8	0,2	0	0,6
t2	0,5	0,1	0	1	0
t3	1	0	0,5	0	1

Wyznaczmy ranking wynikowy dla zapytania: <t1, 1> AND <t2, 0,7> AND <t3, 0>. Wartość parametru *k* w funkcji (2.2.14) przyjęta została jako 0,01.

Rozpoczynamy, podobnie jak w poprzednich przykładach, od wyznaczenia funkcji przynależności indukowanego przez wagi ograniczenia dla termów zapytania w poszczególnych dokumentach, stosując wzór zależność (2.2.14).

I tak, dla termu t1 otrzymujemy:

<t1, 1> = <{(d1, 0); (d2, 0,8); (d3, 0,2); (d4, 0); (d5, 0,6)}, 1> = {(d1, 0,01); (d2, 0,83); (d3, 0,05); (d4, 0,01); (d5, 0,48)}

Następnie dla termu t2:

<t2, 0,7> = <{(d1, 0,5); (d2, 0,1); (d3, 0); (d4, 1); (d5, 0)}; 0,7> = {(d1, 0,83); (d2, 0,19); (d3, 0,10); (d4, 0,66); (d5, 0,10)}

I dla ostatniego termu t3:

<t3, 0> = <{(d1, 1); (d2, 0); (d3, 0,5); (d4, 0); (d5, 1)}; 0> = {(d1, 0,01); (d2, 1); (d3, 0,32); (d4, 1); (d5, 0,01)}

I znów, dla przeliczonych przy pomocy wag termów, obliczamy wyrażenie logiczne zastosowane w zapytaniu:

<t1, 0,5> AND <t2, 0> AND <t3, 1> = {(d1, min(0,01; 0,83; 0,01)); (d2, min(0,83; 0,19; 1)); (d3, min(0,05; 0,10; 0,32)); (d4, min(0,01; 0,66; 1));

(d5, min(0,48; 0,10; 0,01))} = {(d1, 0,01); (d2, 0,19); (d3, 0,05); (d4, 0,01); (d5, 0,01)}

Ostateczny ranking będzie miał więc postać: d2, d3, d1, d4, d5.

Jak więc widzimy, w obrębie modelu rozmytego wyszukiwania informacji, zostało stworzonych kilka metod wykonywania zapytań z wagami przydzielonymi poszczególnym termom. Wagi te mogą mieć różną semantykę, tak więc projektując usługę wyszukiwawczą można dobrać ich sposób interpretacji, stosownie do zamierzeń twórcy systemu.

Liczbowe wagi zmuszają jednak użytkowników do kwantyfikacji dosyć nieprecyzyjnego i nie do końca określonego pojęcia istotności w formie precyzyjnych wartości liczbowych. Ponadto, do właściwego określenia, wymagają jednak sporej wiedzy o semantyce wag, której niuanse mogą czasami poważnie wpływać na jakość wyszukiwania. Dużo bardziej naturalne wydaje się dać możliwość użytkownikowi wyrażenia swoich preferencji co do wag zapytania, w formie lingwistycznej, pozostawiając systemowi przełożenie ich na właściwą reprezentację. Nic więc dziwnego, że taka właśnie możliwość stała się kolejnym rozszerzeniem, jakie pojawiło się w ramach rozmytego modelu wyszukiwania informacji.

Istotność termów w zapytaniu musi być definiowana z niezbędną granularnością, w zależności od charakterystyki usługi wyszukiwawczej, w której mechanizm ten ma być wykorzystywany. W tym przypadku zazwyczaj definiuje się tylko jeden term podstawowy, „istotny", który może być następnie dostrajany przez modyfikatory, takie jak „bardzo", „średnio", czy „minimalnie".

Podobnie jak w przypadku wag liczbowych, również waga lingwistyczna definiuje pewne ograniczenie rozmyte, jakie powinna spełniać waga termu w dokumencie i w zależności od stopnia jego spełniania modyfikować ją na potrzeby danego wyrażenia logicznego, zgodnie z semantyką wagi. Jej funkcja przynależności zdefiniowana zostanie na przedziale [0, 1]. Wagi lingwistyczne stanowią przecież rozmyte wersje wag liczbowych. Bordogna i Pasi zaproponowały dla termu podstawowego „istotny", funkcję przynależności zdefiniowaną na zasadzie semantyki wagi idealnej, opartej na zależności (2.2.14) [Bordogna, Pasi, 1993]:

$$\mu^{ij}_{istotny}\left(dw(t,d)\right)=\begin{cases} e^{\ln(k)\cdot\left(dw(t,d)-i\right)^2} & \text{dla } dw(t,d)<i \\ 1 & \text{dla } i\le dw(t,d)\le j \\ e^{\ln(k)\cdot\left(dw(t,d)-j\right)^2} & \text{dla } dw(t,d)>j \end{cases} \qquad (2.2.15)$$

wartości $i, j \in [0, 1]$, $i \leq j$ są parametrami określającymi umiejscowienie przedziału $[i, j]$ dla którego pojęcie „istotny" jest w pełni spełnione, parametr k ma taką samą definicję jak we wzorze (2.2.14), czyli określa szybkość spadku funkcji przynależności (w formie zbocza funkcji Gaussa) symetrycznie poza oboma krańcami przedziału $[i, j]$. Wszystkie te parametry dobierane mogą być dla konkretnego przypadku. Przykładowe wartości podawane w literaturze, to $i = 0{,}7$, $j = 1$, $k = 0{,}01$ (lub mniej, nawet około 0,001).

Kraft, Bordogna i Pasi [Bordogna, Pasi, 2000] podają także nieco odmienną definicję wagi lingwistycznej, opartą na semantyce progowej:

$$\mu^{ij}_{\text{istotny}}\left(dw(t,d)\right)=\begin{cases}\dfrac{1+i}{2}e^{\ln(k)\cdot\left(dw(t,d)-i\right)^2} & \text{dla } dw(t,d)<i\\ \dfrac{1+dw(t,d)}{2} & \text{dla } i\leq dw(t,d)\leq j\\ \dfrac{1+j}{2}\cdot\left(1+\dfrac{dw(t,d)-j}{2}\right) & \text{dla } dw(t,d)>j\end{cases} \quad (2.2.16)$$

wartości $i, j \in [0, 1]$, $i \leq j$ oraz k mają podobne znaczenie jak w poprzednim wzorze. Funkcja przynależności w tym przypadku dla wag mniejszych od i rośnie w formie krzywej Gaussa, w przedziale $[i, j]$ rośnie liniowo, a po przekroczeniu j, rośnie nadal, lecz w niższym tempie.

Modyfikatory pojęcia „istotny" definiowane są jako operatory przesunięcia

$$\mu_{\text{bardzo istotny}}\left(dw(t,d)\right)=\mu^{i+0{,}2\ j+0{,}2}_{\text{istotny}}\left(dw(t,d)\right) \quad (2.2.17)$$

$$\mu_{\text{średnio istotny}}\left(dw(t,d)\right)=\mu^{i-0{,}3\ j-0{,}3}_{\text{istotny}}\left(dw(t,d)\right) \quad (2.2.18)$$

$$\mu_{\text{minimalnie istotny}}\left(dw(t,d)\right)=\mu^{i-0{,}5\ j-0{,}5}_{\text{istotny}}\left(dw(t,d)\right) \quad (2.2.19)$$

Zilustrujmy teraz proces wyszukiwania na przykładzie. Przyjmijmy, że w systemie mamy następującą macierz term-dokument:

	d1	d2	d3	d4
gwiazda	0,8	0,3	0	0,9
kosmos	0,1	0,2	1	0

Załóżmy dalej, że chcemy wyszukać wszystkie dokumenty dotyczące gwiazd. Niektóre z relewantnych dokumentów w kolekcji mogą akurat nie być zaindeksowane słowem kluczowym „gwiazda", tylko ogólniejszym tematycznie termem „kosmos". Takie dokumenty też mogą być wyszukiwane, ale są one dla nas mniej istotne, ponieważ mogą być zbyt ogólne. Przyjmijmy więc, że zapytanie, które ma wykonać system wyszukiwawczy brzmi następująco:

„<gwiazda; istotny> AND <kosmos; minimalnie istotny>"

Przyjmijmy wartość parametru $k = 0{,}01$. Dla wagi podstawowej „istotny" parametry i, j wynoszą $i = 0{,}7$, $j = 1$. Dla wagi „minimalnie istotny" $i = 0{,}2$, $j = 0{,}5$.

Obliczmy najpierw dla obu termów występujących w zapytaniu, ich stopnie przynależności wynikające z wagi w zapytaniu i wag w dokumentach.

Dla pierwszego termu „gwiazda", ponieważ jest istotny, wykorzystujemy wzór (2.2.16):

<gwiazda, istotny> = <{(d1, 0,8); (d2, 0,3); (d3, 0); (d4, 0,9)}, istotny> = {(d1, 0,9); (d2, 0,41); (d3, 0,09); (d4, 0,95)}

Dla drugiego termu, „kosmos", ponieważ jest minimalnie istotny, korzystamy ze wzoru (2.2.16), z modyfikatorem (2.2.19).

<kosmos; minimalnie istotny> = <{(d1, 0,1); (d2, 0,2); (d3, 1); (d4, 0)}, minimalnie istotny> = {(d1, 0,57); (d2, 0,6); (d3, 0,94); (d4, 0,50)}

Czyli całe zapytanie, po obliczeniu znajdującego się w nim wyrażenia, posiadać będzie następującą funkcję przynależności:

<gwiazda; istotny> AND <kosmos; minimalnie istotny> = {(d1, min(0,9; 0,57)); (d2, min(0,41; 0,6)); (d3, min(0,09; 0,94)); (d4, min(0,95; 0,50))} = {(d1, 0,57); (d2, 0,41); (d3, 0,09); (d4, 0,50)}

Co pozwala nam na sporządzenie następującego rankingu wynikowego dokumentów, dla naszego zapytania: d1, d4, d2, d3.

Na koniec bieżącego punktu, zajmijmy się jeszcze krótko kolejnym rozszerzeniem modelu logicznego wyszukiwania informacji, polegającym na możliwości wykorzystania w zapytaniach kwantyfikatorów.

Z logiki matematycznej pamiętamy że w klasycznej wersji mamy dwa kwantyfikatory: ogólny (dla każdego), wskazujący że coś obowiązuje dla

wszystkich przypadków, oraz szczegółowy (istnieje), wskazujący że coś obowiązuje co najmniej dla jednego przypadku. Ogólnie kwantyfikatory stanowią mechanizm agregacji wyrażeń logicznych, pozwalający na redukcję stopnia ich złożoności. W wyrażeniach języka naturalnego oprócz dwu wyżej wspomnianych, które można językowo określić jako „wszystkie" oraz „co najmniej jeden", stosowanych jest ich cała gama, wymienić tu możemy dla przykładu takie określenia jak: „większość", „co najmniej k", czy też „dokładnie k". Logika rozmyta pozwala definiować znacznie więcej tego rodzaju kwantyfikatorów lingwistycznych, jak np. „zdecydowana większość", „prawie wszystkie", czy też „około k".

Przydatność tego rodzaju mechanizmu agregacji warunków, przy tworzeniu zapytań dla celów wyszukiwawczych, jest dosyć oczywista. Zastanówmy się choćby nad następującym prostym przykładem. Powiedzmy, że chcemy sformułować zapytanie korzystające z czterech słów kluczowych: „gwiazda", „kosmos", „kwazar", „planeta", przy czym wystarczy nam, by wyszukiwanych dokumentach były co najmniej dwa spośród nich. Zapytanie korzystające wyłącznie z operatorów logicznych, musiałoby w tej sytuacji mieć następującą formę:

„(gwiazda AND kosmos) OR (gwiazda AND kwazar)) OR (gwiazda AND planeta) OR (kosmos AND kwazar) OR (kosmos AND planeta) OR (kwazar AND planeta)"

Jak widać, pomimo stosunkowo prostej potrzeby informacyjnej, zapytanie jest całkiem skomplikowane. Znacznie prościej wyglądałoby ono, gdybyśmy mogli skorzystać z kwantyfikatora lingwistycznego, co mogłoby wyglądać mniej więcej tak:

„CO NAJMNIEJ DWA (gwiazda, kosmos, kwazar, planeta)"

Jak więc widzimy, możliwość określenia, że interesują nas dokumenty zawierające, powiedzmy, „większość", czy też „co najmniej k", albo „prawie wszystkie" słowa kluczowe z wymienionego zestawu, znaczenie wzbogaciłaby możliwości wyrażania potrzeb informacyjnych użytkowników, nie komplikując zbytnio samych zapytań.

Nic też dziwnego, że pojawił się szereg prac w kierunku rozszerzenia modelu rozmytego wyszukiwania informacji o możliwość użycia kwantyfikatorów lingwistycznych [Yager, 1996, 2000; Bordogna, Pasi, 1995, 2000]. Wykorzystują one do definiowania kwantyfikatorów w zapytaniach wyszukiwawczych, tzw. uporządkowane ważone operatory uśredniające OWA (ang. *Ordered Weighted Averaging*).

Operatory OWA zaproponowane zostały do modelowania kwantyfikatorów w wyrażeniach logiki rozmytej już w latach osiemdziesiątych ubiegłego wieku [Yager, 1988]. Operator OWA (*n* argumentowy) zdefiniować możemy jako *n* wymiarową funkcję agregującą OWA: $[0, 1]^n \rightarrow [0, 1]$, określoną przy pomocy wektora wag $W = (w_1, \ldots, w_n)$, w następujący sposób:

$$OWA(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} w_i \max_i (x_1, x_2, \ldots, x_n) \qquad (2.2.20)$$

przy czym, aby był to operator uśredniający, suma wag musi spełniać warunek normalizujący:

$$\sum_{i=1}^{n} w_i = 1, w_i \in [0,1]$$

Operator $\max_i (x_1, \ldots, x_n)$ oznacza i-tą wartość maksymalną, czyli np. $\max_1 (0{,}7, 0{,}1, 1) = 1$, $\max_2 (0{,}7, 0{,}1, 1) = 0{,}7$, zaś $\max_3 (0{,}7, 0{,}1, 1) = 0{,}1$.

Dlaczego właśnie operatory OWA nadają się do definiowania kwantyfikatorów? Zwróćmy uwagę, że są to operatory uśredniające, i ich wartość zawsze (dla dowolnego wektora wag) leży między minimum i maksimum:

$$\min(x_1, x_2, \ldots, x_n) \leq OWA(x_1, x_2, \ldots, x_n) \leq \max(x_1, x_2, \ldots, x_n) \qquad (2.2.21)$$

Właściwie, to OWA dla różnych wag tworzą spektrum operatorów agregacji między minimum i maksimum. Przy czym maximum można wyrazić jako operator OWA określony przez zestaw wag (1, 0, …, 0). Wówczas bowiem we wzorze (2.2.20) mamy $OWA(x_1, x_2, \ldots, x_n) = 1 \cdot \max_1 (x_1, x_2, \ldots, x_n) = \max(x_1, x_2, \ldots, x_n)$. Podobnie minimum można wyrazić jako operator OWA z wagami (0, …, 0, 1). Wtedy bowiem $OWA(x_1, x_2, \ldots, x_n) = 1 \cdot \max_n (x_1, x_2, \ldots, x_n) = \min(x_1, x_2, \ldots, x_n)$, gdyż *n*-ty element maksymalny na *n* możliwych, to oczywiście element najmniejszy.

Zauważmy przy tym że w przypadku wyrażeń rozmytych, operacja minimum definiuje koniunkcję. Z kolei koniunkcja wyrażeń, w sposób naturalny, odpowiada kwantyfikatorowi „wszystkie". Podobnie maksimum definiuje alternatywę, która odpowiada kwantyfikatorowi „co najmniej jeden". Jak więc widzimy, przy pomocy operacji OWA możemy zdefiniować pewne spektrum operacji agregujących, kwantyfikatorów, pomiędzy tymi dwoma skrajnościami.

I tak, np. w [Bordogna, Pasi, 2000] do definiowania zapytań w systemie wyszukiwawczym, zaproponowany został następujący zestaw kwantyfikatorów lingwistycznych:

- „wszystkie" – zastępujący koniunkcję „and", zdefiniowany przez wektor wagowy $W_{wszystkie}$, dla którego, jak już wspomnieliśmy wcześniej: $w_n = 1$, $w_i = 0$, dla wszystkich pozostałych i.
- „co najmniej k" – działa jako określenie ostrego progu k, na liczbę warunków wyboru. Określony jest on przez wektor wagowy $W_{co\ najmniej\ k}$, dla którego: $w_k = 1$, $w_i = 0$, dla wszystkich $i \neq k$.
- „około k" – stanowi rozmytą interpretację kwantyfikatora „co najmniej k", w której wartość k nie jest rozumiana jako próg ostry, lecz rozmyty. Oznacza to, że użytkownik powinien być w pełni usatysfakcjonowany z tego że spełnionych jest k lub więcej warunków, ale osiąga pewien (coraz mniejszy) stopień satysfakcji, nawet jeżeli spełnionych jest tylko k–1, k–2, … warunków. Kwantyfikator określony jest przez wektor $W_{około\ k}$, dla którego $w_i = i / \sum_{j=1}^{i} j$ dla $i \leq k$, oraz $w_i = 0$, dla wszystkich $i > k$.
- „większość z" – zdefiniowane jako synonim „co najmniej $2/3n$", gdzie n jest liczbą wszystkich warunków.

2.2.2. Skojarzeniowe wyszukiwanie informacji z wykorzystaniem sieci pojęć

2.2.2.1. Wiedza dziedzinowa w systemach wyszukiwawczych

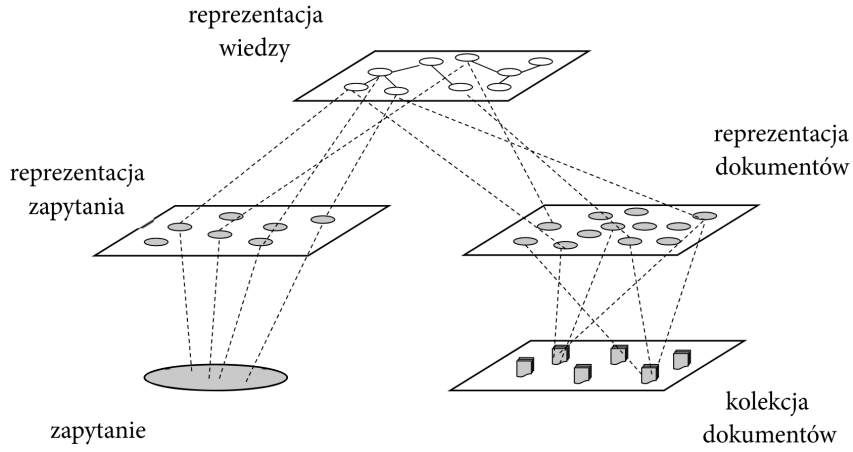
Jednym z głównych problemów na które cierpią prezentowane w punkcie 2.1 klasyczne tekstowe modele wyszukiwawcze jest specyfikacja ich działania na poziomie słownikowym. Zapytania, podobnie jak i reprezentacja dokumentów, określane są przez słowa kluczowe, termy – nieprecyzyjne słowa języka naturalnego. Kwestia dopasowania tych dwu elementów, szczególnie w sytuacji, gdy zarówno opisy dokumentów, jak i zwłaszcza zapytania dalekie są od kompletności i precyzji, niesie ze sobą spore problemy.

W przypadku słów kluczowych dopasowanie, bezpośrednio lub pośrednio, w sposób jawny lub ukryty, opiera się na ich współwystępowaniu w zapytaniach i dokumentach. Dotyczy to w zasadzie wszystkich modeli, zarówno tych klasycznych, podstawowych, przedstawionych przez nas w punkcie 2.1, jak i bardziej zaawansowanych, stanowiących ich rozwinięcie na gruncie samego wyszukiwania informacji. Problem więc pojawia się w sytuacji, gdy nie ma takiego dopasowania na poziomie leksykalnym, kiedy w zapytaniu i w opisie dokumentu nie zostały użyte te same słowa.

Kwestia ta leży u podstaw tak szerokiego zakresu zastosowań sztucznej inteligencji w wyszukiwaniu informacji, ponieważ naprawdę przydatny system wyszukiwawczy musi w jakiś sposób, choćby nawet w niewielkim stopniu, „zrozumieć", czego szuka użytkownik, oraz co zostało zawarte w dokumencie. Do tego, jeśli nawet w zapytaniu i w dokumencie nie pojawiają się dokładnie te same słowa, musi „skojarzyć", że są one ze sobą powiązane i że w związku z tym należy również powiązać dokument i zapytanie.

Aby wykryć tego rodzaju skojarzenia, zazwyczaj niezbędna jest wiedza i umiejętność jej wykorzystania, czyli pewne cechy systemu inteligentnego. W obecnym rozdziale zajmiemy się sytuacją, w której wiedza ta wykorzystywana jest bezpośrednio do wykrycia takich skojarzeń i zastosowania ich w procesie dopasowania obu kawałków naszej układanki.

Tak więc obecnie interesuje nas takie rozbudowanie mechanizmu dopasowującego dokumenty do zapytania, aby dopasowanie to nie odbywało się w prostej warstwie słownikowej termów indeksujących, ale w warstwie pewnej wiedzy dziedzinowej (patrz przykład na rysunku 2.2.1).



Rys. 2.2.1. Dopasowanie dokumentu i zapytania w warstwie wiedzy

Źródło: opracowanie własne.

Wykorzystywana wiedza musi mieć dwojaki charakter [Sparck Jones, 1991; Crestani, van Rijsbergen, 1997]. Pierwszym elementem jest przejście z poziomu słownikowego na poziom pojęciowy. Słowa języka naturalnego są zbyt mało precyzyjne, niejednoznaczne, aby mogły stanowić podstawę do szukania dobrego dopasowania. Pojęcia, czyli pewne klasy obiektów, są znacznie bardziej precyzyjne. Przede wszystkim pojęcia są

jednoznaczne. Jedno pojęcie może być opisywane przez kilka deskryptorów (słów). Na przykład słowa „auto" i „samochód" opisują bez wątpienia to samo pojęcie. Jeden deskryptor może być przypisany kilku różnym pojęciom. Np. słowo „klucz" oznaczać może element otwierający zamek w drzwiach, grupę latających obiektów, czy też kod pozwalający odczytać zaszyfrowaną wiadomość.

Pojęcia są znacznie bardziej jednoznaczne niż słowa, ale nadal znajomość modelu danych nie jest absolutnie precyzyjna. Między pojęciami istnieją różnorakie powiązania, mogące być podstawą do skojarzeń przydatnych na potrzeby wyszukiwania. Drugi segment wiedzy przydatny dla naszych celów musi więc dotyczyć zależności między pojęciami. Ogólnych zależności semantycznych – na przykład między pojęciami ogólniejszymi a bardziej szczegółowymi pojęciami bliskoznacznymi. A także różnego rodzaju powiązań o charakterze dziedzinowym, z zakresu tematyki dokumentu, czy kolekcji dokumentów.

Osią łączącą oba te segmenty wiedzy są pojęcia. Dla potrzeb systemów wyszukiwania informacji, szczególnie przydatne są więc podejścia do reprezentacji wiedzy oparte na pojęciach. Zwróćmy dalej uwagę, że raczej nie będą nas interesowały jakieś bardzo szczegółowe i skomplikowane definicje zachowania się obiektów danej klasy-pojęcia. Potrzebna nam wiedza wyraża się przede wszystkim w samej semantyce pojęć i rozmaitych związków między nimi.

Wiedza ta złożona jest z dosyć prostych elementów, z dużą liczbą powiązań między nimi, jak również miedzy pojęciami a ich deskryptorami słownikowymi. W związku z tym niemal idealnie nadaje się ona do modelowania przy użyciu struktur sieciowych. Węzły sieci reprezentują pojęcia, ich deskryptory (słowa kluczowe) oraz zapytania i dokumenty. Połączenia w sieci reprezentują powiązania między tymi elementami. Jeśli nawet stwierdzenie, że obecnie reprezentacja wiedzy w formie różnego rodzaju struktur sieciowych całkowicie zdominowała dziedzinę systemów wyszukiwania informacji, będzie pewną przesadą, to doprawdy niewielką.

Ogólna koncepcja wykorzystania sieci pojęciowych do skojarzeniowego wyszukiwania informacji staje się jasna, jeśli spojrzymy uważniej na rysunek 2.2.1. Wykonywane przez użytkownika zapytanie zamieniane jest na odpowiednią reprezentację w formie cech, na ogół słów kluczowych, stanowiących pewne deskryptory pojęć w warstwie wiedzy. Następnie aktywowane są pojęcia w warstwie wiedzy, powiązane z tymi słowami kluczowymi. Z kolei na podstawie wiedzy dziedzinowej i zdefiniowanych przez nią połączeń w sieci pojęć aktywowane są dodatkowe

węzły, reprezentujące pojęcia skojarzone z tymi odpowiadającymi termom zapytania. To z kolei pobudza termy w warstwie reprezentacji dokumentów, skojarzone z aktywowanymi pojęciami, co pozwala na określenie wyniku zapytania.

Ranking wynikowy sporządzany jest więc nie tylko na podstawie oryginalnego zapytania, ale także termów skojarzonych na podstawie wiedzy dziedzinowej, pobudzonych w sieci pojęć. Z tego powodu wykorzystany w procesie wnioskowania mechanizm określa się często rozprzestrzenianiem aktywacji w sieci pojęć.

Do reprezentacji wiedzy w systemie wyszukiwawczym możemy wykorzystać bardzo różne metody i podejścia. Określają one charakter połączeń między węzłami reprezentującymi poszczególne pojęcia, sposób przeliczania sieci oraz jej strukturę. Generalnie możemy mówić o trzech podstawowych, najczęściej wykorzystywanych rodzajach sieci pojęciowych [Crestani, van Rijsbergen, 1997]:

- Sieci semantyczne – połączenia między węzłami sieci odzwierciedlają różnego typu powiązania między reprezentowanymi przez nie pojęciami. Połączenia są jednokierunkowe i semantyka każdego z nich musi być rozpoznana i jawnie określona (zazwyczaj w postaci przydzielonej mu etykiety). Np. jeśli wiedza dziedzinowa modelowana w sieci odzwierciedla zależność, między obiektem „Kowalski" i obiektem „samochód", to rodzaj tego powiązania musi być określony (czy jest to relacja typu „jest właścicielem", albo „jest kierowcą", czy też „wypożyczył" itp.) i przypisana połączeniu między węzłami sieci reprezentującymi Kowalskiego i samochód. Aktywacja rozprzestrzenia się od warstwy zapytania poprzez połączenia do warstwy wiedzy, a następnie między węzłami reprezentującymi pojęcia wiedzy dziedzinowej, pobudzając kolejne węzły sieci zgodnie z tą wiedzą. Następnie pobudzane są węzły termów dokumentów i określany stan węzłów dokumentów. Wykorzystanie sieci semantycznych dokładniej przedstawione zostanie w kolejnym punkcie 2.2.2.2.
- Sieci asocjacyjne – połączenia między węzłami odzwierciedlają ogólnie istnienie pewnego, bliżej niesprecyzowanego, związku między reprezentowanymi przez nie pojęciami. Połączenie między węzłami sieci asocjacyjnej zazwyczaj ma charakter dwustronny, aczkolwiek nie jest to bezwzględne wymaganie. Nie określa się charakteru tego powiązania, a jedynie jego siłę, definiowaną przy pomocy współczynnika wagowego połączenia w sieci. Zasadniczo, sieci asocjacyjne modelowane są z wykorzystaniem specjalnych architektur sieci neuronowych, zwykle o lokalnej reprezentacji

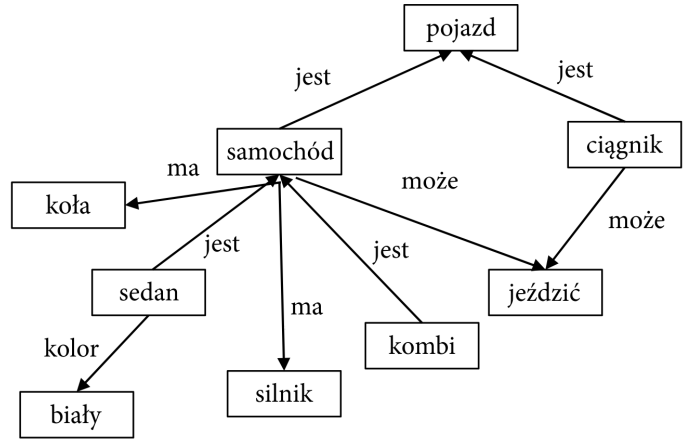
wiedzy (poszczególne elementy wiedzy, przechowywane w węzłach i połączeniach sieci mają swoją określoną interpretację) – czasami określa się tego rodzaju rozwiązania sieciami konekcjonistycznymi. Zastosowania tego rodzaju sieci neuronowych w procesie asocjacyjnego dopasowania zapytań i dokumentów zostaną przedstawione w punkcie 2.2.2.3.

- Sieci wnioskujące – w tym przypadku, powiązania między pojęciami interpretowane są jako zależności logiczne opisane w kategoriach rachunku prawdopodobieństwa. Podobnie jak w sieciach asocjacyjnych, nie określa się typu powiązania. Rozprzestrzenianie aktywacji związane jest z procesami wnioskowania probabilistycznego opartymi na ogół na twierdzeniu Bayesa (rzadziej na rozszerzonych paradygmatach modelowania niepewności wnioskowania, takich jak np. teoria Dempstera-Shafera) i wyznaczaniu prawdopodobieństwa *a posteriori* kolejnych zdarzeń związanych z węzłami sieci. Ranking dokumentów tworzony jest na zasadach rankingowania probabilistycznego. Zastosowania sieci wnioskujących do asocjacyjnego wyszukiwania informacji, zostaną omówione w punkcie 2.2.2.4.

2.2.2.2. Wyszukiwanie skojarzeniowe z użyciem sieci semantycznych

Podstawowe zasady reprezentacji wiedzy w formie sieci semantycznej wprowadzone zostały przez Quillana w 1968 roku [Crestani, 1997]. Sieci semantyczne składają się z węzłów reprezentujących pojęcia, ich konkretne egzemplarze oraz właściwości. Pojęcia układają się w pewne hierarchiczne zależności. Typowe przykłady takich struktur to hierarchie specjalizacji/generalizacji, czy też całość/część. Zależności między pojęciami definiowane są przy pomocy połączeń w sieci, zaetykietowanych informacją o rodzaju zależności.

Główne zależności między pojęciami to relacja „jest egzemplarzem" służąca do definiowania związku między pojęciem a konkretnym obiektem należącym do klasy przez nie definiowanej oraz „jest" służąca do definiowania zależności między pojęciem bardziej szczegółowym i ogólnym. Hierarchia specjalizacji/generalizacji ma specyficzny i zdecydowanie podstawowy charakter. Pojęcia wyżej w tej hierarchii definiują bardziej abstrakcyjne nadklasy, zaś niżej bardziej szczegółowe podklasy. Przyjmuje się, że właściwości przypisane pojęciom nadklasy są również właściwościami wszystkich podklas. Przykład prostej sieci semantycznej przedstawiony został na rysunku 2.2.2.



Rys. 2.2.2. Przykład prostej sieci semantycznej

Źródło: opracowanie własne.

Jeden z podstawowych sposobów korzystania z sieci semantycznej, jak wspomnieliśmy w poprzednim punkcie, polega na tzw. rozprzestrzenianiu aktywacji. Jak wynika z tego, co już sobie powiedzieliśmy, sieć semantyczna jest również siecią asocjacyjną, a więc może być w ten sposób wykorzystana. Procedura jest dosyć prosta: począwszy od pewnych jednostek źródłowych (na przykład reprezentujących termy zapytania), sygnał w sieci przekazywany jest do jednostek z nimi powiązanych, aż wreszcie dotrze do jednostek docelowych (na przykład reprezentujących dokumenty).

Pojawia się tu jednak pewien problem. Połączenia między węzłami w sieci mają charakter lokalny – jednostki powiązane są tylko z tymi, z którymi pozostają w jakimś związku w świecie rzeczywistym. W związku z tym jednokrotne przeliczenie stanu sieci spowoduje pobudzenie tylko węzłów sąsiednich, czyli bezpośrednio połączonych z obudzonymi wcześniej. Aby sygnał dotarł do wszystkich jednostek, do których powinien dotrzeć, zwykle niezbędne jest więc wykonanie kilku przeliczeń sieci, pobudzających kolejne węzły. Rozprzestrzenianie aktywacji jest więc dynamicznym procesem iteracyjnym.

Pojedyncza iteracja sieci składa się z samej fazy rozprzestrzenienia sygnału, nazywanej zwykle pulsem sieci, oraz ze sprawdzenia kryterium zatrzymania procesu iteracji. Ponadto pojedynczy puls składać się może z trzech faz:

1. Fazy korekty wstępnej,
2. Procesu rozprzestrzenienia sygnału (przeliczenia sieci),
3. Fazy korekty końcowej.

Faza wstępna i końcowa są opcjonalne i zwykle zawierają elementy związane z zachowaniem kontroli nad siecią w trakcie całego procesu kolejnych iteracji. Na ogół związane są one z operacjami sterowania aktywacją (np. wytłumiania jej w sytuacjach niepożądanych) zarówno poszczególnych jednostek, jak i większych fragmentów bądź nawet całej sieci.

Samo rozprzestrzenianie aktywacji polega na obliczeniu dla każdej *j*-tej jednostki jej łącznego pobudzenia:

$$I_j(p) = \sum_{i=1}^{k} O_i(p-1) \cdot w_{ij} \tag{2.2.22}$$

Tak więc pobudzenie węzła sieci wyznaczane jest jako iloczyn wyjścia wszystkich węzłów *i* z nim połączonych (ustalonych w poprzedniej iteracji), mnożonych przez wagi połączenia z węzła *i*-tego do *j*-tego. Wagi połączeń mogą mieć charakter rzeczywisty, chociaż w sieciach stricte semantycznych, o zdefiniowanej semantyce połączeń, zwykle przyjmuje się wartości binarne: 0, 1, lub hamujące/wzmacniające: –1, 1. W tego rodzaju sieciach siłę powiązania na ogół determinuje znana semantyka powiązania, w kontekście aplikacji. Wagi o wartościach rzeczywistych częściej stosowane są w sieciach asocjacyjnych, w których mamy do czynienia tylko z jednym, generycznym, typem połączenia [Crestani, 1997].

Po określeniu pobudzenia węzła sieci, kolejnym krokiem jest określenie wartości wyjściowej generowanego przez niego sygnału. Odbywa się to przez zastosowanie właściwej funkcji aktywacji *f*:

$$O_j(p) = f\left(I_j(p)\right) \tag{2.2.23}$$

Typowe funkcje aktywacji wykorzystywane do wyznaczenia wyjścia węzła sieci są takie same, jak w przypadku modelowania neuronów w sieciach neuronowych: funkcja logistyczna, liniowa. Najczęściej jednak wykorzystywana jest funkcja progowa skoku binarnego:

$$O_j(p) = \begin{cases} 0 \text{ gdy } I_j(p) < k_j \\ 1 \text{ gdy } I_j(p) \geq k_j \end{cases} \tag{2.2.24}$$

gdzie k_j jest wartością progu dla jednostki *j*.

Po obliczeniu wartości wyjściowej sygnału danego węzła, wartość ta wysyłana jest do wszystkich węzłów, do których jest on podłączony.

Tak jak powiedzieliśmy, aktywacja w sieci rozprzestrzenia się w kolejnych iteracjach, aż do osiągnięcia kryterium zatrzymania tego procesu.

Taki swobodny proces rozprzestrzeniania się aktywacji charakterystyczny jest jednak bardziej dla sieci asocjacyjnej, o generycznym i jednorodnym charakterze powiązań między jednostkami. W przypadku sieci semantycznej o różnorodnych typach powiązań, trudno byłoby zintegrować swobodnie przepływający sygnał w jednorodny mechanizm wnioskowania. W dodatku stosując wyłącznie mechanizm rozprzestrzeniania aktywacji jako taki, nie wykorzystujemy przecież ważnej informacji, wynikającej ze znajomości rodzaju połączeń.

Jest jeszcze pewien powód techniczny, co wcale nie znaczy, że mniej ważny. Otóż jeżeli przypomnimy sobie właściwości sieci neuronowych, to powinniśmy pamiętać że dla zapewnienia zbieżności procesu relaksacji sieci rekurencyjnej, wagi połączeń muszą spełniać pewne określone warunki, co jest zapewniane przez algorytmy uczenia. Na przykład macierz wag sieci w pełni połączonej musi być symetryczna – co oznacza, w istocie rzeczy, stosowanie połączeń dwustronnych. Warunek ten częściej bywa spełniony dla sieci asocjacyjnych. Założenie, że jakieś bliżej nieokreślone skojarzenie między dwoma pojęciami ma taką samą siłę, jeśli spojrzymy na nie z obu stron, jest często możliwe do zaakceptowania.

W sieci semantycznej połączenia mogą być przecież jednostronne, a jeśli już obowiązują w obie strony, to może to się odbywać z różną siłą. Ponadto sieci semantyczne, mają charakter lokalny. Wagi połączeń mogą wynikać z wiedzy o określonej zależności między zmiennymi i trudno byłoby wymusić przy ich tworzeniu jakiś zupełnie globalny mechanizm kontrolowania ich wartości. W sieci semantycznej, iteracyjny proces swobodnego rozprzestrzeniania aktywacji, w znacznej części wypadków trwałby aż do maksymalnego pobudzenia całości sieci.

Dlatego w sieciach semantycznych wprowadza się zwykle pewien zestaw heurystyk, albo rozprzestrzenia się aktywację zgodnie z pewnymi regułami wnioskowania specyficznymi dla danego rodzaju połączenia. Jedna z podstawowych metod wymuszenia tego rodzaju reguł, polega na wprowadzeniu zestawu ograniczeń na rozprzestrzenianie się aktywacji, kontrolowanego lokalnie dla poszczególnych jednostek, z wykorzystaniem informacji o semantyce połączeń. W sieciach semantycznych stosowany jest więc na ogół proces ograniczonego rozprzestrzeniania aktywacji.

Do typowych rodzajów ograniczeń stosowanych do kontrolowania rozprzestrzeniania aktywacji, należą [Crestani, 1997]:

- Ograniczenia odległości. Przekazywana aktywacja powinna być wygaszana, kiedy dociera do węzłów, które są zbyt odległe w kategoriach liczby połączeń od wstępnie aktywowanych. Odpowiada to prostej regule heurystycznej, że siła związku między dwoma

jednostkami maleje, wraz ze wzrostem semantycznej odległości między nimi. Powiązania mogą być klasyfikowane zgodnie z odległością na którą działają, mierzoną w liczbie połączeń. Zależności między dwoma węzłami bezpośrednio połączonymi, nazywane są związkami pierwszego rzędu. Zależności między dwoma węzłami połączonymi przez jeden węzeł pośredniczący, nazywane są związkami drugiego rzędu itd. Dosyć powszechnie rozważa się jedynie zależności pierwszego, drugiego i co najwyżej trzeciego rzędu, ale zależy to już od aplikacji.

- Ograniczenia rozchodzenia się. Aktywacja powinna być wygaszana w węzłach o bardzo dużej liczbie połączeń wyjściowych do innych jednostek. Celem tego ograniczenia jest zapobieżenie zbyt szerokiemu rozprzestrzenieniu się aktywacji, na znaczne obszary sieci, co mogłoby być skutkiem propagacji sygnału przez węzły o bardzo szerokim znaczeniu semantycznym i dlatego podłączone do wielu innych.
- Ograniczenia ścieżek przesyłania sygnału. Zazwyczaj aktywacja rozprzestrzenia się w sieci, korzystając ze wszystkich dostępnych połączeń. Czasami jednak powinna ona być kanalizowana do preferowanych ścieżek, odzwierciedlających zależne od aplikacji reguły wnioskowania. Może to być modelowane poprzez wykorzystywanie połączeń, jeśli połączenia te są odpowiednio zaetykietowane, kierując przez nie sygnał, wytłumiając go jednocześnie w innych, mniej znaczących ścieżkach.
- Ograniczenia aktywacji. Poszczególnym węzłom mogą być przypisywane dodatkowe progi, ograniczające ich aktywację. Wartości tych progów mogą być zmienne, dostosowując się do łącznego pobudzenia w całej sieci.

Innym jeszcze dodatkiem do procesu rozprzestrzeniania aktywacji może być wykorzystanie informacji zwrotnej ze źródła zewnętrznego. Użytkownik bądź jakiś zewnętrzny proces może obserwować stan aktywacji całej sieci lub wybranych jej elementów i modyfikować je stosownie do swoich potrzeb. W konsekwencji aktywacja rozprzestrzenia się dalej zależnie od doprecyzowanych wskazówek użytkownika. Ponadto użytkownicy mogą mieć możliwość wskazywania odpowiednich dla nich ścieżek dalszego przepływu sygnału w sieci.

Wyżej wymienione (i oczywiście inne, specyficzne) ograniczenia nakładane na proces rozprzestrzeniania aktywacji, zazwyczaj implementowane są w fazie wstępnej lub końcowej korekty, pulsu sieci.

Sieci semantyczne, wykorzystywane w wyszukiwaniu informacji, opierają się zazwyczaj na istniejących powiązaniach między słowami kluczowymi lub dokumentami. Węzły odpowiadają termom, dokumentom, książkom,

artykułom, czasopismom, klasyfikacjom tematycznym, autorom itd. Połączenia wskazują raczej na dosyć ogólne asocjacje między obiektami, na przykład wystąpienie termu w zapytaniu, publikację dokumentu, zaindeksowanie dokumentu danym termem, zaklasyfikowanie dokumentu do danej tematyki itp. Są to powiązania często epizodyczne, nietrwałe i charakterystyczne tylko dla jednej, konkretnej aplikacji (kolekcji dokumentów). Sieć semantyczna zbudowana na potrzeby jednej aplikacji, niemal na pewno nie będzie mogła zostać wykorzystana w innej.

Pierwsze próby zastosowania rozprzestrzeniania aktywacji w sieciach semantycznych do wyszukiwania informacji, miały miejsce w 1981 roku i związane były z osobą S. E. Preece'a [Crestani, 1997].

Preece argumentował, że większość klasycznych w tym okresie podejść może zostać przedstawiona w kategoriach rozmaitych technik rozprzestrzeniania aktywacji w reprezentacji sieciowej, zbudowanej na kolekcji dokumentów. Zestawiając razem różne struktury sieciowe danych z wybranymi rozmaitymi procedurami przetwarzania, pokazał, że możliwa jest reprezentacja z wykorzystaniem sieci semantycznych modelu booleʼowskiego, wektorowego i różnych podejść do ważenia termów, stosowanych w skojarzeniowym wyszukiwaniu informacji.

Koncepcje Preece'a związane były z pojawieniem się pierwszych publikacji na temat sieci neuronowych i widać w nich chęć rozciągnięcia paradygmatu rozprzestrzeniania aktywacji w tym właśnie kierunku. Możliwości obliczeniowe, jakimi dysponował Preece na przełomie lat siedemdziesiątych i osiemdziesiątych, nie pozwoliły mu jednak na powiązanie rozprzestrzeniania aktywacji w obszernych sieciach semantycznych z metodami uczenia maszynowego. Wykorzystywana przez niego do reprezentacji kolekcji dokumentów sieć semantyczna zbudowana była ręcznie. Preece wskazywał jednak na konieczność zastosowania do konstrukcji sieci rozwiązań automatycznych.

Prace Preece'a miały oczywiście charakter bardziej teoretyczny niż duże znaczenie praktyczne. Stworzył on jednak pewien wspólny formalizm łączący tematykę wyszukiwania informacji i sztucznej inteligencji. Prace te stały się inspiracją dla wielu kolejnych badaczy, a niektóre z jego koncepcji wykorzystywane są jeszcze wiele lat później [Crestani, 1997].

Na początku lat osiemdziesiątych, równolegle do Preece'a, P. Shoval prowadził próby nad interaktywnym rozszerzaniem zapytania, przy użyciu rozprzestrzeniania aktywacji w sieciach semantycznych [Shoval, 1981, 1985; Crestani, 1997]. Baza wiedzy zastosowana przez Shovala miała formę sieci semantycznej opartej na tezaurusie – słowniku obejmującym informacje o zależnościach między termami.

Wykorzystywane w niej typy połączeń między słowami kluczowymi miały taki sam charakter jak powszechnie stosowane w tezaurusach: wyrażały zależności hierarchiczne między termami ogólniejszymi i bardziej szczegółowymi, powiązania między synonimami, oraz generalne powiązania w sensie skojarzeniowym. Ponadto Shoval dodał do sieci jeszcze dwa rodzaje powiązań. Jeden – połączenia składające pojedyncze termy w pojęcia opisywane frazami o kilku słowach: np. „information" i „systems" były połączone w „information systems". Drugi rodzaj powiązania, tzw. połączenia modelu, pozwalały na rozszerzenie wiedzy pojęciowej o elementy wiedzy dziedzinowej – np. słowo „business" połączone było z „organizational area".

Sieć semantyczna przetwarzana była przy pomocy metody rozprzestrzeniania aktywacji, o charakterze interakcyjnym, z informacją zwrotną od użytkownika, który obserwując wyniki aktywacji sieci w danej iteracji, przed wykonaniem kolejnej mógł wskazywać w których kierunkach aktywacja mogła zostać zaakceptowana, a w których powinna być wyhamowywana.

System Shovala posiadał szereg interesujących właściwości, a przede wszystkim możliwości automatycznej konstrukcji reprezentacji sieciowej przy pomocy informacji generycznej, zawartej w tezaurusie. Sama metoda rozprzestrzeniania aktywacji, była jednak mocno uproszczona i wymagała dużej ilości informacji zwrotnej od użytkownika, której uzyskanie, oczywiście, jak zwykle jest dosyć trudną kwestią [Crestani, 1997].

Podobne prace związane asocjacyjnym wyszukiwaniem informacji w oparciu o sieci semantyczne wykorzystujące tezaurusy, prowadzone były przez całe lata osiemdziesiąte, wymienić można tu system Coder Foxa [Fox, 1987], badania nad podobieństwem pojęć w sieciach semantycznych.

Jednym z pierwszych praktycznych systemów wykorzystujących do wyszukiwania skojarzeniowego metodę ograniczonego rozprzestrzeniania aktywacji był stworzony w połowie lat osiemdziesiątych system Grant Cohena i Kjeldsena [Cohen, Kjeldsen, 1987; Crestani, 1997].

W systemie Grant stworzona została sieć semantyczna gromadząca wiedzę o wnioskach badawczych i potencjalnych agencjach finansujących badania naukowe. Tematy badawcze określane we wnioskach połączone były z tymi instytucjami gęstą siecią różnego rodzaju powiązań. Zapytanie mogło obejmować jeden lub więcej obszarów badawczych, albo jedną lub więcej agencji finansujących. Wyszukiwanie prowadzone było przy pomocy metody ograniczonego rozprzestrzeniania aktywacji w powstałej w ten sposób sieci, wykorzystując ograniczenia niemal wszystkich

typów z przedstawionych powyżej. Szczególnie szeroko wykorzystywane były ograniczenia ścieżek.

Z ogólnego punktu widzenia Grant można uważać za system wnioskujący, stosujący wielokrotnie pojedynczy schemat wnioskowania:

$$\text{IF } x \text{ AND } R(x, y) \rightarrow y$$

gdzie $R(x, y)$ oznacza ścieżkę łączącą węzły x i y, która mogła składać się jednego lub większej liczby połączeń. W tej określonej aplikacji, dla której zbudowany został Grant, tj. wyszukiwania agencji finansujących dla wniosków badawczych, powyższy schemat wnioskowania równoważny był regule wnioskującej w postaci: „jeżeli agencja finansująca zainteresowana jest tematem x oraz istnieje powiązanie między tematem x i y, wówczas agencja finansująca będzie najprawdopodobniej zainteresowana również tym powiązanym tematem y".

Proces wskazywania ścieżek dawał pewnym ścieżkom preferencje (pozytywne wskazanie) lub pozwalał ich uniknąć (przy negatywnym wskazaniu). Mechanizm oceny ścieżek umożliwiał stworzenie rankingu wyszukanych w sieci węzłów reprezentujących agencje finansujące. Wykorzystanie ograniczonego rozprzestrzeniania aktywacji w stworzonej sieci semantycznej dało bardzo obiecujące efekty, zgodnie z podawanymi przez autorów wynikami testowania systemu lepsze od otrzymywanych przy pomocy prostego wyszukiwania opartego na słowach kluczowych.

System Grant przeznaczony był do realizacji wyszukiwania w jednej określonej dziedzinie, wykorzystywał też w znacznym stopniu wiedzę z jej zakresu. Sieć semantyczna została zbudowana w sposób manualny, przy wykorzystaniu znacznych nakładów związanych z inżynierią wiedzy. Prace wymagały przeprowadzenia dogłębnej analizy dziedziny, w której miał pracować system, w celu określenia właściwych pojęć i związków między nimi – aby można wbudować je w sieć – oraz zdobycia preferencji, umożliwiających rozprzestrzenianie się aktywacji zakodowanymi ścieżkami. Jednym z głównych problemów systemu Grant były trudności z korygowaniem parametrów procesu wskazywania ścieżek.

Kolejnym istotnym krokiem na drodze rozwoju systemów wyszukiwawczych, opartych na rozprzestrzenianiu aktywacji w sieciach semantycznych, był system I^3R, zbudowany przez grupę badaczy Crofta, Lucię, Cohena, Crigeana, Willeta i Thompsona [Croft, Lucia, Cohen, 1988; Croft, Lucia i in., 1989; Croft, Thompson, 1987]. Wykorzystywał on tę metodę do określania pojęć skojarzonych z termami zapytania. Rozprzestrzenianie aktywacji było równoważne propagacji niepewności

z wykorzystaniem wnioskowania wiarygodnego (ang. *plausible inference*) przez system reguł z określonymi współczynnikami wiarygodności (ang. *credibility factors*).

