Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Błażej Zyglarski

Wykorzystanie sieci neuronowych i algorytmów genetycznych w analizie i kategoryzacji dokumentów naukowych

Praca doktorska

dr hab. Piotr Bała, prof. UMK

Wydział Matematyki i Informatyki
Uniwersytet Mikołaja Kopernika

październik 2011

Deklaracja autora:
świadomy odpowiedzialności karnej deklaruję, że napisałem tę pracę samodzielnie a cała jej zawartość została uzyskana zgodnie z prawem.

10 października 2011 .....

date *Błażej Zyglarski*

Deklaracja przełożonego:
praca jest gotowa do recenzji.

10 października 2011 .....

date *dr hab. Piotr Bała, prof. UMK*

Streszczenie

Niniejsza praca jest poświęcona metodom automatycznej analizy dokumentów naukowych. Jej cel to stworzenie dobrej metody na znajdowanie powiązanych ze sobą dokumentów i wyznaczanie słów kluczowych opisujących te dokumenty. Przedstawione w tej rozprawie podejście do tego tematu składa się z zagadnień dotyczących: obliczania miary różnic pomiędzy dokumentami, kategoryzowania dokumentów i wyznaczania słów kluczowych dla dokumentu.

Analiza dokumentów w ogólności jest trudna. Tym trudniejsza jest analiza dokumentów naukowych, które wykorzystują zaawansowaną terminologię, która dodatkowo może występować w dokumencie rzadko. To powoduje, że klasyczne metody wyodrębniania słów kluczowych takich dokumentów stają się nieefektywne. Prezentowane w tej pracy metody są w stanie przeprowadzić analizę różnych danych, ale najlepsze wyniki uzyskują podczas przetwarzania tekstów naukowych.

Pierwszym etapem pracy była kategoryzacja dokumentów w obrębie repozytorium. Na bazie kilku badanych miar różnic pomiędzy dokumentami, opartych na analizie statystycznej słów, analizie n-gramów i miary różnic Kołmogorowa, w niniejszej pracy zdefiniowano nowy sposób podziału dokumentów na kategorie. Kategoryzacja dokumentów została przeprowadzona w oparciu o samoorganizujące się sieci Kohonena, które w celu optymalizacji ich działania, zmodyfikowane zostały z wykorzystaniem idei algorytmów genetycznych i dynamicznych sieci neuronowych.

Kolejnym etapem niniejszej pracy była dokładna analiza dokumentów, wyznaczająca opisujące je słowa kluczowe. Na tym etapie wykorzystano również samoorganizujące się sieci neuronowe Kohonena, które rozszerzono o umiejętność wzmacniania wyniku, bazującą na kategoriach i miarach różnic dokumentów wyznaczonych w pierwszym etapie. Do usprawnienia procesu użyto łańcuchów Markowa wartościujących poszczególne słowa z dokumentów.

Przedstawione w niniejszej rozprawie nowe podejście do problemu analizy tekstu pozwala znacząco przyspieszyć proces wyszukiwania informacji, który dotychczas wymagał sprecyzowania przez szukającego zapytania, a następnie wyboru odpowiednich wyników spośród wielu rezultatów proponowanych przez wyszukiwarki internetowe. Zaprojektowane algorytmy pozwalają na stworzenie kolekcji dokumentów związanych z dokumentami inicjującymi, kilkukrotnie poprawiając wyniki otrzymane za pomocą zwykłych systemów wyszukiwawczych. Zaprojektowane algorytmy przetestowano na różnych zbiorach dokumentów, co potwierdziło poprawność ich konstrukcji i trafność wyników otrzymanych podczas ich działania.

Słowa kluczowe: sieci neuronowe; złożoność Kołmogorowa; algorytmy genetyczne; łańcuchy Markowa;

Klasyfikacja AMS 2000: 92B20, 68T99, 60J22, 68Q30, 91C20

Abstract

In this thesis novel methods of automatic analysis of scientific documents are presented. Main goal of this work was to create a good method for finding related documents and selecting keywords among them. Presented approach consists of computing distances between documents, categorizing documents and selecting keywords.

Analysis of documents can be difficult. Scientific documents analysis is even harder, because advanced and rare terminology is used there. For that reason classic methods of selecting keywords are not effective. Methods presented in this thesis can deal with different data, but the best results can be achieved for scientific texts.

First part of the thesis describes documents categorization. Three distances between documents has been defined and used to perform categorization. Namely we have used: statistical words appearance, n-grams analysis and Kolmogorov distance. A new way of documents comparison with use of these distances is presented. Categorization is performed with use of self organizing Kohonen neural networks, which are modified here with usage of genetic algorithms and dynamic neural networks.

Next part of this thesis is document analysis and keywords selection. Self Organizing Maps are used again with addition of reinforcement upon previously computed documents distances and categories. Markov chains are used for preprocessing words weights.