System I^3R wykorzystywał bazę wiedzy w formie sieci semantycznej, złożonej z węzłów reprezentujących pojęcia, termy i dokumenty, powiązane szeregiem związków semantycznych. Zastosowano formę ograniczonego rozprzestrzeniania aktywacji, opartą na następujących warunkach [Croft, Lucia, Cohen, 1988; Crestani, 1997]:

- Rozprzestrzenianie aktywacji rozpoczyna się od zestawu dokumentów, znajdujących się najwyżej w rankingu będącym wynikiem procedury wyszukiwania opartej na modelu probabilistycznym. Nie korzystano bezpośrednio z węzłów reprezentujących termy zapytania, ani odpowiadających im połączeń term-dokument, term-term, term-pojęcie, ani pojęcie-pojęcie.
- W pierwszym kroku do rozprzestrzeniania aktywacji system wykorzystywał połączenia do dokumentów najbliższych sąsiadów, oraz związane z cytowaniem dokumentów. Te połączenia reprezentowały najsilniejsze wiarygodne powiązania między dokumentami.
- W kolejnych cyklach rozprzestrzeniania aktywacji, korzystano tylko z połączeń do najbliższych sąsiadów. Powiązania związane z cytowaniem były interesujące tylko w odniesieniu do dokumentów startowych.
- W określeniu poziomu aktywności węzła, wykorzystywano wagi na połączeniach. Ustalane one były jako wartości wiarygodności reguły wnioskowania reprezentowanej przez połączenie między węzłami.
- Dokumenty, które były już użyte jako część ścieżki aktywacji nie były wykorzystywane ponownie, jeśli zostały reaktywowane.

Aktywacja w sieci rozprzestrzeniała się aż do osiągniecia kryterium zatrzymania, a następnie ranking dokumentów sporządzany był na podstawie poziomów aktywacji ich węzłów.

Innym przykładem systemu wyszukiwania informacji korzystającego z metody rozprzestrzeniania aktywacji w sieci semantycznej był Metacat [Chen, Dhar, 1991; Chen, 1992]. Był to bardziej klasyczny program, wyszukujący dokumenty w kolekcji dziedzinowej z zakresu medycyny. Poświęcimy mu odrobinę uwagi, ponieważ wykorzystywał nieco inną strategię sterowania rozprzestrzenianiem aktywacji niż wspomniany poprzednio system Grant.

Jak widzieliśmy, metoda ograniczonego rozprzestrzeniania aktywacji, polega generalnie na wytłumianiu sygnału w niepożądanych węzłach i połączeniach. To znaczy, dla nowego pulsu obliczamy pobudzenia

wszystkich jednostek sieci, a potem w miejscach, w których jest to niezbędne, wytłumiamy go. W efekcie ograniczone rozprzestrzenianie aktywacji daje kontrolowany, lecz jednak dosyć szeroki, rozpływ sygnału w sieci. Chen i Dhar zastosowali jeszcze ściślejszą kontrolę rozpływu aktywacji, tylko przez ustalone połączenia, ukierunkowując go wzdłuż określonych ścieżek. Sygnał danej jednostki w nowym pulsie sieci nie rozpływa się do wszystkich połączonych węzłów, ale lokalnie, wyłącznie przez jedno konkretne wybrane połączenie. Do kontroli procesu rozprzestrzeniania aktywacji i wyboru połączeń zastosowany został algorytm oparty na znanej metodzie przeszukiwania grafu sieciowego – metodzie podziału i ograniczania (ang. *branch and bound*).

Sama struktura sieci semantycznej, oparta jest na tezaurusie i sieć zawiera typowe połączenia słownikowe: term o węższym znaczeniu, term o szerszym znaczeniu, term powiązany i term synonim. Działanie systemu wyszukiwawczego również ma charakter klasyczny: termy zapytania (wcześniej jeszcze przetwarzane przy pomocy innych metod) pobudzają węzły źródłowe sieci semantycznej, następnie sygnał przechodzi przez sieć, pobudzając termy indeksujące dokumenty.

W kontrolowaniu procesu rozprzestrzeniania aktywacji w sieci, stosowane są następujące cztery heurystyki [Chen, Dhar, 1991; Chen, 1992]:

1. Termy szczegółowe jako pierwsze. Węzły mające mniej sąsiadów w sieci, zazwyczaj reprezentują pojęcia o bardziej szczegółowym znaczeniu niż węzły o większej liczbie sąsiadów. System stosuje heurystykę odwiedzania najpierw jednostek o mniejszej liczbie połączeń wyjściowych.
2. Połączenia szczegółowe jako pierwsze. System rozprzestrzenia sygnał najpierw w połączeniach do termów o węższym znaczeniu, następnie do termów powiązanych, i dopiero w dalszej kolejności do termów o szerszym znaczeniu.
3. Mniejsza odległość jako pierwsza. W trakcie procesu aktywacji system prześle sygnał do jednostek, które są bliżej węzłów źródłowych (w mniejszej odległości) niż do położonych dalej. Termy dalej położone od źródłowych są mniej relewantne dla danego zapytania w porównaniu do termów znajdujących się bliżej.
4. Rozprzestrzenianie na dwa poziomy. Jak już wspomnieliśmy wcześniej, w sieciach semantycznych liczba połączeń między dwoma węzłami determinuje semantyczną odległość między nimi. W celu wykrycia wyłącznie takich termów, które są blisko powiązane ze źródłowymi, system rozprzestrzenia aktywację każdego węzła źródłowego jedynie o dwa poziomy.

Każdej ścieżce przepływu sygnału, wytyczanej w sieci semantycznej, przypisywany jest pewien koszt oparty na odwiedzonych wcześniej węzłach, rodzaju pokonanych wcześniej połączeń oraz ich liczbie. Funkcja kosztu tworzona jest zgodnie z wymienionymi wcześniej heurystykami, branych jest pod uwagę szereg czynników, między innymi specyfika wchodzących w jej skład węzłów (poprzez analizę ich sąsiadów), typy łączy i długość ścieżki. Koszt ścieżki można traktować jako pewną odległość (w sensie różnic w znaczeniu pojęć) między wyjściowym węzłem źródłowym i nowo odwiedzanym, kończącym ścieżkę. W każdym kroku wydłużana jest tylko ścieżka o najniższym koszcie.

W latach dziewięćdziesiątych dwudziestego wieku trwały prace nad tworzeniem uniwersalnych narzędzi do konstruowania sieci semantycznych na bazie kolekcji dokumentów oraz słowników, najbardziej chyba znanym systemem w tym okresie był zrealizowany w laboratoriach firmy Microsoft projekt MindNet [Richardson, Dolan, Vanderwende, 1998]. Innym przykładem zbliżonego rozwiązania była sieć semantyczna, działająca w oparciu o zasadę rozprzestrzeniania aktywacji ASKNet [Harrington, Clark, 2009; Harrington, 2009, 2010]. Ostatnie lata przyniosły również szereg zastosowań praktycznych, związanych z ontologiami i inicjatywą Semantic Web. Ontologie stanowią formę sieci semantycznych, opisujących zasoby internetowe. Przykładem takiego rozwiązania jest system wyszukiwania informacji turystycznej [Berger, Dittenbach, Merkl, 2004; Wu, Li i in., 2008].

Początkowe rozwiązania w zakresie budowy sieci semantycznych wykorzystywały zwykle wiedzę pozyskiwaną od ekspertów lub istniejącą wiedzę w formie tezaurusów. W latach 90. XX wieku pojawiły się metody generowania zależności między termami w automatycznych tezaurusach, prostych sieciach semantycznych przeznaczonych do wspomagania działania usług informacyjnych [Chen, 1995; Frakes, Baeza-Yates, 1992; Sanderson, Croft, 1999]. W ostatnich dwudziestu latach rozwinięto metody automatycznego generowania zależności semantycznych na podstawie tekstów [Pantel, Pennacchiotti, 2008] [Buitelaar, Cimiano, 2008; Harrington, Clark, 2009]. Dobry przegląd tematyki generowania sieci semantycznych (w formie tzw. ontologii) na podstawie tekstów, znaleźć można w [Cimiano, 2006].

Do zagadnień automatycznego tworzenia zwłaszcza tezaurusów wyszukiwawczych i prostych sieci semantycznych, będziemy jeszcze częściowo wracać w dalszej części bieżącego rozdziału, zwłaszcza w punkcie 2.5, poświęconym rozszerzaniu zapytania w systemach wyszukiwania informacji.

2.2.2.3. Wyszukiwanie skojarzeniowe z wykorzystaniem sieci asocjacyjnych

W sieciach asocjacyjnych, jak już to kilkakroć nadmieniliśmy, połączenia między węzłami odzwierciedlają istnienie ogólnego, bliżej niesprecyzowanego, związku między reprezentowanymi przez nie pojęciami. Nie określamy jego rodzaju, a tylko siłę, definiowaną przy pomocy związanego z nim współczynnika wagowego. Ponadto połączenie między węzłami sieci asocjacyjnej zazwyczaj ma charakter dwustronny i jest ono symetryczne, tzn. jeśli między pewnymi pojęciami a i b istnieje powiązanie o sile w_{ab}, to między pojęciami b i a istnieje również powiązanie o sile $w_{ba} = w_{ab}$.

Cała wiedza o zależnościach między pojęciami w sieci asocjacyjnej, definiowana jest więc przez architekturę połączeń oraz ich wagi. Przepływ sygnału w sieci, odbywa się na zasadach rozprzestrzeniania aktywacji, opisanych zależnościami zbliżonymi do określonych w poprzednim punkcie we wzorach (2.2.22) i (2.2.23). Szczegółowe rozwiązania w różnych systemach mogą się oczywiście różnić od tego ogólnego kanonu, lecz zwykle nie są to różnice radykalne.

Podobieństwo zasad działania sieci asocjacyjnej do paradygmatów stosowanych w sieciach neuronowych jest więc dosyć oczywiste. Co prawda sieci asocjacyjne charakteryzują się zwykle lokalną reprezentacją wiedzy – tzn. węzły, połączenia, a przede wszystkim ich wagi mają określoną interpretację. Węzły reprezentują termy (słowa kluczowe), dokumenty, pojęcia dziedzinowe itp., połączenia odpowiadają istniejącym związkom asocjacyjnym, a wagi określają siłę konkretnego powiązania, między konkretnymi obiektami. Wagi, siły powiązań, mogą być predefiniowane z góry, na podstawie wiedzy dziedzinowej, na ogół jednak określane są w procesie uczenia nienadzorowanego, polegającego na zebraniu pewnych statystyk współwystępowania badanych obiektów. Rzadziej (jak się niedługo przekonamy) stosuje się rozwiązania wykorzystujące uczenie nadzorowane.

Jak widzimy, wymienione wyżej zasady działania sieci asocjacyjnych nie są jednak niezgodne z wymaganiami stawianymi sieciom neuronowym, a nawet podobne rozwiązania stosowane są w powszechnie znanych architekturach neuronowych. Dlatego zwykle sieci asocjacyjne traktuje się jako rodzaj sieci neuronowych. Oczywiście istnieje pewna specyfika, są to jednak systemy stanowiące pewien pomost między modelowaniem neuronowym a symbolicznym, ekspertowym. Stąd czasami określa się tego rodzaju rozwiązania jako sieci konekcjonistyczne.

Podsumowując, w bieżącym punkcie zaprezentujemy przegląd zastosowań sieci neuronowych do rankingowania dokumentów w systemach

wyszukiwawczych. Będą to jednak sieci konekcjonistyczne o lokalnej reprezentacji wiedzy, dodające element wyszukiwania asocjacyjnego, skojarzeniowego do funkcji rankingujących stosowanych w istniejących modelach wyszukiwania informacji. Zastosowania klasycznych modeli neuronowych, których zadaniem jest wypracowanie zupełnie nowej funkcji rankingującej, przy pomocy informacji o relewancji, poprawności wyników wyszukiwania, zostaną omówione w jednym z dalszych punktów podrozdziału 2.2, który poświęcony będzie uczeniu się rankingowania.

Jedną z pierwszych prób zastosowania sieci neuronowych do wyszukiwania informacji stanowiły prace Mozera, na początku lat 80. XX wieku [Mozer, 1984]. Zastosowana przez niego sieć neuronowa składała się z dwu warstw neuronów. Jedna warstwa dla $i = 1, \ldots, n_D$ reprezentowała dokumenty, druga dla $j = 1, \ldots, n_t$ – indeksujące je termy. Struktura sieci Mozera przedstawiona została na rysunku 2.2.3.

Neurony termów i dokumentów połączone są tylko wtedy, gdy dany dokument został zaindeksowany określonym termem. Ogólna idea działania sieci polega na tym, że zapytanie pobudza odpowiednie jednostki związane z występującymi w nim termami. To z kolei pobudza dokumenty z nimi połączone. W następnym kroku pobudzane są termy związane z tymi dokumentami (nawet nie występujące w zapytaniu). W ten sposób sieć iteruje w kolejnych cyklach pobudzania się nawzajem neuronów termów i dokumentów.

Przy czym Mozer zastosował asymetryczny model połączeń, tzn.:

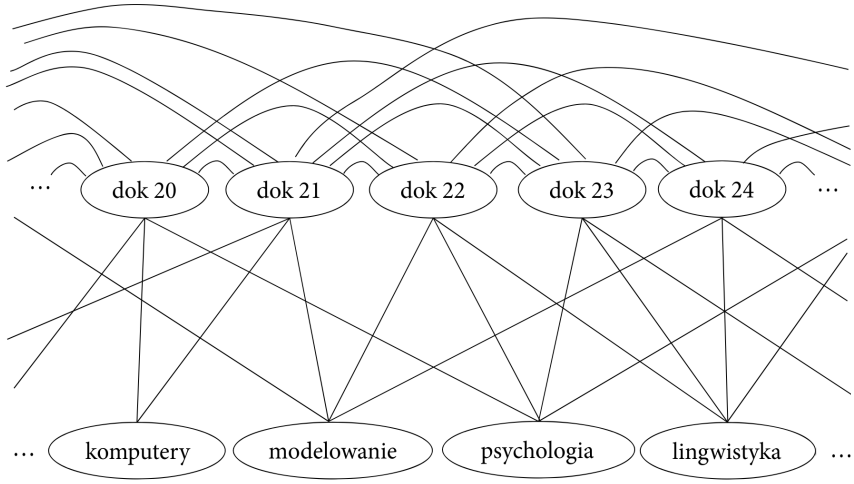
- jest wagą połączenia z *j*-tego termu do *i*-tego dokumentu (0 jeśli brak połączenia, 0,1 w przeciwnym wypadku),
- jest wagą połączenia z *i*-tego dokumentu do *j*-tego termu (0 jeśli brak połączenia, 0,3 w przeciwnym wypadku).

Wagi połączeń w modelu Mozera są więc różne w obu kierunkach. Zwróćmy jednak uwagę, że wszystkie wagi w jednym kierunku mają taką samą wartość. Nie wyrażają więc siły powiązania między obiektami, a tylko brak powiązania (wartość 0) lub jego istnienie (wybrana stała wartość dodatnia – 0,1 lub 0,3, w zależności od kierunku połączenia).

Jak widzimy na rysunku 2.2.3, w modelu Mozera brak jest połączeń między termami. Tym niemniej model wykazywał właściwości wyszukiwania asocjacyjnego. Asocjacje między termami generowane były poprzez ich współwystępowanie w opisach dokumentów oraz naprzemienne cykle wzajemnej aktywacji obu warstw sieci.

Mozer zdawał sobie oczywiście sprawę, że musi zapewnić osiąganie przez sieć stanu stabilnego rekurencji. W tym celu zastosował trzy elementy: hamujące aktywację połączenia między węzłami dokumentów,

częściowe wytłumianie aktywacji z iteracji na iterację oraz utrzymywanie poziomu aktywacji jednostek w zadanym przedziale wartości.



Rys. 2.2.3. Ogólny schemat struktury sieci Mozera

Źródło: opracowanie własne na podstawie [Mozer, 1984].

Jak widzimy na rysunku 2.2.3, w sieci wprowadzono połączenia między neuronami dokumentów, których wagi zdefiniowane zostały następująco:

$w_{D_k D_i}$ jest wagą połączenia z k-tego dokumentu do i-tego dokumentu (0 jeśli $k = i$, 0,0075 w przeciwnym wypadku).

Połączenia między (różnymi) dokumentami działają hamująco na aktywację reprezentujących ich neuronów, co Mozer uzasadnia chęcią wprowadzenia aspektu konkurencyjnego między nimi. Pobudzenie danej grupy termów powinno wspierać dokument najsilniej z nimi powiązany i nie powinno już wspierać innych dokumentów. Silna reakcja jednego dokumentu, stanowi więc przesłankę do osłabienia reakcji innych.

Ponadto, każda jednostka traci w każdym kroku część aktywacji, co ma odpowiadać stopniowemu zanikowi aktywacji w czasie. Przy tym aktywacja nie powinna przekraczać pewnych minimalnych i maksymalnych poziomów. Aby osiągnąć powyższe cele, Mozer wprowadza kolejne cztery parametry systemu:

- θ_t – współczynnik szybkości zanikania aktywacji termu: 0,25,
- θ_D – współczynnik szybkości zanikania aktywacji dokumentu: 0,1,

- M – maksymalny poziom aktywacji: 1,
- m – minimalny poziom aktywacji: –0,2.

Sieć rozpoczyna działanie od aktywowania zbioru jednostek odpowiadających termom zapytania. W eksperymentach Mozera termy w zapytaniu miały charakter pozytywny lub negatywny, co oznaczało ustawienie ich aktywacji, odpowiednio, na maksymalny i minimalny poziom. Jednakże możliwe jest przydzielanie neuronom wejściowym różnych wartości, odpowiadających ich różnej wadze w zapytaniu. Aktywacja jednostek startowych pozostawała stała, nie ulegając zmianom w procesie iteracji sieci.

Następnie system oblicza w danym kroku y łączne pobudzenie wejściowe wszystkich n_D neuronów w warstwie dokumentów, $i = 1, \ldots, n_D$:

$$\eta_{D_i}(y)=\left(\frac{\overline{c}_D}{c_{D_i}}\right)^{n_D}\sum_{j=1}^{n_t} w_{t_j D_i} U\left(t_j(y)\right)-\sum_{k=1}^{n_D} w_{D_k D_i} U\left(D_k(y)\right) \quad (2.2.25)$$

gdzie $\overline{c}_D$ jest średnią liczbą termów indeksujących w dokumencie, w całej kolekcji, liczbą termów indeksujących w dokumencie i, zaś $U(\ )$ jest funkcją tożsamościową z progiem:

$$U(x)=\begin{cases} 0 \text{ dla } x \le 0 \\ x \text{ dla } x > 0 \end{cases}$$

Jak widzimy, łączne pobudzenie wejściowe i-tego neuronu równe jest ważonej sumie aktywacji wszystkich termów, które indeksują reprezentowany przez niego dokument (pozostałe mają zerowe wagi), pomniejszonej o aktywacje innych dokumentów. Składnik aktywacji termów indeksujących skalowany jest przy pomocy współczynnika $\left(\overline{c}_D / c_{D_i}\right)^{n_D}$, który ma wysoką wartość dla dokumentów o niewielkiej liczbie termów indeksujących (w stosunku do średniej), zaś niską wartość, dążącą do 0, dla dokumentów o dużej liczbie termów. Dokumenty o dużej liczbie słów kluczowych są mniej jednorodne tematycznie, w związku z tym powinny generować słabsze asocjacje. Zadaniem tego współczynnika jest oczywiście zapobieganie zbyt szerokiemu rozprzestrzenianiu się aktywacji w sieci i pobudzaniu termów słabo powiązanych z wyjściowymi.

Stan neuronu w następnej iteracji y+1 obliczany jest przy pomocy następującej funkcji aktywacji:

$$D_i(y+1)=(1-\theta_D)D_i(y)+\eta_{D_i}(y)\cdot\begin{cases} M-D_i(y) \text{ gdy } \eta_{D_i}(y)>0 \\ D_i(y)-m \text{ gdy } \eta_{D_i}(y)\le 0 \end{cases} \quad (2.2.26)$$

Czyli, jak widzimy, wyjście neuronu wyznaczane na podstawie jego poprzedniego stanu, z mniejszą od jeden wagą sprzężenia zwrotnego aktywacji ($1 - \theta_D$), oraz poziomu bieżącego pobudzenia wejściowego, forsowanego w stronę maksymalnej lub minimalnej aktywacji, w zależności od znaku.

Po wyznaczeniu stanów neuronów dokumentów, obliczane są aktywacje neuronów warstwy termów. Odbywa się to według zbliżonego schematu. Pobudzenie wejściowe jednostki reprezentującej dane słowo kluczowe zawiera tylko składnik związany z dokumentami, które je wykorzystują:

$$\eta_{t_j}(y) = \left(\frac{\overline{c_t}}{c_{t_j}} \right)^{n_t} \sum_{i=1}^{n_D} w_{D_i t_j} U\left(D_i(y)\right) \tag{2.2.27}$$

Oczywiście, w tym przypadku, c_{tj} jest liczbą dokumentów indeksowanych przez *j*-ty term, zaś oznacza średnią liczbę dokumentów w kolekcji przypadających na jeden term. Idea wykorzystania tych informacji jest taka sama, jak dla jednostek dokumentów.

Funkcja aktywacji również ma podobny charakter:

$$t_j(y+1) = (1-\theta_t) t_j(y) + \eta_{t_j}(y) \cdot \begin{cases} M - t_j(y) \text{ gdy } \eta_{t_j}(y) > 0 \\ t_j(y) - m \text{ gdy } \eta_{t_j}(y) \le 0 \end{cases} \tag{2.2.28}$$

Po wyznaczeniu stanów neuronów w warstwie termów, następuje kolejna iteracja sieci, związana z pobudzaniem jednostek reprezentujących dokumenty itd. Proces ten kontynuowany jest do osiągnięcia przez sieć stanu stabilnego, polegającego na tym, że przyrost netto aktywacji w każdej jednostce wyrównuje się z jej redukcją, to znaczy sytuacji, w której wejścia pobudzające dokładnie odpowiadają wejściom hamującym i zanikaniu aktywacji. Zgodnie z informacjami Mozera [Mozer, 1984], system osiągał równowagę po wykonaniu około 25 kroków. Końcowy ranking dokumentów, dla danego zapytania, sporządzany był po ustabilizowaniu się sieci na podstawie wartości stanów neuronów warstwy dokumentów.

Prototyp sieci został zbudowany na bazie 407 dokumentów i 133 słów kluczowych, przy średnio 3,4 termach na dokument. Przeprowadzono na nim szereg eksperymentów, aby ocenić możliwości zastosowania tego rodzaju rozwiązań w systemach wyszukiwawczych. Badania te pokazały dosyć obiecujące rezultaty, jeśli chodzi o możliwości asocjacyjne

wyszukiwania. Często udało się wyszukać dokumenty, nie posiadające słów kluczowych wspólnych z zapytaniem, a jednak ewidentnie relewantne [Mozer, 1984; Crestani, 1997]. Mozer wskazywał również możliwości wykorzystania systemu w zapytaniach typu query-by-example, czyli poprzez wskazanie dokumentu, lub dokumentów, do których chcemy wyszukać podobne. W takim przypadku należy po prostu rozpocząć proces relaksacji sieci od aktywowania neuronów odpowiadających przykładowym dokumentom.

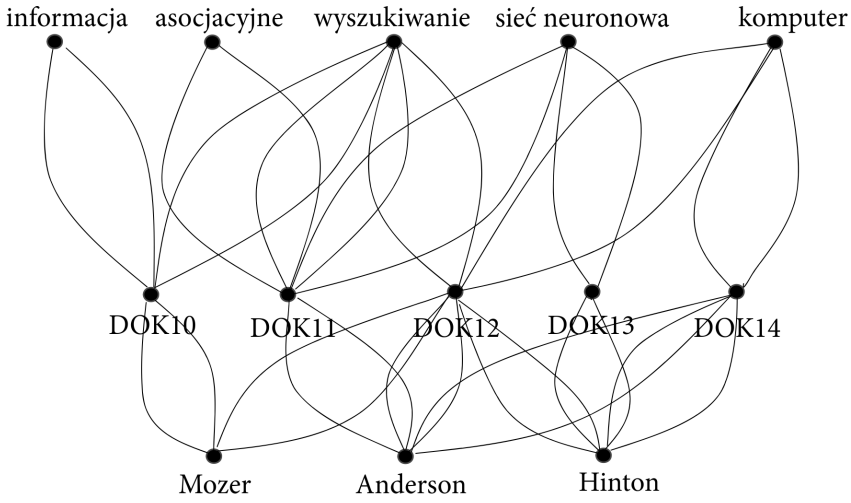
Należy jednak pamiętać, że model Mozera ma wiele istotnych usterek. Testowany był jedynie w warunkach laboratoryjnych, na niewielkiej kolekcji dokumentów. Powstawały pytania, jak będzie się on zachowywał w sytuacji bardziej rzeczywistej. Ponadto istotnym ograniczeniem tej sieci są stałe wagi, nie pozwalające na oddanie różnic w istotności termów indeksujących dla reprezentacji treści dokumentów. Kolejnym problemem jest brak połączeń między neuronami termów. Zależności między termami muszą być identyfikowane poprzez przepływy aktywacji za pośrednictwem powiązanych z nimi dokumentów. Wymaga to, aby dokumenty były indeksowane przez w miarę niewielkie zestawy wysoce skorelowanych słów kluczowych. Budziło to wątpliwości, w jaki sposób model zachowa się w sytuacji indeksowania pełnotekstowego, gdzie w skład reprezentacji dokumentu wchodzą wszystkie słowa z zawartego w nim tekstu.

Tym niemniej pomysły Mozera były bardzo inspirujące, a potencjał który się za nimi krył zachęcał do dalszych badań we wskazanych przez niego kierunkach. Już w kilka lat później, J. Bein i P. Smolensky [Crestani, 1997] kontynuowali prace nad zaproponowanym przez Mozera indukcyjnym podejściem do wyszukiwania informacji. Przeprowadzili oni bardziej rygorystyczne badania modelu Mozera, na większych kolekcjach, zajmując się przede wszystkim jego asocjacyjnymi charakterystykami. Aczkolwiek otrzymane przez nich wyniki nie mogą być tak bezpośrednio porównywane z innymi pracami, ponieważ nie używali oni standardowych kolekcji dokumentów, tym niemniej ich rezultaty wspierają twierdzenie, że model Mozera jest w stanie poprawnie działać także dla większych aplikacji.

Kolejnym systemem wyszukiwawczym, opartym na rozprzestrzenianiu aktywacji, był system AIR R.K. Belewa [Belew, 1987, 1989, 2000].

Sieć neuronowa w systemie AIR złożona jest z trzech warstw: warstwa termów, warstwa dokumentów i warstwa autorów (rysunek 2.2.4). Połączenia między neuronami w sieci istnieją jedynie w przypadku występowania związku między reprezentowanymi przez nie obiektami. Na

przykład między termem i indeksowanym przez niego dokumentem, dokumentem i jego autorem itp. Zwróćmy uwagę, że na rysunku 2.2.4 zawsze widzimy parę połączeń między dwiema jednostkami, ponieważ model AIR wykorzystuje połączenia niesymetryczne.



Rys. 2.2.4. Ogólny schemat struktury sieci systemu AIR

Źródło: opracowanie własne na podstawie [Belew, 1989].

Wagi pomiędzy poszczególnymi warstwami, ustawiane są początkowo na wartości wynikające z odwrotnej częstości, tzn. na przykład jeśli neuron dokumentu połączony jest z m neuronami słów kluczowych, to wagi wszystkich połączeń wyjściowych z tego neuronu będą równe $1/m$. Powody wprowadzenia takiego rozwiązania są dwojakie. Po pierwsze powyższy system ustalania wartości wag pozwala kontrolować rozprzestrzenianie aktywacji. Jeśli któryś z neuronów ma bardzo dużo wartości wyjściowych, wartości związanych z nimi wag są bardzo niskie. Drugim powodem takiego systemu ważenia połączeń jest kwestia zbieżności procesu iteracji sieci. Jeśli suma wartości wag jest równa 1, sieć ma właściwość zachowywania aktywacji. Całkowita aktywność systemu ani nie wzrasta, ani nie maleje [Belew, 1989].

Proces wykonania zapytania w systemie AIR ma charakter interaktywny. Użytkownik wykonuje zapytanie, korzystając z prostego języka. W zapytaniu można wykorzystać wszystkie trzy typy obiektów, czyli słowa kluczowe, autorów dokumentów i same dokumenty. Typowe zapytanie może wyglądać następująco [Belew, 1989]:

((:TERM „ASOCJACYJNE”) (:AUTH „ANDERSON”))

Aktywowane są neurony sieci reprezentujące wszystkie wykorzystane w zapytaniu obiekty. Następnie aktywacja rozprzestrzenia się w pierwszym kroku na węzły z nimi sąsiadujące, w drugim na sąsiednie do sąsiadujących itd. Jednocześnie cały proces relaksacji sieci prezentowany jest użytkownikowi. Sieć, jaką widzimy na rysunku 2.2.4 może być przykładem struktury powstałej po osiągnieciu pewnego stanu końcowego iteracji. Cała struktura rozrasta się stopniowo i połączenia na ekranie dodawane są w miarę rozchodzenia się przez nie sygnału w sieci. Pozwala to użytkownikowi obserwować strukturę powstających asocjacji, lepiej ją rozumieć i wykorzystać tę wiedzę w dalszej konsultacji z systemem.

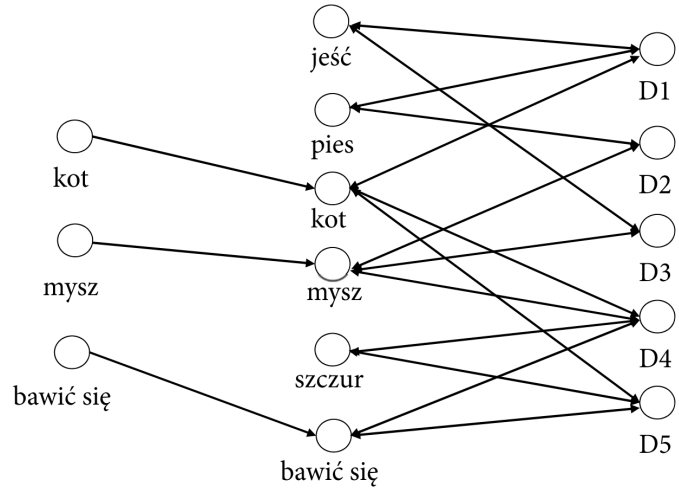
Po zakończeniu relaksacji sieci, proces wyszukiwania się nie kończy. Użytkownik może wskazać na diagramie sieci jednostki reprezentujące elementy (słowa kluczowe, dokumenty, autorów) które według niego są „Bardzo relewantne”, „Relewantne”, Nierelewantne”, „Bardzo nierelewantne” dla jego zapytania. Informacje od użytkownika, wykorzystywane są w kolejnym kroku procesu wyszukiwania, do modyfikacji zapytania użytkownika i ponownego jego wykonania. Taką procedurę interaktywnego wyszukiwania informacji, wykorzystującą do zmiany zapytania informację (pozyskaną od użytkownika) o efektach wyszukiwania w poprzednim kroku, nazywamy zastosowaniem informacji zwrotnej o relewancji, albo sprzężeniem relewancji (ang. *relevance feedback*). Będziemy o niej mówili więcej w podrozdziale 2.5.

W systemie AIR pozyskana informacja na temat relewancji wykorzystywana jest do dynamicznej zmiany wag systemu, czyli do uczenia sieci potrzeby informacyjnej użytkownika związanej z danym zapytaniem. Procesowi temu podlegają zmiany wag wejściowych neuronów wskazanych przez użytkownika z sygnałem o ich relewancji.

Jeśli w_{ij} jest wagą łączącą pewien neuron i (o stanie aktywacji a_i) ze wskazanym przez użytkownika neuronem j (o ustalonym przez użytkownika stanie r_j), to w_{ij} ustawiamy na wartość odpowiadającą korelacji wartości a_i oraz r_j:

$$w_{ij} \propto corr(i,j) = \frac{\mu_{a_i \cdot r_j} - \mu_{a_i} \cdot \mu_{r_j}}{\sigma_{a_i} \cdot \sigma_{r_j}} = \\ = \frac{\sum (a_i \cdot r_j) - \frac{\sum a_i \sum r_j}{N}}{\sqrt{\sum a_i^2 - \frac{\left(\sum a_i\right)^2}{N}} \sqrt{\sum r_j^2 - \frac{\left(\sum r_j\right)^2}{N}}} \qquad (2.2.29)$$

Belew interpretuje wagi systemu jako prawdopodobieństwa warunkowe, że węzeł j będzie relewantny, pod warunkiem, że węzeł i jest relewantny. Zmiana wartości wag powoduje ponowną relaksację sieci, po ustaleniu nowego stanu równowagi użytkownik może przekazać nową informację o relewancji, powodując przeuczenie wag systemu i kolejny krok w sesji wyszukiwawczej itd.



Rys. 2.2.5. Ogólny schemat struktury sieci Wilkinsona i Hingstona

Źródło: opracowanie własne na podstawie [Wilkinson, Hingston, 1991].

Kolejnym przykładem zastosowania neuronowej sieci asocjacyjnej do wyszukiwania informacji jest sieć Wilkinsona i Hingstona [Wilkinson, Hingston, 1991]. Struktura połączeń przedstawiona została na rysunku 2.2.5. Sieć składa się z warstwy termów indeksujących i dokumentów, uzupełnionych o warstwę wejściową termów zapytania. Jak widzimy, jej struktura jest prostsza niż sieci Mozera – brak jest połączeń między neuronami dokumentów. Pamiętajmy jednak, że u Mozera wykorzystywane one były do kontrolowania procesu relaksacji sieci i rozpływu aktywacji. Wilkinson i Hingston rozwiązali ten problem nieco inaczej. Brak jest również warstwy węzłów reprezentujących autorów dokumentów, jak w systemie AIR Belewa – prezentowany obecnie model w reprezentacji dokumentów opiera się wyłącznie na słowach kluczowych.

Model Wilkinsona i Hingstona jest spośród omawianych zdecydowanie najbardziej rozwinięty pod względem funkcjonalnym. Wagi połączeń między termami i dokumentami odzwierciedlają siłę tego powiązania,

zaś cały system został tak zaprojektowany, by w pełni odpowiadać modelowi wektorowemu wyszukiwania informacji, dodając do niego elementy asocjacyjne.

Połączenia między dokumentami i termami w sieci mają charakter symetryczny i wykorzystywane są do przepływów aktywacji w obie strony. Waga w_{ij}, łącząca węzeł i-tego dokumentu i j-tego termu, wyznaczana jest w procesie uczenia nienadzorowanego, na podstawie statystyk z kolekcji:

$$w_{ij} = \frac{d_{ij}}{\sqrt{\sum_{k=1}^{n} d_{ik}^2}} \qquad d_{ij} = tf_{ij} \cdot \log\left(N / f_j\right) \tag{2.2.30}$$

gdzie tf_{ij} jest częstością termu w dokumencie, f_j – liczbą dokumentów zawierającą dany term, N – liczbą wszystkich dokumentów w kolekcji, n – liczbą termów.

Jak widzimy, waga wyznaczana jest zgodnie z regułami modelu wektorowego, przy czym od razu normalizowana jest długością opisu dokumentu. Przypomnijmy, że powyższe wagi stosowane są zarówno przy przepływach sygnału z warstwy termów do dokumentów, jak i w odwrotnym kierunku, z dokumentów do termów.

Stan neuronów warstwy termów zapytania ustalany jest zgodnie z regułą:

$$q_j = \begin{cases} \log N / f_j & \text{jeśli term } j \text{ pojawia się w zapytaniu} \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{2.2.31}$$

Wagi między warstwą zapytania i termów normalizują zapytanie jego długością:

$$wq_j = \frac{1}{\sqrt{\sum_{k=1}^{n} q_k^2}} \tag{2.2.32}$$

Stany neuronów wyznaczane są na podstawie łącznego pobudzenia wejściowego, jako suma wejść mnożonych przez wagi. W pierwszym kroku działania sieci, ponieważ stany neuronów dokumentów są równe 0, jednostki termów otrzymają aktywację wyłącznie od węzłów wejściowych:

$$t_j = q_j \cdot wq_j = \frac{q_j}{\sqrt{\sum_{k=1}^{n} q_k^2}} \tag{2.2.33}$$

Następnie pobudzane są neurony dokumentów, stan *i*-tego dokumentu, a_i, będzie więc wynosił:

$$a_i = \sum_{j=1}^{n} t_j w_{ij} = \frac{\sum_{j=1}^{n} q_j d_{ij}}{\sqrt{\sum_{k=1}^{n} q_k^2} \sqrt{\sum_{k=1}^{n} d_{ik}^2}} \tag{2.2.34}$$

Łatwo zauważyć, że stan neuronu dokumentu jest po prostu równy znanemu nam z modelu wektorowego podobieństwu wektora reprezentacji dokumentu i wektora zapytania. Jak więc widzimy, sieć neuronowa Wilkinsona i Hingstona rzeczywiście realizuje wyszukiwanie zgodnie z regułą działania tego modelu. Jednak działanie sieci w tym miejscu się nie kończy. W kolejnej iteracji, sygnał przepływa z warstwy dokumentów i wejściowej do neuronów termów.

Początkowo Wilkinson i Hingston zdefiniowali pobudzenie każdej jednostki termu t_j jako określane na podstawie łącznego pobudzenia neuronów dokumentów oraz neuronu zapytania:

$$t_j = q_j \cdot wq_j + \sum_{i=1}^{n} a_i w_{ij} = q_j \cdot wq_j + \frac{\sum_{i=1}^{n} a_i d_{ij}}{\sqrt{\sum_{k=1}^{n} d_{ik}^2}} \tag{2.2.35}$$

Okazało się jednak w praktycznych eksperymentach, że taka definicja aktywacji neuronów termów powoduje zbyt szeroką aktywację sieci i stopniowy rozpływ sygnału na wszystkie termy. Dlatego zdecydowano się na wprowadzenie progów aktywacji neuronów dokumentów, i zdefiniowanie funkcji przetwarzania neuronu termu w następujący sposób:

$$t_j = q_j \cdot wq_j + \alpha \cdot \frac{\sum_{a_i > T} a_i d_{ij}}{\sqrt{\sum_{a_i > T} d_{ik}^2}} + \beta \cdot \frac{\sum_{a_i < -T} a_i d_{ij}}{\sqrt{\sum_{a_i < -T} d_{ik}^2}} \tag{2.2.36}$$

gdzie T jest wartością progu z przedziału [0,1], α, β są parametrami dobieranymi dla konkretnego przypadku. Wilkinson i Hingston dobrali w swoich eksperymentach wartości tych stałych jako: $T = 0{,}2$, $\alpha = 0{,}25$, $\beta = 0{,}05$. Przy takich wartościach parametrów sieć stabilizowała się bardzo szybko, praktycznie po dwóch iteracjach.

W bieżącym punkcie przedstawiliśmy kilka architektur asocjacyjnych sieci neuronowych, wykorzystywanych do skojarzeniowego

wyszukiwania informacji. Tego rodzaju rozwiązania okazały się pomocne w polepszeniu jakości wyszukiwania, zwłaszcza w osiągnięciu lepszego dopasowania słów kluczowych, użytych w zapytaniu i występujących w opisach dokumentu.

Nie sposób jednak nie zauważyć, że są to duże systemy, wymagające znacznej pracy przy ich tworzeniu oraz dostrajaniu. Pamiętać również należy, że omawiane propozycje sieci asocjacyjnych miały wbudowaną w swoją strukturę całą kolekcję dokumentów – w formie warstwy neuronów reprezentujących przechowywane teksty. W związku z tym model budowany musi być zawsze dla konkretnej kolekcji i trudno przenieść jakiekolwiek związane z nim doświadczenia na inne aplikacje.

Dlatego, zwłaszcza w przypadku sieci asocjacyjnych, bardziej popularnym podejściem jest stworzenie modelu sieciowego wiedzy dla zależności między termami, w formie tezaurusa asocjacyjnego, który można wykorzystać w procesie pracy nad zapytaniem, a nie bezpośrednio w mechanizmie dopasowania i rankingowania dokumentów. Do tego rodzaju zastosowań sieci asocjacyjnych, będziemy jeszcze wracać w rozdziale 2.5.

2.2.2.4. Wyszukiwanie skojarzeniowe z wykorzystaniem sieci wnioskujących

Siecią bayesowską określamy skierowany, acykliczny graf, w którym węzły reprezentują pewne zdarzenia, zmienne losowe, zaś łączące je krawędzie, warunkowe probabilistyczne zależności przyczynowe między nimi [Cichosz, 2000]. Tak więc przepływ prawdopodobieństwa w sieci jest jednokierunkowy, a ponadto nie może on układać się w zależności cykliczne, tzn. nie może zdarzyć się taka sytuacja, że wyjście któregoś z węzłów bezpośrednio (lub pośrednio, za pośrednictwem innych węzłów) łączy się z jego wejściem.

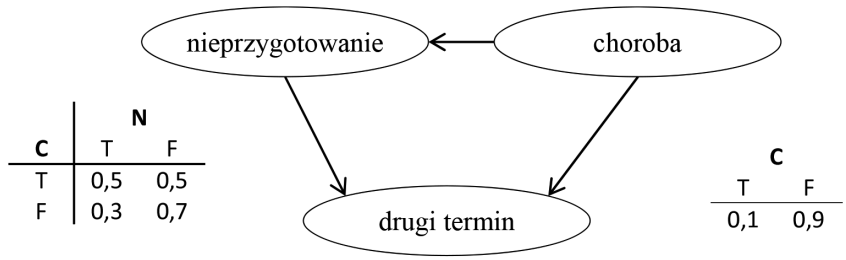
Jeśli pewne zdarzenie p powoduje, implikuje lub w inny sposób wpływa na inne zdarzenie q, to sieć zawiera skierowane połączenie, prowadzące z węzła p do węzła q. Węzły niepołączone w sieci są niezależne od siebie. Tak więc w sieci bayesowskiej dane zdarzenie zależne jest wyłącznie od swoich rodziców w grafie połączeń. Jeśli sieć bayesowska opisuje n niezależnych zdarzeń $q_1, \ldots, q_n$, to definiuje ją łączny rozkład prawdopodobieństwa tych zdarzeń:

$$P(q_1,\ldots,q_n)=\prod_{i=1}^{n}P(q_i \mid \pi(q_i)) \qquad (2.2.37)$$

gdzie $\pi(q_i)$ jest zbiorem rodziców zdarzenia q_i. Jeśli dane zdarzenie q_i nie ma rodziców, to $\pi(q_i)$ jest pusty i stąd $P(q_i \mid \pi(q_i)) = P(q_i)$.

Każdy węzeł q przechowuje macierz połączeń ze swymi węzłami rodzicielskimi $\pi(q)$, określającą prawdopodobieństwa warunkowe $P(q \mid \pi(q))$, dla każdego zestawu wartości zmiennych q oraz $p_j \in \pi(q)$. Mając zbiór prawdopodobieństw *a priori* dla węzłów nieposiadających rodziców (początkowych w grafie), możemy obliczyć prawdopodobieństwo, albo stopień przekonania (ang. *belief*) dla wszystkich pozostały węzłów.

Zilustrujmy ideę działania sieci bayesowskich na prostym przykładzie. Przypuśćmy, że badamy zagadnienie przyczyn konieczności niezdawania przez studentów pewnego egzaminu. Zidentyfikowaliśmy dwa podstawowe powody, które mogą spowodować konieczność podchodzenia do poprawki (zmienna *D* – drugi termin). Są to nieprzygotowanie studentów (zmienna *N*) oraz niedyspozycja zdrowotna (zmienna *C* – choroba). Przy czym nie są to czynniki niezależne. Choroba może również spowodować brak przygotowania. Struktura sieci bayesowskiej, ilustrującej powyższą sytuację, przedstawiona została na rysunku 2.2.6. Jak widzimy krawędzie grafu sieci łączą zarówno węzły nieprzygotowanie" i „choroba" z węzłem „drugi termin", ale również mamy połączenie prowadzące z węzła „choroba" do węzła „nieprzygotowanie".



	N	
C	T	F
T	0,5	0,5
F	0,3	0,7

C	
T	F
0,1	0,9

		D	
N	**C**	T	F
T	T	1,0	0,0
T	F	0,9	0,1
F	T	0,5	0,5
F	F	0,2	0,8

Rys. 2.2.6. Prosta sieć bayesowska dla zagadnienia poprawki z egzaminu

Źródło: opracowanie własne.

Zakładamy, że wszystkie trzy zmienne mają charakter binarny, czyli przyjmują wartości prawda (T) lub fałsz (F). W każdym węźle zdefiniowana została macierz prawdopodobieństw warunkowych: dla wszystkich

układów wartości węzłów rodzicielskich oraz każdej wartości zmiennej reprezentowanej przed dany węzeł, zostało określone prawdopodobieństwo warunkowe takiej sytuacji.

Na tym właśnie polega siła działania tego rodzaju rozwiązań, jak sieci bayesowskie. Można przy ich pomocy, we w miarę nieskomplikowany sposób, opisywać nawet dosyć złożone sytuacje, z dużą liczbą zależnych od siebie czynników, zmiennych, zdarzeń. Ponadto zależności opisywane są w formie lokalnej, związanej z konkretnym połączeniem. Ma to duże znaczenie techniczne, przy większej liczbie czynników globalny opis łącznego prawdopodobieństwa, wymagałby wzięcia pod uwagę znacznie większej liczby możliwych stanów systemu niż po rozkładzie zagadnienia na zestaw prostszych podzależności. Ważniejszą jednak kwestią jest fakt, że w sieci bayesowskiej modelujący może skupić się na każdej z lokalnych zależności z osobna, w oderwaniu od innych, co znacznie ułatwia budowę modelu, a często nawet stanowi jedyną możliwość jego stworzenia – globalnie byłby zbyt skomplikowany.

Korzystając z sieci bayesowskiej możemy wyznaczać interesujące nas prawdopodobieństwa, rozmaitych zdarzeń – prowadzić proces wnioskowania. Wnioskowanie na temat zależności zgodnych z kierunkiem przyczynowości koncepcyjnie jest dosyć proste. Dla dowolnego układu wartości naszych trzech zmiennych *D*, *N*, *C* obliczenie wspólnego prawdopodobieństwa jest proste i wymaga pomnożenia trzech liczb z tabel rozkładów warunkowych:

$$P(D, N, C) = P(D \mid N, C)\, P(N \mid C)\, P(C) \tag{2.2.38}$$

Wszystkie tego rodzaju zdarzenia wymagają w najprostszym podejściu wyliczenia wspólnych prawdopodobieństw dla odpowiednich, związanych z danym zdarzeniem, układów „prawd" i „fałszów" – wartości tych trzech zmiennych i posumowania ich. Oczywiście, przy większych sieciach takie proste podejście mogłoby być dosyć naiwne obliczeniowo.

Sieci bayesowskie pozwalają jednak również na wnioskowanie w drugą stronę, o przyczynach pewnych zachodzących zdarzeń. Na przykład, jeśli wiemy, że student nie zdał egzaminu i będzie potrzebował drugiego terminu, to jakie jest prawdopodobieństwo, że przyczyną tego była choroba? Policzmy:

$$P(C=T|D=T)=\frac{P(D=T,C=T)}{P(D=T)}=$$
$$=\frac{P(D=T,N=T,C=T)+P(D=T,N=F,C=T)}{\sum_{C,N\in\{T,N\}}P(D=T,C,N)}$$

Poszczególne prawdopodobieństwa występujące w powyższej zależności, można obliczyć, korzystając z ich rozwinięcia na prawdopodobieństwa warunkowe (2.2.38). Wystarczy w tym celu pomnożyć prawdopodobieństwa z tablic. Na przykład dla pierwszego z wyrazów w liczniku, $P(D = \mathrm{T}, N = \mathrm{T}, C = \mathrm{T})$, musimy wziąć wartości z kolumny T, z pierwszego wiersza tabeli dla węzła „drugi termin", pierwszego wiersza tabeli węzła „nieprzygotowanie", oraz pierwszego wiersza tabeli „choroba":

$$P(D = \mathrm{T}, N = \mathrm{T}, C = \mathrm{T}) = P(D = \mathrm{T} \mid N = \mathrm{T}, C = \mathrm{T})\, P(N = \mathrm{T} \mid C = \mathrm{T})\, P(C = \mathrm{T}) = 1{,}0 \cdot 0{,}5 \cdot 0{,}1 = 0{,}05$$

Podobnie dla drugiego składnika w liczniku, $P(D = \mathrm{T}, N = \mathrm{F}, C = \mathrm{T})$ – z trzeciego wiersza tabeli węzła „drugi termin", kolumny F dla pierwszego wiersza tabeli węzła „nieprzygotowanie", oraz pierwszego wiersza tabeli „choroba":

$$P(D = \mathrm{T}, N = \mathrm{F}, C = \mathrm{T}) = P(D = \mathrm{T} \mid N = \mathrm{F}, C = \mathrm{T})\, P(N = \mathrm{F} \mid C = \mathrm{T})\, P(C = \mathrm{T}) = 0{,}5 \cdot 0{,}5 \cdot 0{,}1 = 0{,}025$$

Analogicznie dla wyrazu w mianowniku mamy:

$$P(D = \mathrm{T}, N = \mathrm{T}, C = \mathrm{T}) + P(D = \mathrm{T}, N = \mathrm{F}, C = \mathrm{T}) + P(D = \mathrm{T}, N = \mathrm{T}, C = \mathrm{F}) + P(D = \mathrm{T}, N = \mathrm{F}, C = \mathrm{F}) = 1{,}0 \cdot 0{,}5 \cdot 0{,}1 + 0{,}5 \cdot 0{,}5 \cdot 0{,}1 + 0{,}9 \cdot 0{,}3 \cdot 0{,}9 + 0{,}2 \cdot 0{,}7 \cdot 0{,}9 = 0{,}05 + 0{,}025 + 0{,}243 + 0{,}126 = 0{,}444$$

Ostatecznie więc, prawdopodobieństwo że powodem drugiego terminu jest choroba studenta wynosi:

$$P(C = \mathrm{T} \mid D = \mathrm{T}) = (0{,}05 + 0{,}025) / 0{,}444 = 0{,}167$$

Jak więc widzimy, koncepcyjnie nie ma żadnego problemu z prowadzeniem wnioskowania w sieci bayesowskiej. Obliczenia polegają właściwie jedynie na przeszukiwaniu tabel prawdopodobieństwa warunkowego w węzłach. Pojawiają się niestety kłopoty obliczeniowe. Pomimo tego, że nasza przykładowa sieć jest bardzo prosta, ustalenie prawdopodobieństwa wymagało sporej ilości obliczeń. Generalnie, dla dużych sieci, wnioskowanie dokładne w sieci Bayesowskiej jest problemem NP-trudnym. Istnieją jednak algorytmy tzw. wnioskowania przybliżonego, które są efektywne obliczeniowo.

Dalsze szczegóły dotyczące projektowania sieci Bayesowskich i wnioskowania przy ich wykorzystaniu, znaleźć można w podręcznikach

poświęconych tej tematyce. Literatura przedmiotu jest dosyć bogata. Klasycznym podręcznikiem dotyczącym sieci bayesowskich jest książka Pearla [Pearl, 1988], ale można skorzystać z nowszych opracowań, takich jak [Cowell, Dawid i in., 1999] czy [Koski, Noble, 2009]. Omówienie tematyki sieci bayesowskich z punktu widzenia różnych znanych narzędzi analizy decyzji w warunkach ryzyka, znaleźć można w [Kjærulff, Madsen, 2013] (diagramy wpływu), czy też [Jensen, Nielsen, 2007] (drzewa decyzyjne). Ponadto dobry przegląd metod uczenia sieci bayesowskich na podstawie danych, znaleźć można w [Neapolitan, 2003], w literaturze polskojęzycznej [Cichosz, 2000].

Sieci bayesowskie, jak widzimy, stanowią więc dosyć naturalną metodę implementacji asocjacyjnego modelu wyszukiwania informacji. Obiekty systemu wyszukiwawczego, takie jak dokumenty, termy, pojęcia itp. mogą być odwzorowane w struktury sieci w formie węzłów jej grafu, z połączeniami odpowiadającymi zależnościom między nimi, z siłą tych związków mierzoną w formie prawdopodobieństwa. Model opiera się przy tym na dosyć rygorystycznym gruncie formalizmu rachunku prawdopodobieństwa, a cały system może być widziany jako rozszerzenie omawianych wcześniej rozwiązań probabilistycznych, z uwzględnieniem tak kłopotliwych współzależności między modelowanymi elementami.

Nic więc dziwnego, że sieci bayesowskie stanowią chyba najbardziej udane podejście do rozszerzenia modeli wyszukiwawczych o mechanizmy asocjacyjnego dopasowania zapytań i dokumentów, z zastosowaniem sieci pojęciowych. Istnieje szereg konkretnych, udanych rozwiązań w tej dziedzinie, oraz praktycznych systemów wyszukiwawczych, które na nich się opierają. Najważniejsze z nich są trzy podejścia, które omówimy w dalszej części bieżącego punktu:

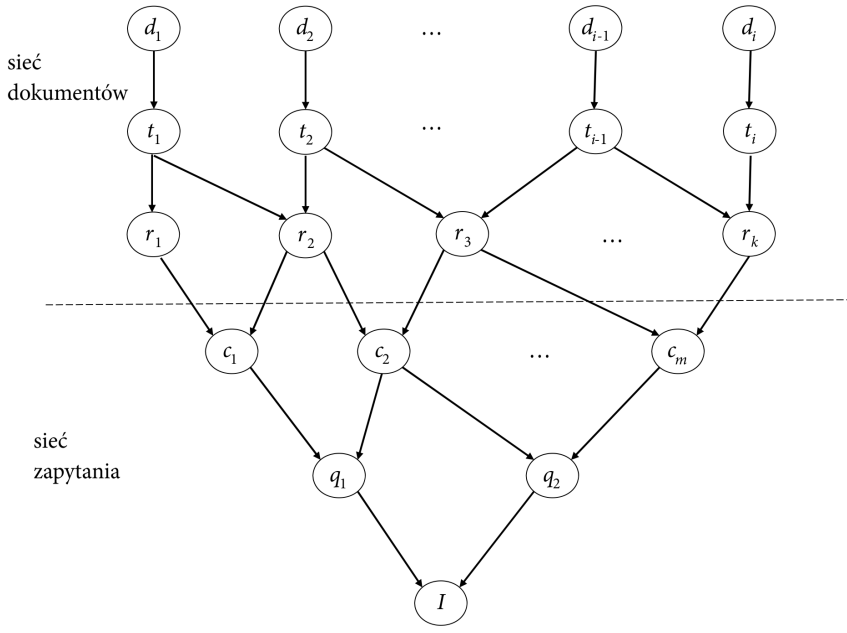
- model sieci wnioskującej (ang. *inference network*),
- model sieci propagacji przekonania (ang. *belief network*),
- model sieci bayesowskiej.

Model sieci wnioskującej stworzony został przez Turtle'a i Crofta [Turtle, Croft, 1990; Turtle, 1991; Turtle, Croft, 1991]. System wyszukiwawczy zbudowany jest z dwu sieci: sieci dokumentów i sieci zapytania. Sieć dokumentów zbudowana jest dla całej kolekcji i reprezentuje informacje na jej temat. Jej struktura jest stała, tj. nie ulega zmianie w trakcie przetwarzania zapytania. Sieć zapytania składa się z jednego węzła reprezentującego potrzebę informacyjną użytkownika, i jednego lub więcej wyrażających tę potrzebę. Sieć zapytania jest tworzona dla danej potrzeby informacyjnej, oraz jest modyfikowana w miarę jak zapytanie jest doprecyzowywane. Sieć dokumentów i zapytania złączone są linkami między pojęciami

reprezentującymi dokumenty i zapytanie. Wszystkie węzły w sieci wnioskującej mogą przyjmować wartości binarne prawda lub fałsz. Schemat ogólny modelu sieci wnioskującej Turtle'a i Crofta przedstawiony został na rysunku 2.2.7.

Sieć dokumentów składa się z węzłów dokumentów d_i, węzłów reprezentacji tekstowej dokumentów t_j, oraz węzłów pojęć r_k. Każdy węzeł dokumentu reprezentuje dokument w kolekcji i odpowiada wydarzeniu, że ten konkretny dokument został zaobserwowany.

Węzły tekstów odpowiadają fizycznym fragmentom tekstu, składającym się na dany dokument. Generalnie węzeł dokumentu może łączyć się z kilkoma węzłami tekstowymi, np. reprezentującymi odrębne fragmenty dokumentu (tytuł, rozdziały, bibliografia itp.). Ponadto różne dokumenty mogą mieć połączenia do tego samego węzła tekstu, np. jeśli pewne fragmenty tekstu powtarzają się w kilku dokumentach. W oryginalnych pracach, Turtle i Croft opisywali reprezentację na poziomie całego dokumentu – czyli powiązanie jeden-do-jednego między węzłem dokumentu i węzłem jego tekstu, tak więc w dalszej części niniejszego podrozdziału węzły t_j będziemy dla uproszczenia pomijać. Jednak ich wprowadzenie jest zawsze możliwe i stanowią one potencjalną możliwość rozwojową modelu.



Rys. 2.2.7. Ogólny schemat podstawowego modelu sieci wnioskującej

Źródło: opracowanie własne na podstawie [Turtle, Croft, 1991].

określony jest w formie połączenia prowadzącego od węzła dokumentu do węzła pojęcia i ma charakter binarny, tzn. definicji sieci, połączenie więc jest, albo go nie ma. Sama struktura grafu połączeń nie uwzględnia częściowych powiązań term/dokument, ważonych powiązań itp.

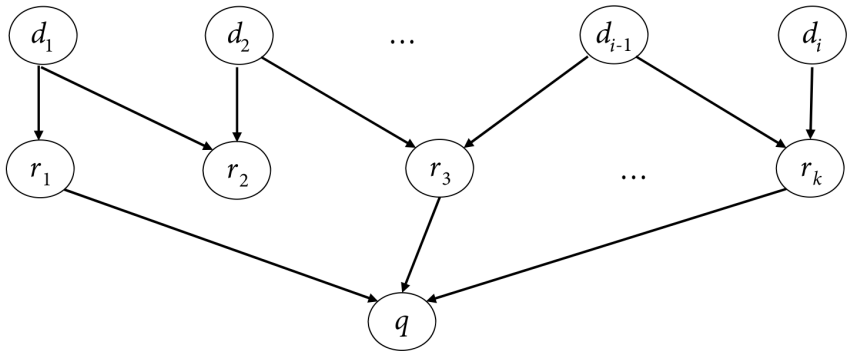
Z każdym dokumentem związane jest prawdopodobieństwo *a priori*, określające prawdopodobieństwo jego obserwacji, generalnie wynosi ono 1/*rozmiar kolekcji*. Każdy węzeł reprezentacji zawiera specyfikację prawdopodobieństwa warunkowego występowania reprezentowanego przez niego pojęcia, pod warunkiem zbioru wszystkich rodzicielskich dokumentów. Prawdopodobieństwa te włączają w siebie efekty dowolnych wag indeksujących (np. częstości termu w każdym z dokumentów rodzicielskich), lub wag termu (np. odwrotność częstości dokumentu) związanych z reprezentowanym pojęciem.

Sieć zapytania tworzona jest dla konkretnej sesji wyszukiwawczej i służy do zdefiniowania potrzeby informacyjnej użytkownika. W żaden sposób nie oddziałuje ona, ani nie zmienia sieci dokumentów, jest z nią tylko połączona. Sieć zapytania składa się z węzłów pojęć reprezentujących zapytania c_m, węzłów reprezentujących zapytania q_k oraz jednego węzła I, reprezentującego potrzebę informacyjną użytkownika.

Węzły pojęć stanowią węzły początkowe grafu sieci zapytania; odpowiadają one podstawowym pojęciom użytym do wyrażenia potrzeby informacyjnej, ich rodzicami w sieci są odpowiadające im pojęcia termów indeksujących dokumenty. W niektórych sytuacjach może tak się zdarzyć, że jedno pojęcie zapytania odpowiadać będzie kilku pojęciom reprezentacji dokumentów. Na przykład jeśli język zapytań jest inny niż język reprezentacji dokumentów. Węzły pojęć zapytania będą wtedy definiować odwzorowanie między słowami kluczowymi wykorzystanymi do reprezentacji dokumentów oraz zapytania. W typowym przypadku jednak, język termów indeksujących i zapytań jest taki sam, i węzły pojęć w sieci dokumentów i zapytań powiązane są połączeniami jeden-do-jednego. W takim przypadku warstwę pojęć zapytania, zazwyczaj po prostu się pomija, łącząc bezpośrednio węzły termów indeksujących z węzłami samych zapytań.

Węzły zapytań definiują same zapytania, reprezentowane przez zestawy pojęć, których węzły są ich rodzicami w sieci. W sieci może występować kilka węzłów zapytań, w przypadku gdy do wyrażenia potrzeby informacyjnej użytkownika, zdefiniowanej w węźle I, potrzebnych jest kilka zapytań. I znów, zazwyczaj mamy do czynienia z prostszą sytuacją, w której występuje tylko jedno zapytanie. Wtedy często rezygnujemy z węzła potrzeby informacyjnej (lub zapytania). Struktura sieci

bayesowskiej w modelu sieci wnioskującej jest więc zazwyczaj prostsza niż ogólna, przedstawiona na rysunku 2.2.7 (patrz rysunek 2.2.8).



Rys. 2.2.8. Typowa struktura sieci bayesowskiej w modelu sieci wnioskującej

Źródło: opracowanie własne na podstawie [Turtle, 1991].

Przy pomocy sieci bayesowskiej, znając wartości prawdopodobieństw *a priori*, związanych z dokumentami i prawdopodobieństw warunkowych dla węzłów wewnętrznych, możemy obliczać prawdopodobieństwa *a posteriori* związane ze zdarzeniami dotyczącymi zależności w kolekcji dokumentów. Sieć, jako całość, reprezentuje zależności w naszym przekonaniu, że potrzeba informacyjna użytkownika odzwierciedlona jest przez dokumenty kolekcji, przy czym ta zależność odbywa się za pośrednictwem reprezentacji dokumentu i zapytania. Wstępna wartość w węźle reprezentującym potrzebę informacyjną (lub reprezentujące ją zapytanie) równa jest prawdopodobieństwu, że potrzeba informacyjna została zaspokojona, pod warunkiem, że nie został zaobserwowany żaden określony dokument kolekcji i wszystkie dokumenty są jednakowo prawdopodobne (lub nieprawdopodobne).

Jeśli zaobserwujemy jeden określony dokument d_i i przekażemy tę informację sieci, przypisując d_i = prawda, to możemy obliczyć nowe prawdopodobieństwa dla każdego węzła w sieci, pod warunkiem że d_i = prawda. W szczególności, możemy obliczyć prawdopodobieństwo, że potrzeba informacyjna została spełniona, pod warunkiem, że zaobserwowany został dokument d_i. Powtarzając ten proces, możemy obliczyć prawdopodobieństwo, że potrzeba informacyjna została zaspokojona przez każdy z dokumentów i na tej podstawie zbudować ich ranking.

Aby to było możliwe, niezbędne jest oczywiście obliczenie dla wszystkich węzłów sieci, poza jej węzłami-korzeniami, czyli węzłami dokumentów, prawdopodobieństw warunkowych dla wszystkich układów wartości

rodziców danego węzła, czyli macierze połączeń. Turtle i Croft [Turtle, Croft, 1991] pokazali metodę tworzenia macierzy połączeń w formie zamkniętych operacji obliczeniowych, pozwalających przy pomocy sieci bayesowskiej realizować operacje wyszukiwania, różnego typu.

Rozpoczniemy od (jak to nazwali autorzy) kanonicznych form macierzy połączeń dla operatorów logicznych. Jako ilustrację, przyjmijmy za [Turtle, Croft, 1991], że węzeł zapytania Q ma trzech rodziców: A, B, C. Oznaczmy dalej prawdopodobieństwa $P(A = prawda) = a$, $P(B = prawda) = b$, $P(C = prawda) = c$.

Rozważmy złożenie w formie alternatywy, operacji „OR", tych węzłów. Alternatywa jest fałszywa tylko, gdy wszystkie jej składniki są fałszywe, zaś w pozostałych przypadkach jej wynikiem jest prawda. To sugeruje następującą macierz połączeń [Turtle, Croft, 1991]:

$$L_{OR} = \begin{pmatrix} 1 & 0 & 0\,0 & 0 & 0\,0 & 0 \\ 0 & 1 & 1\,1 & 1 & 1\,1 & 1 \end{pmatrix} \tag{2.2.39}$$

W macierzy L_{OR} pierwszy wiersz odpowiada przypadkowi $Q = fałsz$, drugi $Q = prawda$. Każda kolumna odpowiada określonej kombinacji wartości rodziców – ich liczba równa jest więc $2^3 = 8$). Ponumerujmy je od 0 do 7, dzięki czemu łatwiej nam będzie zapamiętać ich semantykę, przedstawiając numer kolumny jako liczbę dwójkową. I tak, kolumna $0 = 000_2$ odpowiada sytuacji, gdy wszystkie węzły rodziców, A, B, C, są fałszywe. W takim wypadku prawdopodobieństwo, że Q będzie fałszywe jest równe 1, a że prawdziwe równe 0. Jak widzimy, właśnie te wartości znalazły się w kolumnie 0 macierzy L_{OR}. Kolejna kolumna, o numerze $1 = 001_2$, odpowiada przypadkowi $A = fałsz$, $B = fałsz$, $C = prawda$. Prawdopodobieństwo $P(Q = \text{fałsz})$ w tej sytuacji jest równe 0, zaś $P(Q = \text{prawda})$ jest równe 1. I tak dalej, aż do ostatniej, siódmej kolumny, $7 = 111_2$.

Dla dużych sieci, z powodu znacznych rozmiarów pamięci niezbędnej do przechowywania wszystkich macierzy połączeń, wygodniejsze może być zastąpienie ich otwartej definicji w formie tablicy L_{OR}, funkcją obliczającą prawdopodobieństwo warunkowe wartości w węźle potomnym, dla danego zestawu wartości węzłów rodzicielskich.

Aby obliczyć formułę wyznaczającą prawdopodobieństwo w formie zamkniętego wyrażenia, przypomnijmy sobie to, co mówiliśmy o optymalizacji wyrażeń w zapytaniach i funkcjach algebry Boole'a, w punkcie 2.1.1. Tutaj postąpimy bardzo podobnie, jak przy tworzeniu kanonicznej postaci alternatywnej wyrażenia, przez analogię między działaniami logicznymi a operacjami algebraicznymi, działającymi na zerach i jedynkach. Jeśli

chcemy policzyć prawdopodobieństwo $P(Q = \text{prawda})$ musimy zsumować wyrażenia odpowiadające wartościom węzłów A, B, C w kolumnach macierzy L_{OR}, które w drugim wierszu (odpowiadającym $P(Q = \text{prawda})$) mają wartość 1.

Pierwszą taką kolumną, jest kolumna o numerze $1 = 001_2$, czyli reprezentująca wartości węzłów rodzicielskich $A = \textit{fałsz}$, $B = \textit{fałsz}$, $C = \textit{prawda}$. Zauważmy, że odpowiadające temu warunkowi wyrażenie $(1 - a)(1 - b)c$ ma wartość 1 tylko dla tej sytuacji. Przy każdych innych wartościach węzłów A, B, C wyrażenie to ma wartość 0.

Drugą kolumną, która spełnia nasz warunek jest kolumna $2 = 010_2$, dla $A = \textit{fałsz}$, $B = \textit{prawda}$, $C = \textit{fałsz}$. I znów, wyrażenie $(1 - a)b(1 - c)$, odpowiadające prawdziwościom węzłów A, B, C dla tej kolumny, ma wartość 1 tylko dla wartości z tej kolumny, dla wszystkich innych ma wartość 0.

Postępując tak dla wszystkich kolejnych kolumn, ostatecznie możemy napisać:

$$P(Q = \textit{prawda}) = (1-a)(1-b)c + (1-a)b(1-c) + \\ + (1-a)bc + a(1-b)(1-c) + a(1-b)c + ab(1-c) + abc$$

Powyższe wyrażenie na pewno jest równe 1 dla wszystkich układów wartości węzłów A, B, C, odpowiadających kolumnom macierzy L_{OR}, które w drugim wierszu mają 1 (czyli kolumny 1–7), natomiast dla pozostałych układów wartości tych węzłów (czyli tylko $A = \textit{fałsz}$, $B = \textit{fałsz}$, $C = \textit{fałsz}$, kolumna 0) ma wartość 0.

Po prostych przeliczeniach otrzymamy dla operacji alternatywy

$$P(Q = \textit{prawda}) = 1 - (1-a)(1-b)(1-c) \tag{2.2.40}$$

Oczywiście podobne obliczenia dla $Q =$ fałsz, dają nam:

$$P(Q = \textit{fałsz}) = (1-a)(1-b)(1-c) \tag{2.2.41}$$

W analogiczny sposób musimy postąpić dla operacji koniunkcji. Macierz połączeń ma wówczas postać:

$$L_{AND} = \begin{pmatrix} 1 & 1 & 1\,1 & 1 & 1\,1 & 0 \\ 0 & 0 & 0\,0 & 0 & 0\,0 & 1 \end{pmatrix} \tag{2.2.42}$$

Co odpowiada sytuacji że koniunkcja jest prawdziwa tylko dla ostatniej kolumny, odpowiadającej układowi wartości $A = \textit{prawda}$, $B = \textit{prawda}$, $C = \textit{prawda}$.

Co równoważne jest formule obliczeniowej dla $P(Q = prawda)$:

$$P\left(Q = prawda\right) = abc \tag{2.2.43}$$

oraz dla $P(Q = fałsz)$:

$$\begin{aligned} P\left(Q = fałsz\right) &= (1-a)(1-b)(1-c) + (1-a)(1-b)c + \\ &+ (1-a)b(1-c) + (1-a)bc + a(1-b)(1-c) + \\ &+ a(1-b)c + ab(1-c) = \\ &= 1 - abc \end{aligned} \tag{2.2.44}$$

Operator negacji zdefiniowany jest tylko dla węzłów Q, posiadających pojedynczego rodzica A. Wówczas $Q = prawda$ wtedy i tylko wtedy, gdy $A = fałsz$. Daje to następującą macierz połączeń:

$$L_{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.2.45}$$

oraz jej metodę obliczania:

$$P\left(Q = prawda\right) = 1 - a \tag{2.2.46}$$

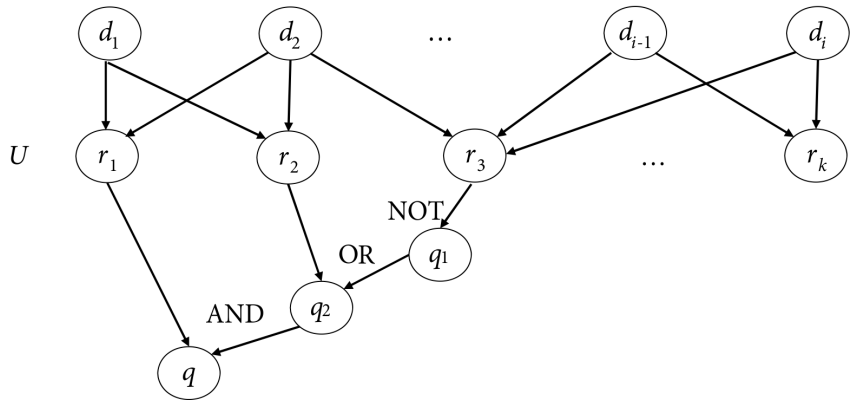
$$P\left(Q = fałsz\right) = a \tag{2.2.47}$$

W ten sposób zdefiniowaliśmy węzły sieci bayesowskiej, wykonujące trzy podstawowe operacje logiczne, za pomocą których możemy zapisać dowolny warunek logiczny. Dla wyszukiwania boolowskiego część sieci definiująca zapytanie ma formę drzewa odpowiadającego zawartemu w nim wyrażeniu logicznemu. Korzeniem drzewa jest finalne zapytanie, a liśćmi słowa kluczowe występujące w zapytaniu. Wszystkie połączenia skierowane są w stronę korzenia. Pośrednie węzły drzewa odpowiadają poszczególnym operatorom logicznym i wykorzystują zdefiniowane powyżej mechanizmy przetwarzania.

Na rysunku 2.2.9 przedstawiona została struktura sieci bayesowskiej, dla przykładowego zapytania $r_1 \wedge (r_2 \vee \neg r_3)$. Jak widać, całe zapytanie, w tym przypadku, reprezentowane jest przez trzy jednostki: węzeł q_1 reprezentuje operację $\neg r_3$, węzeł q_2 reprezentuje koniunkcję r_2 i $\neg r_3$ (jako wyniku działania węzła q_1) oraz węzeł q reprezentuje alternatywę r_1 i wyrażenia w węźle q_2, czyli całość zapytania. Znając prawdopodobieństwa prawdziwości występujących w zapytaniu termów, możemy przy pomocy

tabel połączeń dla poszczególnych operatorów logicznych obliczyć prawdopodobieństwa ich prawdziwości, dochodząc aż do węzła q.

Jeżeli przy realizacji tych operacji logicznych ograniczymy wartości węzłów rodzicielskich do 0 lub 1, to Q musi mieć również wartości binarne. Łatwo jednak możemy rozszerzyć działanie sieci na wyszukiwanie boolowskie, w połączeniu z indeksowaniem wykorzystującym wagi słów kluczowych, po prostu dopuszczając, aby prawdopodobieństwa termów przyjmowały wartości wag z przedziału [0, 1], interpretując je jako prawdopodobieństwo, że dany term został przypisany dokumentowi. Opisane wyżej kanoniczne formy macierzy połączeń, mogą być stosowane bez zmian.



Rys. 2.2.9. Struktura sieci w modelu sieci wnioskowania dla przykładowego zapytania boolowskiego $r_1 \wedge (r_2 \vee \neg r_3)$

Źródło: opracowanie własne na podstawie [Turtle, 1991].

Rozważmy teraz nieco odmienny sposób definiowania prawdopodobieństw warunkowych w węzłach sieci, który pozwoli nam w pewnym stopniu odejść od implementacji wyszukiwania boolowskiego, ale co ważniejsze pozwoli nam zrozumieć inne, bardziej przydatne rozwiązanie do którego przejdziemy za chwilę.

Przyjmijmy więc, że chcemy zdefiniować macierz połączeń tak, aby prawdopodobieństwo prawdziwości węzła potomnego Q, dla danych wartości węzłów rodzicielskich, odpowiadało częstości prawdziwych węzłów rodzicielskich. Nazwijmy tę macierz połączeń L_{SUM}. Ogólnie, w przypadku n węzłów rodzicielskich drugi wiersz macierzy dla $Q = prawda$ miałby postać: $L_{SUM}(1, j) = m_j/n$, gdzie j jest indeksem kolumny, m_j liczbą węzłów rodzicielskich w j-tej kolumnie, mających wartość

prawda. Pierwszy wiersz, dla Q = *fałsz*, miałby w poszczególnych kolumnach wartości dopełniające do 1, czyli $L_{SUM}(0, j) = (n - m_j)/n$.

Dla naszego przykładu trzech węzłów rodzicielskich, macierz połączeń L_{SUM} miałaby więc postać:

$$L_{SUM} = \begin{pmatrix} 1 & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} & 1 \end{pmatrix} \tag{2.2.48}$$

W drugim wierszu, wyraz $L_{SUM}(1, 0)$ ma wartość 0, ponieważ w pierwszej kolumnie, dla $j = 0$ (000_2), wartości w węzłach rodzicielskich wynoszą A = *fałsz, B* = *fałsz, C* = *fałsz*, czyli nie ma żadnego rodzica o wartości prawda. W drugiej kolumnie, dla $j = 1$ (001_2), czyli A = *fałsz, B* = *fałsz, C* = *prawda*, mamy $L_{SUM}(1, 1) = 1/3$, ponieważ jeden rodzic na trzech jest prawdziwy. Podobnie w kolumnie trzeciej. W kolumnie czwartej, dla $j = 3$ (011_2), czyli A = *fałsz, B* = *prawda, C* = *prawda*, mamy dwóch rodziców prawdziwych, czuli $L_{SUM}(1, 3) = 2/3$. I tak dalej, aż w końcu w ostatniej kolumnie, dla układu wartości A = *prawda, B* = *prawda, C* = *prawda*, wszystkie węzły rodzicielskie są prawdziwe, tak więc $L_{SUM}(1, 7) = 3/3 = 1$.

Nietrudno zauważyć, że tym razem, do obliczenia poszczególnych kolumn w drugim wierszu macierzy L_{SUM}, możemy użyć wyrażenia:

$$P(Q = prawda) = \frac{1}{3}(1-a)(1-b)c + \frac{1}{3}(1-a)b(1-c) + \frac{2}{3}(1-a)bc$$
$$+ \frac{1}{3}a(1-b)(1-c) + \frac{2}{3}a(1-b)c + \frac{2}{3}ab(1-c) + abc$$

Łatwo sprawdzić, że dla każdej kolumny macierzy L_{SUM} (z wyjątkiem pierwszej, dla $j = 0$) układ prawdziwości w węzłach rodzicielskich A, B, C odpowiada dokładnie jednemu ze składników tego wyrażenia, zaś pozostałe składniki są dla niego równe 0.

Po prostych obliczeniach otrzymamy więc, że

$$P(Q = prawda) = \frac{a+b+c}{3} \tag{2.2.49}$$

Formuła obliczeniowa dla pierwszego wiersza, musi dawać w wyniku dopełnienia do wartości 1, tak więc:

$$P(Q = fałsz) = 1 - \frac{a+b+c}{3} \tag{2.2.50}$$

Spoglądając na zależność (2.2.49), widzimy, że przy takiej definicji tablicy połączenia dla węzła Q, prawdopodobieństwo prawdziwości tego węzła równe jest po prostu średniej z prawdopodobieństw prawdziwości poszczególnych jego rodziców. Zauważmy, jednak że w ten sposób traktujemy prawdziwość wszystkich jego rodziców jak jednakowo istotną, co może być poważnym ograniczeniem w praktycznych zastosowaniach.

Spróbujmy więc zaradzić temu brakowi. Dla każdego z węzłów rodzicielskich zdefiniujemy wagę, określającą jego istotność, a dokładniej siłę, z jaką prawdopodobieństwo prawdziwości tego węzła wpływa na nasze przekonanie (ang. *belief*) o prawdziwości węzła potomnego Q. W naszym przykładzie z trzema węzłami rodzicielskimi, będziemy więc mieli trzy takie wagi w_a, w_b, w_c, związane z poszczególnymi węzłami rodzicielskimi.

Jako miarę łącznego oddziaływania wszystkich rodziców, przyjmiemy sumę ważoną prawdopodobieństw ich prawdziwości: $w_a a + w_b b + w_c c$. Oznaczmy przez t sumę naszych wag, czyli wartość $t = w_a + w_b + w_c$. Zauważmy, że t jest maksymalnym oddziaływaniem wszystkich rodziców, dla maksymalnych prawdopodobieństw ich prawdziwości $a = 1$, $b = 1$ i $c = 1$. Będziemy więc chcieli, aby w konkretnej sytuacji prawdopodobieństwo prawdziwości Q było proporcjonalne do łącznego oddziaływania węzłów rodzicielskich, w całości możliwego ich wpływu: $(w_a a + w_b b + w_c c)/t$.

Wprowadzimy również jeszcze jedną wagę $w_q \in [0, 1]$, dla węzła Q, określającą nasze przekonanie co do prawdziwości samego wnioskowania o wartości Q. Będziemy przy jej pomocy skalować otrzymane oszacowanie prawdopodobieństwa prawdziwości Q. W takiej sytuacji macierz połączeń L_{WTD_SUM}, dla węzła Q będzie miała postać:

$$L_{WTD\_SUM} = \begin{pmatrix} 1 & 1-\frac{w_c w_q}{t} & 1-\frac{w_b w_q}{t} & \frac{(w_b+w_c)w_q}{t} & 1-\frac{w_a w_q}{t} & 1-\frac{(w_a+w_c)w_q}{t} \\ 0 & \frac{w_c w_q}{t} & \frac{w_b w_q}{t} & \frac{(w_b+w_c)w_q}{t} & \frac{w_a w_q}{t} & \frac{(w_a+w_c)w_q}{t} \\ 1-\frac{(w_a+w_b)w_q}{t} & 1-w_q \\ \frac{(w_a+w_b)w_q}{t} & w_q \end{pmatrix} \quad (2.2.51)$$

W kolumnie pierwszej, odpowiadającej wartościom A = *fałsz*, B = *fałsz*, C = *fałsz*, $L_{WTD\_SUM}(1, 0)$, czyli prawdopodobieństwo prawdziwości,

oczywiście jest równe 0. W kolumnie drugiej, dla *A = fałsz, B = fałsz, C = prawda*, ponieważ tylko węzeł *C* ma wartość *prawda*, wyraz L_{WTD_SUM} (1, 1) jest równy $w_c w_q / t$. W kolumnie trzeciej, dla *A = fałsz, B = prawda, C = fałsz*, prawdziwy jest tylko węzeł *B*, stąd $L_{WTD\_SUM}(1, 2) = w_b w_q / t$. W czwartej kolumnie, dla *A = fałsz, B = prawda, C = prawda*, prawdziwe są węzły *B* i *C*, musimy więc uwzględnić ich łączny wpływ na prawdziwość *Q*, $L_{WTD\_SUM}(1, 3) = (w_b + w_c) w_q / t$. Postępujemy tak dalej, aż wreszcie w ostatniej kolumnie, dla *A = prawda, B = prawda, C = prawda*, otrzymujemy $L_{WTD\_SUM}(1, 7) = (w_a + w_b + w_c) w_q / t = \mathrm{t} w_q / \mathrm{t} = w_q$.

Formułę algebraiczną wyznaczającą wartości elementów macierzy L_{WTD_SUM}, podobnie jak w poprzednim przypadku, dla macierzy L_{SUM}, możemy zbudować z zależności:

$$\begin{aligned}(Q = prawda) = {} & \frac{w_c w_q}{t}(1-a)(1-b)c + \frac{w_b w_q}{t}(1-a)b(1-c) + \\ & + \frac{(w_b + w_c) w_q}{t}(1-a)bc + \frac{w_a w_q}{t}a(1-b)(1-c) + \\ & + \frac{(w_a + w_c) w_q}{t}a(1-b)c + \frac{(w_a + w_b) w_q}{t}ab(1-c) + \\ & + w_q abc\end{aligned}$$

Co po prostych przekształceniach daje nam zależność

$$P(Q = prawda) = \frac{(w_a a + w_b b + w_c c) w_q}{t} \tag{2.2.52}$$

Oraz analogiczną dla pierwszego wiersza macierzy, dla *Q = fałsz*:

$$P(Q = fałsz) = 1 - \frac{(w_a a + w_b b + w_c c) w_q}{t} \tag{2.2.53}$$

Węzły sieci wykorzystujące macierze połączeń L_{WTD_SUM}, w formie sumy ważonej prawdopodobieństw prawdziwości termów rodzicielskich, mogą służyć do definiowania termów indeksujących, reprezentujących strategie ważenia termów, np. znana nam strategia tf*idf.

Niech *A*, *B*, *C* będą węzłami dokumentów, Q węzłem termu indeksującego te dokumenty. Przyjmijmy dalej, że w_a, w_b, w_c są znormalizowanymi częstościami *tf* tego termu w tych dokumentach, zaś $w_q = idf_q$ ($w_a + w_b + w_c$). Jeżeli wówczas, na przykład zaobserwowany został dokument *A*, tzn. *A = prawda*, to wartość *bel*(*Q*), przekonania o prawdziwości w węźle *Q* równa jest

$$bel(Q) = \frac{w_a w_q}{w_a + w_b + w_c} = \frac{tf_a idf_q \left(w_a + w_b + w_c \right)}{w_a + w_b + w_c} = tf_a idf_q \qquad (2.2.54)$$

Jak więc widzimy, macierze połączeń charakterystyce L_{WTD_SUM}, mogą być w sieci bayesowskiej wykorzystywane do reprezentacji węzłów dokumentów, definiując strategie ich ważenia. Mogą być one stosowane również w węzłach zapytania, do reprezentacji zapytań probabilistycznych. W przypadku, gdy system wyszukiwawczy realizować ma zapytania o charakterze boolowskim, w węzłach zapytania należy stosować macierze połączeń L_{OR}, L_{AND} i L_{NOT}.

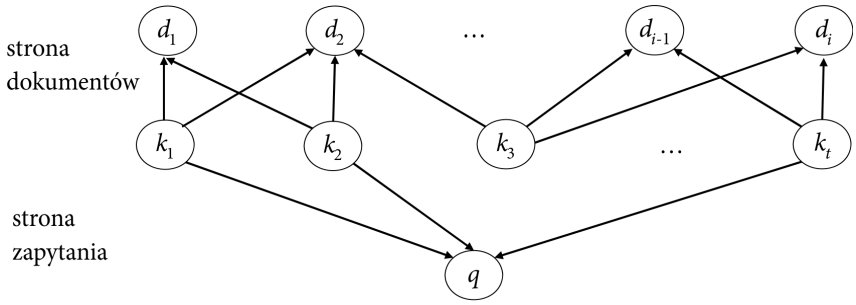
Jak więc widzimy, model wyszukiwawczy oparty na sieci wnioskującej pozwala realizować operacje wyszukiwawcze oparte na bardzo odmiennych zasadach działania, definiując je przy tym w jednolitym środowisku probabilistycznym. Dzięki temu w sieci bayesowskiej granice między różnymi podejściami do wyszukiwania informacji zacierają się, pozwalając w łatwy sposób na łączenie odmiennych technik, ich wymienne czy uzupełniające się działanie. Ponadto, względna prostota samej sieci powoduje, że łatwo jest projektować i dodawać do niej nowej elementy (węzły), rozszerzające funkcjonalność systemu, czy też modyfikujące jego działanie w pożądany sposób. Te dwie właściwości stanowią główne zalety tego modelu, stojące za jego znaczną popularnością i dosyć szerokim zastosowaniem.

Drugim dobrze zdefiniowanym i szeroko akceptowanym modelem wyszukiwania informacji opartym na wykorzystaniu sieci bayesowskich jest tzw. model sieci przekonania (ang. *belief network*), stworzony przez Ribeiro-Neto i Muntza [Ribeiro-Neto, Muntz, 1996; Ribeiro-Neto, Silva, Muntz, 2000; Baeza-Yates, Ribeiro-Neto, 1999].

Podstawowa struktura sieci w modelu propagacji przekonania zaprezentowana została na rysunku 2.2.10. Jak widzimy, jej struktura podobna jest do typowego grafu w modelu sieci wnioskującej, nieco inny jednak jest kierunek przyczynowości w zależnościach między węzłami. Sieć składa się z trzech warstw jednostek, dokumentów, słów kluczowych oraz zapytania. Warstwa termów rozdziela obie pozostałe warstwy. Kierunek przyczynowości prowadzi od słów kluczowych do pozostałych warstw.

Zbiór wszystkich słów kluczowych $U = \{k_1, \ldots, k_t\}$, gdzie t jest liczbą wszystkich wykorzystywanych termów, tworzy przestrzeń zdarzeń elementarnych. Dalej, przez $u \subset U$ będziemy rozumieli podzbiór (określany również jako pojęcie) U, złożony z elementarnych słów kluczowych. Słowa kluczowe mają charakter binarny, są więc zmiennymi losowymi, przyjmującymi wartości {prawda, fałsz}, albo {0, 1}. Tak więc, każdy podzbiór/pojęcie u, może być reprezentowany przez pewien binarny wektor

$u = (k_1, \ldots, k_t)$, gdzie $k_i = 1$ jeśli i-ty term należy do pojęcia u, lub $k_i = 1$, jeśli nie należy. Wprowadźmy funkcję g_i, sprawdzającą stan i-tego termu, tzn. $g_i(u)$ jest równe wartości termu k_i dla pojęcia/podzbioru u.



Rys. 2.2.10. Podstawowa struktura sieci bayesowskiej w modelu propagacji przekonania

Źródło: opracowanie własne na podstawie [Ribeiro-Neto, Muntz, 1996].

Zwróćmy uwagę, że pojęciami/podzbiorami w U będą również dokumenty d, jako reprezentacje indeksujących je słów kluczowych, oraz zapytania q, jako reprezentacje słów kluczowych, które zostały w nich wykorzystane. Jeśli więc term k_i został użyty do indeksowania dokumentu d, to $g_i(d) = 1$. Podobnie jeśli term k_j występuje w zapytaniu q, to $g_j(q)$ także musi być równe 1.

Niech P będzie łącznym rozkładem prawdopodobieństwa, zdefiniowanym w przestrzeni U. Wówczas prawdopodobieństwo dowolnego zbioru/pojęcia $c \subset U$ jest równe:

$$P(c) = \sum_u P(c|u) P(u) \tag{2.2.55}$$

$P(u)$ jest prawdopodobieństwem *a priori* związanym z każdym podzbiorem/pojęciem u, $P(c \mid u)$ definiuje związek pokrywania się między pojęciami c oraz u, w przestrzeni U. $P(c)$ może więc być interpretowane jako stopień pokrywania przestrzeni U przez c. Zwróćmy uwagę, że sumowanie w równaniu (2.2.55) odbywa się po wszystkich 2^t podzbiorach $u \subset U$. Jednak system wyszukiwania informacji, wymaga zazwyczaj wzięcia pod uwagę tylko kilku powiązanych pojęć. Ponieważ wszystkie pojęcia $u \subset U$ są początkowo jednakowo możliwe, prawdopodobieństwo *a priori* $P(u)$ równe jest $P(u) = (1/2)^t$.

Przypomnijmy sobie, że również zapytanie q jest pojęciem w U, tak więc $P(q) = \Sigma_u P(q|u)P(u)$. Podobnie pojęciem jest każdy dokument d_j, czyli $P(d_j) = \Sigma_u P(d_j|u)P(u)$.

Ranking dokumentu d_j w sieci przekonania wykonywany jest na podstawie podobieństwa między dokumentem d_j i zapytaniem q, definiowanego jako stopień pokrywania się d_j oraz q, i mierzonego prawdopodobieństwem $P(d_j|q)$. Stosując definicję prawdopodobieństwa warunkowego: $P(d_j|q) = P(d_j, q)/P(q)$. Ponieważ $P(q)$ jest stałe dla zapytania, wystarczy obliczyć $P(d_j, q)$, przy pomocy wyrażenia $P(d_j, q) = \Sigma_u P(d_j, q|u) P(u)$. Prawdopodobieństwa występowania słów kluczowych w dokumentach i zapytaniach są zupełnie niezależne. W sieci węzły słów kluczowych rozdzielają zapytanie i dokumenty. Czyli węzły d_j *oraz* q są warunkowo niezależne. Tak więc $P(d_j, q|u) = P(d_j|u) P(q|u)$. Co daje nam generyczną formułę rankingowania dokumentu d_j w odniesieniu do zapytania q:

$$P\left(d_j|q\right)=\eta\sum_u P\left(d_j|u\right)P\left(q|u\right)P\left(u\right) \qquad (2.2.56)$$

gdzie stała η ma charakter normalizujący.

Szczegóły realizacji rankingu przy pomocy formuły (2.2.56) zależą już oczywiście od konkretnej architektury sieci. Podobnie jak Turtle i Croft dla modelu sieci wnioskującej, Ribeiro-Neto i Muntz pokazują sposób modelowania przy użyciu sieci propagacji przekonania podstawowych podejść do wyszukiwania informacji [Ribeiro-Neto, Silva, Muntz, 2000]. W przypadku modelu sieci wnioskującej proces sporządzania rankingu opierał się na operacjach niskiego poziomu – przeliczenia wartości w węzłach sieci, przy obserwacji występowania kolejnych dokumentów. Nieco odmienne sformułowanie sieci zaproponowane w modelu propagacji przekonania pozwala na realizację procesu rankingowania dokumentu na wyższym poziomie, co znacznie upraszcza definiowanie systemu.

Rozpoczniemy od określania prawdopodobieństw warunkowych połączeń w węzłach reprezentujących zapytania boolowskie. Na potrzeby sieci bayesowskiej wygodniej nam będzie spojrzeć na proces określania prawdziwości wyrażenia logicznego sformułowanego w zapytaniu, jako na zagadnienie dopasowania wzorca, czy też podobieństwa dwu wektorów. Pamiętajmy bowiem, że w modelu sieci propagacji przekonania, zapytania i opisy dokumentów definiujemy jako pojęcia w przestrzeni U, czyli binarne wektory o wartościach 1 lub 0, w zależności, czy dane słowo kluczowe występuje lub nie w zapytaniu albo w dokumencie.

Pokażmy ideę tego rozwiązania na przykładzie wykorzystywanego już wcześniej zapytania:

$$k_1 \wedge (k_2 \vee \neg\, k_3)$$

Przypomnijmy, że mówiąc o optymalizacji zapytań w modelu wyszukiwania boolowskiego, pokazaliśmy że wyrażenie logiczne może zostać przedstawione w kanonicznej alternatywnej postaci normalnej. Korzystaliśmy z tej właściwości również w modelu sieci wnioskującej. Przypomnijmy, że kanoniczna alternatywna postać normalna zbudowana jest jako alternatywa wyrazów koniunkcyjnych, odpowiadających wartościom „1" lub „prawda" w wyjściowym wyrażeniu. Łatwo sprawdzić, że dla naszego wyrażenia postać ta ma formę:

$$k_1 \wedge (k_2 \vee \neg\, k_3) = k_1 \wedge k_2 \wedge \neg k_3 \vee k_1 \wedge k_2 \wedge k_3 \vee k_1 \wedge \neg k_2 \wedge \neg k_3$$

Aby wyrażenie było prawdziwe, co najmniej jeden z członów jego postaci alternatywnej musi być prawdziwy. W kontekście systemu wyszukiwawczego oznacza to, że dla danego podzbioru/pojęcia u, aby zapytanie było prawdziwe, musi istnieć taki człon w postaci koniunkcyjnej warunku zapytania, który jest dokładnie dopasowany do zestawu słów kluczowych występujących w u.

Właściwość ta została wykorzystana do zdefiniowania prawdopodobieństwa warunkowego połączeń w węźle zapytania q. Oznaczmy przez q_{apn} kanoniczną alternatywną postać normalną, zaś przez q_{ak}, $k = 1, 2, \ldots$ kolejne jej człony. Wówczas:

$$\begin{aligned} P(q|u) &= \begin{cases} 1 \text{ jeżeli } \exists_{q_{ak} \in g_{apn}} \ \forall_{k_i}\ g_i(u) = g_i(q_{ak}) \\ 0 \text{ w przeciwnym razie} \end{cases} \\ P(\overline{q}|u) &= 1 - P(q|u) \end{aligned} \tag{2.2.57}$$

Dla węzłów dokumentów prawdopodobieństwa warunkowe zdefiniowane są jeszcze prościej. Prawdopodobieństwo dokumentu dla danego podzbioru u jest równe 1, jeżeli wzorzec słów kluczowych w dokumencie i w u jest taki sam. W przeciwnym wypadku prawdopodobieństwo to jest równe 0.

$$\begin{aligned} P(d_j|u) &= \begin{cases} 1 \text{ jeżeli } \forall_{k_i}\ g_i(u) = g_i(d_j) \\ 0 \text{ w przeciwnym razie} \end{cases} \\ P(\overline{d}_j|u) &= 1 - P(q|u) \end{aligned} \tag{2.2.58}$$

Prawdopodobieństwo *a priori* poszczególnych termów, jak już wspomnieliśmy wcześniej, określone jest przez $P(u) = (1/2)^t$, co wynika

z założenia o jednostajnym rozkładzie prawdopodobieństwa poszczególnych termów. Korzystając więc z (2.2.57) i (2.2.58), łatwo możemy zastosować do wyznaczenia rankingu dokumentu formułę (2.2.56). Jeżeli istnieje takie pojęcie u, które odpowiada wzorcowi dokumentu d_j, oraz spełnia ono zapytanie q, to $P(d_j \mid q) > 0$ i dokument przyjmowany jest jako relewantny. W przeciwnym wypadku $P(d_j \mid q) = 0$ i dokument odrzucany jest jako nierelewantny.

Równie łatwo możemy zdefiniować wzory dla sieci bayesowskiej realizującej wyszukiwanie w modelu wektorowym.

Przyjmijmy, że w_{iq} oraz w_{ij} są wagami związanymi z termem k_i, odpowiednio w pojęciach reprezentujących zapytanie q oraz dokument d_j. Na przykład mogą być one obliczane przy wykorzystaniu formuły *tf*idf*. Ponadto zdefiniujmy przez u_i pojęcie/podzbiór jednoelementowy, odpowiadający dokładnie określonemu słowu kluczowemu k_i, tj. $g_i(u_i) = 1$ oraz $g_j(u_i) = 0$, dla wszystkich $j \neq i$.

Wówczas prawdopodobieństwa warunkowe połączeń w węzłach sieci bayesowskiej możemy zdefiniować następująco:

$$P(q|u) = \begin{cases} \dfrac{w_{iq}}{\sqrt{\sum_i w_{iq}^2}} \text{ jeżeli } u = u_i \text{ oraz } g_i(q) = 1 \\ 0 \text{ w przeciwnym razie} \end{cases} \tag{2.2.59}$$

$$P(d_j|u) = \begin{cases} \dfrac{w_{ij}}{\sqrt{\sum_i w_{ij}^2}} \text{ jeżeli } u = u_i \text{ oraz } g_i(d_j) = 1 \\ 0 \text{ w przeciwnym razie} \end{cases} \tag{2.2.60}$$

Przy tak zdefiniowanych węzłach sieci dosyć oczywiste jest, że zastosowanie do rankingowania dokumentów formuły (2.2.56), prowadzi do wyznaczania cosinusoidalnej miary podobieństwa dokumentu do zapytania.

Jak więc widzimy, oba przedstawione modele wykorzystujące sieci bayesowskie w procesie wyszukiwania informacji, dzięki wykorzystaniu odmiennej struktury sieci o różnych kierunkach przyczynowości, podchodzą do tego zagadnienia z różnych stron. Model sieci wnioskującej opiera się na reprezentacji przez sieć niskopoziomowej propagacji prawdopodobieństw, model sieci propagacji przekonania na modelowaniu wysokopoziomowych procedur wyszukiwawczych. Obydwa podejścia pozwalają na łączenie różnych metod wyszukiwania informacji w jednym środowisku probabilistycznym, oraz ich rozbudowywanie w pożądanych

kierunkach. Zwróćmy jednak uwagę, że nie wykorzystują one w zasadzie mechanizmów asocjacyjnych, opartych na powiązaniach między dokumentami, lub termami. Nie pozwalają również na rozszerzenie modelu wyszukiwawczego o wiedzę dziedzinową dotyczącą zależności między tymi obiektami.

Dlatego dalsze działania nad wykorzystaniem sieci bayesowskich w wyszukiwaniu, prowadzone były właśnie w tych kierunkach. Wymienić tutaj można takie prace jak [Acid, de Campos i in., 2003; de Campos, Fernandez-Luna, Huete, 2002] wprowadzające w sieci dodatkową warstwę dokumentów, w celu modelowania zależności między nimi. Najbardziej chyba rozwiniętą propozycję rozbudowy bayesowskiego modelu wyszukiwania o powiązania asocjacyjne i dziedzinowe między dokumentami i termami przedstawili de Campos, Fernandez-Luna i Huete, w swoim modelu wyszukiwawczym opartym na sieci bayesowskiej, modelu BNR (ang. *Bayesian Network-based Retrieval*) [de Campos, Fernandez-Luna, Huete, 2003].