Presented in this thesis new approach to scientific documents analysis provides improvement of the information retrieval process, in which user was needed for creating search queries and filtering results given by search engines. This process can be automated with usage of algorithms created in this thesis.

Presented algorithms were tested on various sets of initial data. The results confirmed accuracy of returned results and proved effectiveness of presented approach.

Keywords: neural networks; Kolmogorov complexity; genetic algorithms; Markov chains;

AMS Classification: 92B20, 68T99, 60J22, 68Q30, 91C20

Spis treści

SPIS TREŚCI

Spis rysunków

SPIS RYSUNKÓW

Spis tablic

Wstęp

Cel prowadzonych badań

W ciągu ostatnich lat znacząco wzrosła ilość informacji dostępnych dla dowolnego odbiorcy. Przyczynił się do tego gwałtowny rozwój internetu i ogólny pęd do digitalizacji wszelkich danych. Większość potrzebnych informacji można znaleźć w internecie - w szczególności tyczy się to wszelkich prac i artykułów naukowych. Z gąszczu danych często jednak trudno jest wybrać te interesujące. Okazuje się również, że w popularnych dzisiaj wyszukiwarkach internetowych w większości przypadków odsetek poprawnych wyników jest niewielki - ilość dokumentów i danych dostępnych w internecie jest tak ogromna, że praktycznie niemożliwe jest zindeksowanie i szczegółowe zbadanie ich wszystkich (co jest pokazane w Albert *et al.* (1999)). Użytkownik musi więc samodzielnie przeglądać szereg wyników wyszukiwań tracąc cenny czas.

Również pozyskanie danych do samodzielnej analizy wymaga czasu - zasada działania popularnych serwisów wyszukiwawczych wymaga interakcji z użytkownikiem. Aby otrzymać żądany rezultat, użytkownik musi dostarczyć odpowiednie zapytanie.

Zwykle składa się ono po prostu z odpowiednio dobranego zestawu słów, na podstawie którego wyszukiwarka zwraca listę wyników uporządkowoanych według wyznaczonego przez jej wewnętrzne mechanizmy porządku. Takie ograniczone podejście pozwala na proste, aczkolwiek wymagające od użytkownika wykształcenia pewnego sposobu kreowania zapytań i samodzielnego zweryfikowania jego skuteczności. Jednocześnie utrudniające wyszukiwanie informacji bardziej skomplikowanych, na przykład powiązanych z całym dokumentem, a nie tylko z wybranymi, zawartymi w nim słowami. Samodzielne wyszukiwanie sprowadza się do wytypowania samodzielnie słów kluczowych i urzyciem ich podczas korzystania ze zwykłej wyszukiwarki.

Celem niniejszej pracy jest zaprojektowanie oprogramowania, które pozwala zastąpić człowieka na etapie kreowania zapytań i analizy wyników, jednocześnie dostarczając mu zbiór najtrafniej dobranych rezultatów. W tym celu w pracy podjęto dyskusję na temat algorytmów opartych o sieci neuronowe, algorytmy

genetyczne, aproksymacyjne i teorię prawdopodobieństwa, służących do zaawansowanej analizy, kategoryzacji i kojarzenia dokumentów tekstowych (pisanych w języku naturalnym) oraz wykorzystania ich do stworzenia systemu, który umożliwi wygodną organizację dokumentów i ich analizę. W ramach poniższych rozważań użyte i zmodyfikowane zostały znane metody porównywania dokumentów (Cavnar & Trenkle (1994), Arimura *et al.* (2000)) oraz takie narzędzia, jak sieci neuronowe ze wzmocnieniem (Kohonen (1998)), dynamiczne sieci neuronowe, algorytmy genetyczne i łańcuchy Markowa. Uzyskane wyniki pozwoliły znacząco przyspieszyć proces szukania informacji.

Interakcja użytkownika powinna ograniczać się do dostarczenia tematu wyszukiwania (w postaci kolekcji dokumentów inicjujących system). Po dodaniu do systemu pewnej liczby dokumentów, aplikacja powinna zebrać o nich informacje (słowa kluczowe, relacje między dokumentami), a następnie wyszukać podobne informacje w Internecie (korzystając na przykład z *Google Scholar*[1]). Znalezione dokumenty powinny zostać poddane takiej samej analizie. Dla każdego z dokumentów zgromadzonych w repozytorium aplikacji, system powinien zaproponować zbiór dokumentów (należących do innych użytkowników, pobranych z sieci) tematycznie z nim związanych. Jednocześnie system powinien umożliwić proste funkcje systemu pracy grupowej (Grudin, Jonathan & Poltrock, Steven (1996),Grudin (1994)), pozwalając na współdzielenie dokumentów z innymi użytkownikami.

Prezentowane w tej pracy badania zostały zdominowane przez pracę nad analizą dokumentów tekstowych pisanych językiem naturalnym. Jednak podejście to może zostać przeniesione na dane o szczególnych własnościach i strukturach. Obszary zastosowania wyników niniejszej pracy to, między innymi, analiza treści (w sensie syntaktycznym i częściowo semantycznym) oraz wyszukiwanie podobieństw:

1. dokumentów;

2. kodów źródłowych programów;

Przebieg niniejszej pracy podzielony może zostać na kilka etapów:

1. badania nad sposobami analizy tekstów i metodami ich porównywania oraz wyznaczania stopnia powiązania (miary różnic) między nimi, wykorzystującymi analizę ilościową tekstów oraz metody złożoności Kołmogorowa (Fortnow (2004)), których wyniki zostały opublikowane w Zyglarski *et al.* (2008);

2. badania nad samoorganizującymi się sieciami neuronowymi Kohonena i ich zastosowaniami w kategoryzacji danych z wykorzystaniem wcześniej wy-

[1] http://scholar.google.com

znaczonych miar różnic oraz badania nad możliwościami wykorzystania algorytmów genetycznych i dynamicznych sieci neuronowych do poprawienia wyników tej kategoryzacji, których wyniki zostały opublikowane (Zyglarski (2009)) i zostaną przedstawione w kolejnych publikacjach (Zyglarski (2010));

3. badania nad rozwojem metod wyboru słów kluczowych w dokumentach, oparte o analizę częstości i rozkładu oraz sieci neuronowe Kohonena ze wzmocnieniem (Sutton (1996)) osiągniętym z wykorzystaniem doświadczeń zebranych przez system w trakcie analizy innych dokumentów, których wyniki zostały opublikowane w Zyglarski & Bała (2009) i Zyglarski & Bała (2010b);

4. badania nad wykorzystaniem łańcuchów Markowa do analizy tekstu i wartościowania słów w tekście oraz badania związków pomiędzy tymi słowami w celu usprawnienia metod wyboru słów kluczowych, których wyniki zostały zgłoszone do publikacji (Zyglarski & Bała (2010a));

5. stworzenie zintegrowanego z technologiami portalowymi (w szczególności portalem Liferay[2], na temat którego opublikowana została praca Zyglarski (2008a) i Zyglarski (2008b)) systemu wymiany i zarządzania dokumentami, który pomaga użytkownikom w wyszukiwaniu nowych materiałów, automatycznie odnajdując w dostępnych zasobach (czyli w swoim repozytorium, internecie, czy specyficznych bazach danych) informacje związane z zadanym zagadnieniem, prezentując je użytkownikowi; dzięki temu użytkownik zostaje zwolniony z obowiązku kreowania zapytań (które są wymagane przez popularne dzisiaj systemy wyszukiwawcze i często mogą okazać się nietrywialne) i samodzielnego filtrowania wyników zapytania; integracja algorytmów z systemem zarządzania dokumentami ma na celu dostarczenie użytkownikowi informacji związanych z tematem już w momencie zwykłej pracy z dokumentami;

6. wykorzystanie otrzymanych wyników do analizy innych typów danych i zweryfikowanie poprawności rezultatów pracy na innych typach informacji (w szczególności poprawnego wyznaczania zależności między nimi); wśród nich należy wspomnieć o analizie kodów źródłowych programów czy analizie obrazów.

Przedstawione w tej pracy wyniki dotyczące kategoryzacji danych z wykorzystaniem kilku typów miar różnic, użycia algorytmów genetycznych i przybliżeń do kategoryzacji danych, wyznaczania słów kluczowych opartego o sieci neuronowe Kohonena oraz wzmocnienie korzystające z miar różnich do innych dokumentów

[2] http://www.liferay.com

oraz wykorzystanie łańcuchów Markowa do wyznaczania rankingu słów kluczowych, to wyniki oryginalne, osiągnięte w ramach niniejszej pracy i opublikowane w międzynarodowych czasopismach.

Analiza medycznych tekstów naukowych była realizowana w ramach projektu Kardionet (PL-0262) finansowanego przez Norweski Mechanizm Finansowy (eea grants).

Struktura pracy

Niniejsza praca składa się z czterech rozdziałów i dwóch dodatków, w których omówiono kolejne etapy analizy tekstów zgromadzonych w testowym repozytorium i opisano znane metody tej analizy oraz wprowadzono w ramach niniejszej pracy ich modyfikacje i rozszerzenia.

- **Rozdział 1. Metody analizy tekstu**. W rozdziale tym przedstawiono znane metody analizy tekstu i porównywania dokumentów. Zdefiniowano miary różnic pomiędzy dokumentami i na ich podstawie wprowadzono dodatkowy sposób wyznaczania powiązań, opierający się na wcześniej przedstawionych miarach. W rozdziale tym opisano również oparte o drzewa tekstów, zaprojektowane w ramach niniejszej pracy, szybkie algorytmy obliczania tych miar.

- **Rozdział 2. Kategoryzacja tekstów**. W tej części pracy przedstawiono algorytmy kategoryzacji tekstów, wykorzystujące sieci neuronowe Kohonena oraz, zaprojektowane w ramach tej pracy, modyfikacje algorytmów kategoryzacji, wykorzystujące idee algorytmów genetycznych i dynamicznych sieci neuronowych.