Sieć w modelu BNR złożona jest ze zbioru zmiennych termów $\mathscr{T} = \{T_i, i = 1, \ldots, M\}$, gdzie M jest liczbą termów w słowniku kolekcji, oraz zmiennych dokumentów $\mathscr{D} = \{D_j, j = 1, \ldots, N\}$, gdzie N jest liczbą dokumentów. Zmienne te są dwuwartościowe, przyjmując wartości ze zbioru {*relewantny*, *nierelewantny*}. Dla termów pojęcie relewancji/nierelewancji rozumiane jest jako ich wystąpienie (lub nie) w zapytaniach w subiektywnym świetle potrzeby informacyjnej użytkownika. W celu uproszczenia zapisu będziemy oznaczać, że dany term T_i jest relewantny jako t_i, nierelewantny jako $\overline{t}_i$, oraz analogicznie dla dokumentów: d_j i $\overline{d}_j$.

Model bierze pod uwagę zależności między zmiennymi. Wiedza o nich podzielona jest na dwa rodzaje: wiedzę ekspercką – wykorzystującą zbiór ogólnych kryteriów spójności, oraz wiedzę z kolekcji – otrzymaną na podstawie eksploracji zawartości dokumentów.

Wiedza ekspercka składa się z pewnych założeń, które model musi spełniać. Pozwala ona na stworzenie wstępnego szkieletu sieci bayesowskiej. Można wymienić wśród nich następujące warunki [de Campos, Fernandez-Luna, Huete, 2003]:

1. Istnieją silne zależności między dokumentami oraz indeksującymi je termami. Pozwala to na stworzenie grafu sieci wykorzystującego te związki, tj. w którym od każdego węzła termu prowadzą połączenia skierowane do indeksowanych przez niego dokumentów.
2. Istniejące związki między dokumentami realizują się wyłącznie za pośrednictwem termów, które je indeksują. W modelu BNR nie ma więc bezpośrednich połączeń między dokumentami.

3. Jeśli potrafimy określić relewantność lub nie wszystkich termów związanych z dokumentem D_j, na nasze przekonanie co do relewancji samego dokumentu nie wpływają inne czynniki, np. takie jak wiedza, że jakiś inny dokument D_k lub term T_i jest relewantny (albo nie). To założenie implikuje warunkową niezależność dokumentów przy danych termach, którymi zostały zaindeksowane.

Założenia te są więc dosyć zbliżone do stojących za modelem sieci propagacji przekonania, z wyjątkiem faktu, że nie wykorzystuje się tutaj jawnie określonego węzła zapytania.

Wiedza pozyskiwana z kolekcji służy do określenia najważniejszych związków między termami, w celu doprecyzowania modelu. W praktyce oznacza to, że do budowy warstwy termów niezbędne jest użycie automatycznego algorytmu uczącego. Pierwszy etap procesu tworzenia sieci polega na identyfikacji połączeń w warstwie termów. De Campos, Fernandez-Luna i Huete przyjęli jako podstawowy model reprezentacji zależności między termami, poli-drzewo (albo niby-drzewo, ang. *polytree*), acykliczny graf skierowany, w którym każdą parę węzłów łączy nie więcej niż jedna ścieżka.

Jest to nieco uproszczona struktura, w porównaniu z ogólnie dopuszczalnymi dla sieci bayesowskich. Oczywiście, bardziej złożony model topologii sieci mógłby dokładniej odzwierciedlać związki zależności i niezależności warunkowej pomiędzy termami. Jednocześnie jednak zwiększałby on nakłady obliczeniowe potrzebne zarówno do uczenia, jak i potem do obliczeń propagacji sygnału w sieci. Biorąc pod uwagę rozmiary sieci wynikające z typowej liczby termów i dokumentów w kolekcji oraz istnienie dla poli-drzew efektywnych algorytmów zarówno uczenia, jak i propagacji prawdopodobieństwa o liniowej złożoności obliczeniowej względem liczby węzłów grafu, autorzy zdecydowali się na to uproszczenie struktury modelu BPN, tak by mógł być on stosowany również dla większych baz dokumentów.

Algorytm tworzenia poli-drzewa termów wykorzystuje indeks kolekcji w formie pliku odwrotnego (patrz rozdział 2.1.2) i składa się z trzech głównych kroków [de Campos, Fernandez, Huete, 1998; de Campos, Fernandez-Luna, Huete, 2003]:

1. Obliczenie stopnia zależności dla każdej pary termów.
2. Konstrukcja szkieletu drzewa.
3. Zorientowanie krawędzi w drzewie i ostateczne stworzenie poli-drzewa.

Nie będziemy, oczywiście, szczegółowo analizować całego algorytmu, przyjrzymy się jednak bliżej poszczególnym jego etapom. Dokładny jego opis przedstawiony został w [de Campos, Fernandez, Huete, 1998].

1. Obliczenie stopnia współzależności dla każdej pary termów.

Stopień współzależności pary termów T_i, T_j obliczany jest przy wykorzystaniu miary względnej entropii, opartej na dywergencji Kullbacka-Leiblera:

$$Dep(T_i, T_i \mid \varnothing) = \sum_{T_i^v, T_j^v} P\left(T_i^v, T_j^v\right) \ln\left(\frac{P\left(T_i^v, T_j^v\right)}{P\left(T_i^v\right) P\left(T_j^v\right)}\right) \qquad (2.2.61)$$

Jeżeli współczynnik $Dep(T_i, T_j \mid \varnothing)$ jest równy zero, oznacza to, że T_i i T_j są niezależne od siebie, a wraz ze wzrostem poziomu ich współzależności, jego wartość również rośnie. Indeks v w oznacza różne wartości zmiennych reprezentujących termy, na przykład oznacza prawdopodobieństwo, że term T_i występuje w dokumencie, a T_j nie występuje. Prawdopodobieństwa liczone są na podstawie list dokumentów w indeksie odwrotnym kolekcji, w formie częstości współwystępowania odpowiednich stanów termów. Na przykład to odsetek dokumentów, w których term T_i występuje, a T_j nie występuje, zaś to odsetek dokumentów w których obydwa badane termy współwystępują.

2. Konstrukcja szkieletu drzewa.

W kolejnym kroku należy określić, które termy zostaną połączone w grafie sieci, a które nie. Oczywiście zależy nam, aby w strukturze sieci znalazły się połączenia między słowami kluczowymi o wysokim poziomie współzależności $Dep(T_i, T_j \mid \varnothing)$, natomiast węzły niezależne pozostały niepołączone.

W tym celu wykorzystany został jeden ze standardowych algorytmów konstrukcji minimalnego drzewa rozpinającego, algorytm Prima [Cormen, Leiserson i in., 2017]. Generalnie, algorytm Prima tworzy drzewo, w którym suma wag połączeń między węzłami jest maksymalna. Rozpoczyna od drzewa złożonego z jednego węzła, a następnie w każdym kroku dodaje do drzewa połączenie między węzłem, który już się w nim znajduje a nowym, jeszcze nie wykorzystanym, wybierając go na podstawie największej wartości wagi.

Oczywiście, wykorzystywane w procesie tworzenia drzewa wagi połączeń między węzłami określane są z wykorzystaniem miary ich współzależności $Dep(T_i, T_j \mid \varnothing)$. Pamiętać jednak należy o tym, że zbiór słów kluczowych indeksujących kolekcję dokumentów zawiera zazwyczaj bardzo dużą liczbę termów, w związku z czym wartości zazwyczaj generalnie są dosyć niskie. Algorytm generowania drzewa ma wówczas problemy

z dyskryminacją między termami zależnymi i niezależnymi. Bezpośrednie użycie miary $Dep(T_i, T_j|\emptyset)$ daje zazwyczaj w wyniku graf o bardzo dużej liczbie połączeń, między węzłami o niskim stopniu zależności między nimi, a nawet tak naprawdę niezależnymi.

Dlatego dla obliczonych wartości $Dep(T_i, T_j|\emptyset)$ wykonano test niezależności Chi kwadrat, z jednym stopniem swobody. Jeśli dla danej pary termów T_i, T_j test niezależności dał wynik negatywny (oznaczmy ten fakt przez $\neg I(T_i, T_j|\emptyset)$), oznacza to, że termy są zależne i potencjalne połączenie między nimi jest przydatne. W przypadku pozytywnego wyniku testu tj. przyjęcia hipotezy zerowej o niezależności (oznaczmy taką sytuację przez $I(T_i, T_j|\emptyset)$), termy są niezależne i połączenie między nimi jest zbędne.

Ostatecznie więc, wykorzystane w algorytmie Prima wagi połączeń między węzłami w gałęziach grafu mają wartość:

$$Dep(T_i,T_i)=\begin{cases} Dep(T_i,T_i|\emptyset) & \text{jeżeli} \neg I(T_i,T_i|\emptyset) \\ 0 & \text{jeżeli } I(T_i,T_i|\emptyset) \end{cases} \qquad (2.2.62)$$

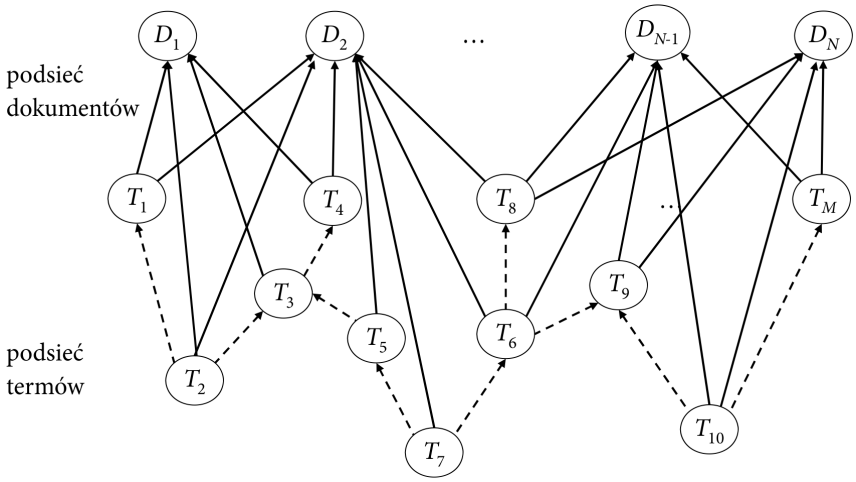
3. Zorientowanie krawędzi w drzewie i ostateczne stworzenie poli-drzewa

Ostatnim krokiem w budowie poli-drzewa jest zorientowanie połączeń w utworzonym grafie. W pierwszym etapie definiujemy wszystkie węzły z połączeniami schodzącymi się. Dla każdej trójki węzłów (termów) tworzącej poddrzewo $T_i - T_k - T_j$ sprawdzamy, czy połączenia między nimi nie powinny mieć kierunków $T_i \rightarrow T_k \leftarrow T_j$. W tym celu korzysta się z właściwości, które powinny posiadać sieci bayesowskie (a nawet wszystkie sieci przyczynowe). Otóż, w układzie schodzącym się $T_i \rightarrow T_k \leftarrow T_j$, zaobserwowanie wartości środkowego węzła powinno zwiększać współzależność między skrajnymi węzłami T_i oraz T_j. W przypadku innych układów, np. rozchodzącego się $T_i \leftarrow T_k \rightarrow T_j$, zaobserwowanie środkowego węzła ma przeciwny efekt – osłabia współzależność między skrajnymi węzłami T_i oraz T_j [Jensen, Nielsen, 2007].

Jeśli więc porównamy stopień współzależności skrajnych węzłów pod warunkiem węzła środkowego i bezwarunkowo, możemy wykorzystać tę wiedzę do wskazania układów węzłów schodzących się. Wykorzystamy jeszcze drugą zależność. Jeśli zdecydujemy się wprowadzić układ schodzący się $T_i \rightarrow T_k \leftarrow T_j$, to termy T_i i T_j nie powinny być warunkowo niezależne względem T_k. Warunek ten sprawdzany jest testem niezależności Chi Kwadrat, dla wartości $Dep(T_i, T_j \mid T_k)$, tym razem z dwoma stopniami swobody. Oznaczmy wynik testu przez $I(T_i, T_j \mid T_k)$.

Sprawdzamy więc wszystkie trójki węzłów (termów) tworzące poddrzewa $T_i - T_k - T_j$. Jeżeli $Dep(T_i, T_j | \emptyset) < Dep(T_i, T_j | T_k)$ oraz $\neg I(T_i, T_j | T_k)$, to wówczas kierujemy połączenia między tymi węzłami $T_i \rightarrow T_k \leftarrow T_j$.

Po określeniu wszystkich połączeń schodzących się, pozostałe kierujemy tak, aby nie powstawały już żadne inne nowe węzły, w których połączenia się schodzą. W wyniku powyższych operacji, otrzymujemy więc sieć o strukturze zbliżonej jak na rysunku 2.2.11. Połączenia między termami i dokumentami (na rysunku zaznaczone ciągłymi liniami) wynikają z zależności związanej z indeksowaniem dokumentów. Połączenia między termami (zaznaczone linią przerywaną) tworzą poli-drzewo, w którym (jak już nadmieniliśmy) każda para termów połączona jest nie więcej niż jedną ścieżką skierowaną. Graf ten ma więc formę szeregu struktur drzewiastych, ale połączonych między sobą, a więc nie w formie lasu.



Rys. 2.2.11. Podstawowa struktura sieci bayesowskiej w modelu BNR

Źródło: opracowanie własne na podstawie [de Campos, Fernandez-Luna, Huete, 2003].

Po zakończeniu procesu identyfikacji strukturalnej i utworzeniu podsieci termów kolejnym krokiem jest oszacowanie prawdopodobieństw związanych z połączeniami w jej obrębie, oraz do węzłów dokumentów. Rozpocznijmy od prawdopodobieństw w węzłach słów kluczowych.

W węzłach termów stanowiących korzenie drzewa, niezbędne jest oszacowanie prawdopodobieństwa *a priori* ich relewancji oraz nierelewancji (czyli, przypomnijmy, wystąpienia lub nie w zapytaniu użytkownika). Dla danego słowa kluczowego T_i, prawdopodobieństwo jego relewantności

przyjmujemy jako $P(t_i) = 1/M$, gdzie M jest liczbą wszystkich termów w kolekcji, zaś prawdopodobieństwo nierelewancji $P(\overline{t_i}) = 1 - P(t_i)$.

Dla każdego termu T_i, niebędącego korzeniem poli-drzewa, o zbiorze węzłów rodzicielskich $\Pi(T_i)$, musimy oszacować zbiór prawdopodobieństw warunkowych $P(T_i \mid \pi(T_i))$, czyli wartości węzła T_i dla $\pi(T_i)$, wszystkich możliwych zestawów wartości rodziców węzła T_i.

Niech $\mathbf{t} \in \pi(T_i)$ będzie dowolnym zestawem wartości rodziców węzła T_i. Na przykład jeśli węzeł T_i ma trzy węzły rodzicielskie, to t może być równe (t_1, $\overline{t_2}$, t_3), co odpowiada wartościom rodziców (*relewantny*, *nierelewantny*, *relewantny*), albo ($\overline{t_1}$, $\overline{t_2}$, t_3) – (*nierelewantny*, *nierelewantny*, *relewantny*). Dla dowolnego układu wartości węzłów termów C, oznaczmy dalej przez $n(C)$ liczbę dokumentów, które zawierają wszystkie termy mające w C wartość *relewantny* i nie zawierają żadnego z termów, mających w C wartość *nierelewantny*. Wówczas prawdopodobieństwa warunkowe połączeń w węzłach termów T_i modelu BNR, zdefiniowane są z wykorzystaniem miary podobieństwa zbiorów opartej na współczynniku Jackarda ($|X \cap Y| / |X \cup Y|$):

$$P(\overline{t_i}|\boldsymbol{t}) = \frac{n(\overline{t_i}, \boldsymbol{t})}{n(\overline{t_i}) + n(\boldsymbol{t}) - n(\overline{t_i}, \boldsymbol{t})} \qquad (2.2.63)$$

$$P(t_i|\boldsymbol{t}) = 1 - P(\overline{t_i}|\boldsymbol{t}) \qquad \text{dla wszystkich } \mathbf{t} \in \pi(T_i)$$

Prawdopodobieństwa te obliczane są w fazie budowy modelu i przechowywane jako macierz połączenia w węźle termu T_i.

Ostatnim elementem, jaki pozostał do zdefiniowania modelu, są prawdopodobieństwa warunkowe połączeń w węzłach dokumentów. Dla każdego D_j, zbiór węzłów rodzicielskich $\Pi(D_j)$, odpowiada zbiorowi wszystkich termów indeksujących ten dokument. Liczba rodziców może więc być w tym przypadku bardzo duża, przeciętnie rzędu 200, 300 termów. Liczba możliwych zestawów ich wartości już ogromna: 2^{200}, 2^{300}. Bezpośrednią ich reprezentację w formie macierzy połączenia należy uznać za mało przydatną z praktycznego punktu widzenia. Dlatego w tym przypadku prawdopodobieństwa połączeń wyznaczane są w locie, z wykorzystaniem specyficznych funkcji prawdopodobieństwa, dla tego konkretnego przypadku. Pomijamy tutaj ich dokładne wyprowadzenie, opiera się ono na podobnej zasadzie co przedstawione wcześniej, dla modelu sieci wnioskującej Turtle'a i Crofta.

Niech, analogicznie jak poprzednio, $\pi(D_j)$ będzie zbiorem wszystkich możliwych zestawów wartości rodziców węzła D_j, zaś $\mathbf{t} \in \pi(D_j)$ dowolnym konkretnym zestawem. Wówczas:

$$P\left(d_j|\boldsymbol{t}\right)=\sum_{T_i\in R_t} w_{ij}$$
$$P\left(\overline{d}_j|\boldsymbol{t}\right)=1-P\left(d_j|\boldsymbol{t}\right) \quad \text{dla wszystkich } \mathbf{t}\in\pi\left(D_j\right) \tag{2.2.64}$$

gdzie R_t jest zbiorem termów relewantnych w **t**, w_{ij} jest wagą termu w dokumencie, $w_{ij} \geq 0$ oraz $\sum_{T_i\in D_j} w_{ij} \leq 1$. Generalnie więc, im więcej w **t** jest termów relewantnych, tym większe prawdopodobieństwo relewancji dokumentu D_j.

Po zbudowaniu sieci bayesowskiej, może ona być wykorzystywana w procesie wnioskowania. Typowe jej użycie z punktu widzenia użytkownika, polega na zainicjowaniu tej procedury, poprzez wprowadzenie do systemu nowych informacji, w formie zaobserwowania jako relewantnych, węzłów termów odpowiadających zapytaniu *Q*. Następnie informacja rozprzestrzenia się w systemie, w końcu osiągając węzły D_j, poprzez wyznaczenie $P(d_j \mid Q)$. Dokumenty prezentowane są następnie użytkownikowi posortowane zstępująco względem prawdopodobieństwa ich relewantności.

Na skutek znacznych rozmiarów sieci, ogólne algorytmy wnioskowania mogą być nieakceptowalne z powodów efektywnościowych, nawet przy niezbyt dużych kolekcjach dokumentów. Dlatego de Campos, Fernandez-Luna i Huete zaproponowali dwuetapowy, przybliżony algorytm propagacji sygnału w sieci [de Campos, Fernandez-Luna, Huete, 2003]:

1. Dokładna propagacja w warstwie termów i obliczenie dla wszystkich węzłów T_i prawdopodobieństw ich relewantności $P(t_i \mid Q)$. Pamiętając, że informacja wprowadzana jest do sieci zawsze w formie obserwacji termów składających się na zapytanie, do obliczenia prawdopodobieństwa *a posteriori* każdego węzła termu może zostać wykorzystany algorytm dokładnej propagacji Pearla [Pearl, 1988]. Wykonywany jest on w czasie wielomianowym.
2. Oszacowanie funkcji prawdopodobieństwa w węzłach dokumentów i obliczenie dla każdego D_j prawdopodobieństwa $P(d_j \mid Q)$ z wykorzystaniem prawdopodobieństw *a posteriori* termów obliczonych w poprzednim etapie.

Obliczenia $P(d_j \mid Q)$ wykonywane są zgodnie ze wzorem:

$$P\left(d_j|Q\right)=\sum_{i=1}^{m_j} w_{ij} P\left(t_i|Q\right) \tag{2.2.65}$$

gdzie m_j jest liczbą termów indeksujących dokument D_j.

De Campos, Fernandez-Luna i Huete pokazują [de Campos, Fernandez-Luna, Huete, 2003], że ten sposób wnioskowania daje wyniki równoważne wnioskowaniu dokładnemu w pełnej sieci.

W bieżącym punkcie przedstawiliśmy kilka podejść do wyszukiwania informacji z wykorzystaniem sieci wnioskujących. Skoncentrowaliśmy się na podstawowym, probabilistycznym paradygmacie ich działania. W literaturze przedmiotu proponuje się również wykorzystanie w procesach wyszukiwawczych sieci wnioskujących opartych na innych metodach propagacji przekonania. Z punktu widzenia tematyki naszej książki, szczególnie ciekawe wydają się być rozwiązania wykorzystujące wnioskowanie rozmyte i miary możliwości [Brini, Boughanem, Dubois, 2005; Brini, de Campos i in., 2005; Boughanem, Brini, Dubois, 2009].

2.2.3. Metody uczenia maszynowego w identyfikacji funkcji rankingującej

2.2.3.1. Charakterystyka modeli uczenia się rankingowania

W poprzednich punktach bieżącego rozdziału koncentrowaliśmy się na wykorzystaniu w procesie wyszukiwania informacji standardowych funkcji rankingujących, wynikających z określonych podstaw teoretycznych związanych z danym modelem wyszukiwania: logiki, miar podobieństwa, czy też rachunku prawdopodobieństwa. W punkcie 2.1 przedstawiliśmy podstawowe podejścia do tworzenia tego rodzaju modeli wyszukiwawczych, w punktach 2.2.1 i 2.2.2 pokazywaliśmy rozwiązania pozwalające na poprawę jakości procesu wyszukiwania, przy pomocy metod sztucznej inteligencji.

Parametry standardowych funkcji rankingujących wykorzystywanych w tych podejściach, na ogół szacowane były w procesie uczenia nienadzorowanego, polegającego na zbieraniu określonych statystyk zależności występowania cech dokumentów i zapytań (słów kluczowych) w kolekcji. Jako przykłady tego rodzaju procedur uczących wymienić możemy tutaj ustalanie wag termów w dokumentach w modelu wektorowym, czy też statystyk współwystępowania słów kluczowych potrzebnych do określenia prawdopodobieństw w modelach probabilistycznych.

Jednakże już w punkcie 2.1.3, dotyczącym podejścia probabilistycznego, pojawiła się sytuacja, w której pewne parametry modeli wyszukiwawczych wymagały do ich oszacowania zastosowania metod uczenia nadzorowanego, wykorzystujących w sposób jawny informację o relewancji dokumentów dla danego zapytania. Obecnie pójdziemy dalej w tym

kierunku, rezygnując z teoretycznych funkcji rankingujących, natomiast identyfikując je przy pomocy nadzorowanych metod uczenia maszynowego. Podejście to często określa się mianem Uczenia się Rankingowania (LTR, ang. *Learning To Rank*) [Mitra, Craswell, 2017].

Funkcja rankingująca uczona jest przy mocy zbioru treningowego, zazwyczaj składającego się z zestawu zapytań treningowych. Dla każdego zapytania określony jest zbiór dokumentów, oraz związane z nimi oceny ich relewantności. Informacja o relewantności może mieć charakter bezpośredniej opinii użytkownika, albo zostać uzyskana na podstawie oszacowania korzystającego z innych danych (w środowisku internetowym np. na podstawie kliknięć). W bieżącym punkcie zakładamy, że reprezentacje zapytań i dokumentów, w formie wektorów cech, są określone.

W większości wypadków zakładać będziemy również, że uczenie odbywa się w trybie off-line. Na podstawie zbioru treningowego tworzony jest model wyszukiwawczy, wykorzystywany następnie do tworzenia rankingów wynikowych dla nowych zapytań.

Tak jak wspomnieliśmy, uczenie funkcji rankingującej związane jest z identyfikacją pewnego odwzorowania, z wykorzystaniem zbioru zapytań treningowych, dla których znane są oceny relewancji dokumentów w kolekcji treningowej. Możemy mówić o trzech różnych podejściach do tworzenia modeli LTR, w zależności od postawionych celów uczenia [Liu, 2009, 2011]:

- Podejście punktowe, w którym pary treningowe definiowane są przez pojedyncze dokumenty, dla poszczególnych zapytań: $(\mathbf{x}_{qd}, r_q(d))$. Z każdym wektorem wejściowym $\mathbf{x}_{qd}$, złożonym z połączonych cech zapytania q i dokumentu d, związany jest stopień relewancji tego dokumentu dla danego zapytania, $r_q(d)$.
- Podejście oparte na parach, w którym informacja dotycząca relewancji przyjmuje formę preferencji między parami dokumentów, w odniesieniu do poszczególnych zapytań.
- Podejście oparte na listach, w którym dla poszczególnych zapytań i zbioru związanych z nimi dokumentów informacja o relewancji ma formę rankingowanej listy tych dokumentów.

Proces tworzenia modelu LTR wymaga dosyć rygorystycznego środowiska jego przygotowania oraz weryfikacji. Kwestie wiarygodnej weryfikacji modeli wyszukiwania informacji były przedmiotem intensywnych badań w zasadzie niemal od początku prac w tym zakresie. Jak wspomnieliśmy w punkcie 2.1.3, podstawowe mierniki jakości systemu wyszukiwawczego: dokładność i precyzja wyszukiwania, wymagają pomiaru wyników jego działania w formie zbierania ocen relewancji od użytkowników – procesu nieco mgliście określonego i mocno subiektywnego.

Aby zachować jak największy obiektywizm uzyskanej oceny oraz porównywalność między różnymi badanymi systemami wyszukiwawczymi, wypracowano pewne standardowe podejścia, wśród których podstawową rolę gra tzw. metodologia eksperymentalna Cranfield, opracowana już w latach 60. ubiegłego wieku, przez Cyrila W. Cleverdona na Uniwersytecie Cranfield [Cleverdon, 1991]. Przyjrzymy jej się nieco dokładniej, ponieważ zdefiniowane w jej ramach reguły mają istotne znaczenie nie tylko dla procesu testowania, ale również zbierania danych na potrzeby modeli Learning To Rank.

Zastosowanie metodologii Cranfield do testowania modelu wyszukiwawczego, ma postać następującej procedury [Liu, 2011]:

- Należy zebrać dużą liczbę (losowo wybranych) zapytań, które utworzą zbiór testowy.
- Dla każdego zapytania q:
  - należy zebrać zbiór dokumentów $\{d_j\}$, $j = 1, \ldots, m$, związanych z zapytaniem,
  - uzyskać ocenę relewantności każdego dokumentu na podstawie opinii człowieka,
  - wykorzystać badany model rankingujący do utworzenia rankingu dokumentów,
  - pomierzyć różnicę między wynikami rankingowania a ocenami relewancji, korzystając z określonej miary jakości.
- Do oceny działania modelu rankingującego wykorzystać średnią wartość miary jakości dla wszystkich zapytań testowych.

W przypadku wyboru dokumentów związanych z zapytaniem, celem jest ograniczenie liczby dokumentów, które będą musiały zostać ocenione przez ludzi, poprzez odrzucenie tych, które z pewnością są nierelewantne dla zapytania. Możliwe są tutaj różne strategie. Na przykład: wybierane są wszystkie dokumenty zawierające słowo kluczowe spośród użytych w zapytaniu. Inna często stosowana strategia polega na wykorzystaniu pewnego predefiniowanego, wzorcowego systemu rankingującego, ewentualnie scaleniu wyników działania kilku różnych systemów tego rodzaju.

Jeśli chodzi o zbieranie ocen relewancji, stosowane są trzy strategie:

- Ocena stopnia relewantności. Oceniający ludzie określają, czy dokument jest relewantny dla zapytania, czy nie (ocena binarna), albo dokładniej specyfikują stopień tej relewancji (zazwyczaj w formie kliku kategorii porządkowych, np. Idealny, Doskonały, Dobry, Niezły, Zły).
- Ocena preferencji poprzez porównanie parami. Oceniający ludzie określają, czy dla danego zapytania jeden dokument jest bardziej relewantny niż drugi.

- Ocena pełnego uporządkowania. Oceniający ludzie, dla danego zapytania, określają pełny porządek wszystkich dokumentów, czyli odpowiednie ich ustawienie (permutację), zgodnie z relewancją.

Ręczna ocena relewancji jest zawsze procesem wymagającym czasowo i niemal niemożliwym jest ocenić wszystkie dokumenty, które są relewantne dla zapytania. Tak więc, niemal zawsze mamy do czynienia z sytuacją, że model rankingujący zwraca w rankingu dokumenty, które nie zostały ocenione przez przygotowujących dane ludzi. Na ogół nie ocenione dokumenty przyjmuje się w procesie testowania jako nierelewantne.

Kolejnym krokiem weryfikacji systemu wyszukiwawczego jest porównanie wyników jego działania z uzyskanymi ocenami relewancji, a następnie obliczenie określonej miary jakości jego działania. Podstawowymi miernikami jakościowymi są oczywiście, zdefiniowane w punkcie 2.1.3, dokładność i precyzja wyszukiwania. W systemach opartych na podejściu LTR mają one jednak zdecydowanie niewystarczający charakter i niezbędne jest sięgnięcie po inne miary jakości wyszukiwania, głębiej wchodzące w naturę celów, które chcemy osiągnąć w procesie uczenia tego rodzaju modeli.

Poniżej przedstawimy kilka z najważniejszych miar oceny systemów wyszukiwawczych [Liu, 2011]. Większość mierników działania definiowana jest w odniesieniu do każdego zapytania, jako funkcje listy rankingującej tworzonej przez model i rankingu wynikającego z ocen relewancji, a następnie uśredniana jest po wszystkich zapytaniach. Przedstawiane mierniki mają charakter kryteriów maksymalizowanych, błędy rankingowania dla danej miary określa się zwykle jako 1 – wartość tej miary.

Aby zilustrować omawiane miary jakości, posłużymy się pewnym przykładem. Przyjmijmy, że dla danego zapytania q ranking stworzony przez model wyszukiwawczy składa się z dokumentów d_8, d_3, d_5, d_9, przy czym uzyskały one następujące binarne oceny relewancji $r(d_8) = relewantny$, $r(d_3) = nierelewantny$, $r(d_5) = relewantny$, $r(d_9) = nierelewantny$.

- **Średnia odwrotności rang, MRR** (ang. *Mean Reciprocal Rank*).

Dla danego zapytania przez rangę, $ranga_q$, rozumiemy w tym przypadku pozycję pierwszego dokumentu relewantnego w zwróconej przez model liście wynikowej. MRR jest średnią odwrotności rang dla poszczególnych zapytań.

$$MRR = \frac{1}{Q}\sum_{q}\frac{1}{ranga_q} \tag{2.2.66}$$

gdzie Q jest liczbą zapytań w zbiorze testowym.

Jak widzimy, w przypadku miary MRR, na jakość wyszukiwania ma wpływ tylko pozycja pierwszego relewantnego dokumentu w rankingu. Dokumenty znajdujące się poniżej pozycji $ranga_q$ są nieistotne.

Oczywiście w naszym przykładzie $ranga_q = 1$, więc dla tego jednego pytania MRR = 1.

- **Średnia Przeciętna Dokładność, MAP** (ang. *Mean Average Precision*).

Aby zdefiniować miarę MAP, określmy najpierw „Dokładność na pozycji *k*" (*P@k*). Załóżmy, że dysponujemy binarnymi ocenami relewancji dokumentów, tj. ocena 1 jeśli dokument jest relewantny, 0 w przeciwnym wypadku. Wówczas dla listy rankingowej π, oraz listy ocen relewancji *r*:

$$P@k(\pi,r)=\frac{\sum_{t\le k} r\left(\pi^{-1}(t)\right)}{k} \tag{2.2.67}$$

gdzie $\pi^{-1}(t)$ oznacza dokument znajdujący się w rankingu wynikowym modelu π, na pozycji t. Wartość $r(\pi^{-1}(t))$ oznacza przypisaną temu dokumentowi ocenę relewancji (0 lub 1).

Przeciętna dokładność dla listy rankingowej π, oraz listy ocen relewancji r, otrzymanych dla danego zapytania q, określona jest wówczas następująco:

$$AP(\pi,r)=\frac{\sum_{k=1}^{m} P@k\cdot r\left(\pi^{-1}(k)\right)}{m_1} \tag{2.2.68}$$

gdzie m jest liczbą dokumentów związanych z zapytaniem q, zaś m_1 jest liczbą dokumentów ocenionych na 1 (jako relewantne).

Średnia przeciętna dokładność MAP, równa jest średniej z AP, dla wszystkich zapytań.

Dla naszego przykładu widzimy, że ponieważ pierwszy dokument jest relewantny, więc *P*@1 = 1. Drugi dokument jest nierelewantny, więc *P*@2 = (1+0)/2 = 1/2. Trzeci dokument jest ponownie relewantny, więc *P*@3 = (1+0+1)/3 = 2/3. I, analogicznie *P*@4 = (1+0+1+0)/4 = 1/2. Przeciętna dokładność dla zapytania jest więc równa AP = (1 + 2/3)/2 = 5/6.

- **Zdyskontowany Skumulowany Zysk, DCG** (ang. *Discounted Cumulative Gain*)

Jest to miara oceny jakości wyszukiwania, która wykorzystuje ważone oceny relewantności dokumentów, wyrażone w formie wielu kategorii

porządkowych. Waga ma za zadanie dyskontować coraz dalszą pozycję dokumentów w rankingu.

Niech π będzie listą rankingową dla zapytania q, zaś r listą ocen relewancji dokumentów. Wówczas miara DCG dla pozycji k w rankingu, zdefiniowana jest następująco:

$$DCG_k(\pi,r)=\sum_{j=1}^{k} G(r(\pi^{-1}(j)))(j) \tag{2.2.69}$$

gdzie $G(\cdot)$ jest zyskiem-oceną dokumentu (zazwyczaj przyjmuje się $G(z) = (2^z - 1)$, gdzie z jest numerem kategorii relewancji liczonym od 0), zaś $\eta(j)$ jest współczynnikiem dyskontującym pozycję (zazwyczaj przyjmuje się $\eta(j) = 1/\log_2(j+1)$).

Dla naszego przykładu, mamy więc:

$DCG_1 = (2^1 - 1)/\log_2 2 = 1$
$DCG_2 = (2^1 - 1)/\log_2 2 + (2^0 - 1)/\log_2 3 = 1 + 0 = 1$
$DCG_3 = (2^1 - 1)/\log_2 2 + (2^0 - 1)/\log_2 3 + (2^1 - 1)/\log_2 4 = 1 + 1/2 = 1{,}5$
$DCG_4 = (2^1 - 1)/\log_2 2 + (2^0 - 1)/\log_2 3 + (2^1 - 1)/\log_2 4 + (2^0 - 1)/\log_2 5 = 1 + 1/2 = 1{,}5$

- **Znormalizowany Zdyskontowany Skumulowany Zysk, NDCG** (ang. *Normalized Discounted Cumulative Gain*)

Aby znormalizować współczynnik DCG_k, i obliczyć $NDCG_k$, musimy wyznaczyć wartość $IDCG_k$, czyli tzw. idealny współczynnik DCG, albo inaczej współczynnik DCG dla idealnego rankingu. W idealnym rankingu najbardziej relewantne dokumenty powinny znajdować się jak najbliżej początku rankingu. W celu jego otrzymania, powinniśmy więc uporządkować wszystkie dokumenty z rankingu, aż do pozycji k, zstępująco według ich ocen relewancji.

Zauważmy, że DCG otrzymane dla rankingu idealnego miałoby maksymalną możliwą wartość tego współczynnika dla wyszukanych k dokumentów, którą oznaczymy właśnie przez $IDCG_k$. Ponadto jeśli w na końcu rankingu idealnego występują dokumenty nierelewantne (z oceną 0) to w obliczeniach IDCG możemy je pominąć, ponieważ nie wpływają na wartość współczynnika.

Współczynnik $NDCG_k$, możemy wyznaczyć zgodnie z formułą.

$$NDCG_k(\pi,r)=\frac{DCG_k(\pi,r)}{IDCG_k(\pi,r)} \tag{2.2.70}$$

Oczywiście, $NDCG_k$ przyjmuje wartości od 0 do 1. Jeśli do obliczenia NDCG wykorzystamy wszystkie ocenione dokumenty (czyli m), to otrzymamy $NDCG_m$, czyli w skrócie NDCG.

Dla naszego przykładu ranking dla $k = 4$ (czyli dla m) miałby więc postać: d_8, d_5, d_3, d_9, z ocenami relewancji $r(d_8)$ = *relewantny*, $r(d_5)$ = *relewantny*, $r(d_3)$ = *nierelewantny*, $r(d_9)$ = *nierelewantny*. Dwa ostatnie dokumenty możemy więc pominąć. Otrzymujemy zatem:

$IDCG = (2^1 - 1)/\log_2 2 + (2^1 - 1)/\log_2 3 = 1 + 1/1{,}58 = 1 + 0{,}63 = 1{,}63.$
$NDCG = 1{,}5/1{,}63 = 0{,}92$

- **Korelacja rang, RC** (ang. *Rank Correlation*)

Korelacja między listą rankingową sporządzoną przez model (π) i oceny relewancji (π_r). Na przykład, kiedy zostanie użyty współczynnik τ Kendalla, korelacja rang mierzy ważoną niezgodność parami między obiema listami:

$$\tau_K(\pi,\pi_r) = \frac{\sum_{u<v} w_{uv}\left(1+\operatorname{sgn}\left(\left(\pi(u)-\pi(v)\right)\left(\pi_r(u)-\pi_r(v)\right)\right)\right)}{2\sum_{u<v} w_{uv}} \qquad (2.2.71)$$

gdzie w_{uv} jest wagą, $\pi(u)$ jest pozycją w rankingu π dokumentu d_u.

Zauważmy, że powyższy współczynnik zdefiniowany jest jednoznacznie przy założeniu, że oceny relewancji określone są w formie pełnego uporządkowania π_r. Kiedy oceny mają inną formę, stopni relewancji, czy też ocen parami, może istnieć wiele rankingów, zgodnych z tymi ocenami. Na przykład, w tym pierwszym przypadku, dla dokumentów o tej samej ocenie relewancji każde ich ustawienie jest równie poprawne.

W takiej sytuacji możemy zdefiniować współczynnik τ Kendalla jako:

$$\tau_K(\pi,\Omega_r) = \max_{\pi_r \in \Omega_r} \tau_K(\pi,\pi_r) \qquad (2.2.72)$$

gdzie Ω_r jest zbiorem dopuszczalnych rankingów relewancji.

Podsumowując właściwości przedstawionych miar oceny jakości wyszukiwania, trzeba zauważyć, że wszystkie one obliczane są na poziomie zapytania i w dalszej kolejności uśredniane dla całego zbioru zapytań. Może to czasami powodować pewne problemy w sytuacji, gdy istniejące w kolekcji zbiory dokumentów relewantnych są słabiej powiązane z niektórymi zapytaniami. Ich udział w średniej jest taki sam, jak w innych przypadkach.

Ponadto miary te zależne są od pozycji dokumentów w rankingu. Przy małych zmianach wartości funkcji rankingującej dokumentu jego pozycja w rankingu będzie pozostawała stała, a dopiero po osiągnięciu pewnego poziomu wynik dokumentu przeskakiwać będzie wynik innych. Zmiany pozycji w rankingu, a co za tym idzie zmiany wartości miar oceny systemu, mają więc charakter nieciągły i nieróżniczkowalny względem wartości funkcji rankingującej i w konsekwencji względem jej parametrów.

Patrząc z punktu widzenia modeli LTR, bezpośrednie zastosowanie miar oceny systemu wyszukiwawczego do uczenia funkcji rankingującej jest więc raczej problematyczne. Zamiast nich, w procesie uczenia, niezbędne jest zastosowanie innych funkcji celu, dających wyniki zgodne z przedstawionymi wyżej miarami testowymi. W dalszej części bieżącego podrozdziału przyjrzymy się bliżej niektórym rozwiązaniom wykorzystywanym w modelach uczenia funkcji rankingującej, z trzech wymienionych wcześniej grup: podejście punktowe, oparte na parach oraz na listach.

2.2.3.2. Podejście punktowe

W podejściu punktowym, rankingowanie odbywa się na poziomie dokumentu, podobnie jak w tradycyjnych, omawianych w poprzednich podrozdziałach, systemach wyszukiwawczych. Tak więc, ranking wynikowy dla zapytania, tworzony jest na podstawie oceny dopasowania do niego każdego dokumentu w kolekcji.

Przestrzeń wejściowa modelu składa się z wielowymiarowych wektorów, złożonych z połączonych cech zapytania i dokumentu. Wyjściem modelu jest stopień relewancji dokumentu dla zapytania lub wartości pewnej miary, która jest do niego proporcjonalna.

Zasady tworzenia zbioru treningowego dla punktowych systemów LTR, nie różnią się specjalnie od wymienianych w poprzednim punkcie, dla zbiorów testowych. Tworzony jest pewien zbiór zapytań treningowych. Dla każdego zapytania treningowego q określany jest zestaw dokumentów treningowych, dla których pozyskiwane są oceny relewancji. Pary treningowe $(\mathbf{x}_{qd}, r_q(d))$ tworzone są przez wzorzec wejściowy cech każdego zapytania q i dokumentu d, $\mathbf{x}_{qd}$ oraz odpowiadającą temu dokumentowi ocenę relewancji dla zapytania $r_q(d)$.

Możliwe są przy tym trzy sytuacje [Liu, 2011]:

- Ocena relewancji $r_q(d)$ może być bezpośrednio pozyskiwana od człowieka, jako stopień relewancji (binarny, w formie kategorii porządkowych lub rzeczywisty, czyli wartość z przedziału [0, 1]).

- Jeśli ocena relewancji dokonywana jest jako preferencja w parach $r_q(d_i, d_j)$, można na jej podstawie wyznaczyć stopień relewancji dokumentu, obliczając częstość jego zwycięstw nad innymi dokumentami.
- Jeśli ocena relewancji ma formę pełnej uporządkowanej listy π_r, możemy wyznaczyć stopień relewancji dokumentu wykorzystując funkcję przeliczającą. Na przykład, jako stopień relewancji dokumentu może zostać wykorzystana jego względna pozycja w rankingu.

Podejście punktowe ma więc charakter predykcyjny. Na podstawie zbioru par treningowych $T = \{(\mathbf{x}_{qd}, r_q(d))\}$ staramy się zbudować pewne odwzorowanie f, służące do predykcji stopnia relewancji nowego dokumentu dla nowego zapytania. Zadanie LTR, uczenia się rankingowania, polega więc w tym przypadku na optymalizacji problemu [Onal, Zhang i in., 2018]:

$$\arg\min_{f \in F} \sum_{(\mathbf{x}_{qd}, r_q(d)) \in T} L\Big(\big(f\big(\mathbf{x}_{qd}\big), r_q(d)\big)\Big) \tag{2.2.73}$$

gdzie F jest zbiorem rozważanych funkcji, zaś $L(\cdot)$ funkcją kosztu lub błędu między rzeczywistą wartością stopnia relewancji dokumentu d dla zapytania q a jej predykcją otrzymaną przy użyciu funkcji f.

Procedura szukania odwzorowania f polega na ogół na estymacji parametrycznej, czyli znalezieniu zestawu jego parametrów minimalizującego funkcję kosztu (2.2.73). Forma i stopień złożoności odwzorowania określany jest przez przyjęty zbiór F i może się zmieniać od prostych funkcji liniowych, poprzez warstwowe jednokierunkowe sieci neuronowe, aż do rozbudowanych architektur deep-learningu. W zależności od formy ocen relewancji (wartości rzeczywiste, binarne lub w formie kategorii porządkowych) oraz przyjętej metody modelowania, możemy mówić tu o trzech rodzajach podejść: regresyjnym, klasyfikacyjnym oraz opartym na regresji porządkowej (w której wartości zmiennej zależnej mierzone są w skali porządkowej).

W podejściu opartym na regresji zakładamy, że oceny relewancji przypisane dokumentów w odniesieniu do zapytań mają charakter ciągłych wartości liczbowych z przedziału [0, 1], gdzie 0 oznacza dokument nierelewantny, 1 zaś relewantny. Funkcja kosztu ma charakter funkcji kwadratowej:

$$L\Big(f\big(\mathbf{x}_{qd}\big), r_q(d)\Big) = \Big(r_q(d) - f\big(\mathbf{x}_{qd}\big)\Big)^2 \tag{2.2.74}$$

Co oznacza, że poszukujemy odwzorowania rankingującego f metodą najmniejszych kwadratów.

Modele regresyjne należały do najwcześniejszych zastosowań LTR. Jedną z pierwszych prac z tego zakresu w dziedzinie wyszukiwania informacji, były badania Fuhra, związane z wykorzystaniem modeli regresji wielomianowej [Fuhr, 1989]. Dokładną analizę wykorzystania kwadratowej funkcji kosztu do budowy rankingu relewancji przeprowadzili Cossock, Zhang [Cossock, Zhang, 2006]. Badali oni teoretyczne właściwości metody najmniejszych kwadratów dla liniowych ważonych modeli regresji, w kontekście przedstawionych w poprzednim podpunkcie miar oceny jakości rankingu. W pracy udowodniono, że kwadratowa funkcja kosztu nakłada górne ograniczenie na błąd rankingowania, oparty na NDCG. Oznacza to, że jeśli błąd kwadratowy modelu dąży do zera, to również błąd NDCG rankingów tworzonych z wykorzystaniem funkcji f dąży do zera. Kwadratowa funkcja celu uczenia modelu daje więc wyniki zgodne z NDCG [Cossock, Zhang, 2006; Liu, 2011].

Należy jednak być świadomym, że stosowanie kwadratowej funkcji celu uczenia modelu LTR, może powodować pojawienie się pewnych problemów. Na przykład, związanych z jej symetrycznym charakterem. W parach treningowych występować będzie wiele dokumentów relewantnych dla zapytania, dla których $r_q(d) = 1$. Zupełnie błędna ocena takiego dokumentu przez system, jako nierelewantny, czyli $f(\mathbf{x}_{qd}) = 0$, daje w wyniku błąd kwadratowy o wartości 1. Problem polega na tym, że ocena dokumentu przez system $f(\mathbf{x}_{qd}) = 2$, również daje w wyniku błąd kwadratowy o wartości 1, mimo że z punktu widzenia celu naszego uczenia jest ona może raczej „hiperpoprawna" niż błędna.

Poprzedni problem, można oczywiście rozwiązać stosując postać funkcjonalną modelu, która daje znormalizowane wartości wyjściowe z przedziału od 0 do 1, przykład ten jednak powinien nam uświadomić pewną ważną kwestię, związaną nie tylko z kwadratową funkcją kosztu modelu, ale generalnie z podejściem punktowym do LTR. Otóż funkcja kwadratowa tylko w pewnym przybliżeniu jest zgodna z celami uczenia modelu LTR. Celem metody najmniejszych kwadratów jest minimalizacja błędu oceny dokumentu, celem uczenia modelu LTR jest uzyskanie jak najlepszego rankingu dokumentów. Brzmi to nieco paradoksalnie, ale w pewnych przypadkach metoda regresji może być za dokładna. Łatwo wyobrazić sobie sytuacje w których dalsze poprawianie dokładności oceny dokumentu nie wnosi kompletnie nic do jakości rankingu, ponieważ nie powoduje przestawień w nim dokumentów. I z drugiej strony, w wielu przypadkach nawet przy dosyć dużym poziomie błędu kwadratowego oceny dokumentu przez model, ranking może być optymalny, ponieważ względne pozycje dokumentów w liście, otrzymane na podstawie tej oceny, dokładnie odpowiadają pozycjom wynikającym z treningowych ocen relewancji.

Podsumowując nasze rozważania, kwadratowa funkcja kosztu uczenia modelu pozwala na realizację celów LTR i uzyskiwanie modeli rankingujących o wysokiej jakości w sensie miary NDCG. Koszt kwadratowy stanowi bowiem ograniczenie z góry na błąd oparty na tej mierze i jego redukcja powoduje zwiększanie optymalności rankingu. W wielu zadaniach jednak stosowanie modeli regresji niekoniecznie musi być najlepszym rozwiązaniem. Błąd kwadratowy oceny dokumentów może bowiem dawać dosyć luźne ograniczenie na błąd rankingu oparty na mierze NDCG. Tak jak powiedzieliśmy, problem ten ma bardziej generalny charakter i dotyczy w zasadzie ogólnie rozwiązań LTR wykorzystujących podejście punktowe, w którym oceniamy relewancję dokumentu względem zapytania, w pewnym oderwaniu od tworzonego dalej rankingu wynikowego.

W przypadku gdy oceny relewancji dokumentów mają charakter kategorii jakościowych, zagadnienie uczenia się funkcji oceny dokumentu przyjmuje charakter zadania klasyfikacji. Przyjrzymy się więc obecnie wykorzystaniu w procesie rankingowania kilku metod uczenia maszynowego, należących do tej kategorii.

Rozpoczniemy od wykorzystania w zagadnieniu LTR binarnych ocen relewancji, czyli klasyfikacji dokumentów na dwie kategorie: relewantny w odniesieniu do zapytania i nierelewantny. Istnieje oczywiście szereg klasyfikatorów należących do tej grupy, z których możemy skorzystać w zadaniu uczenia funkcji rankingującej. Jedną z tradycyjnych metod tworzenia klasyfikatorów binarnych jest regresja logistyczna.