- **Rozdział 3. Wyznaczanie słów kluczowych**. W tej części pracy pokazano metody wyznaczania słów kluczowych na podstawie ich liczebności i położenia w tekście. Opisano w niej wykorzystane do tego celu algorytmy samoorganizujących się sieci neuronowych Kohonena, poszerzone w ramach tej pracy o możliwość wzmocnienia procesu uczenia się, zależną od informacji zgromadzonych przez algorytm w trakcie pracy z kolejnymi dokumentami. Opisano tu również nowy sposób na stworzenie wstępnego rankingu słów w oparciu o ich pozycje w tekście i łańcuchy Markowa.

- **Rozdział 4. Podsumowanie**. Ta część pracy przedstawia podsumowanie uzyskanych wyników oraz wprowadza do proponowanej tematyki dalszych badań.

- **Dodatek A. Struktura aplikacji**. W tym dodatku opisano krótko techniczną stronę projektowania aplikacji.
- **Dodatek B. Ewaluacja zaprojektowanych algorytmów**. W dodatku tym przedstawiono przykładowe wyniki uzyskane podczas testowania zaprojektowanych algorytmów na różnych zbiorach testowych danych.

Rozdział 1

Metody analizy tekstu

W rozdziale tym przedstawiono znane metody analizy tekstu i porównywania dokumentów. Zdefiniowano miary różnic pomiędzy dokumentami i na ich podstawie wprowadzono oryginalny sposób wyznaczania powiązań opierający się na kilku wcześniej przedstawionych miarach i spostrzeżeniu, że podobieństwo dokumentów naukowych jest związane z wieloma ich cechami możliwymi do sprawdzenia w różny sposób. Sposób ten polega na wykorzystaniu wielu różnych miar różnic pomiędzy dokumentami do ich porównywania i kojarzenia tylko dokumentów, które są podobne względem wszystkich sprawdzanych cech.

Terminem miara różnic zostały nazwane zdefiniowane w niniejszej pracy funkcjie działające na parach dokumentów i przyporządkowujące im wartości rzeczywiste.

W rozdziale tym opisano również oparte o drzewa tekstów zaprojektowane w ramach niniejszej pracy szybkie algorytmy obliczania zaproponowanych miar.

Zdefiniowane miary różnic pomiędzy dokumentami nie spełniają nierówności trójkąta, jednak w praktyce nie jest to warunek konieczny do porównywania dokumentów.

1.1 Wprowadzenie

Wśród wszystkich dostępnych dla odbiorcy danych można zidentyfikować dane o charakterystycznych strukturach. Oczywistym jest że taką strukturę posiadają wszelkie bazy danych (mają ścisłe definicje tabel, kolumn i więzów między nimi). Strony internetowe również zawierają szereg metadanych (w pracy Cruz *et al.* (1998) pokazane jest kojarzenie informacji na stronach internetowych) wyznaczonych przez format html/xml. Kolejną grupę stanowią dokumenty naukowe, które charakteryzują się specyficzną strukturą wykorzystywaną do szukania zależności i powiązań pomiędzy nimi. Głównym elementem tej struktury są cytowania,

które tworzą swoistą sieć powiązań tematycznie podobnych dokumentów. Druga ważna cecha, to zespół autorów, którzy tworzą jakąś grupę dokumentów naukowych, zwykle o podobnej tematyce - tak powstaje kolejna sieć powiązań - sieć społeczna. Takimi powiązaniami zajmował się między innymi Barabasi, który w Barabási *et al.* (2002) opisuje sieć społeczną współautorów. Ostatnia ważna cecha, to często dodatkowo podane słowa kluczowe opisujące dokument.

Badania przeprowadzone w niniejszej pracy skupiają się na tekstach, które nie mają wyznaczonych słów kluczowych, podanych cytowań, czy wspólnych autorów. Na podstawie znanych metod analizy czystego tekstu (Rajman & Besançon (1997)), bez korzystania z metadanych w ramach tej pracy zaproponowano nową metodę porównywania dokumentów. Metoda ta polega na połączeniu trzech różnych sposobów porównywania tekstów w celu zwrócenia uwagi na różne jego właściwości.

Zadanie porównywania informacji zawartych w tekście naturalnym jest jednym z obszarów zainteresowania szerokiej dziedziny, jaką jest *text mining*. Jak wynika z Tan (1999) *text mining* jest w obecnym czasie uważany za ważniejszy od *data miningu*, działającego zwykle na bazach danych posiadających dobrze zdefiniowaną strukturę i znajduje szerokie zastosowania w biznesie oraz przedsięwzięciach komercyjnych. Właśnie brak zdefiniowanej struktury języka naturalnego jest największym problemem utrudniającym wybieranie z tekstu istotnych danych. Metody analizy danych (wskazywane między innymi w Mannila *et al.* (1994)), które sprawdzają się dla danych konkretnie zdefiniowanych (jak na przykład ściśle określone zbiory danych z bioinformatyki - Lord *et al.* (2003)) nie są wystarczająco elastyczne dla danych nie posiadających ustalonej struktury. Dlatego prowadzonych jest wiele badań nad metodami pozwalającymi określić tę strukturę automatycznie, jak chociażby Ferrer i Cancho & Solé (2001), gdzie na podstawie grafów słów autorzy szukają związków leksykalnych między nimi, czy Nahm & Mooney (2000), gdzie autorzy wykorzystują szablony, dopasowując je do konkretnych fragmentów tekstu, które później tworzą ustrukturyzowaną bazę danych.

1.2 Porównywanie dokumentów na podstawie liczebności słów

Rozważania na temat analizy tekstów należy zacząć od próby odpowiedzi na pytanie, w jaki sposób człowiek rozpoznaje zawartość treściową tekstu. Ważną rolę odgrywają słowa w nim użyte, co pociąga za sobą potrzebę analizy ilościowej tych słów. W tej sekcji zaprezentowano metodę analizy i porównywania tekstów na podstawie ilości wystąpień słów w nich zawartych.

Podczas gdy człowiek nie ma problemu z wyodrębnieniem istotnych, a zatem

bogatych w treść fragmentów tekstu, automatyzacja tego procesu nie jest prosta. Zwykłe metody ilościowe zawodzą, ponieważ 80% słów występujących w tekście nie wpływa na jego rzeczywistą wartość a stanowi po prostu jego uzupełnienie. W Cherfi *et al.* (2003) pokazana jest metoda, która bierze pod uwagę poza ilościową analizą słów również ich analizę gramatyczną, wartościującą słowa w zależności od ich znaczenia z punktu widzenia gramatycznego i określającą zasady powiązań pomiędzy poszczególnymi słowami.

W poniższych rozważaniach użyto następującej definicji *słowa*.

Definicja 1 *Słowem nazywamy każdy ciągły ciąg liter.*

1.2.1 Częstość występowania słów w tekstach

Rozważmy przykładowy tekst (oznaczony przez T). Załóżmy, że tekst ten został wstępnie pozbawiony wszelkich symboli różnych od liter (zostały zastąpione znakami białymi), oraz że wszystkie duże litery zostały zamienione na litery małe. Przez L oznaczmy tablicę ilości wystąpień poszczególnych słów.

Algorytm 1 ***Wyznacz częstość słów***

$T = tekst$

$WHILE\ t = przeczytajSłowo(tekst)$

$IF\ (t \in L)$

$L[t] \Leftarrow L[t] + 1$

$ELSE$

$insert\ t\ into\ L \wedge L[t] \Leftarrow 1$

$END\ IF$

$END\ WHILE$

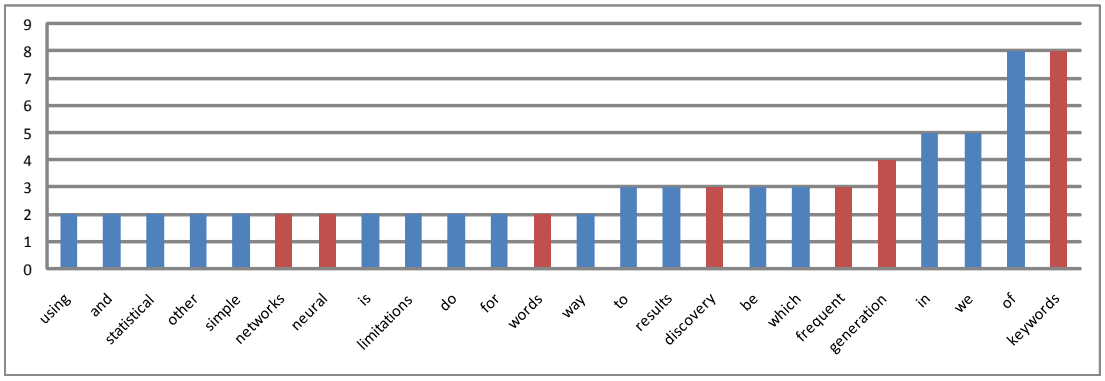
This is simple text about **keywords** generation (**discovery**). Of course all **keywords** are difficult for automatic generation (or **discovery**), but in limited way we could achieve results, which will fullfil our needs and retrieve proper **keywords**. During implementation of document management system we've **discovered** mentioned previously limitations in using simple statistical methods for **keywords** generation.

Main goal of further contemplations was to pay attention of words context. In other words we wanted to select most **frequent keywords**, which occur in common neighborhood. Using **neural networks** we show disadvantages of statistical **keywords discovery**.

Neural networks based **keywords** generation is way much reliable and gives better results, including promotion of less **frequent keywords**. Others, which can be more **frequent** do not have to be part of the results. Limitations were defeated.

Rysunek 1.1: Przykładowy tekst z zaznaczonymi pogrubioną czcionką wybranymi empirycznie słowami kluczowymi.