Podstawowa forma zastosowania tej metody do identyfikacji funkcji rankingującej polega na wykorzystaniu zbioru par treningowych $T=\{(\mathbf{x}_{qd}, r_q(d))\}$, do bezpośredniego dopasowania zależności logistycznej [Ailon, 2009]. Podobnie jak poprzednio $\mathbf{x}_{qd}$ jest wzorcem cech pozyskanych z danego dokumentu d i zapytania q, zaś $r_q(d)$ binarną oceną relewancji tego dokumentu dla tego zapytania ($r_q(d) = 1$ jeśli jest on relewantny, $r_q(d) = 0$ jeśli nierelewantny). Biorąc pod uwagę cele uczenia i właściwości funkcji logistycznej, możemy przyjąć, że w ten sposób modelujemy prawdopodobieństwo relewancji dokumentu, przy danej reprezentacji dokumentu i zapytania:

$$P\left(R \mid \mathbf{x}_{qd}\right) = \frac{1}{1+e^{-\left(w_0+\sum w_i x_{qd}(i)\right)}} \tag{2.2.75}$$

W celu oszacowania parametrów regresji w_0, w_1, …, w_p, gdzie p jest liczbą cech wejściowych (wymiarem wektora $\mathbf{x}_{qd}$), możemy zlinearyzować model (2.2.75):

$$P\left(R\,|\,\mathbf{x}_{qd}\right)\left(1+e^{-w_0-\sum w_i x_{qd}(i)}\right)=1$$

$$P\left(R\,|\,\mathbf{x}_{qd}\right)e^{-w_0-\sum w_i x_{qd}(i)}=1-P\left(R\,|\,\mathbf{x}_{qd}\right)$$

$$\frac{P\left(R\,|\,\mathbf{x}_{qd}\right)}{1-P\left(R\,|\,\mathbf{x}_{qd}\right)}=e^{w_0+\sum w_i x_{qd}(i)}$$

$$\ln\left(\frac{P\left(R\,|\,\mathbf{x}_{qd}\right)}{1-P\left(R\,|\,\mathbf{x}_{qd}\right)}\right)=w_0+\sum w_i x_{qd}(i) \qquad (2.2.76)$$

Parametry modelu (2.2.76) można już oczywiście łatwo wyznaczyć liniową metodą najmniejszych kwadratów, korzystając ze zbioru treningowego $T=\{(\mathbf{x}_{qd}, r_q(d))\}$. Zwróćmy uwagę, że wielkość występująca po lewej stronie zależności (2.2.76) to logarytm szansy relewancji dokumentu przy danym wzorcu cech jego oraz zapytania, $\ln S(R \mid \mathbf{x}_{qd})$.

Dla nowego zapytania, nie występującego w zbiorze treningowym T, jeśli będziemy chcieli zbudować ranking dokumentów, to: korzystając z oszacowanych wcześniej parametrów $w_0, w_1, \ldots, w_p$, przy pomocy zależności (2.2.76) obliczamy logarytm szansy relewancji dokumentów $\ln S(R \mid \mathbf{x}_{qd})$, a następnie na podstawie (2.2.75), wyznaczamy prawdopodobieństwo relewancji $P(R \mid \mathbf{x}_{qd})$. Funkcja oceny dokumentów, stanowiąca podstawę tworzenia rankingu, może także bezpośrednio opierać się na logarytmie szansy relewantności, $\ln S(R \mid \mathbf{x}_{qd})$. Szansa jest proporcjonalna do prawdopodobieństwa $P(R \mid \mathbf{x}_{qd})$ i pozycje dokumentów w rankingach korzystających z obydwu tych miar są takie same.

Nieco odmienne podejście do zastosowania regresji logistycznej w procesie uczenia funkcji rankingującej zaproponowali Cooper i Gey [Cooper, Gey, Dabney, 1992; Gey, 1994]. Na podstawie ocen relewancji dokumentów dla zbioru zapytań treningowych sporządzili zbiór treningowy do uczenia relewancji na poziomie poszczególnych termów w kolekcji. To jest, dla każdego zapytania q, dokumentu d oraz termu t, pozyskiwany był wzorzec cech wejściowych $\mathbf{x}_{qdt}$, któremu w parze treningowej przypisywana była relewancja dokumentu dla zapytania $r_q(d)$. Jako cechy wejściowe $\mathbf{x}_{qdt}$, wykorzystywane były takie statystyki jak logarytm liczby wystąpień termu t w dokumencie d, liczby jego wystąpień w zapytaniu q, logarytm idf, czyli odwrotności liczby dokumentów w których term t występuje w kolekcji, długość opisu zapytania, długość opisu dokumentu

itp. Liczba i szczegółowy zestaw zmiennych (cech) wejściowych zależny był od konkretnej kolekcji dla której tworzono model.

Następnie korzystając ze zbioru treningowego $T = \{(\mathbf{x}_{qdt}, r_q(d))\}$, szacowano parametry modelu regresji logistycznej dla szansy relewancji dokumentu przy danym zapytaniu, dokumencie i termie.

$$\ln S\left(R \mid \mathbf{x}_{qdt}\right) = w_0 + w_1 x_{qdt}(1) + \ldots + w_p x_{qdt}(p) \tag{2.2.77}$$

Dla nowego zapytania i dokumentu, przy pomocy modelu (2.2.77) wyznaczano $\ln S(R \mid \mathbf{x}_{qdt})$ dla wszystkich termów występujących w zapytaniu, a następnie scalano je w $\ln S(R \mid \mathbf{x}_{qd})$, korzystając ze wzoru:

$$\ln S\left(R \mid \mathbf{x}_{qd}\right) = \sum_{t \in q} \left(\ln S\left(R \mid \mathbf{x}_{qdt}\right) - \ln S(R)\right) \tag{2.2.78}$$

Wartość $\ln S(R)$ jest stałą, obliczaną dla danej kolekcji.

Jako funkcję rankingującą wykorzystywano $P(R \mid \mathbf{x}_{qd})$, obliczane na podstawie $\ln S(R \mid \mathbf{x}_{qd})$ przy pomocy zależności (2.2.75).

Regresja logistyczna jest przykładem tradycyjnej metody klasyfikacyjnej opartej na metodzie najmniejszych kwadratów. Często stosowanym w zadaniu uczenia funkcji rankingującej klasyfikatorem, opartym na nieco odmiennej zasadzie, są maszyny wektorów wspierających, SVM (ang. *Support Vector Machines*). Przykład wykorzystania SVM do binarnej klasyfikacji relewancji dokumentów przedstawia Nallapati [Nallapati, 2004; Liu, 2011].

W swej podstawowej formie, maszyny wektorów wspierających są klasyfikatorami liniowymi, tzn. funkcja dyskryminacyjna modelu ma charakter liniowy:

$$g\left(R \mid \mathbf{x}_{qdt}\right) = w_1 x_{qd}(1) + \ldots + w_p x_{qd}(p) = \mathbf{w}^T \mathbf{x}_{qd} \tag{2.2.79}$$

Klasyfikacja nowych wzorców odbywa się zgodnie z zasadą: je**żeli** dla nowego zapytania q i dokumentu d, wartość funkcji dyskryminacyjnej $g(R \mid \mathbf{x}_{qd}) > 0$, to przyjmujemy że dokument ten jest relewantny dla zapytania, w przypadku gdy $g(R \mid \mathbf{x}_{qd}) < 0$, przyjmujemy że jest on nierelewantny.

Jeśli spojrzymy na równania (2.2.79) oraz (2.2.76), to łatwo zauważyć, że zasadniczo pod względem funkcji modelu klasyfikator SVM nie różni się specjalnie od przedstawionego wyżej podejścia opartego na regresji logistycznej. Różnica polega na sposobie zdefiniowania hiperpłaszczyzny dyskryminacyjnej dla obu modeli, która w przypadku SVM posiada lepsze właściwości.

Zauważmy bowiem, że równanie liniowe $g(R \mid \mathbf{x}_{qd}) = 0$ definiuje pewną hiperpłaszczyznę w przestrzeni wejść $\mathbf{x}_{qd}$, rozdzielającą pozytywne (relewantne) i negatywne (nierelewantne) przykłady w zbiorze treningowym. W przestrzeniach o dużych wymiarach (a taką zwykle jest przestrzeń cech zapytań i dokumentów), przy skończonej liczbie przykładów w zbiorze treningowym, istnieć może wiele różnych hiperpłaszczyzn spełniających warunek separacji obu podzbiorów.

W przypadku regresji logistycznej parametry $w_1, \ldots, w_p$ funkcji modelu (gdzie p jest liczbą cech dokumentów/zapytań, wymiarem przestrzeni wejściowej), dobierane są tak, by hiperpłaszczyzna rozdzielająca minimalizowała błąd oceny szansy relewancji dokumentu dla zapytania. Rozwiązaniem będą parametry definiujące jedną z hiperpłaszczyzn rozdzielających dokumenty relewantne od nierelewantnych. Dla SVM warunek wyboru płaszczyzny dyskryminacyjnej jest silniejszy. Otóż naszym celem jest znalezienie płaszczyzny rozdzielającej, maksymalizującej margines klasyfikacji (czyli jak najbardziej odległej zarówno od zbioru relewantnych przykładów treningowych, jak i od zbioru nierelewantnych).

Przyjmijmy wobec tego, jak wcześniej, że dysponujemy zbiorem przykładów treningowych $T = \{(\mathbf{x}_{qd}, r_q(d))\}$, gdzie $\mathbf{x}_{qd}$ jest p-wymiarowym wzorcem cech dokumentu d i zapytania q, zaś $r_q(d)$ jest oceną jego relewancji (tzn. $r_q(d) = 1$ dla przykładów relewantnych, $r_q(d) = 0$ dla nierelewantnych). Aby osiągnąć efekt maksymalizacji marginesu klasyfikacji, obok parametrów funkcji modelu $w_1, \ldots, w_p$ każdemu przykładowi treningowemu $(\mathbf{x}_{qd}, r_q(d))$ przyporządkowujemy dodatkowy parametr ξ_{qd} (jest to właśnie ten występujący w nazwie typu modelu wektor wspierający). Będziemy przy tym chcieli, aby dla modelu spełnione były warunki:

- Parametry ξ_{qd} miały wartości nieujemne:

$$\xi_{qd} \geq 0 \tag{2.2.80}$$

- Dla treningowych przykładów pozytywnych, dla których $r_q(d) = 1$, wartość funkcji modelu (2.2.79) była nie tylko dodatnia, ale nie spadała poniżej poziomu $1 - \xi_{qd}$, tzn.:

$$\mathbf{w}^T \mathbf{x}_{qd} \geq 1 - \xi_{qd} \tag{2.2.81}$$

- Dla treningowych przykładów negatywnych, dla których $r_q(d) = 0$, wartość funkcji modelu (2.2.79) była nie tylko ujemna, ale nie większa niż $-1 + \xi_{qd}$:

$$\mathbf{w}^T \mathbf{x}_{qd} \leq -1 + \xi_{qd} \tag{2.2.82}$$

Będziemy chcieli znaleźć takie parametry $\mathbf{w} = (w_1, \ldots, w_p)$ oraz ξ_{qd}, aby wartości funkcji modelu były jak najbardziej „rozpychane" do 1 dla przykładów relewantnych oraz do –1 dla przykładów nierelewantnych. W tym celu wartości ξ_{qd} muszą być jak najmniejsze. Ponadto margines klasyfikacji (między –1 a 1) wynosi 2, co w przestrzeni wejściowej odpowiada marginesowi cech dokumentów/zapytań, równemu $2/||\mathbf{w}||$. Dlatego będziemy również minimalizować $||\mathbf{w}||$, co skutkuje maksymalizacją marginesu. Aby określić wartości parametrów, musimy więc znaleźć minimum następującej funkcji:

$$\arg\min\left(\frac{1}{2}\mathbf{w}^2 + \lambda \sum_{(\mathbf{x}_{qd}, r_q(d))\in T} \xi_{qd}\right) \tag{2.2.83}$$

Znalezienie parametrów $\mathbf{w} = (w_1, \ldots, w_p)$ oraz ξ_{qd}, wymaga więc rozwiązania problemu optymalizacyjnego, z funkcją celu (2.2.83) i ograniczeniami (2.2.80) do (2.2.82).

Przejdźmy do sytuacji, gdy oceny dokumentów definiowane są w formie kilku kategorii porządkowych, stopniujących poziom ich relewancji dla zapytania (takich jak np. *Idealny*, *Doskonały*, *Dobry*, *Niezły*, *Zły*). W tym przypadku w zadaniu LTR musimy skorzystać z klasyfikatora wieloklasowego, w którym kategorie wyjściowe odpowiadać poszczególnym poziomom oceny stopnia relewancji.

Zakładamy więc, że klasyfikator dla danego wzorca wejściowego $\mathbf{x}_{qd}$, cech dokumentu d i zapytania q ma za zadanie przypisać mu numer odpowiedniej kategorii stopnia relewancji ze zbioru $\{1, 2, \ldots, K\}$. W naszym przykładzie $K = 5$, i kategoria numer 1 odpowiada poziomowi *Idealny*, kategoria numer 2, poziomowi *Doskonały*, numer 3 – *Dobry* itd. Otrzymane wyniki klasyfikacji wykorzystywane są do budowy rankingu dokumentów dla zapytania: dokumenty zakwalifikowane do kategorii 0 umieszczane są na początku rankingu, następnie te należące do kategorii 1, 2 itd. Kolejność uporządkowania dokumentów w obrębie kategorii jest dowolna.

Podobnie jak w przypadku regresji, można wskazać [Li, Burges, Wu, 2008; Mitra, Craswell, 2017, 2018; Zhang, Rahman i in., 2017] na pewną niezgodność celów uczenia stosowanych przy tworzeniu klasyfikatorów z potrzebami rankingowania dokumentów. Celem uczenia klasyfikatorów jest uzyskanie jak najdokładniejszej predykcji kategorii wyjściowej dla danego wzorca danych wejściowych.

Błąd klasyfikacji, w sensie liczby błędnie sklasyfikowanych wzorców, jest oczywiście zgodny z miarami błędu rankingowania, opartymi

na NDCG, ponieważ doskonale dokładna klasyfikacja daje w wyniku idealnie dokładny raport. Natomiast łatwo podać przykłady sytuacji, w których nawet duży błąd klasyfikacji może dać w wyniku doskonale idealny raport. Na przykład, jeśli dla danej kolekcji dokumentów wszystkie predykcje kategorii relewancji będą konsekwentnie przesunięte o jedną pozycję, to błąd klasyfikacji będzie równy 100%, a utworzony na jej podstawie ranking będzie doskonale poprawny [Li, Burges, Wu, 2008].

Li, Burges, Wu [Li, Burges, Wu, 2008; Liu, 2011] przedstawili sposób interpretacji wyjścia klasyfikatora wieloklasowego, tak by wyniki jego działania były lepiej dostosowane do potrzeb rankingowania dokumentów, oparty na oszacowaniu wartości oczekiwanej stopnia relewancji dokumentu dla zapytania.

Przyjmijmy jak poprzednio, że klasyfikator tworzony jest na podstawie zbioru treningowego $T = \{(\mathbf{x}_{qd}, r_q(d))\}$, gdzie $\mathbf{x}_{qd}$ jest wzorcem cech dokumentu d i zapytania q, $r_q(d)$ oceną relewancji tego dokumentu dla zapytania, przy czym $r_q(d) \in \{1, \ldots, K\}$, czyli jest jedną z kategorii porządkowych stopnia relewancji dokumentu d dla zapytania q. Sam klasyfikator ma postać odwzorowania $f(\mathbf{x}_{qd}, \mathbf{w})$, gdzie $\mathbf{w}$ jest zestawem parametrów, szacowanych w procesie uczenia.

Błąd klasyfikacji, w sensie liczby błędnych klasyfikacji, możemy wówczas zapisać:

$$\sum_{(\mathbf{x}_{qd}, r_q(d)) \in T} I_{f(\mathbf{x}_{qd}, \mathbf{w}) \neq r_q(d)} \tag{2.2.84}$$

gdzie funkcja I jest równa 1 gdy $f(\mathbf{x}_{qd}, \mathbf{w}) \neq r_q(d)$, 0 w przeciwnym przypadku.

W praktyce błąd (2.2.84) jest raczej stosowany do oceny modelu, zaś do jego uczenia wykorzystuje się którąś z zastępczych funkcji celu, opartą na koszcie kwadratowym, wykładniczym lub innym. Li, Burges i Wu proponują użycie jako funkcji błędu jednej z często stosowanych w zagadnieniach klasyfikacji tekstów funkcji kosztu, opartej na entropii względnej [Li, Burges, 2008; Mitra, Craswell, 2018; Zhang, Rahman i in., 2017]:

$$\sum_{(\mathbf{x}_{qd}, r_q(d)) \in T} \sum_{k=1}^{K} -\log P\Big(f(\mathbf{x}_{qd}, \mathbf{w}) = k\Big) I_{r_q(d)=k} \tag{2.2.85}$$

gdzie $P(f(\mathbf{x}_{qd}, \mathbf{w}) = k)$ oznacza prawdopodobieństwo wskazania przez klasyfikator kategorii wyjściowej k, przy danym wzorcu wejściowym $\mathbf{x}_{qd}$.

Oznaczmy teraz przez $f_k(\mathbf{x}_{qd}, \mathbf{w})$ wyjście modelu klasyfikacyjnego, odpowiadające k-tej kategorii. Może to być na przykład wartość funkcji dyskryminacyjnej modelu związanej z tą kategorią albo wartość neuronu wyjściowego warstwowej sieci neuronowej, odpowiadająca tej kategorii. Na podstawie wszystkich wartości wyjściowych klasyfikatora prawdopodobieństwo $P(f(\mathbf{x}_{qd}, \mathbf{w}) = k)$ możemy oszacować, korzystając z funkcji softmax:

$$P\left(f(\mathbf{x}_{qd}, \mathbf{w}) = k\right) = \frac{e^{f_k(\mathbf{x}_{qd}, \mathbf{w})}}{\sum_{s=0}^{K-1} e^{f_s(\mathbf{x}_{qd}, \mathbf{w})}} \tag{2.2.86}$$

Podczas tworzenia rankingu dla nowego zapytania i dokumentu, wyjścia klasyfikatora przekształcane są na wartości funkcji rankingującej przy pomocy następującej zależności wyznaczającej wartość oczekiwaną poziomu relewancji dokumentu dla zapytania:

$$rank\left(\mathbf{x}_{qd}\right) = \sum_{k=1}^{K} g(k) P\left(f\left(\mathbf{x}_{qd}, \mathbf{w}\right) = k\right) \tag{2.2.87}$$

gdzie $g(\cdot)$ jest dowolną monotoniczną (rosnącą) funkcją, przyporządkowującą wartości liczbowe kategoriom porządkowym stopnia relewancji. Jeśli stopień relewancji dokumentu wzrasta liniowo wraz w wyborem kolejnych kategorii porządkowych wykorzystywanych do jego oceny, na ogół przyjmuje się $g(k) = k - 1$.

Metody klasyfikacyjne nie biorą pod uwagę porządkowego charakteru kategorii ocen relewancji. Zakładają one po prostu, że wyjściem modelu jest pewna wybrana kategoria poziomu relewancji, do której kwalifikujemy wzorzec wejściowy danego dokumentu dla konkretnego zapytania. W stosowanych do tworzenia klasyfikatorów metodach nie ma znaczenia, czy klasy wyjściowe mają charakter porządkowy czy nominalny.

Powstaje więc pytanie, czy w jakiś sposób nie możemy wykorzystać informacji o relacji porządku, występującej w ocenach relewancji. Prowadzi nas to do trzeciej z typowych grup metod wykorzystywanych do tworzenia modeli LTR przy punktowych ocenach relewancji.

Jedną z bardziej znanych metod należących do tej grupy, jest algorytm PRank, wykorzystujący liniowy model regresji porządkowej, uczony przy pomocy reguły perceptronowej delta, zaproponowany przez Crammera i Singera [Crammer, Singer, 2002; Harrington, 2003; Liu, 2011].

Przyjmijmy ponownie, że do budowy modelu wykorzystujemy zbiór par treningowych $T = \{(\mathbf{x}_{qd}, r_q(d))\}$, gdzie $\mathbf{x}_{qd}$ jest wzorcem cech dokumentu

d i zapytania q, $r_q(d) \in \{1, \ldots, K\}$, oceną relewancji tego dokumentu dla zapytania, w formie kategorii porządkowych stopnia relewancji. Ponieważ kategorie te mają charakter porządkowy, możemy je reprezentować w rzeczywistej przestrzeni ciągłej oceny relewancji, jako pewien jej podział na podprzedziały, odpowiadające poszczególnym kategoriom. Przyjmijmy, że r-tą klasę relewancji reprezentuje przedział $[b_{r-1}, b_r]$, zaś cały podział przestrzeni określony jest przez K elementowy zbiór wartości progowych krańców przedziałów $b_1 \leq \ldots \leq b_{K-1} \leq b_K = \infty$.

Dla każdego nowego wzorca cech dokumentu i zapytania $\mathbf{x}$, wyznaczana jest wartość liniowej funkcji oceny $\mathbf{w}^T\mathbf{x}$ (wektor $\mathbf{w}$ jest wektorem parametrów modelu), przy czym wzorcowi temu przydzielana jest kategoria relewancji r, taka, że $b_{r-1} < \mathbf{w}^T\mathbf{x} < b_r$. Aby określić kategorię relewancji danego wzorca wejściowego $\mathbf{x}$, należy więc znaleźć najmniejszą wartość b_r, taką, że $\mathbf{w}^T\mathbf{x} < b_r$. Funkcja rankingująca przyjmuje więc postać:

$$\begin{aligned} f(\mathbf{x}, \mathbf{w}, \mathbf{b}) &= \min_{r \in \{1,\ldots,K\}} \{r : \mathbf{w}^T\mathbf{x} < b_r\} = \\ &= \min_{r \in \{1,\ldots,K\}} \{r : \mathbf{w}^T\mathbf{x} - b_r < 0\} \end{aligned} \tag{2.2.88}$$

Parametry modelu $\mathbf{w}$ (parametry funkcji oceniającej), $\mathbf{b}$ (granice przedziałów oceny odpowiadające poziomom relewancji), uczone są przy pomocy reguły uczenia perceptronu (stąd nazwa algorytmu PRank – ang. *Perceptron Ranking*) [Crammer, Singer, 2002]. Dla każdej pary treningowej $(\mathbf{x}_{qd}, r_q(d))$ przy pomocy zależności (2.2.88) wyznaczana jest prognoza kategorii relewancji $r_{qd} = f(\mathbf{x}_{qd}, \mathbf{w}, \mathbf{b})$. Jeśli $r_{qd} \neq r_q(d)$, występuje błąd klasyfikacji i dokonywana jest korekta parametrów $\mathbf{w}$ oraz $\mathbf{b}$, tak by przesunąć wartości $\mathbf{w}^T\mathbf{x}_{qd}$ oraz b_r (dla kategorii r dla których występował błąd) bliżej do siebie. Dokładniej mówiąc, wyznaczany jest błąd klasyfikacji (liczba kategorii niedoszacowania lub przeszacowania modelu) $r_q(d) - r_{qd}$, a następnie:

- parametry funkcji liniowej zmieniane są proporcjonalnie do wielkości błędu zgodnie z formułą $\mathbf{w} = \mathbf{w} + (r_q(d) - r_{qd})\,\mathbf{x}_{qd}$.
- wszystkie progi b_r na pozycjach, na których wystąpił błąd (czyli między r_{qd} i $r_q(d)$ lub $r_q(d)$ i r_{qd}, w zależności od kierunku błędu) zmieniane są o 1 w kierunku przeciwnym do znaku błędu): $b_r = b_r - \mathrm{sgn}(r_q(d) - r_{qd})$.

Na przykład, przyjmijmy, że poprawna relewancja dla danej pary treningowej wynosi $r_q(d) = 4$, czyli wartość $\mathbf{w}^T\mathbf{x}_{qd}$ powinna wpadać w przedział między b_3 a b_4, zaś prognoza systemu dla tego wzorca była równa tylko $r_{qd} = 1$ (a więc $\mathbf{w}^T\mathbf{x}_{qd}$ jest mniejsze od b_1). Wyjście systemu jest więc niedoszacowane, błąd wynosi $r_q(d) - r_{qd} = 4 - 1 = 3$. Z jednej strony więc będziemy

się starali zwiększyć wartość funkcji liniowej, z drugiej obniżyć krańce przedziałów dla błędnie ocenionych kategorii.

Zwiększamy więc wartość parametrów funkcji $\mathbf{w} = \mathbf{w} + 3\mathbf{x}_{qd}$. W ten sposób wartość funkcji liniowej zwiększy się, ponieważ $(\mathbf{w} + 3\mathbf{x}_{qd})^{\mathrm{T}}\mathbf{x}_{qd} = \mathbf{w}^{\mathrm{T}}\mathbf{x}_{qd} + 3(\mathbf{x}_{qd})^{\mathrm{T}}\mathbf{x}_{qd} = \mathbf{w}^{\mathrm{T}}\mathbf{x}_{qd} + 3||\mathbf{x}_{qd}||^2$. Po zmianie podnosimy więc ocenę relewancji dokumentu o $3||\mathbf{x}_{qd}||^2$. Zwróćmy uwagę, że gdyby model przeszacowywał, błąd $r_q(d) - r_{qd}$ byłby ujemny, czyli po korekcie parametrów, wartość funkcji liniowej zostałaby zmniejszona.

Po drugie, ponieważ błąd jest dodatni, obniżymy progi błędnych kategorii $b_1 = b_1 - 1$, $b_2 = b_2 - 1$, $b_3 = b_3 - 1$, jeszcze zwiększając możliwość poprawnej klasyfikacji, poprzez przesunięcie w dół przedziałów dla niedoszacowanych przez system kategorii relewancji. Oczywiście, gdyby system przeszacowywał, znak błędu byłby ujemny i krańce odpowiednich przedziałów zostałyby podniesione.

Inne podejście do uczenia rankingowania z zastosowaniem regresji porządkowej przedstawili Shashua i Levin [Shashua, Levin, 2003]. Zaproponowali oni do uczenia parametrów liniowej funkcji oceniającej oraz krańców przedziałów reprezentujących poszczególne kategorie relewancji dwa rozwiązania oparte na maksymalizacji marginesu i maszynach wektorów wspierających SVM.

Podobnie więc, jak w przypadku algorytmu PRank, celem jest znalezienie zestawu krańców przedziałów (wartości progowych) $b_1, \ldots, b_{K-1}$, przy czym $b_1 \leq \ldots \leq b_{K-1} \leq b_K = \infty$, oraz parametrów funkcji liniowej $\mathbf{w}$, dzielących przestrzeń wejściową cech obiektów/zapytań na obszary odpowiadające poszczególnym kategoriom poziomu relewancji, czyli spełniających warunek (2.2.88). Przypomnijmy, że warunek ten oznacza, iż wszystkim wektorom wejściowym $\mathbf{x}$, spełniającym nierówność $b_{r-1} < \mathbf{w}^{\mathrm{T}}\mathbf{x} < b_r$, przypisuje się ocenę relewancji odpowiadającą kategorii o numerze r. Zwróćmy uwagę, że poszczególne równania $\mathbf{w}^{\mathrm{T}}\mathbf{x} = b_r$, $r = 1, \ldots, K-1$, definiują w przestrzeni wejść hiperpłaszczyzny, tak więc zagadnienie tworzenia modelu regresji porządkowej i zestawu parametrów $\mathbf{b}, \mathbf{w}$ modelu, możemy przedstawić również jako zadanie znalezienia rodziny K-1 równoległych hiperpłaszczyzn, separujących kolejne poziomy porządkowe relewancji.

Do uczenia modelu wykorzystujemy zbiór par treningowych $T = \{(\mathbf{x}_{qd}, r_q(d))\}$, gdzie $\mathbf{x}_{qd}$ jest wzorcem cech dokumentu d i zapytania q, $r_q(d) \in \{1, \ldots, K\}$ oceną relewancji tego dokumentu, jedną z kategorii porządkowych stopnia relewancji.

Jak wspomnieliśmy, Shashua i Levin [Shashua, Levin, 2003] zaproponowali dwa możliwe podejścia do uczenia parametrów modelu metodą

maksymalizacji marginesu klasyfikacji: strategię „ustalonego marginesu" oraz strategię „sumy marginesów".

W strategii „ustalonego marginesu" maksymalizowany jest margines określony przez dwie najbliższe sąsiednie klasy. Niech ($\mathbf{w}$, b_j) określają hiperpłaszczyznę rozdzielającą dwie sąsiednie klasy (powiedzmy, że **są to klasy** j oraz j+1). Jeżeli funkcja separująca dla tych dwu kategorii $\mathbf{w}^T\mathbf{x} - b_j < 0$, to klasyfikujemy wzorzec $\mathbf{x}$ do kategorii j, jeśli zaś $\mathbf{w}^T\mathbf{x} - b_j > 0$, do j+1.

Będziemy przy tym chcieli znaleźć takie wartości parametrów $\mathbf{w}$ i b_j, aby odległość wzorców treningowych od hiperpłaszczyzny rozdzielającej, w przestrzeni kategorii, była jak najbliższa 1. Tak więc, naszym celem jest znalezienie takich wartości parametrów, by dla par treningowych ($\mathbf{x}_{qd}$, $r_q(d)$) należących do kategorii j (tj. dla których $r_q(d) = j$) wartość funkcji separującej $\mathbf{w}^T\mathbf{x} - b_j$ była jak najbliższa –1, zaś dla par należących do kategorii j+1 (dla których $r_q(d) = j$+1), wartość $\mathbf{w}^T\mathbf{x} - b_j$ była jak najbliższa 1. Łączny margines w przestrzeni kategorii wynosić więc będzie 2, co odpowiada marginesowi w przestrzeni wejść (równemu odległości między hiperpłaszczyznami $\mathbf{w}^T\mathbf{x} = b_j$ **– 1 oraz** $\mathbf{w}^T\mathbf{x} = b_j$ **+ 1), o wartości** $2/||\mathbf{w}||$. Dlatego jednym z podstawowych celów uczenia modelu będzie minimalizacja normy $||\mathbf{w}||$, co zapewnia maksymalizację marginesu.

Ponadto, w ograniczeniach separowalności, dla danej hiperpłaszczyzny separującej ($\mathbf{w}$, b_j) wprowadzamy dwie zmienne: zmienną $\xi_{qd}(j)$ – kontrolującą margines wzorców z klasy j, z lewej strony hiperpłaszczyzny separującej, tj. w warunku $-1 + \xi_{qd}(j)$, oraz zmienną $\xi^*_{qd}(j+1)$ – kontrolującą margines klasy j+1, z prawej strony hiperpłaszczyzny separującej, tj. w warunku $1 - \xi^*_{qd}(j+1)$.

Wyznaczenie parametrów $\mathbf{w}$, $\mathbf{b}$, $\boldsymbol{\xi}_{qd}$, $\boldsymbol{\xi}^*_{qd}$, polega wtedy na rozwiązaniu zadania programowania kwadratowo-liniowego:

$$\min \frac{1}{2}\mathbf{w}^2 + \lambda \sum_{j=1}^{K-1} \left(\sum_{\substack{(\mathbf{x}_{qd}, r_q(d)) \in T \\ r_q(d)=j}} \xi_{qd}(j) + \sum_{\substack{(\mathbf{x}_{qd}, r_q(d)) \in T \\ r_q(d)=j+1}} \xi^*_{qd}(j+1) \right) \tag{2.2.89}$$

$$\mathbf{w}^T\mathbf{x}_{qd} - b_j \le -1 + \xi_{qd}(j) \quad \xi_{qd}(j) \ge 0 \quad \text{jeżeli } r_q(d) = j$$

$$\mathbf{w}^T\mathbf{x}_{qd} - b_j \ge 1 - \xi^*_{qd}(j+1) \quad \xi^*_{qd}(j+1) \ge 0 \quad \text{jeżeli } r_q(d) = j+1$$

Chu i Keerthi [Chu, Keerthi, 2005] zauważają, że model (2.2.89) Shashui i Levina nie zapewnia spełnienia naturalnych nierówności dla krańców przedziałów $b_1 \le \ldots \le b_{K-1}$, dlatego proponują dodanie jawnych ograniczeń, które je wymuszają:

$$b_{j-1} \le b_j,\ j=2,\ldots,K-1 \tag{2.2.89a}$$

Druga strategia zaproponowana przez Shashuę i Levina [Shashua, Levin, 2003], „sumy marginesów", polega na maksymalizacji sumy wszystkich K–1 marginesów klasyfikacji między poszczególnymi kategoriami porządkowymi. Uwzględnia ona problem porządkowania progów b_j, definiujących przedziały klas porządkowych regresji, bez konieczności wprowadzania dodatkowych jawnych ograniczeń, włączając ten warunek w proces optymalizacji.

Wprowadza się w niej zestaw dodatkowych K-1 progów a_j, j = 1, …, K–1, dla każdej kategorii, $a_1 \le b_1 \le a_2 \le b_2 \le \ldots \le a_{K-1} \le b_{K-1}$, przy czym $\mathbf{w}^T\mathbf{x}_{qd} \le a_j$ jeżeli $r_q(d) = j$, oraz $\mathbf{w}^T\mathbf{x}_{qd} \ge b_j$ jeżeli $r_q(d) = j$+1. Egzemplarze par treningowych dla każdej kategorii j, mają więc znaleźć się pomiędzy dwiema równoległymi hiperpłaszczyznami ($\mathbf{w}$, b_{j-1}) i ($\mathbf{w}$, a_j) (przyjmując $b_0 = -\infty$, oraz $a_K = \infty$). Marginesy między poszczególnymi klasami są więc równe $b_j - a_j$, j = 1, …, K–1, i zasadniczym celem uczenia jest maksymalizacja ich sumy (czy też raczej minimalizacja ujemnej sumy wartości $a_j - b_j$).

Podobnie jak w poprzednim przypadku, warunki separowalności wzorców treningowych między hiperpłaszczyznami rozdzielającymi zapewnia się poprzez wprowadzenie minimalizowanych w procesie uczenia zmiennych błędów $\xi_{qd}(j)$, $\xi^*_{qd}(j+1)$, definiujących dozwolone przekroczenie marginesu.

Model optymalizacyjny, umożliwiający wyznaczenie parametrów $\mathbf{w}$, $\mathbf{a}$, $\mathbf{b}$, $\boldsymbol{\xi}_{qd}$, $\boldsymbol{\xi}^*_{qd}$, ma w tym przypadku następującą postać:

$$\begin{aligned}
&\min \sum_{j=1}^{K-1}\left(a_j - b_j\right) + \lambda \sum_{j=1}^{K-1}\left(\sum_{\substack{(\mathbf{x}_{qd}, r_q(d))\in T \\ r_q(d)=j}} \xi_{qd}(j) + \sum_{\substack{(\mathbf{x}_{qd}, r_q(d))\in T \\ r_q(d)=j+1}} \xi^*_{qd}(j+1) \right) \\
&a_j \le b_j \\
&b_j \le a_{j+1} \\
&\mathbf{w}^T\mathbf{x}_{qd} \le a_j + \xi_{qd}(j) \quad \text{jeżeli } r_q(d) = j \\
&\mathbf{w}^T\mathbf{x}_{qd} \ge b_j - \xi^*_{qd}(j+1) \quad \text{jeżeli } r_q(d) = j+1 \\
&\mathbf{w}^T\mathbf{w} \le 1,\ \xi_{qd}(j) \ge 0, \xi^*_{qd}(j+1) \ge 0
\end{aligned} \tag{2.2.90}$$

Jak widzimy, w tym przypadku minimalizacja normy wektora parametrów $||\mathbf{w}||$ nie jest elementem funkcji celu modelu optymalizacyjnego.

Wynika to z faktu, że hiperpłaszczyzny tworzące marginesy separacji grup wzorców wejściowych odpowiadających różnym kategoriom relewancji mają w tym przypadku równania: $\mathbf{w}^T\mathbf{x} = b_j$ oraz $\mathbf{w}^T\mathbf{x} = a_j$. Odległość między nimi, czyli rozmiar marginesu separacji w przestrzeni wzorców wejściowych, wynosi więc $(b_j - a_j)/||\mathbf{w}||$. Skoro w funkcji celu maksymalizujemy marginesy $(b_j - a_j)$, dla osiągnięcia celu uczenia wystarczy utrzymać normę wektora parametrów $||\mathbf{w}||$ na ograniczonym poziomie, stąd też ograniczenie modelu $\mathbf{w}^T\mathbf{w} \leq 1$.

Chu i Keerthi [Chu, Keerthi, 2005] zaproponowali jeszcze odmienne podejście do wykorzystania maszyn wektorów wspierających i uczenia opartego na maksymalizacji marginesu w zadaniu regresji porządkowej. Również i w tym przypadku procedura uczenia modelu SVM bezpośrednio uwzględnia problem porządkowania wartości progów b_j, definiujących krańce przedziałów klas porządkowych regresji. Chu i Keerthi dla każdego z progów b_j budują jednak proces uczenia w oparciu o wszystkie próbki treningowe, a nie tylko należące do sąsiednich klas j oraz $j+1$.

Jeśli bowiem weźmiemy dowolny z progów b_j wyznaczających koniec przedziału klasowego dla klasy j, to zwróćmy uwagę, że warunek $\mathbf{w}^T\mathbf{x}_{qd} \leq b_j$ powinien być spełniony dla wszystkich par treningowych, dla których $r_q(d) \leq j$, zaś warunek $\mathbf{w}^T\mathbf{x}_{qd} \geq b_j$ dla wszystkich, dla których $r_q(d) > j$. Przyjmując standardowo margines 2 (czyli ±1), dla danego progu b_j, będziemy więc w procesie uczenia maksymalizacji marginesu minimalizować zmienne błędu $\xi_{qd}(j) = \mathbf{w}^T\mathbf{x}_{qd} - (b_j - 1)$ dla $r_q(d) \leq j$ oraz $\xi^*_{qd}(j+1) = (b_j + 1) - \mathbf{w}^T\mathbf{x}_{qd}$ dla $r_q(d) > j$.

Wyznaczenie parametrów $\mathbf{w}$, $\mathbf{b}$, ξ_{qd}, ξ^*_{qd}, polega więc wtedy na rozwiązaniu zadania optymalizacyjnego:

$$\begin{aligned}
&\min \frac{1}{2}\mathbf{w}^2 + \lambda \sum_{j=1}^{K-1} \left(\sum_{k=1}^{j} \sum_{\substack{(\mathbf{x}_{qd}, r_q(d)) \in T \\ r_q(d)=k}} \xi_{qd}(j) + \sum_{k=j+1}^{K} \sum_{\substack{(\mathbf{x}_{qd}, r_q(d)) \in T \\ r_q(d)=k}} \xi^*_{qd}(j+1) \right) \\
&\mathbf{w}^T\mathbf{x}_{qd} - b_j \leq -1 + \xi_{qd}(j), \quad \xi_{qd}(j) \geq 0 \\
&\text{dla } r_q(d) = k,\ k = 1, \ldots, j \\
&\mathbf{w}^T\mathbf{x}_{qd} - b_j \geq 1 - \xi^*_{qd}(j+1), \xi^*_{qd}(j+1) \geq 0 \\
&\text{dla } r_q(d) = k,\ k = j+1, \ldots, K
\end{aligned} \tag{2.2.91}$$

Porównując model (2.2.91) z podejściami zaproponowanymi przez Shashuę i Levina (2.2.89) oraz (2.2.90), widzimy, że rozwiązanie to chyba najbardziej bezpośrednio opiera się na paradygmacie działania maszyn wektorów wspierających, wykorzystując go wprost do wyznaczenia parametrów modelu regresji porządkowej. Z drugiej jednak strony, wymaga ono wyznaczenia znacznie większej liczby parametrów ξ_{qd}, ξ^*_{qd}, oraz ograniczeń w modelu optymalizacji.

Na koniec przeglądu wykorzystania SVM do regresji porządkowej dla celów rankingowania zwróćmy jeszcze uwagę na pewien istotny fakt. W przytoczonych modelach wykorzystywane były liniowe funkcje klasyfikujące. Natomiast stosując jądro w postaci odpowiedniej rodziny funkcji, możemy przekształcić SVM w klasyfikator nieliniowy. Czytelników odsyłamy tutaj do literatury poświęconej tej klasie modeli [Koronacki, Ćwik, 2015].

Przedstawione wyżej modele, nie wyczerpują oczywiście katalogu wszystkich możliwości. Chu i Ghahramani, dla przykładu, przedstawili zastosowanie klasyfikatora probabilistycznego opartego na procesach Gaussowskich do regresji porządkowej i tworzenia rankingu [Chu, Ghahramani, 2005, 2005a]. Rennie i Srebro prowadzili również badania zastosowań do tego celu dosyć szerokiej klasy popularnych modeli klasyfikacyjnych opartych na regresji binarnej [Rennie, Srebro, 2005].

Rennie i Srebro zaproponowali użycie do zadania rankingowania względem zestawu porządkowych kategorii relewancji rozszerzonej wersji ogólnego klasyfikatora binarnego o liniowej funkcji oceny $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$, gdzie w_0 jest wyrazem wolnym. Tradycyjnie klasyfikator uczony jest z wykorzystaniem zbioru par treningowych $T = \{(\mathbf{x}_{qd}, r_q(d))\}$, gdzie $\mathbf{x}_{qd}$ jest wzorcem cech dokumentu d i zapytania q, przy czym w zadaniach klasyfikacji binarnej przyjmujemy że $r_q(d) = \pm 1$. Stosowana w procesie uczenia funkcja kosztu minimalizuje błąd treningowy wraz elementem regularyzacyjnym normy wektora parametrów:

$$E(\mathbf{w}) = \sum_{qd} L\left(f\left(\mathbf{x}_{qd}, \mathbf{w}\right), r_q(d)\right) + \frac{\lambda}{2}\mathbf{w}^2 \tag{2.2.92}$$

Zauważmy, że powyższe warunki spełnia dosyć obszerna grupa metod klasyfikacyjnych opartych na regresji binarnej, które różnią się między sobą właściwie tylko definicją członu kosztu klasyfikacji dla pojedynczej pary treningowej $L(f(\mathbf{x}_{qd}, \mathbf{w}), r_q(d))$.

Dla kosztu zerojedynkowego:

$$L\left(f\left(\mathbf{x}_{qd},\mathbf{w}\right),r_q(d)\right)=h\left(f\left(\mathbf{x}_{qd},\mathbf{w}\right)\cdot r_q(d)\right)$$
$$h(z)=\begin{cases}0\,\text{dla}\,z>0\\1\,\text{dla}\,z\le 0\end{cases} \tag{2.2.93}$$

Dla kosztu marginesu:

$$L\left(f\left(\mathbf{x}_{qd},\mathbf{w}\right),r_q(d)\right)=h\left(f\left(\mathbf{x}_{qd},\mathbf{w}\right)\cdot r_q(d)\right)$$
$$h(z)=\begin{cases}0\,\text{dla}\ge 1\\1\,\text{dla}\,z<1\end{cases} \tag{2.2.94}$$

Dla kosztu zawiasowego (ang. *hinge loss*):

$$L\left(f\left(\mathbf{x}_{qd},\mathbf{w}\right),r_q(d)\right)=h\left(f\left(\mathbf{x}_{qd},\mathbf{w}\right)\cdot r_q(d)\right)$$
$$h(z)=\begin{cases}0\ \text{dla}\ z\ge 1\\1-z\ \text{dla}\ z<1\end{cases} \tag{2.2.95}$$

Dla zmodyfikowanego kosztu kwadratowego:

$$L\left(f\left(\mathbf{x}_{qd},\mathbf{w}\right),r_q(d)\right)=h\left(f\left(\mathbf{x}_{qd},\mathbf{w}\right)\cdot r_q(d)\right)$$
$$h(z)=\begin{cases}0\ \text{dla}\ z\ge 1\\(1-z)^2\ \text{dla}\ z<1\end{cases} \tag{2.2.96}$$

Dla regresji logistycznej

$$L\left(f\left(\mathbf{x}_{qd},\mathbf{w}\right),r_q(d)\right)=h\left(f\left(\mathbf{x}_{qd},\mathbf{w}\right)\cdot r_q(d)\right)$$
$$h(z)=\log\left(1+e^{-z}\right) \tag{2.2.97}$$

Jak widzimy, dla wszystkich tych typów modeli klasyfikacji binarnej funkcje straty w procesie uczenia mogą być widziane jako kary $L(f(\mathbf{x}_{qd}, \mathbf{w}), r_q(d)) = h(f(\mathbf{x}_{qd}, \mathbf{w})\cdot r_q(d))$ dla marginesu klasyfikacji $z = f(\mathbf{x}_{qd}, \mathbf{w})\cdot r_q(d)$ i różnią się między sobą jedynie funkcją kary nałożoną $h(z)$, nałożoną na ten margines [Rennie, Srebro, 2005].

Powyższe wnioski dla klasyfikacji binarnej Rennie i Srebro wykorzystają do rozszerzenia dowolnej z powyższych metod na regresję porządkową,

poprzez zastosowanie odpowiedniej funkcji kosztu $L(f(\mathbf{x}_{qd}, \mathbf{w}), r_q(d))$. Zakładamy więc obecnie, jak w poprzednio dyskutowanych przypadkach, że wartości relewancji w parach treningowych mają charakter K kategorii porządkowych: $r_q(d) \in \{1, \ldots, K\}$, przy czym poszczególne kategorie określone są przez przedziały $-\infty = b_0 \leq b_1 \leq \ldots \leq b_{K-1} \leq b_K = \infty$.

Rennie i Srebro [Rennie, Srebro, 2005] pokazują dwa podejścia i dwie funkcje kosztu, pozwalające rozszerzyć regresję binarną na przypadek kategorii porządkowych.

Pierwsze z proponowanych podejść, oparte na tzw. „bezpośrednim progu", polega na określeniu błędu uczenia dla danej pary treningowej $(\mathbf{x}_{qd}, r_q(d))$ z wykorzystaniem progów definiujących „poprawny" przedział treningowy:

$$\begin{aligned} &L\Big(f\left(\mathbf{x}_{qd}, \mathbf{w}\right), r_q(d)\Big) = \\ &= h\Big(f\left(\mathbf{x}_{qd}, \mathbf{w}\right) - b_{r_q(d)-1}\Big) + h\Big(b_{r_q(d)} - f\left(\mathbf{x}_{qd}, \mathbf{w}\right)\Big) \end{aligned} \tag{2.2.98}$$

gdzie $h()$ jest dowolną z wymienionych wcześniej funkcji kary dla marginesu klasyfikacji binarnej.

Funkcja bezpośredniego progu uwzględnia przekroczenia wyłącznie bezpośrednich progów. Nie bierze natomiast pod uwagę tego, czy błąd nie został popełniony o więcej niż jedną kategorię wyjściową, a jeżeli tak, to ile kategorii on wynosił. Tego rodzaju rozróżnienie czyni natomiast drugie z zaproponowanych przez Rennie i Srebro podejść [Rennie, Srebro, 2005], oparte na „wszystkich progach". Funkcja kosztu uczenia w tym przypadku sumuje kary za naruszenia każdego z progów reprezentujących krańce przedziałów kategorii porządkowych:

$$\begin{aligned} &L\Big(f\left(\mathbf{x}_{qd}, \mathbf{w}\right), r_q(d)\Big) = \sum_{j=1}^{K-1} h\Big(s\big(j, r_q(d)\big)\Big(b_{r_q(d)} - f\left(\mathbf{x}_{qd}, \mathbf{w}\right)\Big)\Big) \\ &s(j, y) = \begin{cases} -1 \text{ gdy } j < y \\ +1 \text{ gdy } j \geq y \end{cases} \end{aligned} \tag{2.2.99}$$

Podsumowując rozważania w niniejszym podrozdziale, zaprezentowaliśmy szereg rozwiązań do uczenia funkcji rankingującej z wykorzystaniem podejścia punktowego, opartych na różnych metodach modelowania i możliwych do zastosowania przy różnych podejściach do sposobu oceny relewancji dokumentów dla zapytania.

Należy jednak zwrócić uwagę, jak już wcześniej wspominaliśmy, że podejście punktowe skupione jest na ocenie pojedynczego dokumentu, w związku z tym problem względnego uporządkowania różnych dokumentów nie może być w naturalny sposób włączony do procesów uczenia tego rodzaju modeli [Liu, 2011]. Pozycje poszczególnych dokumentów nie są widoczne dla funkcji celu wykorzystywanych do ich uczenia i poprawność rankingu jest tylko pośrednim skutkiem poprawności oceny pojedynczego dokumentu.

W dodatku, ponieważ punktowe funkcje nie są świadome pozycji dokumentu w rankingu, z taką samą istotnością traktują one błędy dokumentów ważnych, lokowanych na początku listy rankingowej, co mniej ważnych, znajdujących się na jej końcu i bardzo słabo wpływających na potrzeby informacyjne użytkowników. Te ostatnie, jako znacznie liczniejsze, mogą zdominować proces uczenia modelu [Liu, 2011].

Kolejnym problemem, o którym należy pamiętać jest fakt, że z poszczególnymi zapytaniami może być związana różna liczba dokumentów. W przypadkach gdy liczebności dokumentów powiązanych z zapytaniami występującymi w zbiorze treningowym są silnie zróżnicowane, powodować to może nadreprezentację próbek treningowych dla niektórych zapytań i zdominowanie przez nie funkcji kosztu uczenia modelu.

2.2.3.3. Podejście oparte na parach

W przypadku podejścia opartego na parach, przestrzeń wejściowa modelu rankingowania obejmuje wzorce cech dwu dokumentów oraz zapytania. Wyjściem modelu jest , czyli preferencja dokumentu w parze wejściowej, w odniesieniu do danego zapytania, definiowana zazwyczaj jako wartość ze zbioru {–1, 0, 1} lub {0, 1/2, 1}, a w niektórych przypadkach jako ciągła miara preferencji między powyższymi granicami.