Proste zliczenie słów nie przynosi oczekiwanych rezultatów, co widać na przykładowym tekście (rysunek 1.1 i wyrkes 1.2) .



Rysunek 1.2: Ilość wystąpień poszczególnych słów w przykładowym tekście. Kolorem czerwonym oznaczono wybrane empirycznie słowa kluczowe.

Na wykresie 1.2 widać najczęściej występujące w przykładzie słowa. Słowa kluczowe (oznaczone kolorem czerwonym) wciąż stanowią mniejszość. Przyjmując 20 najlepszych wyników za domniemane słowa kluczowe, skuteczność algorytmu

na podanym przykładzie wynosi 35%. W innych przypadkach może to być znacznie mniej.

1.2.2 Słowa nieistotne

Można zauważyć, że słowa, które zajmują wysokie miejsca w rankingu są w większości, albo bardzo pospolite, albo są zaimkami, przyimkami i spójnikami, czyli słowami technicznymi, konstrukcyjnymi. Łatwo widać, że występują one również w wielu innych, dowolnych dokumentach niezależnie od ich tematyki.

Słowniki i stop-listy

Istnieje wiele słowników, na podstawie których można wyznaczyć słowa nieistotne. Jest to dobra metoda, zakładając, że zbiór danych na którym pracuje algorytm jest zbiorem o bardzo zróżnicowanej tematyce. W przypadku analizy zbiorów, które składają się z dokumentów o specyficznym charakterze (na przykład dokumentów naukowych), okazuje się, że jest w nim wykorzystywanych bardzo wiele specyficznych sformułowań, które nie są słowami kluczowymi i jednocześnie nie są często używane w języku naturalnym. Dlatego może się zdarzyć, że sformułowanie takie nie zostanie odrzucone poprzez analizę słów ze słownika.

Jednocześnie istnieją również algorytmy pozwalające na wyznaczanie stop-list (*ang: stoplists*) (Rzeczi & Mościński (2009)) na podstawie własnego zestawu dokumentów. K. Rzecki prezentuje metodę generowania stop-list na podstawie dokumentów znalezionych w serwisach internetowych, bazującą na częstości słów.

Ze względu na specyfikę projektowanego systemu, który może być wykorzystywany do analizy dokumentów charakterystycznych, zdecydowano się samodzielnie wyznaczyć słowa nieistotne bazując na częstości ich występowania w repozytorium zgromadzonych dokumentów. Zakładając, że są to dokumenty naukowe zgromadzone w niezbyt dużej ilości (baza początkowa w ilości około 1000 dokumentów (zobacz A.3)), oraz biorąc pod uwagę dalsze rozszerzenie wybranej metody w celu wyznaczania słów kluczowych, wybrano prostą metodę analizy częstości występowania słów.

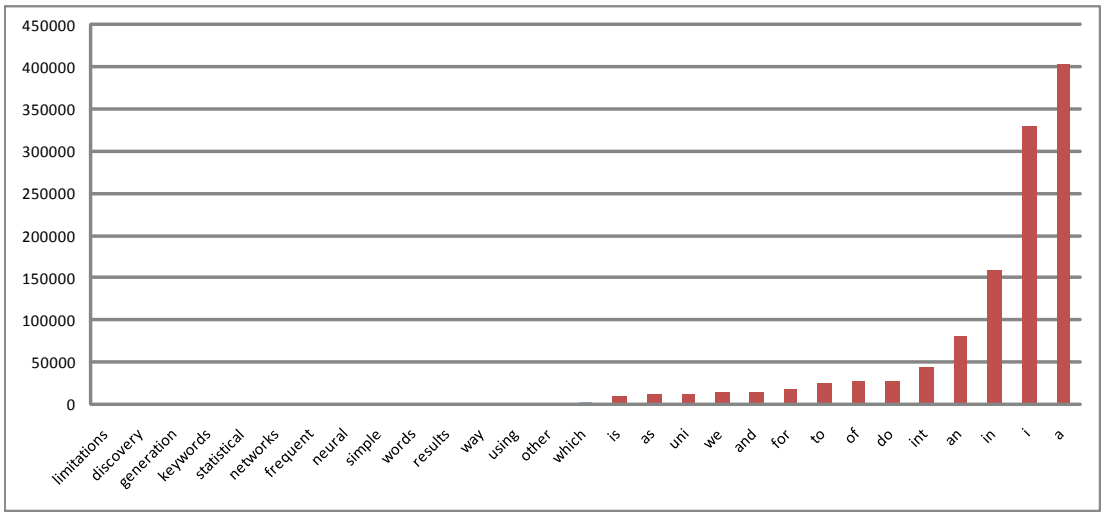
Uwaga 1 *Za słowo nieistotne w grupie dokumentów uznajemy słowo, które pojawia się bardzo często we wszystkich dokumentach z tej grupy.*

Jak pokazują testy, powyższe podejście gwarantuje dobre selekcjonowanie słów w obrębie analizowanych dokumentów naukowych.

Samodzielne wyznaczanie słów nieistotnych

Naturalnym następstwem powyższych spostrzeżeń jest przeprowadzenie zliczenia wystąpień słów w obrębie całego repozytorium dokumentów (zbiór tych słów

oznaczono przez T^s). Zakładając, że w repozytorium występują dokumenty z różnych dziedzin otrzymano gwarancję, że na liście (oznaczonej przez LSN) najwyżej punktowane będą słowa nieistotne. Na wykresie 1.3 [1]widać w jaki sposób słowa z przykładu 1.2 pojawiają się na liście LSN. Wykres ten przedstawia fragment listy słów zliczonych w obrębie całego repozytorium, porównując słowa występujące najczęściej i ilości ich wystąpień (prawa część wykresu) z ilością wystąpień przykładowych słów kluczowych z przykładowych dokumentów. Jak widać najczęściej występujace słowa to części zdań nie wpływające w istotny sposób na treść dokumentów.

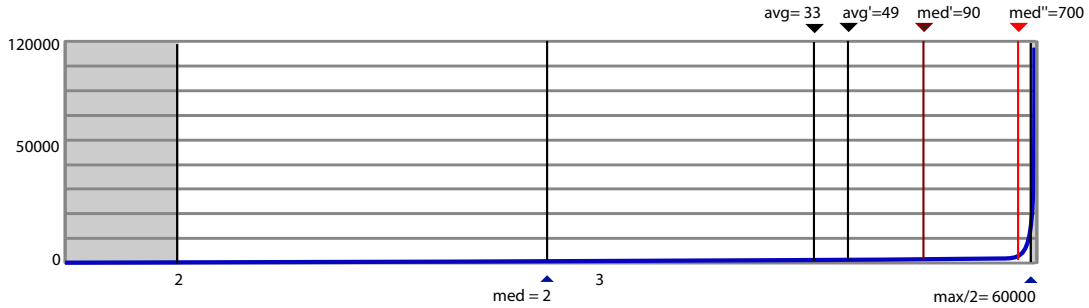


Rysunek 1.3: Ilość wystąpień słów w testowym zbiorze 1000 tekstów. Pokazano ilości słów z przykładu 1.2

Próg dla słów nieistotnych

Ponieważ na liście LSN występują wszystkie słowa pojawiające się w dokumentach, a zatem również te kluczowe, należy wyznaczyć granicę ilości wystąpień, powyżej której słowa zostaną uznane za nieistotne.

[1]Prezentowana lista została opracowana na podstawie analizy około 1000 dokumentów, z których większość była dokumentami pobranymi przez system automatycznie z internetu. Duża część z nich dotyczyła analizy tekstów i ekstrakcji danych. Doświadczenie zostało jednak przeprowadzone na listach różnej wielkości - począwszy od repozytoriów zawierających około 500 dokumentów, po repozytoria zawierające nawet ponad 1000 dokumentów. Repozytoria te były również zróżnicowane tematycznie. We wszystkich testach wyniki prezentowały się na podobnym poziomie.



Rysunek 1.4: Ilości wystąpień słów w repozytorium dokumentów. Oznaczono proponowane progi poza którymi słowa mogą być uznane za nieistotne.

Naturalnym wydaje się usunięcie tych słów, które występują częściej niż połowa z największej liczby wystąpień. Z przeprowadzonych badań i wykresu 1.4 wynika, że nie jest to dobre ograniczenie, ponieważ bardzo niewiele słów spełnia to kryterium (zbiór takich słów oznaczmy przez $T^s_{>\frac{1}{2}max}$).

Okazuje się,że na podstawie wyników zliczania słów dla całego, rozbudowanego repozytorium dokumentów (rysunek 1.4), ilość takich słów można oszacować w następujący sposób:

$$\left\|T^s_{>\frac{1}{2}max}\right\| < \frac{\|T^s\|}{1000}$$

co daje zbyt małą możliwość ograniczenia listy słów kluczowych, czyli zbyt małą wielkość generowanej stop-listy. Jednocześnie złym podejściem okazuje się próba usunięcia tych słów, które występują częściej niż mediana z ilości wystąpień.

$$\|T^s_{>med}\| < \|T^s\| \frac{9}{10}$$

. Oszacowanie to generuje zbyt długą stop-listę.