Jeżeli nie zostanie powiedziane inaczej, zakładać będziemy, że zbiór treningowy ma postać $T = \{(\mathbf{x}_{qu}, \mathbf{x}_{qv}), r_q(u, v))\}$, gdzie $(\mathbf{x}_{qu}, \mathbf{x}_{qv})$ jest wektorem cech dla pary dokumentów treningowych d_u, d_v oraz zapytania q, zaś $r_q(u, v)$ jest oceną preferencji odnośnie do relewancji w tej parze dokumentów względem tego zapytania. Oceny preferencji mogą być ustalane w różny sposób, lecz zasadniczo mamy do czynienia z trzema wymienianymi już w bieżącym podrozdziale przypadkami [Liu, 2009, 2011]:

- Oceny relewancji mogą być zbierane odrębnie dla poszczególnych dokumentów w parze, w formie ich stopnia relewancji dla zapytania $r_q(d)$. Wówczas muszą zostać przeliczone do preferencji między poszczególnymi parami dokumentów. Na przykład jeśli preferencja w parach ma przyjmować wartości ±1, to może być to formuła

postaci: $r_q(u, v) = 2I_{rq(u) < rq(v)} - 1$, gdzie $I_{rq(u)< rq(v)}$ jest funkcją wskaźnikową równą 1, gdy warunek jest spełniony, 0 w przeciwnym przypadku.

- Jeśli ocena relewancji dokonywana jest jako preferencja w parach $r_q(d_u, d_v)$, można bezpośrednio (lub po odpowiednich przeskalowaniach) przypisać $r_q(u, v) = r_q(d_u, d_v)$.
- Jeśli ocena relewancji ma formę pełnej uporządkowanej listy π_r, możemy zdefiniować dla przykładu .

W większości wypadków podejście oparte na parach, redukuje się do zadania klasyfikacji dokumentów dla danego zapytania. Jeśli dla danego zapytania q chcemy określić preferencję w parze dokumentów d_u, d_v, $pref(\mathbf{x}_{qu}, \mathbf{x}_{qv})$, to w takim przypadku tworzymy model funkcji oceny relewancji dokumentu $f()$, oceniamy przy jej pomocy odrębnie oba dokumenty: $f(\mathbf{x}_{qu})$, $f(\mathbf{x}_{qv})$, a następnie wskazujemy jako preferowany dokument oceniony jako bardziej relewantny. W innych przypadkach model może oceniać bezpośrednio funkcję preferencji $pref(\mathbf{x}_{qu}, \mathbf{x}_{qv})$ w parach dokumentów (dla uproszczenia zapisu, będziemy również używać formy $pref(u, v)$, pamiętać jednak należy, że preferencja oceniana jest zawsze dla wzorców cech dokumentów względem danego zapytania).

Rozpoczniemy jednak od kwestii samego porządkowania dokumentów, w sytuacji gdy dla danego zapytania dysponujemy funkcją preferencji $pref(u, v)$, innymi słowy dla każdej pary dokumentów potrafimy wskazać, ich względne uporządkowanie w rankingu. W ogólności zadanie to nie jest wcale takie trywialne, jak mogłoby się wydawać. Pamiętajmy bowiem, że funkcja preferencji $pref(u, v)$ uczona jest na podstawie ocen zaszumionych preferencji, które mogą mieć charakter niespójny, w związku z czym mogą generować cykliczne uporządkowania, na przykład takie jak $a < b < c < a$.

Problem ten ogólnie, w odniesieniu do zadań porządkowania, znany był już od lat 70. ubiegłego wieku. Cohen, Schapire i Singer pokazali jednak w kontekście relewancji dokumentów, że nawet jeśli funkcja preferencji jest kombinacją stosunkowo dobrze zachowujących się preferencji bazowych, to jeśli ranking dokumentów sporządzany jest przy więcej niż dwu ocenach relewancji (tj. relewantny/nierelewantny), zadanie uporządkowania dokumentów na podstawie preferencji jest NP-zupełne [Cohen, Schapire, Singer, 1998, 1999].

Stworzyli oni jednak również stosunkowo prosty, efektywny obliczeniowo algorytm oparty na podejściu zachłannym, który pozwala na uzyskanie dobrego przybliżenia optymalnego uporządkowania dokumentów. Zakłada on, że dysponujemy oszacowaniami preferencji $pref(u, v)$ dla

każdej pary dokumentów u, v, przy czym wartości oszacowanych preferencji niekoniecznie muszą być zgodne. Możliwe jest więc, że $pref(u, v) \neq pref(v, u)$. Zakłada się również, że preferencja ta oceniana jest jako wartość w przedziale [0, 1], tj. $pref(u, v)$ bliskie 1 interpretowane jest jako silna rekomendacja, że u powinno znaleźć się przed v, bliskie 0, że v przed u, zaś około 1/2 – brak rekomendacji.

Rozpoczynamy od całego zbioru wszystkich rankingowanych dokumentów V. Dla każdego dokumentu $v \in V$ obliczamy $\pi(v) = \Sigma_{u \in V} pref(v, u) - \Sigma_{u \in V} pref(u, v)$, czyli siłę łącznej preferencji dla dokumentu v względem wszystkich pozostałych dokumentów ze zbioru V, stanowiącą jego potencjalną ocenę rankingującą.

Następnie znajdujemy dokument o najwyższej ocenie w V, tzn. $t = \arg\max_{u \in V} \pi(u)$, ustawiamy go w rankingu przed wszystkimi dokumentami pozostającymi w V, a następnie usuwamy z V. Korygujemy oceny wszystkich dokumentów $v \in V$, usuwając z nich wpływ dokumentu t: $\pi(v) = \pi(v) - pref(v, t) + pref(t, v)$.

I ponownie wyznaczamy dokument o maksymalnej potencjalnej ocenie, wstawiamy go na kolejne miejsce w rankingu (przed wszystkimi dokumentami pozostałymi w V) i usuwamy ze zbioru V, korygując oceny pozostałych dokumentów. Czynności te powtarzamy do chwili, gdy ze zbioru V nie usuniemy wszystkich dokumentów.

Jak wspomnieliśmy Cohen, Schapire i Singer przyjmują, że funkcja preferencji $pref(u, v)$ jest kombinacją zestawu pewnych preferencji bazowych:

$$prev(u,v) = \sum_{i=1}^{M} w_i R_{f_i}(u,v) \tag{2.2.100}$$

gdzie parametry w_i, $i = 1, \ldots, M$ mają charakter wag strukturalnych, tj. $w_i \in [0, 1]$ oraz $\sum_{i=1}^{M} w_i = 1$. Preferencje bazowe otrzymywane są przy wykorzystaniu funkcji oceniających relewancję dokumentów dla danego zapytania $f_i()$:

$$R_{f_i}(u,v) = \begin{cases} 1 \text{ gdy } f_i(u) > f_i(v) \\ 0 \text{ gdy } f_i(u) < f_i(v) \\ \frac{1}{2} \text{ w innym wypadku} \end{cases} \tag{2.2.101}$$

Tak więc funkcja preferencji dla pary dokumentów przy danym zapytaniu *pref*(*u*, *v*) uzyskiwana jest jako mieszanina wyników działania pewnych ekspertów rankingujących $f_i()$. Mogą być to funkcje oceniające, otrzymane w wyniku zastosowania różnych metod wyszukiwania albo różnych metod klasyfikacji pod względem relewancji. Cohen, Schapire i Singer [Cohen, Schapire, Singer, 1998, 1999] przedstawiają metodę uczenia parametrów w_i, funkcji preferencji *pref*(*u*, *v*), z wykorzystaniem ocen preferencji od użytkownika.

Proces uczenia przebiega według następującego schematu. Dla danego zbioru dokumentów *X* tworzymy bazowe funkcje preferencji dla poszczególnych ekspertów oraz przyjmujemy wartości początkowe wag $w_i = 1/M$, $i = 1, \ldots, M$. Wyznaczając *pref*(*u*, *v*) przy pomocy zależności (2.2.100), tworzymy ranking dokumentów, korzystając z opisanego wyżej algorytmu przybliżonego uporządkowania dokumentów. Na podstawie oceny przedstawionego użytkownikowi rankingu uzyskujemy od niego ocenę preferencji w formie zbioru *F* par dokumentów (*u*, *v*) takich że „*u* powinno być preferowane względem *v*" (to znaczy $r_q(u, v) = 1$).

Dla wskazanych przez użytkownika par dokumentów w zbiorze *F* preferencja uzyskana przy użyciu ekspertów powinna również wynosić 1. Dla danego eksperta , wyznaczamy więc wartość popełnianego przez niego błędu (kosztu), jako średnie niedoszacowanie preferencji w zbiorze *F*:

$$L\left(R_{f_i}, F\right) = \frac{\sum_{(u,v)\in F}\left(1 - R_{f_i}(u,v)\right)}{|F|} \tag{2.2.102}$$

Wagi poszczególnych ekspertów modyfikujemy tak, by promować ekspertów tworzących funkcje preferencji w dużej mierze zgodne z ocenami użytkownika:

$$w_i = w_i \cdot \beta^{L\left(R_{f_i}, F\right)} / Z \tag{2.2.103}$$

gdzie $\beta \in [0,1]$ jest parametrem uczenia, zaś *Z* stałą normalizującą, zapewniającą sumowanie się wag do wartości 1. Ponieważ β jest mniejsze od 1, więc w przypadku ekspertów, dla których koszt *L*() jest duży, wagi będą zmniejszane znacznie bardziej, niż jeśli koszt *L*() jest mały.

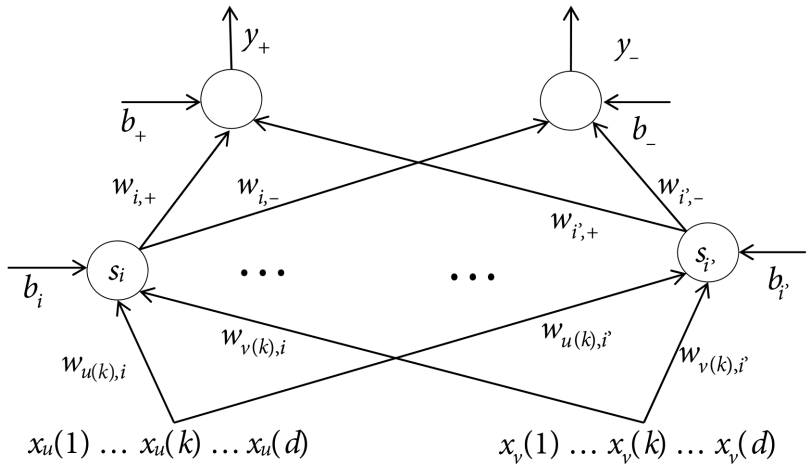
Przedstawiona wyżej metoda wykorzystuje do modelowania funkcji preferencji *pref*(*u*, *v*) dla pary dokumentów, przy danym zapytaniu, rozwiązanie polegające na scalaniu wyników działania kilku „ekspertów", czyli modeli oceny relewancji porównywanych dokumentów dla tego zapytania. Wymaga ona więc wcześniejszej budowy tego rodzaju modeli.

W literaturze proponuje się również bardziej bezpośrednie podejścia do modelowania funkcji *pref*(*u*, *v*). Między innymi zaproponowano szereg metod wykorzystujących sieci neuronowe.

Przykładem takiego podejścia jest algorytm SortNet [Rigutini, Papini i in., 2008] [Rigutini, Papini i in., 2011], porządkujący zbiór dokumentów zgodnie ze zstępującym poziomem relewancji. Do porównywania dokumentów w tej metodzie wykorzystywana jest warstwowa jednokierunkowa sieć neuronowa.

Sieć neuronowa porównująca obiekty zbudowana jest z trzech warstw: warstwy wejściowej, warstwy ukrytej i warstwy wyjściowej (rysunek 2.2.12). W warstwie wejściowej znajduje się $2 \cdot n$ neuronów, reprezentujących poszczególne cechy pary porównywanych dokumentów (u, v), dla danego zapytania q. Wejścia sieci odpowiadają elementom składowym wektora $(\mathbf{x}_{qu}, \mathbf{x}_{qv}) = (x_{qu}(1), \ldots, x_{qu}(n), x_{qv}(1), \ldots, x_{qv}(n))$, gdzie n jest liczbą cech porównywanych dokumentów.

Warstwa wyjściowa obejmuje dwa neurony: y_+, którego stan dla danej pary wejściowej dokumentów (u, v) interpretować będziemy jako prawdopodobieństwo faktu, że dokument u jest preferowany w stosunku do v, $P(u \succ v)$, oraz y_- określający prawdopodobieństwo, że w parze tej preferowany dokument v, $P(u \prec v)$. Aby uniknąć w fazie tworzenia rankingu części problemów z niezgodnością porównań preferencji, dyskutowanych w poprzednio prezentowanym modelu, sieć wymuszać będzie zgodność bezpośrednich preferencji, to znaczy warunek $y_+(u, v) = y_-(v, u)$.



Rys. 2.2.12. Struktura sieci neuronowej porównującej dokumenty w algorytmie SortNet

Źródło: opracowanie własne na podstawie [Rigutini, Papini i in., 2011].

Aby zapewnić powyższą zgodność, warstwa ukryta zawiera zawsze parzystą liczbę neuronów, tzn. dla każdego neuronu ukrytego s_i, występuje neuron dualny $s_{i'}$. Ponadto neurony w sieci współdzielą ze sobą wagi (oznaczenia takie same jak na rysunku 2.2.12), zgodnie z następującym schematem [Rigutini, Papini i in., 2011]):

- $w_{u(k),i'} = w_{v(k),i}$ oraz $w_{v(k),i'} = w_{u(k),i}$, tj. wagi połączeń z $x_u(k)$, $x_v(k)$ do neuronu $s_{i'}$, są nawzajem zamienione w stosunku do wag połączeń do neuronu s_i.
- $w_{i',+} = w_{i,-}$ oraz $w_{i',-} = w_{i,+}$, a więc tutaj również wagi do neuronu wyjściowego „+" oraz „–" są takie same, tylko nawzajem zamienione.
- $b_{i'} = b_i$ oraz $b_- = b_+$, tj. wartości progów dla ukrytych neuronów dualnych są takie same, podobnie jak dla obu neuronów wyjściowych.

Dzięki powyższym regułom współdzielenia wag, jeśli oznaczymy przez $s_i(u, v)$ wyjście i-tego neuronu ukrytego dla wejściowej pary dokumentów (u, v), to otrzymujemy:

$$\begin{aligned} s_i(u,v) &= \varphi\left(\sum_k \left(w_{u(k),i} \cdot x_u(k) + w_{v(k),i} \cdot x_v(k)\right) + b_i\right) = \\ &= \varphi\left(\sum_k \left(w_{v(k),i'} \cdot x_u(k) + w_{u(k),i'} \cdot x_v(k)\right) + b_{i'}\right) = \\ &= s_{i'}(v,u) \end{aligned} \tag{2.2.104}$$

gdzie $\varphi()$ jest funkcją aktywacji neuronów ukrytych.

Ponadto, jeśli dalej oznaczymy przez $y_+(u, v)$, $y_-(v, u)$ wartości neuronów wyjściowych sieci, to znów korzystając zasad współdzielenia wag, oraz (2.2.104), otrzymujemy:

$$\begin{aligned} y_+(u,v) &= \varphi\left(\sum_i \left(w_{i,+} \cdot s_i(u,v) + w_{i',+} \cdot s_{i'}(u,v)\right) + b_+\right) = \\ &= \varphi\left(\sum_i \left(w_{i',-} \cdot s_i(u,v) + w_{i,-} \cdot s_{i'}(u,v)\right) + b_-\right) = \\ &= \varphi\left(\sum_i \left(w_{i',-} \cdot s_{i'}(v,u) + w_{i,-} \cdot s_i(v,u)\right) + b_-\right) = \\ &= y_-(v,u) \end{aligned} \tag{2.2.105}$$

Jak więc widzimy, powyższa architektura sieci wraz z regułami współdzielenia wag zapewnia spełnienie warunku zgodności porównań $y_+(u, v) = y_-(v, u)$. Pokazać również można [Rigutini, Papini i in., 2008, 2011], że

przedstawiony model sieci spełnia właściwości uniwersalnej aproksymacji (co w dużej mierze wynika z tej właściwości dla warstwowych sieci jednokierunkowych).

Uczenie sieci neuronowej ma charakter w zasadzie standardowy, pamiętając oczywiście, że jedynie połowa wag ma charakter adaptacyjny – pozostałe są przyjmowane na podstawie ich wartości. Zbiór treningowy ma postać: $T = \{((\mathbf{x}_{qu}, \mathbf{x}_{qv}), \mathbf{t}_{quv})$, gdzie wyjścia treningowe sieci $\mathbf{t}_{quv}$ są dwuelementowymi wektorami binarnymi, których wartości składowe określane są na podstawie uzyskanych od użytkowników ocen preferencji w parach treningowych:

$$\mathbf{t}_{quv} = \begin{cases} [10] \text{ jeżeli } u \succ_q v \\ [01] \text{ jeżeli } u \prec_q v \end{cases} \tag{2.2.106}$$

Do uczenia sieci wykorzystywana jest również zwykła kwadratowa funkcja błędu:

$$L(u,v) = \left(t^1_{quv} - y_+(u,v)\right)^2 + \left(t^2_{quv} - y_-(u,v)\right)^2 \tag{2.2.107}$$

Nauczona sieć wykorzystywana jest następnie do porównywania par dokumentów, przy czym stosowana jest następująca prosta reguła:

$$\begin{aligned} u \succ_q v &= y_+(u,v) > y_-(u,v) \\ u \prec_q v &= y_+(u,v) < y_-(u,v) \end{aligned} \tag{2.2.108}$$

Jak widzieliśmy, architektura sieci zapewnia bezpośrednią zgodność porównań. Natomiast sieć nie zabezpiecza pełnego uporządkowania, ponieważ pary dokumentów porównywane są niezależnie i niekoniecznie musi być spełniona relacja przechodniości preferencji, tzn. jeśli $u \succ v$ i $v \succ z$, to niekoniecznie musi być spełniony warunek $u \succ z$. Badania eksperymentalne wskazują jednak, że jeśli porównania preferencji w zbiorze treningowym T są zgodne, to sieć potrafi nauczyć się właściwości przechodniości porównań.

W wypadku gdyby działanie sieci neuronowej miało charakter dokładny, do uporządkowania dokumentów i utworzenia rankingu przy jej wykorzystaniu, wystarczyłoby zastosowanie dowolnej procedury sortowania. Oczywiście, sieci neuronowe uczą się właściwych reakcji jedynie w przybliżeniu, dlatego też algorytm SortNet wykorzystuje procedurę iteracyjnego wielokrotnego uczenia sieci neuronowej, na coraz większych,

przyrostowych zbiorach treningowych, rozszerzanych w każdym kroku o błędnie uporządkowane przykłady. Wynikiem działania algorytmu jest sieć generująca najlepszy ranking (w sensie dyskutowanych na początku bieżącego podrozdziału miar poprawności rankingu) na zbiorze walidacyjnym [Rigutini, Papini i in., 2008, 2011].

Nieco odmienne podejście do uczenia porównywania dokumentów pod kątem relewancji dla danego zapytania prezentuje algorytm RankNet [Burges, Shaked i in., 2005; Matveeva, Burges i in., 2006; Burges, 2010]. Stworzony został w firmie Microsoft i wykorzystywany jest (lub metody powstałe na jego bazie) w niektórych jej rozwiązaniach.

W przypadku algorytmu RankNet odchodzi się od wykorzystania ocen preferencji do modelowania bezpośrednio funkcji preferencji *pref*$(\mathbf{x}_{qu}, \mathbf{x}_{qv})$ na rzecz modelowania funkcji rankingującej $f()$, dla której jeżeli $u \succ v$, wówczas $f(\mathbf{x}_{qu}) > f(\mathbf{x}_{qv})$.

Zbiór treningowy składa się więc z par dokumentów $(\mathbf{x}_{qu}, \mathbf{x}_{qv})$ dla danego zapytania oraz dokonanych przez użytkownika ocen preferencji $r_q(u, v) \in \{-1, 0, 1\}$ (w skrócie oznaczać będziemy je przez r_{uv}), przy czym, oczywiście, wartość –1 oznacza że w parze dokument u został oznaczony jako mniej relewantny od v, wartość 1 że u jest bardziej relewantny niż v, zaś 0, że poziom relewancji obu dokumentów jest taki sam. Oceny te przeliczane są do ocen $R_{uv} \in \{0, ½, 1\}$ przy pomocy zależności $R_{uv} = (1 + r_{uv})/2$. Wartości R_{uv} mogą być traktowane jako oceny prawdopodobieństwa, że dokument u jest bardziej relewantny niż dokument v. Model może być również bezpośrednio stosowany dla ciągłych ocen wartości R_{uv}.

Przy pomocy modelu funkcji rankingującej f możemy wyznaczyć wartości oceny relewancji dla obu dokumentów w parze treningowej. Oznaczmy więc wartości wyjścia modelu przez $y_u = f(\mathbf{x}_{qu})$ oraz $y_v = f(\mathbf{x}_{qv})$. Dla danych wartości oceny relewancji przez model, prawdopodobieństwo preferencji dla dokumentu w parze, w metodzie RankNet, modelowane jest przy wykorzystaniu funkcji logistycznej:

$$p_{uv} = P(u \succ v) = \frac{1}{1 + e^{-\sigma(y_u - y_v)}} \tag{2.2.109}$$

gdzie parametr σ krzywej logistycznej, dobierany jest dla konkretnego przypadku. Do uczenia modelu $f()$ stosowana jest następnie funkcja kosztu (błędu) oparta na entropii. Dla pojedynczej pary treningowej jest ona równa:

$$L(u,v) = -R_{uv} \ln p_{uv} - (1 - R_{uv}) \ln(1 - p_{uv}) \tag{2.2.110}$$

Podstawiając prawdopodobieństwo p_{uv} z (2.2.109) do (2.2.110), po krótkich przeliczeniach otrzymujemy:

$$\begin{aligned} L(u,v) &= (1-R_{uv})\sigma(y_u - y_v) + \ln\left(1+e^{-\sigma(y_u-y_v)}\right) = \\ &= \frac{1}{2}(1-r_{uv})\sigma(y_u - y_v) + \ln\left(1+e^{-\sigma(y_u-y_v)}\right) \end{aligned} \tag{2.2.111}$$

Wyznaczenie pochodnych funkcji kosztu $L(u, v)$ względem wyjść modelu, y_u oraz y_v, dla dokumentów w parze, również nie sprawia poważniejszego kłopotu. Łatwo obliczyć, że:

$$\begin{aligned} \frac{\partial L}{\partial y_u} &= \sigma\left(\frac{1}{2}(1-r_{uv}) - \frac{1}{1+e^{\sigma(y_u-y_v)}}\right) \\ \frac{\partial L}{\partial y_v} &= -\frac{\partial L}{\partial y_u} \end{aligned} \tag{2.2.112}$$

Jeśli w_i, $i = 1, \ldots, p$ są parametrami modelu $f()$, to potrzebne do procedur uczenia pochodne funkcji kosztu $L(u, v)$ względem dowolnego parametru w_i, wyznaczyć można stosując wzór na pochodną funkcji złożonej:

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y_u}\frac{\partial y_u}{\partial w_i} + \frac{\partial L}{\partial y_v}\frac{\partial y_v}{\partial w_i} \tag{2.2.113}$$

Pochodne wyjścia modelu względem parametrów $\frac{\partial y_u}{\partial w_i}$ i $\frac{\partial y_v}{\partial w_i}$, dla obu dokumentów w parze, mają charakter standardowy. W metodzie RankNet do modelowania relewancji dokumentów $f()$ wykorzystuje się warstwową sieć jednokierunkową. W takim wypadku można je obliczyć wykorzystując klasyczną metodę wstecznej propagacji błędu.

Inne interesujące podejście do uczenia się rankingu, z wykorzystaniem informacji o preferencjach **użytkowników**, polega na tworzeniu w każdym kroku kolejnego modelu rankingującego na podstawie zbioru ważonych wzorców treningowych, przy czym wagi próbek dostosowywane są do błędu w poprzednim kroku. Wyznaczenie finalnego rankingu odbywa się na podstawie połączonych wyników działania wszystkich modeli. Podejście takie określane jest w uczeniu maszynowym jako tzw. „boosting". Algorytm RankBoost [Freund, Iyer i in., 2003], który przedstawiamy

poniżej, jest adaptacją do celów rankingowania znanego algorytmu klasyfikacyjnego AdaBoost.

Zakładamy więc, jak poprzednio, że mamy do dyspozycji treningowe oceny preferencji dla par dokumentów dla danego zapytania $r_q(u, v)$, przy czym przyjmujemy, że jeśli $r_q(u, v) > 0$, dokument u oceniony został jako bardziej relewantny od v, jeśli $r_q(u, v) < 0$ – odwrotnie, zaś wartość $r_q(u, v) = 0$ oznacza brak preferencji między tymi dokumentami. Im wyższa wartość $|r_q(u, v)|$, tym ważniejsze jest, aby jeden dokument znalazł się w rankingu przed drugim. Zakłada się ponadto, iż oceny relewancji są zgodne w parach, tj. $r_q(u, v) = -r_q(v, u)$.

Celem algorytmu jest zbudowanie funkcji rankingującej $f()$, takiej, że jeśli $f(u) > f(v)$, to dokument u powinien zostać umieszczony w rankingu powyżej v.

Oceny preferencji przekształcane są do postaci rozkładu preferencji dokumentów:

$$D(u,v)=\frac{\max\{0;r(u,v)\}}{\sum_{ij}\max\{0;r(i,j)\}} \tag{2.2.114}$$

Algorytm RankBoost polega na wykonaniu szeregu kolejnych $t = 1, \ldots T$ kroków. W danym kroku t:

- Na podstawie rozkładu preferencji D_t (przy czym $D_1 = D$) tworzony jest model rankingujący $f_t()$ oraz budowany jest przy jego użyciu ranking dokumentów.
- Ustalany jest dodatni współczynnik α_t (możliwe sposoby jego wyznaczania przedstawione zostaną niżej).
- Tworzony jest nowy rozkład preferencji D_{t+1}:

$$D_{t+1}(u,v)=\frac{D_t(u,v)\exp\left(\alpha_t\left(f_t(v)-f_t(u)\right)\right)}{Z_t} \tag{2.2.115}$$

gdzie Z_t jest współczynnikiem normalizującym:

$$Z_t=\sum_{u,v}D_t(u,v)\exp\left(\alpha_t\left(f_t(v)-f_t(u)\right)\right) \tag{2.2.116}$$

Po wykonaniu wszystkich T kroków finalna funkcja rankingująca tworzona jest jako kombinacja liniowa wszystkich modeli $f_t()$:

$$f_t(x) = \sum_{t=1}^{T} \alpha_t f_t(x) \tag{2.2.117}$$

Bez wątpienia kluczowe znaczenie w działaniu algorytmu RankBoost ma zależność (2.2.115) modyfikująca treningowy rozkład preferencji dokumentów. Zwróćmy uwagę, że jakiekolwiek zmiany w wartości preferencji nastąpią jedynie, gdy wartość preferencji $D_t(u, v) > 0$, czyli dokument u oceniony powinien być jako bardziej relewantny od dokumentu v. Jeżeli w tej sytuacji funkcja $f_t()$ uporządkuje te dokumenty w rankingu błędnie, to wyrażenie $\exp(\alpha_t(f_t(v) - f_t(u)))$ ma wartość większą od 1, czyli preferencja $D_{t+1}(u, v)$ wzrasta, aby w następnym kroku starać się wymusić poprawne uporządkowanie. Jeśli natomiast u i v uporządkowane były poprawnie, to $\exp(\alpha_t(f_t(v) - f_t(u))) < 1$, co zmniejsza wagę tego wzorca treningowego w uczeniu następnego modelu.

Pokazać można, że istotną rolę dla działania algorytmu RankBoost i otrzymania rankingu o niskiej wartości błędu ma dobór współczynnika α_t oraz modelu konstrukcji funkcji rankingującej $f_t()$ w taki sposób, aby minimalizowały one wartość współczynnika Z_t określonego przez (2.2.116). Algorytm RankBoost nie czyni co prawda założeń odnośnie wykorzystywanych modeli $f_t()$, traktując je jako czarną skrzynkę, tym niemniej dosyć oczywiste jest, że minimalizacja wartości Z_t jest zgodna z optymalizacją jakości rankingu jako takiego. Natomiast warunek ten może być wykorzystany do doboru wartości współczynników α_t. W [Freund, Iyer i in., 2003] dyskutowane są trzy następujące podejścia do tego zagadnienia:

- Dla danego ustalonego modelu rankingującego $f_t()$, Z_t może być potraktowane jako funkcja α_t. Można pokazać, że posiada ona jedno minimum, które może zostać znalezione przez prostą metodę przeszukiwania binarnego (bisekcji).
- Druga metoda minimalizacji Z_t może zostać zastosowana, gdy $f_t()$ ma zakres wartości $\{0, 1\}$. Oznaczmy, dla $b \in \{-1, 0, 1\}$,

$$W_{t,b} = \sum_{u,v} D_t(u,v) I_{f_t(v)-f_t(u)=b} \tag{2.2.118}$$

gdzie $I_{f_t(v)-f_t(u)=b}$ jest równe 1, jeśli warunek jest spełniony, 0 w przeciwnym przypadku. Wówczas $Z_t = W_{t,-1}e^{-\alpha_t} + W_{t,0} + W_{t,1}e^{\alpha_t}$. Po prostych obliczeniach otrzymujemy, że Z_t ma minimum dla:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{W_{t,-1}}{W_{t,1}}\right) \tag{2.2.119}$$

- Trzecie oszacowanie α_t, może zostać zastosowane, gdy $f_t()$ przyjmuje wartości z zakresu [0, 1]. Wówczas można pokazać, że Z da się aproksymować z góry przez wyrażenie:

$$Z \le \left(\frac{1-r}{2}\right)e^{\alpha} + \left(\frac{1+r}{2}\right)e^{-\alpha}, \text{ gdzie } r = \sum_{u,v} D(u,v)\left(f(u) - f(v)\right) \tag{2.2.120}$$

Prawa strona nierówności (2.2.120) przyjmuje minimum dla

$$\alpha = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right) \tag{2.2.121}$$

Wartość tę więc możemy wykorzystać do oszacowania α.

Do uczenia funkcji rankingującej, przy ocenach preferencji relewancji dokumentów parami, zastosować możemy również metody oparte ma maszynach wektorów wspierających [Herbrich, Graepel, Obermayer, 1999; Herbrich, Obermayer, Graepel, 2000; Joachims, 2002]. Użycie SVM do tego rodzaju regresji porządkowej, polega przy tym na dosyć naturalnej modyfikacji standardowego modelu.

Dla par dokumentów u, v, ze zbioru treningowego T, dla których w ocenach użytkowników została wskazana preferencja relewancji dla dokumentu u, np. $r_q(u, v) = 1$, chcemy zbudować taki model funkcji rankingującej dokumenty f, dla którego zachodzi nierówność $f(\mathbf{x}_{qu}) > f(\mathbf{x}_{qv})$. Przyjmując liniowy model funkcji rankingującej $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$, mamy więc dla par dokumentów u, v, dla których $r_q(u, v) = 1$, następującą zależność:

$$\begin{aligned} &\mathbf{w}^T\mathbf{x}_{qu} > \mathbf{w}^T\mathbf{x}_{qv} \\ &\mathbf{w}^T\mathbf{x}_{qu} - \mathbf{w}^T\mathbf{x}_{qv} > 0 \\ &\mathbf{w}^T\left(\mathbf{x}_{qu} - \mathbf{x}_{qv}\right) > 0 \end{aligned} \tag{2.2.122}$$

Co pozwala na zbudowanie modelu SVM, o funkcji dyskryminacyjnej $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$, spełniającego warunki:

$$\begin{aligned} &\min \frac{1}{2}\mathbf{w}^2 + \lambda \sum_{\substack{(\mathbf{x}_{qu}, \mathbf{x}_{qv}, r_q(u,v)) \in T: \\ r_q(u,v)=1}} \xi_{quv} \\ &\mathbf{w}^T\left(\mathbf{x}_{qu} - \mathbf{x}_{qv}\right) \ge 1 - \xi_{quv}, \quad \xi_{quv} \ge 0, \text{ dla } r_q(u,v) = 1 \end{aligned} \tag{2.2.123}$$

Dla tego rodzaju modeli SVM zbudowane zostały również efektywne algorytmy optymalizacyjne, rozwiązujące problem (2.2.123) w liniowym czasie [Joachims, 2006; Chapelle, Keerthi, 2010].

Analizując przedstawione wyżej metody uczenia funkcji rankingującej na podstawie preferencji relewancji w parach dokumentów, można ocenić, że w porównaniu do podejścia punktowego mają one poważne zalety, ponieważ modelowana w nich jest raczej względna pozycja dokumentu w rankingu, a nie bezwzględna wartość relewancji. Tym niemniej jednak, można wskazać również pewne ich wady [Liu, 2011]:

- Przejście od ocen relewancji pojedynczych dokumentów do preferencji w parach może prowadzić do pewnej utraty informacji o relewancji. Przedstawione wyżej metody z tej ostatniej grupy w zdecydowanej większości wykorzystują tylko informację o kierunku preferencji, a nie o jej sile, tzn. o tym jak bardzo dokument „wygrywający" w parze jest oceniany jako bardziej relewantny od „przegrywającego".
- Większość z przedstawionych wyżej metod rankingowania w oparciu o preferencje w parach dokumentów nie bierze pod uwagę pozycji dokumentu w finalnym rankingu, tylko porównania pojedynczych par dokumentów. Pamiętać jednak należy, że zazwyczaj dokumenty na początkowych pozycjach listy rankingowej są znacznie ważniejsze dla użytkowników niż te umieszczone niżej.
- Ponieważ liczba porównywanych par dokumentów może być równa kwadratowi ich liczby, różnice w liczbie par dokumentów między zapytaniami mogą być nawet jeszcze poważniejsze niż w przypadku podejścia punktowego [Cao, Xu i in., 2006; Qin, Zhang i in., 2008]. Aby rozwiązać ten problem, w niektórych opracowaniach [Cao, Xu i in., 2006] proponuje się normalizację funkcji celu stosowanych modeli na poziomie zapytania poprzez liczbę par dokumentów związanych z danym zapytaniem.
- Podejście oparte na preferencji w parach jest bardziej wrażliwe na zaszumione oceny użytkowników niż punktowe. Błędna ocena pojedynczego dokumentu może prowadzić do wielu błędnie ocenionych par. Problem ten może w pewnym stopniu złagodzić transformacja funkcji kosztów uczenia modeli rankingujących, „spłaszczająca" efekty oddziaływania dużych odchyleń wprowadzanych przez dane odstające, np. przy wykorzystaniu funkcji kosztu logistycznego [Carvalho, Elsas i in., 2008].

Przyjrzymy się więc teraz nieco dokładniej kilku metodom próbującym wyeliminować część z przedstawionych wyżej problemów. Zajmiemy się

przy tym przede wszystkim dwoma pierwszymi punktami z przedstawionej listy. Na problemy trzeci i czwarty należy w rozważanych w bieżącym punkcie zadaniach zwrócić szczególną uwagę, mają one jednak charakter bardziej generalny i rozwiązywane są najczęściej metodami ogólnie stosowanymi w zagadnieniach modelowania.

Rozpoczniemy od problemu przedstawionego na początku listy. Pierwsza z prezentowanych metod umożliwia wykorzystanie do uczenia rankingowania preferencji określanych na podstawie ocen relewancji wyrażonych w formie kilku kategorii porządkowych, o coraz wyższej intensywności [Qin, Liu i in., 2007]. Oryginalnie została ona co prawda zastosowana do rozszerzenia ostatnio dyskutowanego rozwiązania opartego na maksymalizacji marginesu i modelach SVM, nie ma jednak powodów, aby nie mogła zostać wykorzystana również i w przypadku innych algorytmów.

Przyjmijmy więc, że dla par dokumentów u, v oceny preferencji wyrażone zostały w formie kilku kategorii porządkowych, określających narastający poziom tej preferencji, ewentualnie relewancja każdego z dokumentów w parze oceniona została przez użytkowników w skali porządkowej obejmującej kilka kategorii. W tym drugim przypadku, jeśli do oceny dokumentów wykorzystano K kategorii relewancji, potrafimy określić preferencje w parach dla $K(K\text{-}1)/2$ poziomów. Na przykład dla czterech poziomów relewancji {1, 2, 3, 4}, otrzymamy 6 różnych poziomów siły preferencji dla dokumentów u, v pochodzących z grup {(1, 2), (1, 3), (1, 4), (2, 3), (2, 4) (3, 4)}.

Dla każdej grupy poziomu natężenia preferencji dokumentów w parach możemy zbudować rankingujący model SVM określony przez (2.2.123). Będziemy je określać bazowymi modelami rankingującymi. Idea proponowanej metody polega na zbudowaniu dla każdego modelu bazowego wynikającego z niego rankingu dokumentów, a następnie agregacji powstałych rankingów w jedną całość.

W literaturze poświęconej zagadnieniom wyszukiwania informacji zaproponowanych zostało szereg podejść do scalania rankingów [Liu, 2011], takich jak agregacja metodą Bordy [Aslam, Montague, 2001; Dwork, Kumar i in., 2001], metodą mediany rangi [Fagin, Kumar, Sivakumar, 2003], z wykorzystaniem metod *soft computingu* – algorytmu genetycznego [Beg, 2004], logiki rozmytej [Ahmad, Beg, 2002], agregacja oparta na łańcuchach Markova [Dwork, Kumar i in., 2001]. Do scalenia rankingów otrzymanych przy pomocy modeli bazowych zastosowana może być praktycznie dowolna z nich. W [Qin, Liu i in., 2007] wykorzystano algorytm agregacji oparty na metodzie Bordy i ważonej metodzie Bordy.

Przyjmijmy, że zbudowanych zostało l modeli bazowych. Oznaczmy przez $\boldsymbol{\pi}_1$, …, $\boldsymbol{\pi}_l$ listy rankingowe utworzone przez każdy z nich. Metoda Bordy, nazwana na cześć XVII-wiecznego, matematyka francuskiego Jeana-Charlesa de Borda, polega na ocenie wyniku każdego dokumentu na podstawie jego pozycji w rankingu. Na przykład jako wynik dokumentu możemy przyjąć liczbę dokumentów, które znalazły się w rankingu poniżej niego we wszystkich listach rankingów bazowych:

$$s(u) = \sum_{k=1}^{l} s_k(u) \tag{2.2.124}$$

gdzie , oraz oznacza, że w rankingu π_k dokument u został uszeregowany wyżej od dokumentu v. Następnie dokumenty porządkowane są zgodnie ze swoimi wynikami $s(u)$.

W ważonej metodzie Bordy wynik dokumentu ustalany jest na podstawie ważonych wyników dla poszczególnych modeli bazowych:

$$s(u) = \sum_{k=1}^{l} \alpha_k s_k(u) \tag{2.2.125}$$

gdzie α_k jest wagą przypisaną k-temu modelowi bazowemu. Wagi ustalane mogą być na przykład na bazie działania modelu na dodatkowym zbiorze walidacyjnym. Mogą również być określane przez ekspertów, aby odzwierciedlać pewną wiedzę teoretyczną [Qin, Liu i in., 2007]. Wagi mogą być również przydzielane na podstawie siły preferencji, dla której tworzony był model bazowy.

Do kwestii uwzględnienia siły preferencji w modelach rankingowania opartych na porównaniu dokumentów parami można podejść również od strony wykorzystania w procesie uczenia specjalnych funkcji celu zachowujących rozmiary tej różnicy. Zagadnienie to analizowali Cortes, Mohri, Rastogi [Cortes, Mohri, Rastogi, 2007]. Zaproponowali oni kilka funkcji kosztu, które mogą być zastosowane do tego celu. Będziemy przy tym przyjmować, że porównywania w parach dokonywane są na podstawie ocen relewancji poszczególnych dokumentów dla zapytania, tzn. zbiór treningowy ma postać: $T = \{(\mathbf{x}_{qu}, \mathbf{x}_{qv}), (r_q(u), r_q(v))\}$.

Pierwsza z zaproponowanych funkcji celu uczenia jest naturalnym rozszerzeniem niesymetrycznego jednostronnego kosztu „zawiasowego" (ang. *hinge loss*), nakładającym karę na błędną klasyfikację w parze dokumentów, z uwzględnieniem różnicy w poziomach oceny ich relewancji przez model:

$$L_{HR}\left(f,\mathbf{x}_{qu},\mathbf{x}_{qv}\right)=\begin{cases}0 \text{ jeżeli } \left(f\left(\mathbf{x}_{qv}\right)-f\left(\mathbf{x}_{qu}\right)\right)\left(r_q(v)-r_q(u)\right)\geq 0\\ \left|f\left(\mathbf{x}_{qv}\right)-f\left(\mathbf{x}_{qu}\right)\right|^n \text{ w innym wypadku}\end{cases} \quad (2.2.126)$$

gdzie $n = 1$ lub $n = 2$ daje, oczywiście wersję bezwzględną lub kwadratową błędu.

Widzimy jednak, że funkcja kosztu L_{HR}, dla danej pary dokumentów u, v, bierze pod uwagę wielkość różnicy oceny relewancji przez model, ale nie „prawdziwej" różnicy w ocenach użytkowników $r_q(v) - r_q(u)$. Kolejna z zaproponowanych funkcji spełnia już powyższy warunek (ponownie dla $n = 1, 2$).

$$L_{MP}\left(f,\mathbf{x}_{qu},\mathbf{x}_{qv}\right)=\left|\left(f\left(\mathbf{x}_{qv}\right)-f\left(\mathbf{x}_{qu}\right)\right)-\left(r_q(v)-r_q(u)\right)\right|^n \quad (2.2.127)$$

Jednostronna, niesymetryczna wersja funkcji L_{MP}, nakładająca karę tylko na błędne klasyfikacje, będzie miała postać ($n = 1, 2$):

$$L_{HMP}\left(f,\mathbf{x}_{qu},\mathbf{x}_{qv}\right)=\begin{cases}0 \text{ jeżeli } \left(f\left(\mathbf{x}_{qv}\right)-f\left(\mathbf{x}_{qu}\right)\right)\left(r_q(v)-r_q(u)\right)\geq 0\\ \left|\left(f\left(\mathbf{x}_{qv}\right)-f\left(\mathbf{x}_{qu}\right)\right)-\left(r_q(v)-r_q(u)\right)\right|^n \text{ w in. wyp.}\end{cases} \quad (2.2.128)$$

Cortes, Mohri, Rastogi [Cortes, Mohri, Rastogi, 2007] zaproponowali ponadto funkcję kosztu dla uczenia z marginesem, stanowiącą rozszerzenie funkcji celu wykorzystywanej w klasycznym regresyjnym modelu SVM (SVR):

$$L_{SVR}\left(f,\mathbf{x}_{qu},\mathbf{x}_{qv}\right)= \\ =\begin{cases}0 \text{ jeżeli } \left|\left(f\left(\mathbf{x}_{qv}\right)-f\left(\mathbf{x}_{qu}\right)\right)-\left(r_q(v)-r_q(u)\right)\right|\leq\epsilon\\ \left|\left|\left(f\left(\mathbf{x}_{qv}\right)-f\left(\mathbf{x}_{qu}\right)\right)-\left(r_q(v)-r_q(u)\right)\right|-\epsilon\right|^n \text{ w in. wyp.}\end{cases} \quad (2.2.129)$$

Dla $n = 1$ oraz liniowej funkcji oceny $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$ funkcja kosztu (2.2.129) związana jest z modelem maszyny wektorów nośnych, określonym przez następujący problem optymalizacji z ograniczeniami [Cortes, Mohri, Rastogi, 2007]:

$$\begin{aligned}
&\min \frac{1}{2}\mathbf{w}^2 + \lambda \sum_{(u,v)\in T} \left(\xi_{quv} + \xi^*_{quv}\right) \\
&\mathbf{w}^T\left(\mathbf{x}_{qv} - \mathbf{x}_{qu}\right) - \left(r_q(v) - r_q(u)\right) \le \epsilon + \xi_{quv} \quad \xi_{quv} \ge 0 \\
&\left(r_q(v) - r_q(u)\right) - \mathbf{w}^T\left(\mathbf{x}_{qv} - \mathbf{x}_{qu}\right) \le \epsilon + \xi^*_{quv} \quad \xi^*_{quv} \ge 0
\end{aligned} \tag{2.2.130}$$

Przejdziemy obecnie do drugiego z podstawowych problemów, pojawiających się przy tworzeniu rankingu na podstawie informacji o preferencjach użytkowników odnośnie do relewancji w parach porównywanych dokumentów. Przypomnijmy, że związany jest on z faktem jednolitego traktowania wszystkich porównywanych dokumentów, bez uwzględnienia ich pozycji w rankingu wynikowym, podczas gdy dla użytkowników zazwyczaj znacznie ważniejsze są dokumenty na początkowych pozycjach listy niż te umieszczone niżej.

Sugeruje to oczywiście rozwiązanie polegające na tym, aby wykorzystywana w procesie uczenia modelu funkcja kosztu nakładała większe kary na błędy pojawiające się na początkowych pozycjach rankingu. Co prawda, na podstawie porównań parami dokumentów nie można określić ich ostatecznej pozycji w liście rankingowej, możliwe jest jednak oszacowanie jej. Pierwsza z prezentowanych metod, funkcja kosztu uczenia zastosowana w modelu „P-norm push" [Rudin, 2006, 2009], szacuje pozycję dokumentu w rankingu analizując wszystkie pary zawierające ten dokument.

Przyjmujemy binarny charakter oceny relewancji, tak więc dla par dokumentów u, v, ze zbioru treningowego T, dla których w ocenach użytkowników została wskazana preferencja relewancji dla dokumentu u, $r_q(u, v) = 1$, chcemy zbudować taki model funkcji rankingującej dokumenty f, dla którego zachodzi nierówność $f(\mathbf{x}_{qu}) > f(\mathbf{x}_{qv})$. Oznacza to, że dla danego dokumentu v chcemy redukować liczbę dokumentów ocenionych lepiej, które znalazłyby się w rankingu pod nim. Innymi słowy naszym celem jest minimalizacja błędu:

$$\sum_{u:r_q(u,v)=1} 1_{f\left(x_{qu}\right) \le f\left(x_{qv}\right)} \tag{2.2.131}$$

Oczywiście dla celów uczenia modelu zastępujemy zero-jedynkową funkcję błędu, z powodu jej właściwości, pewną przyjętą funkcją błędnej pozycji w rankingu L:

$$\sum_{u:r_q(u,v)=1} L\left(f\left(x_{qv}\right) - f\left(x_{qu}\right)\right) \tag{2.2.132}$$

co ostatecznie daje funkcję celu uczenia $L(f)$

$$L(f)=\sum_{v}\sum_{u:r_q(u,v)=1} L\Big(f\big(x_{qv}\big)-f\big(x_{qu}\big)\Big) \tag{2.2.133}$$

Algorytm P-norm push, proponuje dodatkowe „pchnięcie" dla zbyt nisko rankingowanego dokumentu, w sensie zwiększenia kary za duże różnice we właściwej pozycji w rankingu, z wykorzystaniem nieujemnej, wypukłej i monotonicznie rosnącej funkcji g:

$$L(f)=\sum_{v} g\left(\sum_{u:r_q(u,v)=1} L\Big(f\big(x_{qv}\big)-f\big(x_{qu}\big)\Big)\right) \tag{2.2.134}$$

Jako funkcję kary g można zastosować dla przykładu funkcję wykładniczą $g(r) = e^r$, albo potęgową $g(r) = r^p$, dla dużych wartości p.

Funkcja kosztu (2.2.134) posiada szereg pozytywnych właściwości, w stosunku do tradycyjnie stosowanych funkcji kosztu postaci (2.2.133). Na przykład znacznie bardziej karane są błędy w górnej, początkowej części rankingu, niż na końcowych jego pozycjach. Aby to pokazać, wykorzystamy prosty przykład ilustracyjny, przedstawiony w [Rudin, 2009].

Rozważmy następujący przykładowy zbiór ocen dokumentów:

$$(-1,+1,-1,+1,-1,-1,+1,+1)$$

gdzie wartość +1 oznacza, że dany dokument został oceniony wyżej od wszystkich dokumentów, z wartością −1. Przyjmijmy ponadto funkcję $g(r) = r^4$. Dla wyjściowego ustawienia dokumentów, zakładając rosnący porządek rankingowania, tzn. kolejne dokumenty znalazłyby się na coraz wyższych pozycjach rankingu, błąd (2.2.134) (przy zwykłej zerojedynkowej funkcji kosztu L) wynosiłby:

$$0^4 + 1^4 + 2^4 + 2^4 = 33$$

Jego wyznaczenie jest dosyć proste (możemy pominąć pozycje oznaczone +1, ponieważ nie ma żadnych dokumentów ocenionych lepiej od nich, a więc ich błąd jest równy 0):

- Dokument 1 – nie ma żadnego dokumentu ocenionego wyżej, za nim w rankingu.
- Dokument 3 – jest 1 dokument oceniony wyżej, za nim w rankingu.

- Dokument 5 – są 2 dokumenty ocenione wyżej, za nim w rankingu.
- Dokument 6 – są 2 dokumenty ocenione wyżej, za nim w rankingu.

Powiedzmy, że niewłaściwie przestawimy sąsiadujące dokumenty w dole rankingu, np. pierwszy i drugi:

$$(+1,-1,-1,+1,-1,-1,+1,+1)$$

Błąd wynosi:

$$1^4 + 1^4 + 2^4 + 2^4 = 34$$

Zmiana błędu jest więc niewielka, jego wzrost wynosi tylko 1. Natomiast przy niewłaściwym przestawieniu sąsiadujących dokumentów w górze rankingu, np. sześć i siedem:

$$(-1,+1,-1,+1,-1,+1,-1,+1)$$

Błąd wyniósłby:

$$0^4 + 1^4 + 2^4 + 3^4 = 98$$

W obydwu przypadkach przestawiliśmy (niewłaściwie) elementy sąsiednie. Widzimy jednak że przestawienie dokumentów w górze rankingu spowodowało znacznie większy przyrost błędu, niż przestawienie w dole.

Inne podejście do wprowadzenia do funkcji błędu pozycji dokumentu w rankingu związane jest z bezpośrednim wykorzystaniem pozycyjnych miar użyteczności rankingu, zdefiniowanych w punkcie 2.2.3.1 (takich jak MRR, MAP czy NDCG). Problem polega na tym, że miary te mają charakter nieciągły, tak więc w procesie uczenia modelu musimy korzystać z przybliżonych funkcji celu, zgodnych z nimi. Przykładem tego typu rozwiązań jest algorytm LambdaRank [Burges, Ragno, Le, 2007; Burges, 2010], będący modyfikacją prezentowanego już wcześniej w tym punkcie algorytmu RankNet (zależności (2.2.112) i (2.2.113)).

Przypomnijmy, że algorytm RankNet wykorzystywał metodę gradientową do uczenia funkcji rankingującej dokumenty *f*, dla zbioru treningowego ocen relewancji w parach dokumentów r_{uv}. Algorytm LambdaRank wykorzystuje przybliżone oszacowanie gradientu funkcji uczenia modelu *L*, dla pary dokumentów *u* i *v*, z wykorzystaniem formuły pewnej funkcji λ nazwanej λ-gradientem:

$$\lambda_{uv} = \frac{\partial L(y_u - y_v)}{\partial y_u} = \frac{-\sigma}{1 + e^{\sigma(y_u - y_v)}} \left|\Delta_{NDCG}\right| \qquad (2.2.135)$$

gdzie $y_u = f(\mathbf{x}_{qu})$, $y_v = f(\mathbf{x}_{qv})$, zaś $|\Delta_{NDCG}|$ zmianą miary NDCG, przy zamianie w rankingu dokumentów u i v. Zbiór uczący składa się z par dokumentów, dla których $r_{uv} = 1$. Pochodna funkcji celu względem parametrów modelu wyznaczana jest oczywiście ze wzoru na pochodną funkcji złożonej:

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y_u}\frac{\partial y_u}{\partial w_i} \tag{2.2.136}$$

Można pokazać, że optymalizacja powyższego modelu daje wyniki zgodne z optymalizacją miary NDCG. Ponadto wartość gradientu (2.2.135) zależy od wartości tej miary. W [Donmez, Svore, Burges, 2009] pokazano metody konstrukcji λ-gradientów dla innych miar pozycyjnych rankingu.

2.2.3.4. Podejście oparte na listach

W przypadku podejścia opartego na listach, bierzemy pod uwagę zbiór dokumentów, związanych z zapytaniem q, reprezentowanych przez wektory cech tych dokumentów, np. $\left\{x_{qj}\right\}_{j=1}^{m}$. Naszym celem jest określenie listy rankingowej (permutacji) tych dokumentów.

Oceny relewancji mogą być zbierane w różny sposób, a następnie przeliczane na odpowiadające im listy rankingowe, tzw. fundamentalnie prawdziwe listy lub permutacje [Liu, 2011]:

- Jeśli oceny relewancji zbierane są w formie stopnia relewancji każdego z dokumentów $r_j = r(x_{qj})$, to wszystkie permutacje zgodne z tymi ocenami (tzn. takie permutacje π_y że dla wszystkich u, v, jeśli $r_u > r_v$ to $\pi_y(u) < \pi_y(v)$) są fundamentalnie prawdziwe. W tym przypadku zestawowi ocen relewancji może odpowiadać wiele permutacji fundamentalnie prawdziwych. Zbiór tych permutacji oznaczymy przez Ω_y.
- Jeśli oceny relewancji zbierane są w formie preferencji w parach $r_{uv} = r(x_{qu}, x_{qv})$, wtedy ponownie wszystkie permutacje zgodne z tymi ocenami są fundamentalnie prawdziwe. W tej sytuacji przez permutację zgodną z preferencjami użytkownika, rozumiemy taką permutację I_y, że dla wszystkich u, v dla których $r_{uv} = 1$ otrzymujemy $\pi_y(u) < I_y(v)$. I znów tego rodzaju permutacji fundamentalnie prawdziwych może być wiele i oznaczamy je przez Ω_y.
- Jeśli ocena relewancji zbierana jest w formie uporządkowania dokumentów przez użytkownika π_r, to po prostu przyjmujemy $\pi_y = \pi_r$.

Metody uczenia rankingu oparte na listach, generalnie podzielić możemy na dwie grupy:

- wykorzystujące funkcje kosztu bezpośrednio związane z pozycyjnymi miarami użyteczności rankingu, zdefiniowanymi w punkcie 2.2.3.1;
- niezwiązane bezpośrednio z tymi miarami.

Podstawowym problemem pojawiającym się przy próbie wykorzystania do uczenia modelu rankingującego, funkcji kosztu opartych bezpośrednio na miarach użyteczności rankingu, jest, jak już niejednokrotnie wspomnieliśmy, ich nieciągły charakter. Wartości miar takich, jak NDCG, MRR, czy MAP, zmieniają się w sposób skokowy przy przestawianiu pozycji dokumentów w rankingu. W podejściach należących do pierwszej z wymienionych stosuje się kilka podejść dla rozwiązania tej kwestii. Pierwsze z nich polega na wykorzystaniu jako funkcji kosztu w procesie uczenia, ciągłej aproksymanty miary jakości modelu.

Przykładem tego rodzaju rozwiązania jest algorytm SoftRank [Taylor, Guiver i in., 2008]. Do modelowania funkcji rankingującej wykorzystywano w nim warstwową sieć jednokierunkową. Jako funkcję celu w procesie uczenia sieci zastosowano wygładzoną aproksymację miary NDCG, nazwaną SoftNDCG, opartą na wartości oczekiwanej tej miary.

W metodzie przyjmuje się, że wartości oceny poszczególnych dokumentów przez model mają charakter niepewny i określone są w formie rozkładów prawdopodobieństwa. Przyjmuje się model rozkładu normalnego wokół wartości wyjściowej funkcji rankingującej, tzn. dla danych dokumentów treningowych $\{x_{qj}\}_{j=1}^{m}$ związanych z zapytaniem q, ocena każdego z dokumentów y_j określona jest przez rozkład normalny prawdopodobieństwa o wartości oczekiwanej równej wartości wyjściowej modelu $f(x_{qj})$ oraz odchyleniu standardowym σ_y określonym przez jego błąd.

$$p(y_j) = N(y_j \mid f(x_{qj}), \sigma_y) \tag{2.2.137}$$

Ponieważ ocena dokumentu jest zmienną losową, w związku z tym również jego pozycja w rankingu także staje się zmienną losową. Każdy dokument może być z pewnym prawdopodobieństwem umieszczony na każdej z pozycji rankingu.

Prawdopodobieństwo, że dokument i zostanie umieszczony w rankingu powyżej dokumentu j (znaczmy je przez p_{ij}) odpowiada prawdopodobieństwu tego, że otrzyma on wyższą ocenę, lub równoważnie prawdopodobieństwu, że różnica ich ocen będzie większa od 0. Tak więc:

$$p_{ij} = P(y_i - y_j > 0) = \int_0^{\infty} N(y \mid f(x_{qi}) - f(x_{qj}), 2\sigma_y) dy \qquad (2.2.138)$$

Rozkład prawdopodobieństwa pozycji w rankingu dokumentu j, r_j otrzymywany jest z rozkładu dwumianowej zmiennej losowej, dla $m-1$ prób Bernoulliego, gdzie prawdopodobieństwo sukcesu odpowiada prawdopodobieństwu, że dokument j zostanie pokonany przez inny dokument i, p_{ij}. Obliczany jest on przy pomocy następującej procedury rekurencyjnej, ze względu na liczbę dokumentów.

Dla jednego dokumentu, pozycja rankingowa może wynosić tylko 0 (co oznacza najlepszy ranking). Tak więc $p_j^{(1)}(r) = \delta(r)$, gdzie $\delta(r) = 1$ tylko dla $r = 0$, dla pozostałych r jest równa zero.

Dla każdego kolejnego dokumentu i, rozkład prawdopodobieństwa wynosi:

$$p_j^{(i)}(r) = p_j^{(i-1)}(r-1)\, p_{ij} + p_j^{(i-1)}(r)(1 - p_{ij}) \qquad (2.2.139)$$

Jeżeli teraz wyznaczymy miarę NDCG dla rankingu wszystkich m dokumentów treningowych:

$$NDCG = \left(NDCG_{max}\right)^{-1} \sum_{j=1}^{m} (j) G\left(r_j\right) \qquad (2.2.140)$$

gdzie, przypomnijmy, $NDCG_{max}$ jest miarą DCG dla rankingu idealnego, zazwyczaj przyjmujemy $G(z) = (2^z - 1)$ oraz $\eta(j) = 1/\log_2(j+1)$. W sytuacji, gdy ocena każdego z dokumentów r_j jest zmienną losową, miarę *SoftNDCG* zdefiniujemy korzystając z wartości oczekiwanej funkcji G tej zmiennej:

$$\begin{aligned} SoftNDCG &= \left(NDCG_{max}\right)^{-1} \sum_{j=1}^{m} (j) E\left(G\left(r_j\right)\right) \\ &= \left(NDCG_{max}\right)^{-1} \sum_{j=1}^{m} (j) \sum_{r=0}^{m-1} G(r)\, p_j(r) \end{aligned} \qquad (2.2.141)$$

gdzie $p_j(r) = p_j^{(m)}(r)$ i obliczane jest przy pomocy (2.2.139).

Funkcja *SoftNDCG* jest miarą jakości (użyteczności), a nie kosztu, w związku z tym jako funkcję celu uczenia modelu oceny f, wykorzystuje

się funkcję 1 – *SoftNDCG*. Tak jak powiedzieliśmy, wykorzystywana jest ona do treningu modelu warstwowej sieci neuronowej [Taylor, Guiver i in., 2008].

Nieco odmienny rodzaj aproksymacji zastosowany został przez Qina, Liu i Li [Qin, Liu, Li, 2009; Liu, 2011]. Proponują oni aproksymację pozycji rankingowych dokumentów funkcjami ciągłymi względem wartości funkcji ich oceny.

Zmieniając w definicji miary NDCG (wzory (2.2.69) i (2.2.70)) indeks sumowania elementów z pozycji w rankingu na indeksy dokumentów, możemy zapisać NDCG jako:

$$NDCG = \left(NDCG_{max}\right)^{-1} \sum_{j=1}^{m} \frac{G\left(r_j\right)}{\log(1+\pi\left(x_{qj}\right)} \tag{2.2.142}$$

gdzie $\pi(x_{qj})$ jest pozycją j-tego dokumentu w rankingu π, którą można obliczyć na podstawie liczby wyżej ocenionych dokumentów:

$$\pi\left(x_{qj}\right) = 1 + \sum_{i \neq j} I_{\{f\left(x_{qj}\right) - f\left(x_{qi}\right) < 0\}} \tag{2.2.143}$$

Zależność (2.2.143) ma charakter nieciągły, ale może zostać aproksymowana ciągłą funkcją wartości funkcji oceny dokumentów:

$$\hat{\pi}\left(x_{qj}\right) = 1 + \sum_{i \neq j} \frac{\exp\left(-\alpha\left(f\left(x_{qj}\right) - f\left(x_{qi}\right)\right)\right)}{1 + \exp\left(-\alpha\left(f\left(x_{qj}\right) - f\left(x_{qi}\right)\right)\right)} \tag{2.2.144}$$

gdzie $\alpha > 0$ jest stałą skalującą.

Podstawiając (2.2.144) do (2.2.142) otrzymujemy ciągłą i różniczkowalną aproksymację miary NDCG, ApprNDCG. Jako funkcję kosztu uczenia modelu oceny dokumentów możemy wykorzystać funkcję 1 – ApprNDCG.

Inne podejście do wykorzystania funkcji celu uczenia bezpośrednio opartej na miarach użyteczności rankingu może polegać na zastosowaniu metod optymalizacji nieciągłych funkcji celu. Przykładem tego rodzaju rozwiązania jest metoda AdaRank [Xu, Li, 2007], wykorzystująca boosting do optymalizacji funkcji wykładniczej, miary użyteczności

rankingu. Ponieważ funkcja wykładnicza jest monotoniczna, jest to równoważne optymalizacji samej miary oceny rankingu.

Algorytm AdaRank, jak już wspomnieliśmy wykorzystuje podejście oparte na boostingu. Przypomnijmy, że generalnie boosting jest metodą polegającą na wielokrotnym uczeniu modelu, z wykorzystaniem zbiorów uczących zawierających ważone próbki z wyjściowego zbioru treningowego.

Oznaczmy przez $S(q_i, \mathbf{x}_i, \mathbf{r}_i)_{i=1}^{k}$ zbiór treningowy złożony z k zapytań q_i, odpowiadających im zestawów cech dokumentów $\mathbf{x}_i$, oraz zestawów ocen relewancji $\mathbf{r}_i$. Oznaczmy dalej przez $E\left(\pi\left(q_i, \mathbf{x}_i, f\right), \mathbf{r}_i\right)$ wartość miary oceny rankingu związanych z danym zapytaniem dokumentów $\mathbf{x}_i$, utworzonym przy pomocy funkcji oceny f.

Algorytm AdaRank możemy zreasumować następująco [Xu, Li, 2007]:

Inicjujemy rozkład wag próbek treningowych $P_1(i) = 1/k$.

Kolejne kroki wykonujemy dla $t = 1, \ldots, T$:

- Tworzymy słaby model rankingujący h_t, przy pomocy zbioru treningowego, ważonego rozkładem P_t.
- Obliczamy α_t:

$$\alpha_t = \frac{1}{2}\ln\frac{\sum_{i=1}^{k} P_t(i)\left(1+E\left(\pi\left(q_i, \mathbf{x}_i, h_t\right), \mathbf{r}_i\right)\right)}{\sum_{i=1}^{k} P_t(i)\left(1-E\left(\pi\left(q_i, \mathbf{x}_i, h_t\right), \mathbf{r}_i\right)\right)} \tag{2.2.145}$$

Tworzymy model w danym kroku f_t:

$$f_t(x) = \sum_{l=1}^{t} \alpha_l h_l(x) \tag{2.2.146}$$

Modyfikujemy rozkład P_{t+1}:

$$P_{t+1}(i) = \frac{\exp\left(-E\left(\pi\left(q_i, \mathbf{x}_i, f_t\right), \mathbf{r}_i\right)\right)}{\sum_{l=1}^{k} \exp\left(-E\left(\pi\left(q_l, \mathbf{x}_l, f_t\right), \mathbf{r}_l\right)\right)} \tag{2.2.147}$$

Ostatecznym modelem oceny dokumentów jest $f_T(x)$.

Inne podejścia z tej grupy stosują bezpośrednią optymalizację miary jakości rankingu, wykorzystując metody optymalizacji funkcji nieciągłych, takie jak np. algorytm genetyczny.

Druga grupa podejść opartych na listach, wykorzystuje do uczenia rankingu funkcje uczenia niezwiązane bezpośrednio z miarami jakości

rankingu, lecz niezgodność między wyjściowym rankingiem modelu, a fundamentalnie prawdziwą permutacją, zgodną z ocenami relewancji użytkownika. Przykładem takiego rozwiązania może być algorytm ListNet [Cao, Qin i in., 2007].

Podstawą metody jest założenie, że z predykcją list rankingowych (permutacji) dokumentów przy użyciu funkcji rankingujących związana jest niepewność. Zakłada się, że dowolna permutacja jest możliwa, ale różne permutacje mają różne prawdopodobieństwa obliczane na podstawie funkcji rankingujących.

Jeśli więc mamy zbiór dokumentów z wartościami funkcji rankingującej $\mathbf{y} = \{y_j\}$, $j = 1, \ldots, m$, gdzie $y_j = f(x_{qj})$, zaś φ jest rosnącą funkcją dodatnią, to prawdopodobieństwo dowolnej permutacji π może zostać zdefiniowane jako:

$$P(\pi|\mathbf{y}) = \prod_{j=1}^{m} \frac{\varphi\left(y_{\pi(j)}\right)}{\sum_{k=j}^{m} \varphi\left(y_{\pi(k)}\right)} \tag{2.2.148}$$

gdzie oznacza ocenę dokumentu na pozycji j w permutacji π.

W algorytmie ListNet funkcja rankingująca budowana jest na podstawie modelu neuronowego. Dla danego zapytania q sieć neuronowa f generuje oceny dokumentów. Następnie wybieranych jest k najwyżej ocenionych dokumentów i tworzona jest podgrupa możliwych rankingów G_k, zawierająca wszystkie permutacje zawierające na k najwyższych miejscach najwyżej ocenione dokumenty.

Dla każdej permutacji $g \in G_k$, prawdopodobieństwo tej permutacji obliczane jest przy pomocy wzoru (2.2.148) dla k najlepszych dokumentów, z zastosowaniem jako φ funkcji wykładniczej:

$$P(g|f) = \prod_{t=1}^{k} \frac{\exp f\left(x_{qj_t}\right)}{\sum_{l=t}^{m} \exp f\left(x_{qj_l}\right)} \tag{2.2.149}$$

Jako funkcja kosztu uczenia dla zapytania q wykorzystywana jest funkcja entropii, opartej na dywergencji Kullbacka-Leiblera:

$$-\sum_{g \in G_k} P\left(g|r_q\right) \ln P(g|f) \tag{2.2.150}$$

Podobne rozwiązania [Liu, 2011] znaleźć można w algorytmach ListMLE [Xia, Liu i in., 2008] i BolzRank [Volkovs, Zemel, 2009].

Literatura

Acid S., de Campos L.M., Fernandez-Luna J.M., Huete J.F. (2003), *An information retrieval model based on simple Bayesian networks*, „International Journal of Intelligent Systems", vol. 18, s. 251–265.

Ahmad N., Beg M. (2002), *Fuzzy logic based rank aggregation methods for the World Wide Web*, „Proc. of the Intl. Conference on Artificial Intelligence in Engineering and Technology (ICAIET02)", s. 363–368.

Ailon N. (2009), *A Simple Linear Ranking Algorithm Using Query Dependent Intercept Variables*, [w:] M. Boughanem, C. Berrut, J. Mothe, C. Soule-Dupuy (red.) *Advances in Information Retrieval (ECIR 2009)*, „Lecture Notes in Computer Science", vol. 5478, Springer, s. 685–690.

Aslam J.A., Montague M. (2001), *Models for metasearch*, „Proc. of the 24th ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR 01)", s. 276–284.

Baeza-Yates R., Ribeiro-Neto B. (1999), *Modern information retrieval*, Addison Wesley, Harlow.

Beg M.M.S. (2004), *Parallel rank aggregation for the World Wide Web*, „World Wide Web Journal", vol. 6, no. 1, s. 5–22.

Belew R.K. (1987), *A Connectionist Approach to Conceptual Information Retrieval*, „Proceedings of the International Conference on Artificial Intelligence and Law", Baltimore, s. 116–126.

Belew R.K. (1989), *Adaptive information retrieval: Using a connectionist representation to retrieve and learn about documents*, „Proc of the ACM SIGIR conference on Research and Development in Information Retrieval", s. 11–20.

Belew R.K. (2000), *Finding out about. A cognitive perspective on search engine technology and the WWW*, Cambridge University Press, Cambridge.

Berger H., Dittenbach M., Merkl D. (2004), *An Adaptive Information Retrieval System Based on Associative Networks*, *Conceptual Modelling*, „APCCM'04: Proceedings of the First Asia-Pacific Conference on Conceptual Modelling", Vol. 31, s. 18–22.

Bookstein A. (1980), *Fuzzy requests: an approach to weighted Boolean searches*, „Journal of the American Society for Information Science", vol. 31, no. 4, s. 240–247.

Bordogna G., Pasi G.A. (1993), *Fuzzy linguistic approach generalizing Boolean IR: a model and its evaluation*, „Journal of the American Society for Information Science", vol. 44, no. 2, s. 70–82.

Bordogna G., Pasi G. (1995), *Linguistic Aggregation Operators of Selection Criteria in Fuzzy Information Retrieval*, „International Journal of Intelligent Systems", vol. 10, no. 2, s. 233–248.

Bordogna G., Pasi G. (2000), *Application of Fuzzy Set Theory to Extend Boolean Information Retrieval*, [w:] F. Crestani, G. Pasi (red.), *Soft Computing in Information Retrieval. Techniques and Applications*, Springer, s. 21–47.

Boughanem M., Brini A., Dubois D. (2009), *Possibilistic networks for information retrieval*, „International Journal of Approximate Reasoning", vol. 50, s. 957–968.

Brini A., Boughanem M., Dubois D. (2005), *A model for information retrieval based on possibilistic networks*, „Proc. of the Symposium on String Processing and Information Retrieval, Buenos Aires, Argentina, November 2–4", s. 271–282.

Brini A., de Campos L.M., Dubois D., Boughanem M. (2005), *Query propagation in possibilistic information retrieval networks*, „Proc. of the Joint 4th Conference of the European Society for Fuzzy Logic and Technology and the 11th Rencontres Francophones sur la Logique Floue et ses Applications, Barcelona, Spain, September 7–9", s. 1281–1286.

Buitelaar P., Cimiano P. (red.) (2008), *Ontology Learning and Population – Bridging the Gap from Text to Knowledge*, IOS Press.

Burges, C.J. (2010), *From RankNet to LambdaRank to LambdaMART*, Microsoft Research Technical Report MSR-TR-2010-82.

Burges C.J., Ragno R., Le Q.V. (2007), *Learning to rank with nonsmooth cost functions*, „Advances in Neural Information Processing Systems 19 (NIPS 2006)", s. 395–402.

Burges C.J., Shaked T., Renshaw E., Lazier A., Deeds M., Hamilton N., Hullender G. (2005), *Learning to rank using gradient descent*, „Proc. of the 22nd International Conference on Machine Learning (ICML05)", s. 89–96.

Cambazoglu B.B., Aykanat C. (2006), *Performance of query processing implementations in ranking-based text retrieval systems using inverted indices*, „Information Processing and Management", vol. 42, no. 4, s. 875–898.

Campos L.M. de, Fernandez J.M., Huete J.F. (1998), *Query expansion in information retrieval systems using a Bayesian network-based thesaurus*, „Proc. of the 14th Uncertainty in Artificial Intelligence Conference", s. 53–60.

Campos L.M. de, Fernandez-Luna J.M., Huete J.F. (2002), *A layered Bayesian network model for document retrieval*, [w:] F. Crestani, M. Girolami, van C.J. Rijsbergen (red.), *Advances in Information Retrieval*, *Lecture Notes in Computer Science 2291*, Springer, s. 169–182.

Campos L.M. de, Fernandez-Luna J.M., Huete J.F. (2003), *The BNR model: foundations and performance of a Bayesian network-based retrieval model*, „International Journal of Approximate Reasoning", vol. 34, s. 265–285.

Cao Y., Xu J., Liu T.-Y., Li H., Huang Y., Hon H.-W. (2006), *Adapting ranking SVM to document retrieval*, „Proc. of the 29th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR06)", s. 186–193.

Cao Z., Qin T., Liu T.Y., Tsai M.F., Li H. (2007), *Learning to rank: from pairwise approach to listwise approach*, „Proc. of the 24th International Conference on Machine Learning (ICML 2007)", s. 129–136.

Carvalho V.R., Elsas J.L., Cohen W.W., Carbonel J.G. (2008), *A meta-learning approach for robust rank learning*, „SIGIR 2008 Workshop on Learning to Rank for Information Retrieval (LR4IR08)", s. 2352–2360.

Chapelle O., Keerthi S.S. (2010), *Efficient algorithms for ranking with SVMs*, „Information Retrieval Journal", vol. 13, no. 3, s. 201–215.

Chen H. (1992), *Knowledge-Based Document Retrieval: Framework and Design*, „Journal of Information Science: Principles & Practice", vol. 18, no. 3, s. 293–314.

Chen H. (1995), *Machine learning for information retrieval: neural networks, symbolic learning and genetic algorithms*, „Journal of the American Society for Information Science", vol. 46, no. 3, s. 194–216.

Chen H., Dhar V. (1991), *Cognitive Process as a Basis for Intelligent Retrieval Systems Design*, „Information Processing and Management", vol. 27, no. 5, s. 405–432.

Chu W., Ghahramani Z. (2005), *Gaussian processes for ordinal regression*, „Journal of Machine Learning Research", vol. 6, s. 1019–1041.

Chu W., Ghahramani Z. (2005a), *Preference learning with Gaussian processes*, „Proc. of the 22nd International Conference on Machine Learning (ICML05)", s. 137–144.
Chu W., Keerthi S.S. (2005), *New approaches to support vector ordinal regression*, „Proc. of the 22nd International Conference on Machine Learning (ICML 05)", s. 145–152.
Cichosz P. (2000), *Systemy uczące się*, WNT, Warszawa.
Cimiano P. (2006), *Ontology Learning and Population from Text. Algorithms, Evaluation and Applications*, Springer.
Cleverdon C.W. (1991), *The significance of the Cranfield tests on index languages*, „Proc. of the ACM SIGIR Conference on Research and Development in Information Retrieval", s. 3–12.
Cohen P.R., Kjeldsen R. (1987), *Information Retrieval by constrained spreading activation on Sematic Networks*, „Information Processing & Management", vol. 23, no. 4, s. 255–268.
Cohen W.W., Schapire R.E., Singer Y. (1998), *Learning to order things*, „Advances in Neural Information Processing Systems 10 (NIPS97)", MIT Press, s. 451–457.
Cohen W.W., Schapire R.E., Singer Y. (1999), *Learning to order things*, „Journal of Artificial Intelligence Research", vol. 10, s. 243–270.
Cooper W.S., Gey F.C., Dabney D.P. (1992), *Probabilistic retrieval based on staged logistic regression*, „Proc. of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval", s. 198–210.
Cormen T.H., Leiserson Ch.E., Rivest R.L, Stein C. (2017), *Wprowadzenie do algorytmów*, Wydawnictwo Naukowe PWN, Warszawa.
Cortes C., Mohri M., Rastogi A. (2007), *Magnitude-preserving ranking algorithms*, „Proc. of the 24th Intl Conf. on Machine Learning (ICML07)", s. 169–176.
Cossock D., Zhang T. (2006), *Subset ranking using regression*, „Proc. of the 19th Annual Conference on Learning Theory (COLT 2006)", s. 605–619.
Cowell P.G., Dawid A.P., Lauritzen S.L., Spiegelhalter D.J. (1999), *Probabilistic Networks and Expert Systems*, Springer.
Crammer K., Singer Y. (2002), *Pranking with ranking*, „Proc. of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS 01)", MIT Press, s. 641–647.
Crestani F. (1997), *Application of spreading activation techniques in information retrieval*, „Artificial Intelligence Review", vol. 11, no. 6, s. 453–482.
Crestani F., Pasi G. (1999), *Soft information retrieval: Applications of fuzzy set theory and neural networks*, [w:] N. Kasabov, R. Kozma (red.), *Neuro-Fuzzy Techniques for Intelligent Information Systems*, Springer, s. 287–315.
Crestani F., van Rijsbergen C.J. (1997), *A Model for Adaptive Information Retrieval*, „Journal of Intelligent Information Systems", vol. 8, s. 29–56.
Croft W., Thompson R.H. (1987), *I3R: a new approach to the design of Document Retrieval Systems*, „Journal of the American Society for Information Science", vol. 38, no. 6, s. 389–404.
Croft W., Lucia T., Cohen P. (1988), *Retrieving documents by plausible inference: a preliminary study*, „Proc. of ACM SIGIR conference on Research and Development in Information Retrieval", s. 481–494.
Croft W., Metzler D., Strohman T. (2009), *Search Engines: Information Retrieval in Practice*, Pearson.

Croft W., Lucia T., Crigean J., Willet P. (1989), *Retrieving documents by plausible inference: an experimental study*, „Information Processing & Management", vol. 25, no. 6, s. 599–614.
Dominich S. (2001), *Mathematical Foundations of Information Retrieval*, Kluwer.
Dominich S. (2008), *The Modern Algebra of Information Retrieval*, Springer Verlag, Berlin–Heidelberg.
Donmez P., Svore K.M., Burges C.J. (2009), *On the local optimality of Lambda Rank*, „Proc. of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009)", s. 460–467.
Dwork C., Kumar R., Naor M., Sivakumar D. (2001), *Rank aggregation methods for the web*, „Proceedings of the 10th International Conference on World Wide Web (WWW01)", s. 613–622.
Fagin R., Kumar R., Sivakumar D. (2003), *Efficient similarity search and classification via rank aggregation*, „Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD03)", s. 301–312.
Fox E.A. (1987), *Development of the Coder system: A testbed for artificial intelligence methods in information retrieval*, „Information Processing & Management", vol. 23, no. 4, s. 341–366.
Frakes W.B., Baeza-Yates R. (1992), *Information Retrieval: Data Structures & Algorithms*, Prentice Hall.
Freund Y., Iyer R., Schapire R., Singer Y. (2003), *An efficient boosting algorithm for combining preferences*, „Journal of Machine Learning Research", vol. 4, s. 933–969.
Fuhr N. (1989), *Optimum polynomial retrieval functions based on the probability ranking principle*, „ACM Transactions on Information Systems", vol. 7, no. 3, s. 183–204.
Gey F.C. (1994), *Inferring probability of relevance using the method of logistic regression*, „Proc. of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval", s. 222–231.
Harrington B. (2009), *ASKNet: Automatically Creating Semantic Knowledge Networks from Natural Language Text*, PhD Thesis, University of Oxford.
Harrington B. (2010), *A Semantic Network Approach to Measuring Relatedness*, „Proc. of the 23rd International Conference on Computational Linguistics (COLING 2010), Beijing China".
Harrington B., Clark S. (2009), *ASKNet: Creating and Evaluating Large Scale Integrated Semantic Networks*, „International Journal of Semantic Computing", vol. 2, no. 3, s. 343–364.
Harrington E.F. (2003), *Online ranking/collaborative filtering using the perceptron algorithm*, „Proceedings of the 20th International Conference on Machine Learning (ICML 2003)", s. 250–257.
Herbrich R., Graepel T., Obermayer K. (1999), *Support Vector Learning for Ordinal Regression*, „Proceedings of the 9th International Conference on Artificial Neural Networks, Edinburgh", s. 97–102.
Herbrich R., Obermayer K., Graepel T. (2000), *Large margin rank boundaries for ordinal regression*, [w:] A.J. Smola, P.L. Bartlett, B. Schölkopf, D. Schuurmans (red.), *Advances in Large Margin Classifiers*, MIT Press, Cambridge, MA, s. 115–132.
Jensen F.V., Nielsen T.D. (2007), *Bayesian Networks and Decision Graphs*, Springer.
Jo H. (2019), *Text Mining. Concepts, Implementation, and Big Data Challenge*, Springer.

Joachims T. (2002), *Optimizing search engines using clickthrough data*, „Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD02)", s. 133–142.
Joachims T. (2006), *Training linear SVMs in linear time*, „Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD06)", s. 217–226.
Kjærulff U.B, Madsen A.L. (2013), *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*, Springer.
Koronacki J., Ćwik J. (2015), *Statystyczne systemy uczące się*, Exit, Warszawa.
Koski T., Noble J.M. (2009), *Bayesian Networks. An Introduction*, Wiley.
Kowalski G. (2011), *Information Retrieval. Architecture and Algorithms*, Springer.
Li P., Burges C., Wu Q. (2008), *McRank: learning to rank using multiple classification and gradient boosting*, „Advances in Neural Information Processing Systems 20 (NIPS 07)", s. 845–852.
Liu T.-Y. (2009), *Learning to Rank for Information Retrieval*, „Foundation and Trends in Information Retrieval", vol. 3, no. 3, s. 225–331.
Liu T.-Y. (2011), *Learning to Rank for Information Retrieval*, Springer.
Manning C.D., Raghavan P., Shütze H. (2007), *An introduction to information retrieval*, Cambridge University Press.
Matveeva I., Burges C., Burkard T., Laucius A., Wong L. (2006), *High accuracy retrieval with multiple nested ranker*, „Proc. of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR06)", s. 437–444.
Mitra B., Craswell N. (2017), *Neural Models for Information Retrieval*, arXiv:1705.01509.
Mitra B., Craswell N. (2018), *An Introduction to Neural Information Retrieval*, „Foundations and Trends in Information Retrieval", vol. 13, no. 1, s. 1–126.
Miyamoto S. (1990), *Fuzzy Sets In Information Retrieval And Cluster Analysis*, Kluwer.
Mozer M.C. (1984), *Inductive Information Retrieval using parallel distributed computation*, Technical Report, Institute for Cognitive Science, University of California, San Diego, USA.
Nallapati R. (2004), *Discriminative models for information retrieval*, „Proc. of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval", s. 64–71.
Neapolitan R.E. (2003), *Learning Bayesian Networks*, Prentice-Hall.
Onal K.D., Zhang Y., Altingovde I.S., Rahman M.M., Karagoz P., Braylan A., Dang B., Chang H.L., Kim H., McNamara Q., Angert A., Banner E., Khetan V., McDonnell T., Nguyen A.T., Xu D., Wallace B.C., de Rijke M., Lease M. (2018), *Neural Information Retrieval: At the End of the Early Years*, „Information Retrieval", vol. 21, no. 2–3, s. 111–182.
Pantel P., Pennacchiotti M. (2008), *Automatically Harvesting and Ontologizing Semantic Relations*, „Proc. of the 2008 conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge", IOS Press, s. 171–195.
Pearl J. (1988), *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann.
Qin T., Liu T.-Y., Lai W., Zhang X.-D., Wang D.-S., Li H. (2007), *Ranking with multiple hyperplanes*, „Proc. of the 30th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR07)", s. 279–286.

Qin T., Liu T.Y., Li H. (2009), *A general approximation framework for direct optimization of information retrieval measures*, „Information Retrieval", vol. 13, no. 4, s. 375–397.
Qin T., Zhang X.-D., Tsai M.-F., Wang D.-S., Liu T.-Y., Li H. (2008), *Query-level loss functions for information retrieval*, „Information Processing and Management", vol. 44, no. 2, s. 838–855.
Radecki T. (1979), *Fuzzy set theoretical approach to document retrieval*, „Information Processing and Management", vol. 15, no. 5, s. 247–260.
Rennie J.D.M., Srebro N. (2005), *Loss functions for preference levels: regression with discrete ordered labels*, „IJCAI 2005 Multidisciplinary Workshop on Advances in Preference Handling" ACM, New York.
Ribeiro-Neto B., Muntz R. (1996), *A belief network model for IR*, „Proc. of the 19th annual international ACM SIGIR Conference on Research and Development in Information Retrieval", s. 253–260.
Ribeiro-Neto B., Silva I., Muntz R. (2000), *Bayesian Network Models for Information Retrieval*, [w:] F. Crestani, G. Pasi (red.), *Soft Computing in Information Retrieval. Techniques and Applications*, Physica-Verlag, Heidelberg, s. 259–291.
Richardson S.D., Dolan W.B., Vanderwende L. (1998), *MindNet: acquiring and structuring semantic information from text*, „Proc. of the 36th Annual Meeting of the Association for *Computational Linguistics* and 17th International Conference on *Computational Linguistics*, Vol. 2, s. 1098–1102.
Rigutini L., Papini T., Maggini M., Scarselli F. (2008), *Learning to rank by a neural-based sorting algorithm*, „Proceedings of the SIGIR 2008 Workshop Learning to Rank for Information Retrieval", s. 1–8.
Rigutini L., Papini T., Maggini M., Scarselli F. (2011), *Learning to rank by a neural preference function*, „IEEE Trans. on Neural Networks", vol. 22 , no. 9, s. 1368–1380.
Robertson S.E. (1997), *The probability ranking principle in IR*, „Journal of Documentation", vol. 33, no. 4, s. 294–304, przedruk w: K. Sparck Jones, P. Willett (red.), *Readings in information retrieval*, Morgan Kaufmann, s. 281–286.
Rudin C. (2006), *Ranking with a p-norm push*, „Proc. of the 19th Annual Conference on Learning Theory (COLT06)", s. 589–604.
Rudin C. (2009), *P-norm push: A Simple Convex Ranking Algorithm that Concentrates at the Top of the List*, „Journal of Machine Learning Research", vol. 10, s. 2233–2271.
Salton G. (1968), *Automatic information organization and retrieval*, McGraw-Hill.
Salton G. (1991), *The smart project in automatic document retrieval*, „Proceedings of the 14th ACM SIGIR Conference on Research and development in information retrieval", s. 356–358.
Salton G., Buckley C. (1987), *Term-weighting approaches in automatic text retrieval*, „Information Processing and Management", vol. 24, no. 5, s. 513–523.
Salton G., McGill M.J. (1983), *Introduction to modern information retrieval*. McGraw-Hill.
Salton G., Wong A., & Yang C.S. (1975), *A vector space model for automatic indexing*, „Communications of the ACM", vol. 18, no. 11, s. 613–620.
Sanderson M., Croft W.B. (1999), *Deriving concept hierarchies from text*, „Proc. of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval", s. 206–213.
Shashua A., Levin A. (2003), *Ranking with large margin principles: two approaches*, „Advances in Neural Information Processing Systems" 15, s. 937–944.

Shoval P. (1981), *Expert/consultation system for a retrieval data-base with semantic network of concepts*, „ACM SIGIR Forum", vol. 16, no. 1, s. 145–149.
Shoval P. (1985), *Principles, procedures and rules in an expert system for information retrieval*, „Information Processing & Management", vol. 21, no. 6, s. 475–487.
Sparck Jones K. (1991), *The role of Artificial Intelligence in information retrieval*, „Journal of the American Society for Information Science", vol. 42, no. 8, s. 558–565.
Taylor M., Guiver J., Robertson S., Minka T. (2008), *Softrank: optimising non-smooth rank metrics*, „Proc. of the 1st International Conference on Web Search and Web Data Mining (WSDM 2008)", s. 77–86.
Turtle H. (1991), *Inference Networks for Document Retrieval*, Ph.D. Thesis, University of Massachusetts.
Turtle H., Croft W. (1990), *Inference networks for document retrieval*, „Proc. of the 13th ACM SIGIR Conference on Research and Development in Information Retrieval", s. 1–24.
Turtle H., Croft W.B. (1991), *Evaluation of an inference network-based retrieval model*, „ACM Transactions on Information Systems", vol. 9, no. 3, s. 187–222.
Volkovs M.N., Zemel R.S. (2009), *Boltzrank: learning to maximize expected ranking gain*, „Proc. of the 26th International Conference on Machine Learning (ICML 2009)", s. 1089–1096.
Wilkinson R., Hingston P. (1991), *Using the cosine measure in a neural network for document retrieval*, „Proc. of the ACM SIGIR Conference on Research and Development in Information Retrieval", s. 202–210.
Wu G., Li J., Feng L., Wang K. (2008), *Identifying potentially important concepts and relations in an ontology*, „Proc. of the 7th International Semantic Web Conference, Lecture Notes in Computer Science", vol. 5318, Springer, s. 33–49.
Xia F., Liu T.Y., Wang J., Zhang W., Li H. (2008), *Listwise approach to learning to rank – theorem and algorithm*, „Proc. of the 25th International Conference on Machine Learning (ICML 2008)", s. 1192–1199.
Xu J., Li H. (2007), *Adarank: a boosting algorithm for information retrieval*, „Proc. of the 30th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)", s. 391–398.
Yager R.R. (1987), *A note on weighted queries in information retrieval systems*, „Journal of the American Society for Information Science", vol. 38, no. 1, s. 23–24.
Yager R.R. (1988), *On ordered weighted averaging aggregation operators in multi-criteria decision making*, „IEEE Trans. on Systems, Man and Cybernetics", vol. 18, no. 1, s. 183–190.
Yager R.R. (1996), *Quantifier guided aggregation using OWA operators*, „International Journal of Intelligent Systems", vol. 11, no. 1, s. 49–73.
Yager R.R. (2000), *A Framework for Linguistic and Hierarchical Queries in Document Retrieval*, [w:] F. Crestani, G. Pasi (red.), *Soft Computing in Information Retrieval. Techniques and Applications*, Springer, s. 3–20.
Zadrożny S., Kacprzyk J. (2005), *An Extended Fuzzy Boolean Model of Information Retrieval Revisited*, „Proc. of the 14th IEEE International Conference on Fuzzy Systems", s. 1020–1025.
Zhang Y., Rahman M., Braylan A., Dang B., Chang H.-L., Kim H., McNamara Q., Angert A., Banner E., Khetan V., McDonnell T., Nguyen A.T., Xu D., Wallace B., Lease M. (2017), *Neural Information Retrieval: A Literature Review*, Raport Techniczny, arXiv:1611.06792.

Spis rysunków

Rozdział 3

Nowe technologie w sieci Internet – krótka charakterystyka i możliwe zastosowania

3.1. Business Intelligence

W 2002 roku jedna z popularniejszych sieci amerykańskich marketów Target, postawiła przed sobą nowy cel, który miał zagwarantować jej wielomilionowe dochody. Analitycy firmy słusznie zauważyli, że pary, którym właśnie urodziło się dziecko stanowią bardzo dochodową grupę konsumencką. Rodziny takie na przykład w mniejszym bądź większym stopniu zmieniają swoje nawyki konsumenckie, przechodząc na zdrowszą, a tym samym droższą, żywność. Zmianie ulega także ilość środków wydawanych na rodzinę. Ponieważ nie byli jedynymi, którzy doszli do takich wniosków, dlatego też należało zrobić coś, co pozwoliłoby wyprzedzić konkurencję w walce o „rodzinnego" klienta.

Pomysłem, który miał pomóc w realizacji tego celu, była próba „odgadnięcia" na podstawie nawyków zakupowych klientek, która z nich była spodziewała się dziecka. Zbierając dane zakupowe, które firma zbierała przy pomocy specjalnych kart kredytowych, które wcześniej wręczano klientom[1], analitycy Target odkryli pewne zależności. Kobiety, które spodziewały się dziecka, diametralnie zmieniały swoje nawyki żywieniowe, skłaniając się ku zdrowszej, bogatej w magnes czy żelazo diecie.

1 Chodzi tutaj o karty, które kupujący daje sprzedawcy podczas dokonywania płatności. Są one przeznaczone nie tylko do uiszczania opłat. Stanowią swojego rodzaju wizytówkę posiadacza. Dzięki temu sklep wie, jakie produkty zostały przez daną osobę nabyte czy za jaką kwotę. Na podstawie historii transakcji można wysyłać indywidualne oferty z promocjami dla zachęcenia do kolejnej wizyty.

Aby pokazać, jak bardzo wnikliwe były te analizy, Charles Duhigg opisał pewien przypadek, gdy pracownicy z Target wysłali do jednej z klientek katalog z artykułami dla młodych matek. Niedługo po tym, jak pakiet reklamowy trafił do domu adresata, menedżer firmy odebrał telefon. Rozmówcą okazał się ojciec dziewczyny (jak się okazało, dziewczyna była jeszcze uczennicą liceum), który w dosadny sposób skrytykował postępowanie firmy, skarżąc się jednocześnie, że tak agresywnym marketingiem, wręcz zachęcają ją do zajścia w ciążę. Jak pisze dalej Duhigg, menedżer wylewnie przeprosił, po czym po kilku dniach ponownie zadzwonił, żeby przeprosić raz jeszcze. Ku jego zaskoczeniu, zamiast kolejnej fali oskarżeń to ojciec dziewczyny przeprosił za swój wcześniejszy wybuch, ponieważ rozmawiał z córką, która rzeczywiście spodziewała się dziecka [Duhigg, 2012].

Historia opisana powyżej stanowi idealny przykład Business Intelligence, czyli procesu wykorzystania strategii i technologii do zebrania danych w informacje, które można wykorzystać do zoptymalizowania procesu zarządzania, zwiększenia konkurencyjności, a tym samym zysków.

W dzisiejszych czasach liczba informacji rośnie cztery razy szybciej niż światowa gospodarka, a moc obliczeniowa komputerów dziesięć razy szybciej. Trudno się dziwić, że ludzie zaczynają narzekać na przeładowanie informacjami [Mayer-Schonberge, Cukier, 2014]. Dzieje się tak ponieważ znacznie wzrosła sama ilość źródeł, z których dane można pobierać.

Można do nich zaliczyć chociażby wspomniane wyżej dane sprzedażowe, czyli pośrednio generowane przez ludzi. Jednakże dochodzą do tego także te, które są w całości wytwarzane przez maszyny. Przykładem tutaj mogą być chociażby stosowane przez jedną z firm przewozowych, czujniki instalowane w silnikach, które badają szczegółowo temperaturę, częstotliwość drgań, poziom oleju itp. Informacje te są zbierane, a następnie wysyłane do kontroli, dzięki czemu praca silnika jest monitorowana 24 godziny na dobę.

Nasze komputery osobiste również generują dane, czy to w formie logów, czy też plików cookies (tzw. ciasteczek), które związane są z naszą aktywnością w sieci.

Opisane powyżej dane mają charakter strukturalny i można je łatwo zmierzyć, np. temperatura silnika będzie podana w stopniach Celsjusza, a nasza aktywność w Internecie będzie badana przy pomocy tematyki stron, które odwiedzamy, dlatego też dane takie nazywamy danymi ustrukturyzowanymi.

Web 2.0 (a już niedługo Web 3.0) definiujący współczesną wersję Internetu, którego podstawowymi cechami są; interaktywność, ciągła

obecność w sieci dużych grup internautów (zwłaszcza dzięki portalom społecznościowym), możliwość współtworzenia treści, rozwój blogosfery powszechna dostępność do sieci [Gonciarski, 2010], Wstawiając zdjęcie z wakacji na Instagramie, udostępniając filmik na Facebooku, pisząc komentarz na Twitterze, tworzymy informacje, które nas w pewnym stopniu definiują.

Łatwo zauważyć, że mamy tutaj do czynienia z dwoma typami informacji. Pierwszy to dane generowane świadomie przez człowieka, które monitorują jego aktywność w Internecie. Drugi to informacje wysyłane do sieci, generowane przez urządzanie. Można powiedzieć, że stąd właśnie pochodzi określenie Internet rzeczy (ang. *Internet of Things*) [Thomas, McSharry, 2015]. Wiesław Gonciarski w swojej pracy z kolei pisze, że Internet rzeczy to bardziej koncepcja niż technologia, gdyż wykorzystuje różne rozwiązania technologiczne, umożliwiające urządzeniom komunikowanie się między sobą, a następnie analizowanie różnorodnych danych, by samodzielnie podejmować decyzje i inicjować działanie tych urządzeń lub innych połączonych do sieci [Gonciarski, 2017].

Biorąc pod uwagę fakt, że informacji stale przybywa, Internet z jednej strony staje się źródłem niemal nieograniczonej liczby zasobów z różnych kanałów oraz w różnych formach. Stawia to pewne wyzwanie, mianowicie zebrania i wyfiltrowania z tego morza informacyjnego tylko danych, które można wykorzystać. Wszystko to składa się na definicję terminu, jakim jest Big Data, czyli technologii, która stworzona jest właśnie jako remedium na przedstawiony wyżej problem. Przedsiębiorstwo META Group zdefiniowało Big Data jako koncepcję 3V:

Wielkość (Volume)

Szybkość (Velocity)

Różnorodność (Variety)

Główną wartością Big Data jest zatem nie sam zbiór danych, ale konsekwencje jego wykorzystania [Mazurek, 2019]. Umiejętne wykorzystanie danych, które codziennie dostarcza Internet, z różnych jego kanałów, czyli zasobów wewnętrznych (np. historie sprzedaży internetowej oraz stacjonarnej), jak również zewnętrznych (obserwowanie użytkowników portali społecznościowych, ich zainteresowań, nawyków itp.), pozwala w łatwy sposób dostarczyć informacji o aktualnych preferencjach konsumenckich oraz w miarę możliwości przewidywać je.

Ponieważ, jak wspomnieliśmy już wcześniej, materiały do analizy pochodzą z różnych źródeł, nie jest już zatem możliwe tradycyjne gromadzenie danych w osobnych bazach. Tracąc bowiem czas na samą ich integrację przy tak szybkim wzroście transferu danych w Internecie, ryzykujemy, że

otrzymane rezultaty ich przetwarzania będą już mocno nieaktualne. Dlatego też wraz z końcem XX wieku można zaobserwować znaczny wzrost zainteresowania technologią chmury obliczeniowej. National Institute of Standards and Technology (NIST) definiuje chmurę jako model umożliwiający wszechobecny, wygodny dostęp sieciowy do wspólnych, konfigurowalnych zasobów (np. sieci, serwerów, dysków danych, aplikacji oraz usług) [Mell, Grance, 2019]. Kumulacja danych z kilku źródeł stanowi niewątpliwie zaletę takiego rozwiązania. Dodatkowym plusem wykorzystania chmury obliczeniowej jest zwiększenie lub zmniejszenie zapotrzebowania na moc obliczeniową [Czerwonka, 2016]. Z rozwiązań chmurowych korzystają na przykład firmy specjalizujące się w branży RPA (*Robotic Process Automation*) właśnie dzięki możliwościom, jakie dają narzędzia do tworzenia robotów, jak chociażby Blue Prism czy UiPath. Dzięki temu stworzone roboty nie są przypisane bezpośrednio do konkretnej maszyny i mogą być uruchamiane (oraz kontrolowane) z dowolnego miejsca na ziemi przy pomocy wirtualnych pulpitów.

Rozwój technologii BI pociąga za sobą konieczność stworzenia nowych algorytmów i technologii, które będą w stanie poradzić sobie z opisanym powyżej wyzwaniem. Coraz częściej w tego typu analizach wykorzystuje się sztuczną inteligencję.

3.2. Nowy model interfejsów

3.2.1. Rozszerzona rzeczywistość

W 2012 roku firma Google zapowiedziała swój najnowszy produkt. Google Glass z charakterystycznym prostopadłościanem zamiast tradycyjnych szkieł, miało stanowić pierwszą próbę stworzenia czegoś, co dziś określa się Rzeczywistością Rozszerzoną (AR – A*ugmented Reality*). Jest to technologia, której przeznaczeniem jest integracja świata wirtualnego ze światem realnym.

W pierwszej kolejności dziedzina ta kojarzyć się może z przemysłem rozrywkowym, a dokładniej rynkiem gier komputerowych. Porównanie to jest jak najbardziej trafne. Znana firma branży gier konsolowych Nintendo wydała produkt bazujący na popularnej marce japońskich stworzeń „Pokemon GO". Gra polegała na tym, że gracz z uruchomioną grą na telefonie (bądź tablecie) oraz włączoną opcją lokalizacji mógł w konkretnych miejscach spotkać wirtualne stwory i próbować je złapać.

System gry bazuje na technologii GPS. Twórcy umieścili w aplikacji informacje na temat współrzędnych, gdzie znajdują się tzw. „punkty spotkań". Gdy użytkownik znajdzie się w pobliżu takiego punktu, wówczas gra wywołuje określone zdarzenie przypisane danemu punktowi na telefonie użytkownika (np. pojedynek z danym stworem).

Jest to tylko jeden z przykładów, a jak w swoim artykule napisał Norbert Biedrzycki, AR to nie tylko Pokemon GO [Biedrzycki, 2018]. Należy także pamiętać, że rozszerzona rzeczywistość nie ogranicza się tylko do obrazu, ale również obejmuje dźwięki czy wrażenia dotykowe. Daje to szeroki wachlarz możliwości, dzięki czemu technologia ta znajduje szerokie zastosowanie także w innych dziedzinach, spośród których poniżej wymienimy kilka przykładowych.

- **Logistyka –** wszystkie aplikacje GPS, które z jednej strony pokazują w miarę możliwości nasze jak najdokładniejsze położenie, ale również wyznaczają optymalną trasę podróży z jednego punktu do drugiego. Kolejnym przykładem jest zastosowanie w transporcie, gdzie za pomocą wirtualnego modelu paczek jest możliwe wizualizowanie ich rozmieszczenia np. na samochodzie.
- **Projektowanie** – AR jest bardzo przydatną technologią podczas projektowania wnętrz. Znana sieć sklepów meblowych we współpracy z Apple stworzyła aplikację, dzięki której klient może sprawdzić, czy dany mebel pasuje do danego pomieszczenia. Tworzy się wirtualny model takiego mebla w 3D, następnie bardzo dokładnie są wprowadzane jego parametry. W ten sposób powstaje wirtualny odpowiednik towaru, który następnie renderowany na tablecie w czasie rzeczywistym, przy wykorzystaniu technologii aparatu urządzenia, jest nanoszony na realne tło, które ów aparat obejmuje. Daje to złudzenie, jakby dany przedmiot znajdował się w tym miejscu fizycznie. Pozwala to łatwo określić czy dany mebel dobrze się prezentuje w określonej scenerii.
- **Edukacja** – Rzeczywistość rozszerzona może znacznie przyczynić się do rozwoju takich gałęzi medycyny jak chirurgia czy neuroanatomia, ponieważ AR oferuje możliwość wzbogacenia świata rzeczywistego o wirtualne informacje sensoryczne, takie jak dźwięk, dotyk lub obrazy [Henssen, van den Heuvel i in., 2020]. Nietrudno zatem wyobrazić sobie, jak nauka w niedalekiej przyszłości będzie wykorzystywała tę technologię do nauki, tak jak dzisiaj wykorzystuje się komputery.
- **Turystyka** – W miejscowości Oblęgorek pod Kielcami znajduje się muzeum Henryka Sienkiewicza, w którym zwiedzający może mieć

swojego osobistego przewodnika w postaci słuchawek i odtwarzacza dźwiękowego. Wykorzystywana jest tutaj technologia Bluetooth. W urządzeniu odtwarzającym został umieszczony odbiornik, który reaguje na nadajniki zamieszczone w wejściu do kolejnych pomieszczeń. Gdy tylko zwiedzający znajdzie się w odpowiedniej odległości od nadajnika, ten przesyła sygnał, do odbiornika z informacją, który plik z nagraniem ma się w danej chwili odtwarzać. Pójściem krok dalej jest stworzenie odpowiedniego oprogramowania, dzięki którym wraz z wykorzystaniem specjalistycznych gogli możliwe będzie na bieżąco otrzymywanie interesujących informacji na wyświetlaczu przed oczami użytkownika.

3.2.2. Wspomagane wyszukiwanie

Oczywiście opisana powyżej technologia, podobnie jak rzeczywistość wirtualna (*Virtual Reality*, *VR*) to nie jedyne rozwiązania wzbogacające zdobywanie informacji. Przykładem może być tu algorytm PageRank firmy Google, który dostosowuje wyniki wyszukiwania interesujących nas treści do popularności stron. Popularność wynika z jednej strony bezpośrednio z treści zawartych na samej stronie, jak również z zamieszczenia odnośników do niej na innych stronach. Im więcej linków prowadzi do danej strony, tym wyższy PageRank danej strony. Uwzględniany jest także wskaźnik popularności stron, na których taki odnośnik się znajduje. Suma tych oraz kilku dodatkowych czynników składa się na rezultat wyszukiwania, czyli miejsca na liście stron o danej tematyce.

Kolejnym elementem pomagającym w wyszukiwaniu informacji są podpowiedzi generowane przez wyszukiwarkę podczas wpisywania hasła (rysunek 3.2.1).

Podpowiedzi mają kilka źródeł. Pierwszym z nich jest nasza własna historia wyszukiwania, zapisywana w historii przeglądarki oraz ciasteczkach (*cookies*). Kolejnym źródłem są wyszukiwania innych użytkowników. Wyniki są rejestrowane i zapisywane na serwerze, następnie za każdym razem, gdy użytkownik wpisuje daną frazę, automatycznie zaciągane są najbardziej popularne wyniki dla danej frazy i wyświetlane w formie podpowiedzi. Dla zwiększenia efektywności działania przeglądarki, Google dało możliwość zgłoszenia danego wyszukiwania jako nieodpowiedniej podpowiedzi. Należy pamiętać o tym, że podpowiedzi w różnych okresach mogą być różne. To, że jednego dnia dana fraza miała takie a nie inne wyniki, nie oznacza, że za tydzień również będą takie same,

ponieważ przy ich doborze są także uwzględniane trendy. Trendy czyli frazy cieszące się w danym czasie największą popularnością[2].

Rys. 3.2.1. Przykład podpowiedzi wyświetlanych dla hasła „Londyn" w wyszukiwarce Google

Źródło: Google.pl.

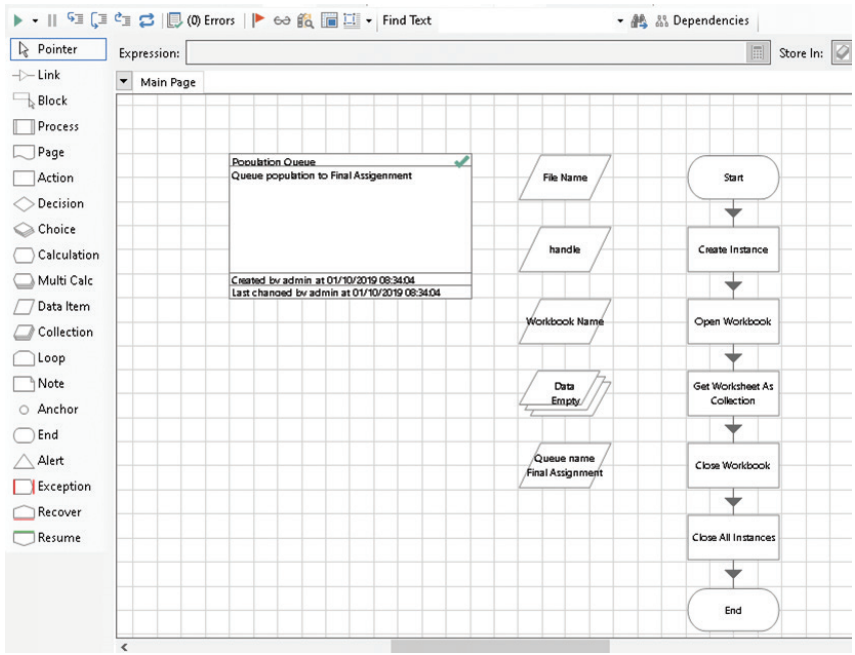
Metody opisane powyżej znacznie usprawniają wynajdywanie informacji w sieci, jednak o ile są to ułatwienia przewidziane dla człowieka, powoli zaczęto rozwijać także technologię, która może w pełni być wykorzystana przez użytkowników wirtualnych (boty). Projekt ten zwany „Semantic Web" zostanie szerzej opisany w rozdziale 3.4.

3.2.3. Robotyzacja procesów biznesowych

Robotic Process Automation (w skrócie RPA) kiedyś uznawany był za przelotną modę, teraz stanowi coraz szybciej rozwijającą się dziedzinę w branży usługowej. Trzon tej dziedziny stanowią specjalnie stworzone do tego środowiska deweloperskie, które w założeniu mają swobodnie kooperować z innymi aplikacjami. Stworzone w tej technologii roboty mają za zadanie dokładne odtwarzanie pracy człowieka, jednak błędem jest łączenie tego typu wirtualnego pracownika z pojęciem sztucznej

2 Google pod adresem https://trends.google.com/trends/?geo=US udostępniło wgląd w informacje, które trendy są w danej chwili najbardziej popularne.

inteligencji. Jeśli mielibyśmy rozdzielić rodzaje oprogramowania na dwa typy [Zieliński, 2000], na programy tradycyjne i programy AI, to roboty (w swojej zdecydowanej większości) będą się zawierały jeszcze w tej pierwszej kategorii.



Rys. 3.2.2. Przykład prostego robota, którego zadaniem jest otwieranie pliku Excel oraz kopiowanie jego zawartości do kolekcji (odpowiednik tablic w programowaniu tradycyjnym). Gdy tylko dane zostaną zapisane, program zamyka Excela i zamyka jego instancję

Źródło: opracowanie własne.

Blue Prism to obecnie jedna z popularniejszych technologii branży RPA, stworzona w środowisku .Net. Zawiera ona specjalnie stworzone biblioteki, które dają możliwość po pierwsze identyfikacji elementów aplikacji, której czynności robot ma automatyzować, a w dalszej części manipulowanie nimi w taki sam sposób, w jaki może to robić człowiek. Sam proces tworzenia robota, w przypadku Blue Prism, sprowadza się do układania, w przeznaczonym do tego studio, algorytmu ze specjalnie zaprogramowanych bloków (rysunek 3.2.2). Programowanie takie nazywa się programowaniem blokowym.

Oczywiście programowanie blokowe stanowi tylko jeden ze sposobów tworzenia robotów. Alternatywna dla Blue Prism technologia o nazwie

Automation Anywhere (AA) wykorzystuje gotowe skrypty, które układane jeden po drugim tworzą złożone instrukcje, wykonywane przez robota.

W dalszym jednak ciągu roboty poruszają się tylko po tym obszarze, który wyznaczy im człowiek, nie są bowiem zdolne do samodzielnego podejmowania decyzji. Jasno określone reguły oraz przewidziane możliwości wyboru to dwa z trzech czynników (trzecim jest powtarzalność wykonywanych czynności), aby można było mówić o możliwości zautomatyzowania procesu.

Jednakże, jak już wspomniano wcześniej, roboty mają możliwość współpracowania prawie z każdą aplikacją, także z tymi, które wykorzystują technologię AI. Przykładem tutaj mogą być inteligentne aplikacje OCR (*optical character recognition*), które służą do rozpoznawania tekstu na przykład z plików PDF. Innym przykładem są aplikacje analizujące gromadzone dane. Robot, jeśli automatyzowany proces to przewiduje, może stanowić nić łączącą działanie kilku lub nawet kilkunastu aplikacji o różnym przeznaczeniu, tworząc tym samym potężne oprogramowanie, którego moc obliczeniowa równa jest maszynie, na której jest uruchomiony.

Już dziś daje to ogromne możliwości, w wyręczaniu człowieka z wielu żmudnych oraz czasochłonnych czynności. Program nie tylko wykona je dokładniej (poprzez wyeliminowanie czynnika ludzkiego), ale również szybciej. Odciążając w ten sposób człowieka, który ma tym samym możliwości rozwoju między innymi poprzez bardziej efektywne wykorzystanie swoich talentów.

3.3. Inteligentne roboty i agenty sieciowe Chat-boty

Postęp technologiczny jest nieodłącznym elementem rozwoju ludzkości. Od wynalezienia pierwszych narzędzi, do rewolucji przemysłowej, której zalążkiem było stworzenie pierwszej maszyny parowej, minęło nieporównywalnie więcej czasu niż od połowy XIX wieku do obecnej epoki, internetowej. Jeremy Rifkin w swojej książce *Koniec pracy* nazwał to zjawisko rewolucją technologiczną [Rifkin, 2001].

W roku 1950 brytyjski matematyk Alan Turing stworzył słynny test. Polegał on na sprawdzeniu, czy przez odpowiednio długi czas maszyna jest zdolna prowadzić konwersację z człowiekiem w taki sposób, aby ten nie zorientował się, że jego rozmówcą nie jest człowiek. O ile w tamtych

czasach idea maszyny potrafiącej prowadzić konwersację z człowiekiem znajdowała się jedynie w sferze marzeń i naukowych dociekań, o tyle dziś staje się zjawiskiem jak najbardziej realnym.

Coraz częściej można spotkać się ze zjawiskiem, które potocznie określa się mianem „chatbota". Chatbot, inaczej „elektroniczny rozmówca", to aplikacja komputerowa, która posługując się elementami języka naturalnego, projektowana jest z myślą o prowadzeniu konwersacji z człowiekiem. Jeżeli mielibyśmy zdefiniować czym jest język naturalny, posłużymy się definicją profesora Malinowskiego z Uniwersytetu Łódzkiego, który w swojej pracy opisuje język naturalny jako język, którego reguły zostały ukształtowane zwyczajowo w sposób spontaniczny. Ma otwarty i płynny słownik, stale podlega modyfikacjom znaczeniowym i składniowym. Za zmiany i rozwój języka naturalnego odpowiedzialne są liczne czynniki, w tym zmiany społeczne i kulturowe [Malinowski, 2010].

Ze względu na różnorodność pod względem konstrukcji jak również działania, chatboty możemy podzielić na proste oraz złożone. Te pierwsze mają zakodowane wysyłanie prostych powiadomień mailowych czy też wiadomości sms. Zostały one stworzone głównie w celach informacyjnych np. o promocjach. Bardzo często w korespondencji mailowej można natknąć się na SPAM, który w pierwszej chwili, z racji skomponowanej treści, może sprawiać wrażenie, że został skonstruowany przez człowieka. Jest to najbardziej prymitywna forma chatbota, którego rola ogranicza się tylko do wysyłania gotowych wiadomości. Dla zwiększenia wiarygodności niektórzy twórcy pokusili się o nadanie im bardziej „ludzkiej" formy, poprzez celowe umieszczanie w treści komunikatu błędów interpunkcyjnych lub składniowych, co ma sugerować, że taka wiadomość została skonstruowana i wysłana przez człowieka.

Bardziej złożone boty są w stanie prowadzić proste konwersacje i te również można podzielić na dwie grupy. Pierwsze z nich bazują na słowach kluczowych lub jasno sprecyzowanych zwrotach.

Gdy użytkownik posłuży się zwrotem, który został przewidziany przez twórcę, na przykład powita robota słowem „Witaj", aplikacja jest w stanie odpowiedzieć poprzez wyświetlenie frazy, która została zaimplementowana jako odpowiedź np. „Dzień dobry". Tego typu chatboty działają w oparciu o ograniczoną bazę danych, która sprowadzona jest do listy wyrazów i zdań oraz prostych algorytmów, które odpowiadają za cały mechanizm działania. Dodatkowo, ponieważ roboty te nie są zaprojektowanie do zdobywania doświadczenia podczas konwersacji i zamykają się w swojej ograniczonej liście odpowiedzi, po pewnym czasie stają się przewidywalne i tym samym można się ich „nauczyć".

Z technologii tej korzysta na przykład Biuro Promocji Miasta Wrocławia. Jest to prosty mechanizm, który odpowiada na pytania klientów dotyczące atrakcji turystycznych oraz ciekawych miejsc w okolicy.

Niektóre biura podróży również zaczęły korzystać z „usług" wirtualnych pracowników. Poprzez aplikację Messenger użytkownik jest w stanie za pomocą prostych zdań uzyskać interesujące go informacje.

Taka forma komunikacji z klientem jest lepsza od tradycyjnej formy, która sprowadza się do rozmowy telefonicznej z realną osobą lub poprzez wymianę korespondencji mailowej z kilku powodów:

1) Dostępność 24 godziny na dobę.
2) Odpowiedź przychodzi najdłużej po kilku sekundach.
3) Ilość rozmów prowadzonych przez aplikację w jednym momencie znacznie przewyższa ilość rozmów, które w tym samym czasie odbywają się w tradycyjnej formie.
4) Personel, który do tej pory musiał zajmować się każdym, nawet najprostszym zapytaniem od klientów, teraz może się bardziej skupić na zadaniach wymagających większej kreatywności, z którymi robot z racji swoich ograniczeń, nie byłby w stanie sobie poradzić.

Druga grupa rozwiązań z tej dziedziny to chatboty, które nastawione są na bardziej złożoną konwersację. Zamiast bazy zaimplementowanych gotowych zdań posiadają one wbudowaną bibliotekę słów oraz ich znaczenie wraz z podziałem na rodzaje, odmiany przez przypadki, jak również semantykę oraz zbiór reguł obowiązujących w języku, dla którego jest przewidziana aplikacja. Dodatkowo aplikacja taka zaopatrzona jest w pewnego rodzaju wzorce, a więc nie sztywno skonstruowane zdania, a formy, które robot przy pomocy zaawansowanych algorytmów stosuje do konstrukcji odpowiedzi i tym samym do prowadzenia konwersacji.

Prostym przykładem tego, jak funkcjonuje robot, jest zaprezentowanie przykładu, w którym zadajemy aplikacji pytanie. Można założyć, że zdanie pytające zawsze zakończone jest pytajnikiem[3]. Nasze przykładowe zapytanie brzmi:

„Gdzie jest Kraków?"

Przy pomocy wbudowanych reguł językowych, nasz robot już wie, że ma do czynienia z pytaniem. Analizując zdanie zna jego formę oraz

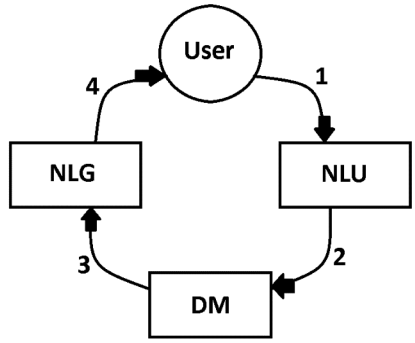
3 Autor posłużył się tutaj pewnym uproszczeniem. O ile w językach, których alfabet bazuje na łacińskim lub cyrylicy, zdanie pytające zakończone jest pytajnikiem, o tyle w językach dalekiego oraz bliskiego wschodu sprawa już nie jest tak oczywista. Dla tych języków musiałyby być napisane osobne reguły co do rozróżniania rodzajów zdań.

słowo kluczowe „Gdzie", które jasno wskazuje, że rozmówca może pytać o to, w którym miejscu znajduje się jego podmiot. Zatem aby podtrzymać kontekst wypowiedzi, robot musi w odpowiedni sposób dobrać słowa do tych, które otrzymał. Nie ma tutaj do końca znaczenia jaki dokładnie wyraz znajduje się pomiędzy słowem „gdzie" a „Kraków". Dopóki zawiera się ono w pewnej „rodzinie" słów, które wskazują na umiejscowienie czegoś, zdanie to pasuje do pewnego wzorca pytania o miejsce położenia. Podobnie jak człowiek, robot nie będzie skupiał się na różnicy między zdaniami „Gdzie jest Kraków?", „Gdzie leży Kraków?" lub „Gdzie znajduje się Kraków?", gdyż ich sens jest taki sam.

Robot otrzymując wypowiedzi od użytkownika, parsuje je (przetwarza dane wejściowe tak, aby były one „zrozumiałe" dla języka maszyny) przy pomocy modułu zwanego *Natural Language Understanding* (NLU) i wyodrębnia znaczenie wypowiedzi. Stąd dane przesyłane są dalej to modułu, który jest odpowiedzialny za prawidłowe odpowiedzi jak i prowadzenie konwersacji przez aplikację, który nazywa się *Dialogue Manager* (w dalszej części nazwa ta będzie określana skrótem DM).

DM jest ważnym modułem, którego celem jest koordynacja przepływu dialogu i komunikowanie się z innymi podsystemami i komponentami. DM jest meta-komponentem chatbota, który ułatwia interakcję między chatbotem a użytkownikiem [Galitsky, 2019] i to jego zadaniem jest właściwe dobranie odpowiedzi w taki sposób, aby wypowiedź nie tylko nawiązała sensem wypowiedzi do otrzymanej danej wejściowej, ale również został podtrzymany kontekst wypowiedzi. To oraz umiejętne formułowanie bardziej złożonych zdań są warunkami niezbędnymi do tego, aby była zachowana ciągłość konwersacji. DM tworzy zdanie-odpowiedź w języku maszynowym, który dla człowieka może być niezrozumiały.

Gdy nastąpi rozpoznanie formy zdania oraz jego sensu, robot jest gotowy do sformułowania odpowiedzi poprzez zastosowanie odpowiedniego wzorca. Aby odpowiedź była poprawna, aplikacja taka musi dysponować odpowiednią bazą wiedzy. Baza taka z jednej strony zawiera oczywiście wszelkie informacje, które są potrzebne robotowi (np. chatbot biura turystycznego zaopatrzony będzie w bazę danych, która zawiera jak najwięcej informacji o miejscach, do których wycieczki wspomniane biuro organizuje). Z drugiej zaś stanowi zbiór reguł, którymi robot posługuje się zarówno przy odczytywaniu danych wejściowych, jak i konstruowaniu danych wyjściowych. To, w jaki sposób DM zbuduje wypowiedź, będzie zależało od tego, jaka strategia została przyjęta przez twórcę, który z kolei starał się ją dobrać pod kątem funkcji jaką ma pełnić chatbot.



Rys. 3.3.1. Graficzne przedstawienia funkcjonowania chatbota. 1. Pobieranie danych wejściowych od użytkownika i przesyłanie ich do NLU, gdzie następuje konwersja z języka naturalnego na język maszynowy. 2. Parsowane dane przesyłane są do DM. Tam następuje interpretacja znaczeniowa oraz kontekstowa otrzymanego sformułowania, po czym generowana jest odpowiedź. 3. Gotowe sformułowanie przesyłane jest do NLG, gdzie z postaci maszynowej, zostaje przekonwertowane na język naturalny. 4. Gotowa odpowiedź, jako dana wyjściowa przesyłana jest do użytkownika

Źródło: opracowanie własne.

Jedna z popularniejszych metod tworzenia odpowiedzi nazywana jest „Dialog oparty na przykładach" (*Dialogue Management Based on Example--Based*) [Galitsky, 2019]. Chatboty posługujące się tą metodą mają zawarte w swojej bazie danych przykładowe wypowiedzi początkowe, które aplikacja może otrzymać od użytkownika, oraz przypisane do nich odpowiedzi. Ważną cechą tej metody jest fakt, że baza ta może być w dalszej części łatwo aktualizowana poprzez dodawanie nowych form wypowiedzi, oraz modyfikowana poprzez edycję tych już istniejących. Żeby chatbot posługujący się tą metodą funkcjonował optymalnie, potrzebna jest bardzo duża liczba przykładowych dialogów, które developer musi wprowadzić ręcznie.

Po skompletowaniu informacji robot korzystając z wbudowanych form zdaniowych jest gotowy udzielić odpowiedzi, np.

„Kraków znajduje się na południu Polski"

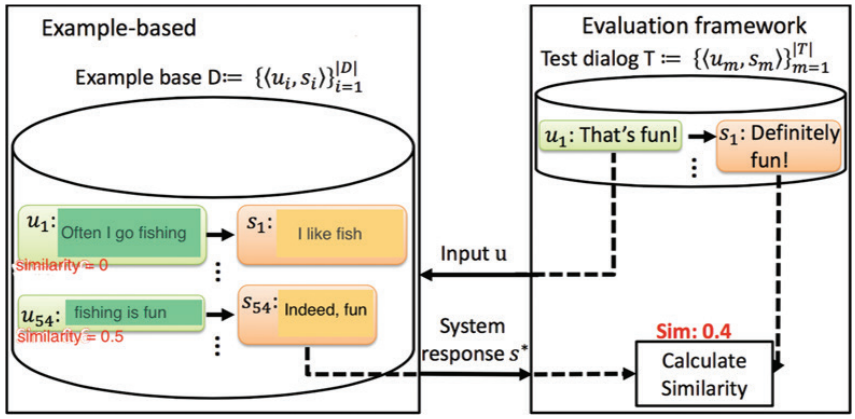
Brak któregoś z elementów opisujących gramatykę może sprawić, że chatbot mylnie zinterpretuje zadane mu pytanie i udzieli błędnej odpowiedzi. Podobnie jak człowiek, któremu obca jest reguła pisania nazw własnych z wielkiej litery po natrafieniu na słowo, które pisane z dużej litery znaczy zupełnie co innego niż pisane z małej, może się łatwo pomylić podczas odpisywania swojemu rozmówcy. Pomimo, iż poprawnie zinterpretuje ów wyraz, to sens zdania będzie zupełnie odbiegał od treści konwersacji.

Posłużmy się naszym zdaniem jednak lekko je zmodyfikujmy, zamieniając słowo „Kraków" na „Łódź". Sens zdania pozostaje bez zmian, ponieważ w dalszym ciągu pytamy o położenie miasta, ale tym razem położonego w centralnej części kraju. W tym momencie gdyby robot w wyniku niedopatrzenia jego twórcy, nie posiadał zaimplementowanej reguły o nazwach własnych, udzieliłby co prawda poprawnej odpowiedzi, ale nie dokładnie takiej, jakiej oczekiwaliśmy.

Podany powyżej przykład przedstawia chatbota, który działa na prostej zasadzie kojarzenia i porównywania danych wejściowych z tymi, które są już zawarte w jego bazie danych, która, jak już wspomnieliśmy wcześniej, jest mocno rozbudowana. Rozwiązanie takie daje ciekawe efekty, dając wrażenie, że rozmówca naprawdę prowadzi konwersację z człowiekiem, jednak jest ono niezwykle pracochłonne. Wymaga bowiem od developera na etapie tworzenia robota zgromadzenia ogromnej ilości przykładowych dialogów i wypowiedzi, które mogą być wykorzystane przez aplikację, a i tak może to się okazać niewystarczające. Nie chodzi tylko o to, że sam język ulega modyfikacjom, dochodzą nowe słowa i zwroty, ale trzeba także wziąć pod uwagę to, że każdy rozmówca jest inny, a co za tym idzie inny będzie także sposób wypowiedzi każdego z użytkowników. Wprowadzanie i aktualizowanie tych wszystkich reguł zabrałoby twórcy robota jeśli nie całe życie, to na pewno dużo czasu.

Dlatego też zamiast uzbrajać chatbota w potężną bazę wiedzy, można zastosować rozwiązania oparte na wykorzystaniu w robocie mechanizmów uczących. Przykładem może być tu metoda zaproponowana w [Hiraoka, Neubig i in., 2017]. Polega ona na interakcji z użytkownikiem, gdzie robot otrzymuje zapytania w formie nieoznakowanej, aby następnie nadać im odpowiednie oznaczenia w bazie danych. Jest to o tyle efektywne, że robot nie wymaga wbudowania dużej liczby danych, z których każda ma przypisaną odpowiednią etykietę, ale może dysponować małą bazą uporządkowanych danych, którą w dalszej części będzie ją sukcesywnie powiększał [Settles, 2019].

Posługując się przykładem naszego chatbota, mechanizm uczący polegać będzie na tym, że robot otrzymując dane wejściowe, wyszukuje w swojej bazie danych frazę, która jest najbardziej zbliżona do tej otrzymanej (posiada największy współczynnik podobieństwa). Następnie robot odczytuje odpowiedź (etykietę), która jest przypisana do tej szukanej i wysyła ją do użytkownika/rozmówcy. Jednocześnie zapisuje w swojej pamięci stworzoną w ten sposób kombinację, której będzie mógł używać w przyszłości (rysunek 3.3.2).



Rys. 3.3.2. Graficzna prezentacja funkcjonowania modelu uczenia Active Learning Example-Based

Źródło: [Galitsky, 2019].

Powyższy model jest oczywiście przedstawiony w pewnym uproszczeniu, ponieważ bardziej zaawansowane chatboty nie skupiają swoich zasobów jedynie na tworzeniu nowych kwestii dialogowych, ale także używają ich do zbierania informacji, które mogą być wykorzystane w przyszłości.

Jak wiadomo pamięć komputera jest mniej zawodna niż ludzka, w której dochodzi czynnik fałszywych wspomnień. Sposób zapamiętywania właściwy dla ludzkiego mózgu przypomina metodę przechowywania obrazów w pamięci komputerów, z tą różnicą, że nasza pamięć oferuje dodatkową funkcję – zapamiętane dane zmieniają się z czasem [Mlodinow, 2016]. Dlatego też nie będzie przesadą stwierdzenie, że w niedalekiej przyszłości to nie ludzie, a właśnie maszyny mogą pełnić funkcje doradców politycznych lub gospodarczych, tak samo jak dziś pełnią funkcje pomocników w wyborze wycieczki.

Kolejnym krokiem w rozwijaniu technologii wirtualnych rozmówców stanowi połączenie omówionych powyżej metod formowania odpowiedzi, wraz z oprogramowaniem służącym do rozpoznawania mowy (już teraz słownik Google dla języka angielskiego posiada ponad milion słów, a jednocześnie jest ciągle aktualizowany przez użytkowników, którzy wysyłają kolejne zapytania, wzbogacając ty, samym bazę danych portalu) oraz syntezatorem mowy. Sam syntezator nie jest niczym nowym, ponieważ już pod koniec lat 30. ubiegłego wieku Homer Dudley pracujący w Bell Telephone Laboratory zaprezentował pierwsze urządzenie, które potrafiło symulować ludzką mowę.

Do największych trudności przy tworzeniu sztucznego rozmówcy należy kwestia rozumienia przez maszynę emocji, stworzenia systemu

komputerowego będącego w stanie odczytywać emocje z tonu głosu czy mowy ciała, a więc coś, co trudno zamknąć w sztywnych ramach definicji. Prawidłowe odczytanie uczuć, które towarzyszą słowom stanowi kluczowy problem do rozwiązania, na drodze do stworzenia pełnoprawnego rozmówcy dla człowieka.

3.4. Technologia Semantic Web

Internet w dzisiejszej formie wraz ze swoimi zasobami jest pełen informacji, wśród których każdy użytkownik stara się znaleźć te, które go najbardziej interesują. W tym celu posługuje się wyszukiwarką, która wpisaną frazę wyszukuje bądź w całości, bądź też rozbija ją na słowa i poprzez wyszukiwanie słów-kluczy wyświetla odpowiednie wyniki. Dochodzą do tego oczywiście zaawansowane algorytmy, które między innymi korzystają z historii poszukiwań innych użytkowników. Algorytm analizuje, która strona dla danego wyszukiwania była najczęściej otwierana i wyświetla wyniki w odpowiedniej kolejności. Kolejnym elementem optymalizującym wyszukiwanie jest konstrukcja samej strony. Szczególnie rejestrując swojego bloga mamy możliwość przypisania do niego słów-kluczy tzw. tagów, które „definiują" naszą stronę. Dane z takiej tradycyjnej strony są zrozumiałe tylko dla człowieka.

Semantic Web jest to technologia, która poprzez zmiany budowy stron czy interfejsów serwisów, zmieni dotychczasowy sposób wyszukiwania informacji w sieci. Semantyczne rozszerzenie jest nadbudówką dla obecnej sieci zaprojektowaną z myślą o prezentowaniu informacji w formacie do odczytu maszynowego [Golbeck, Hendler, 2007]. Innymi słowy sieć semantyczna służy do tego, aby dane zawarte w sieci były możliwe do przetworzenia przez inne aplikacje (np. boty). Jest to możliwe poprzez rozszerzenie istniejących już technologii komunikacyjnych w sieci o kolejne standardy internetowe, takie jak:

UNICODE
URI
XML i XML Schema
RDF i RDF Schema
OWL
RIF
SPARQL

UNICODE jest to uniwersalny standard określający spójny sposób kodowania wielojęzycznego tekstu, który umożliwia międzynarodową wymianę danych tekstowych i stwarza podstawy dla globalnego oprogramowania [Unicode, 2019]. Stanowi on skuteczną alternatywę dla różnych języków, z których każdy posiada własny system kodowania znaków w nim występujących. Dzięki temu standardowi każdy symbol – niezależnie od tego, w którym języku – ma przypisaną stałą wartość.

URI (*Uniform Resource Identifier*) jest identyfikatorem składającym się z sekwencji znaków, która spełnia określone standardy składniowe. Służy do identyfikowania określonych zasobów, zarówno materialnych, jak i abstrakcyjnych. Budowa URI ma charakter hierarchiczny, który jest bardzo podobny do URL (z czym jest często mylony). URL bowiem to adres miejsca w Internecie, gdzie można znaleźć dany zasób. URI z kolei stanowi pewne rozszerzenie URL o URN (*Uniform Resource Name*), dzięki czemu potrafimy dokładnie określić konkretny elementu, który nas interesuje. Jako przykład weźmy adres URL http://poligon-net.cba.pl/?page_id=234, gdzie:

1) http oznacza protokół,
2) poligon-net.cba.pl jest alternatywną nazwą serwera,
3) ?page_id=234 jest adresem konkretnej podstrony.

URI stanowi rozszerzenie i wygląda następująco **http://poligon-net.cba.pl/?page_id=234#OkladkaKsiazki**. Fraza **#OkladkaKsiazki** zdefiniowana jest jako fragment i odnosi się – w tym przypadku – do konkretnego fragmentu strony (tzw. kotwicy). Może to być na przykład konkretny fragment tekstu, który został oznaczony kotwicą.

XML (*Extensible Markup Language*) to uniwersalny język znaczników, za pomocą którego można opisać dane w ustrukturalizowany sposób [Kempa, 2017]. Innymi słowy jest on wykorzystywany do definiowania interfejsu użytkownika. W swojej budowie przypomina język HTML, czyli dane, które chcemy zdefiniować, zapisywane są w znacznikach. Poniżej przykładowy zapis w formacie XML:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SpisPracownikow>
    <Pracownik>
        <NumerPracownika>1</NumerPracownika>
        <NazwaPracownika>Jan Kowalski</NazwaPracownika>
        <Wiek>45</Wiek>
    </Pracownik>
    <Pracownik>
```

```
        <NumerPracownika>2</NumerPracownika>
        <NazwaPracownika>Andrzej Nowak</NazwaPracownika>
        <Wiek>35</Wiek>
    </Pracownik>
</SpisPracownikow>
```

Efekt kodu widoczny poniżej

Numer Pracownika	Nazwa Pracownika	Wiek
1	Jan Kowalski	45
2	Andrzej Nowak	35

Łatwo zauważyć, że budowa kodu XML (podobnie jak HTML) ma charakter hierarchiczny, gdzie można wyróżnić element główny (w naszym wypadku jest to <**SpisPracownikow**>) oraz elementy potomne, z czego każdy kolejny element w hierarchii może także posiadać elementy pochodne w stosunku do siebie, tym samym będąc dla nich *rodzicem*.

Język ten wykorzystywany jest zarówno przy tworzeniu aplikacji webowych, jak i typu desktop (na przykład aplikacje WPF), a ponieważ nie jest on przypisany do konkretnego języka programowania, możne być wykorzystywany do przesyłania dokumentów pomiędzy różnymi systemami i tym samym ich integracji.

Innym przykładem zastosowania standardu XML jest chociażby technologia RPA. Oprogramowanie Blue Prism daje możliwość zapisu w taki formacie, dzięki czemu roboty mogą być łatwo przenoszone pomiędzy maszynami.

RDF (*Resource Description Framework*) jest to struktura (framework) bazująca na języku XML, wykorzystywana do ustandaryzowania informacji zawartych w sieci, której celem jest umożliwienie m.in. wyszukiwania i przetwarzania wirtualnych danych [RDF, 2009]. Technologia ta powstała z myślą o robotach działających w sieci i stanowi informacje, które owe boty wykorzystują do przeszukiwaniach zasobów internetowych. Tak jak człowiek, który szuka w księgarni interesującej go książki, jest w stanie ją zidentyfikować poprzez takie atrybuty jak tytuł czy autor, tak samo maszyna jest w stanie „rozpoznać" konkretną stronę przy pomocy ustandaryzowanych opisów, które stanowią pewnego rodzaju metajęzyk dla użytkownika maszynowego.

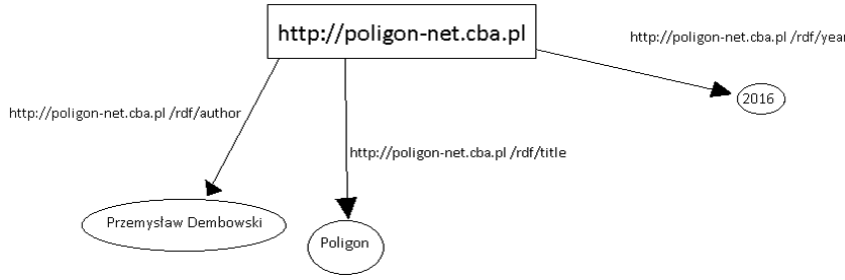
Podobnie jak w języku naturalnym, także i tutaj składnia RDF posiada podmiot (*subject*), czyli główną część zdania, predykat (*predicate*), który pełni funkcję funktora zdaniotwórczego oraz obiekt (*object*), czyli konkretną cechę predykatu. Prostym przykładem będzie zdanie „Słoń jest

duży". Podmiotem jest „słoń", predykatem słowo „jest" natomiast funkcję obiektu pełni tutaj słowo „duży".

Poniżej przykład opisu strony za pomocą metajęzyka RDF:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:site=" http://poligon-net.cba.pl /rdf/">
<rdf:Description rdf:about=" http://poligon-net.cba.pl">
        <site:title>Poligon</si:title>
        <site:author>Przemysław Dembowski</si:author>
        <site:year>2016</si:year>
</rdf:Description>
</rdf:RDF>
```
[4]

Zapis ten można przedstawić w postaci grafu, który można uporządkować przy pomocy RDF Schema, czyli zbioru reguł opisujących język RDF.



Rys. 3.4.1. Przedstawienie graficzne informacji zapisanej w formacie RDF. Główny atrybut, podmiot, ma przypisanych kilka atrybutów (author, title, year) o określonych cechach. Jest to oczywiście proste przedstawienie, dlatego że dla bardziej zaawansowanych stron podmioty mogą być również atrybutami dla podmiotów wyższego rzędu i jednocześnie posiadać własne atrybuty (tzw. zagnieżdżenie)

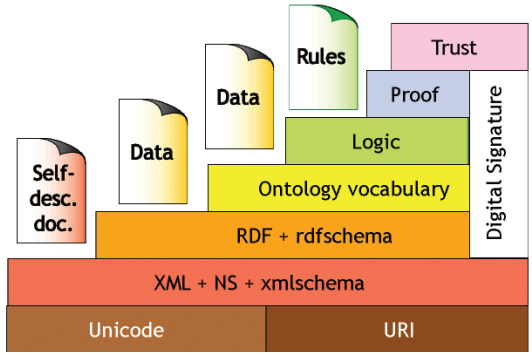
Źródło: opracowanie własne.

OWL (*Web Ontology Language*) jest to standard, który poprzez posiadanie bogatego słownika, stanowiącego zbiór reguł ontologicznych dla sieci Internet, jest rozszerzeniem dla RDF i XML. Korzystający z niego program komputerowy potrafi nie tylko zdefiniować rzeczy, ale także je pogrupować

4 Więcej informacji na temat RDF wraz z przykładami można znaleźć na stronie https://www.w3schools.com/xml/xml_rdf.asp (dostęp: 15.09.2019).

i wykazać relacje pomiędzy nimi. Klasyfikacja taka odbywa się poprzez zastosowanie kategorii (*classes*) oraz podkategorii (*subclasses*), gdzie każda jej instancja nazywana jest jednostką (*individual*). Przypomina to programowanie obiektowe, gdzie programista tworzy najpierw pewien obiekt ze zdefiniowanymi atrybutami, który służy za matrycę do tworzenia instancji tegoż obiektu, z przypisanymi wartościami dla każdego z atrybutów. Jest to bardzo ważne w procesie „porządkowania" informacji zawartych w sieci dla funkcjonowania maszyn i programów, których celem jest pracowanie właśnie na materiałach, które są zamieszczone w Internecie[5].

RIF (*Rule Interchange Format*). Wraz z rozwojem sieci semantycznej powstają różne systemy reguł, z których każda ma inną semantykę. Potrzebne jest zatem rozwiązanie, które zapewni optymalną wymianę tych reguł, pomiędzy różnymi systemami [Kifer, 2008], zamiast skupiać się na definiowaniu zunifikowanych zasad. Aby rozwiązanie takie było możliwe, twórcy przyjęli podejście stworzenia rodziny języków zwanych dialektami o ściśle określonej składni i semantyce, które przy tym mają być jednolite i rozszerzalne [Kifer, Boley, 2010].



Rys. 3.4.2. Graficzne przedstawienie hierarchii standardów. Niżej znajdują się te bardziej podstawowe, które służą do opisywania i definiowania elementów sieci. Wyższe partie służą już nie tylko do opisu, ale również do ustalania reguł ontologicznych oraz logicznych dla warstw niższych

Źródło: [Berners-Lee, 2000].

SPARQL (Protocol and RDF Query Language) jest językiem kwerend przeznaczonym dla RDF, w którym, korzystając z jego standardów, możliwe jest stworzenie zapytania w celu pobrania informacji na aplikacjach

5 Ciekawy przykład konstrukcji semantycznej wykorzystującej składnię OWL można zobaczyć na stronie http://www.linkeddatatools.com/introducing-rdfs-owl (dostęp: 15.09.2019).

semantycznych. Ponieważ, jak już pisaliśmy wcześniej, struktura RDF jest trójelementowa (podmiot–predykat–obiekt), dlatego też budowa zapytań do złudzenia przypomina zapytania NoSQL o strukturze dokument–klucz–wartość.

Opisane powyżej standardy tworzą pewnego rodzaju stos, który stanowi strukturę Semantic Web. Począwszy od najniższych (które tworzą podstawy), każda kolejna stanowi nadbudówkę poprzedniej, wykorzystując i rozszerzając jej elementy.

Na zakończenie należy wspomnieć, że opisana powyżej technologia jest w dalszym ciągu rozwijana, toteż w niedługim czasie struktura ta może ulec ewolucji, w toku której dojdą nowe elementy, a stare zostaną zastąpione bardziej wydajnymi. Niemniej świat technologii konsekwentnie dąży do ustrukturyzowania zawartości Internetu. Daje to nowe możliwości rozwoju technologii, które stworzone są właśnie z myślą o pracy w sieci, jak chociażby roboty tworzone w branży RPA.

Literatura

Berners-Lee T. (2000), *Semantic Web – XML2000*, prezentacja on-line, W3C, https://www.w3.org/2000/Talks/1206-xml2k-tbl/ (dostęp: 30.10.2020).

Biedrzycki N. (2018), *Rzeczywistość stapia się ze światem cyfrowym. AR to nie tylko Pokémon Go*, https://businessinsider.com.pl/technologie/nowe-technologie/arczym-jest-rozszerzona-rzeczywistosc/qn6173n, 04.03.2018, (dostęp: 01.10.2019).

Czerwonka P. (2016), *Zastosowanie chmury obliczeniowej w polskich organizacjach*, Wydawnictwo Biblioteka, Łódź.

Duhigg C. (2012), *Siła Nawyku. Dlaczego robimy to, co robimy i jak można to zmienić w życiu i biznesie*, Wydawnictwo Naukowe PWN, Warszawa.

Galitsky B. (2019), *Developing Enterprise Chatbots*, Springer.

Golbeck J., Hendler J. (2007), *A Semantic Web approach to the provenance challenge*, „Concurrency and Computation: Practice and Experience", vol. 20, Issue 5, s. 431–439.

Gonciarski W. (2010), *Gospodarka cyfrowa – powstanie i etapy rozwoju*, [w:] W. Gonciarski (red.), *Zarządzanie w warunkach gospodarki cyfrowej*, WAT, Warszawa, s. 11–38.

Gonciarski W. (2017), *Koncepcja zarządzania 2.0 jako konsekwencja rewolucji cyfrowej*, *Studia Ekonomiczne*, „Zeszyty Naukowe Uniwersytetu Ekonomicznego w Katowicach", nr 338, s. 38–53.

Henssen D.J.H.A., van den Heuvel L., De Jong G., Vorstenbosch M.A.T.M., van Cappellen van Walsum A.-M., Van den Hurk M.M., Kooloos J.G.M., Bartels R.H.M.A. (2020), *Neuroanatomy Learning: Augmented Reality vs. Cross-Sections*, „Anatomical Sciences Education", vol. 13, no. 3, s. 353–365.

Hiraoka T., Neubig G., Yoshino K., Toda T., Nakamura S. (2017), *Active Learning for Example-based Dialog Systems*, [w:] K. Jokinen, G. Wilcock (red.), *Dialogues with Social Robots*, Springer, s. 67–78.
Kempa A. (2017), *Wprowadzenie do WPF – Tworzenie aplikacji w WPF przy użyciu XAML i C#*, Helion, Gliwice.
Kifer M. (2008), *Rule Interchange Format: The Framework*, [w:] D. Calvanese, G. Lausen (red.) *Web Reasoning and Rule Systems. Proceedings of Second International Conference, RR 2008, Karlsruhe, Germany, 31 October–1 November 2008*, „Lecture Notes in Computer Science", vol. 5341, s. 1–11.
Kifer M., Boley H. (2010), *RIF Overview (Documentation)*, https://www.w3.org/ TR/2010/ NOTE-rif-overview-20100622 (dostęp: 16.09.2019).
Malinowski G. (2010), *Logika ogólna*, Wydawnictwo Naukowe PWN, Warszawa, s. 18.
Mayer-Schonberge V., Cukier L. (2014), *Big-Data – Rewolucja, która zmieni nasze myślenie, pracę i życie*, MT Biznes, Warszawa.
Mazurek G. (2019), *Transformacja Cyfrowa – Perspektywa Marketingu*, Wydawnictwo Naukowe PWN, Warszawa.
Mell P., Grance T. (2019), *The NIST Definition of Cloud Computing – Recommendation od the National Institute of Standards and Technology*, https://nvlpubs.nist.gov/ nistpubs/Legacy/SP/nistspecialpublication800-145.pdf (dostęp: 28.09.2019).
Mlodinow L. (2016), *Nieświadomy mózg – jak to, co dzieje się za progiem świadomości, wpływa na nasze życie*, Prószyński Media, Warszawa, s. 98.
RDF (2009), *RDF (Documentation)*, https://www.w3.org/RDF/ (dostęp: 14.09.2019).
Rifkin J. (2001), *Koniec pracy*, Wydawnictwo Dolnośląskie, Wrocław.
Settles B. (2019), *Active Learning Literature Survey*, „Computer Sciences Technical Report", 1648, University of Wisconsin–Madison, Updated on: January 26, 2019, www.burrsettles.com/pub/settles.activelearning.pdf (dostęp: 15.05.2019).
Thomas R., McSharry P. (2015), *Big Data Revolution*, Wiley, Chichester.
Unicode (2019), Unicode, https://www.unicode.org/versions/Unicode12.0.0/ch01.pdf (dostęp: 14.09.2019).
Zieliński J.S. (2000), *Zarys badań nad sztuczną inteligencją*, [w:] J.S. Zieliński (red.), *Inteligentne systemy w zarządzaniu. Teoria i praktyka*, Wydawnictwo Naukowe PWN, Warszawa.

Spis rysunków

Epilog

Książka ukaże się na rynku w 2021 r. po jednym roku od pojawienia się na całej kuli ziemskiej pandemii Covid-19 niszczącej gospodarkę szczególnie mocno w niektórych krajach przodujących w rozwoju nauki (USA, Europa Zachodnia), które musiały zająć się produkcją nowych, nieznanych szczepionek, w których zachodzą nieoczekiwane zjawiska pauperyzacji społeczeństw, a jednocześnie wielokrotny wzrost dochodów jednostek, co spowoduje nie-ewolucyjne zjawiska społeczne i zmianę priorytetów badawczych. Autorzy spodziewają się, że za parę lat potrzebne będzie napisanie przewodnika umożliwiającego wybór narzędzi i ułatwiającego orientację w zaistniałej nowej rzeczywistości.

Książka poświęcona jest inteligentnym systemom informatycznym w organizacji, wykorzystującym rozproszone źródła informacji niestrukturalnej dostępne w sieci Internet.

W przeciwieństwie do strukturalnych źródeł danych (w formie plikowej lub baz danych) wykorzystywanych przez tradycyjne systemy informatyczne zarządzania, źródła internetowe mają charakter przede wszystkim dokumentów nieposiadających ściśle zdefiniowanej struktury semantycznej i schematu zawartości. Uzyskanie wyższej jakości przetwarzania tego rodzaju informacji niestrukturalnej wymaga inteligentnego zachowania systemu informatycznego, przede wszystkim w formie „zrozumienia" analizowanego dokumentu.

WYDAWNICTWO
UNIWERSYTETU
ŁÓDZKIEGO

wydawnictwo.uni.lodz.pl
ksiegarnia@uni.lodz.pl
(42) 665 58 63

Książka dostępna również
jako e-book

ISBN 978-83-8220-353-0
9 788382 203530