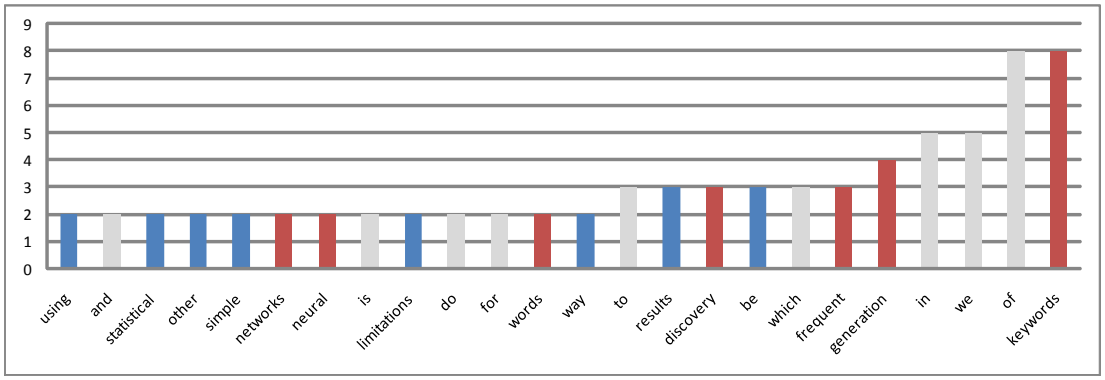
W ramach niniejszej pracy poczyniono **empiryczne** spostrzeżenie, że granica powinna przebiegać pomiędzy tymi dwoma wartościami. Taką wartością jest na przykład średnia ($T^s_{>avg}$) ze wszystkich ilości wystąpień słów.

$$\left\|T^s_{>avg}\right\| < \|T^s\| \frac{1}{10}$$

Innym przybliżeniem granicy może być średnia wystąpień elementów, ze zbioru $T^s \setminus (T^s_{<med} \cup T^s_{>\frac{1}{2}max})$ (oznaczona przez avg' a zbiór elementów do usunięcia $T^s_{>avg'}$).

$$\|T^s\| \geq \left\|T^s_{>\frac{1}{2}max}\right\| \geq \left\|T^s_{>avg'}\right\| \geq \left\|T^s_{>avg}\right\| \geq \|T^s_{>med}\| > 0$$

Przeprowadzone w ramach tej pracy testy pokazują, że najlepszym rozwiązaniem jest utworzenie stop-listy ze zbioru słów $T^s_{>avg'}$[2]. Dalsze testy wskazują również, że powyższe oszacownia są wystarczające do badania kolejnych repozytoriów tekstów naukowych.



Rysunek 1.5: Częstość wystąpień poszczególnych słów w przykładowym tekście. Kolorem szarym oznaczono wybrane przez algorytm słowa nieistotne. Kolorem czerwonym oznaczono wybrane empirycznie słowa kluczowe.

Po odrzuceniu słów uznanych za nieistotne skuteczność wyboru wzrasta nawet do 58%[3].

Szybki algorytm wyznaczania częstości słów

Prezentowany algorytm 1 jest nieefektywny, ponieważ wymaga dla każdego ze słów znalezionych w tekście wyszukania go na liście słów już przeczytanych i zwiększenia ilości jego wystąpień. Dlatego w ramach tej pracy opracowany został algorytm oparty o budowę drzewa prefiksowego o złożoności O(n). Założenia są takie same jak w algorytmie 1. Wynikiem algorytmu jest drzewo G o korzeniu w g. Przykładowe drzewo pokazano na rysunku 1.7. Liście drzewa reprezentują wszystkie słowa występujące w tekście T.

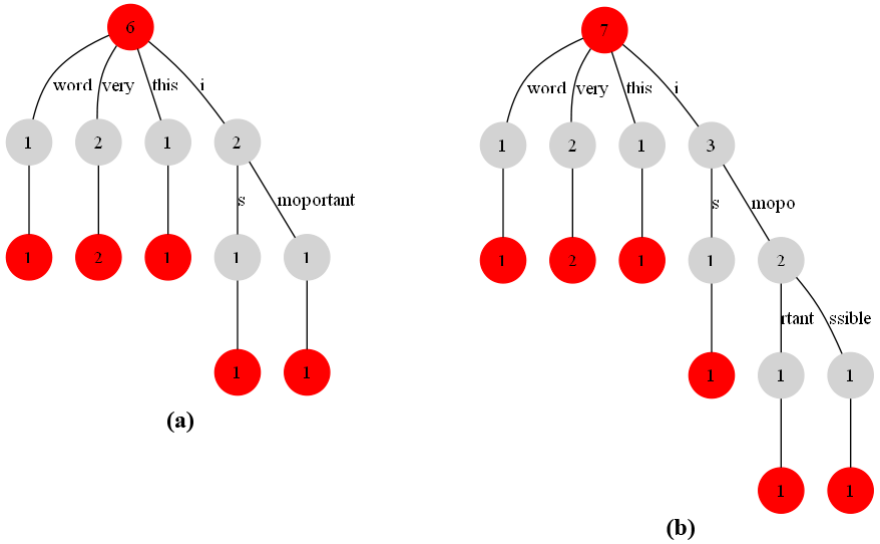
Algorytm 2 ***Zbuduj drzewo prefiksowe*** G ***tekstu*** T

Przez WC *oznaczymy zbiór znaków białych.*
$g \Leftarrow nowyWierzchołek()$
$G \Leftarrow \{g\}$
$p \Leftarrow g$

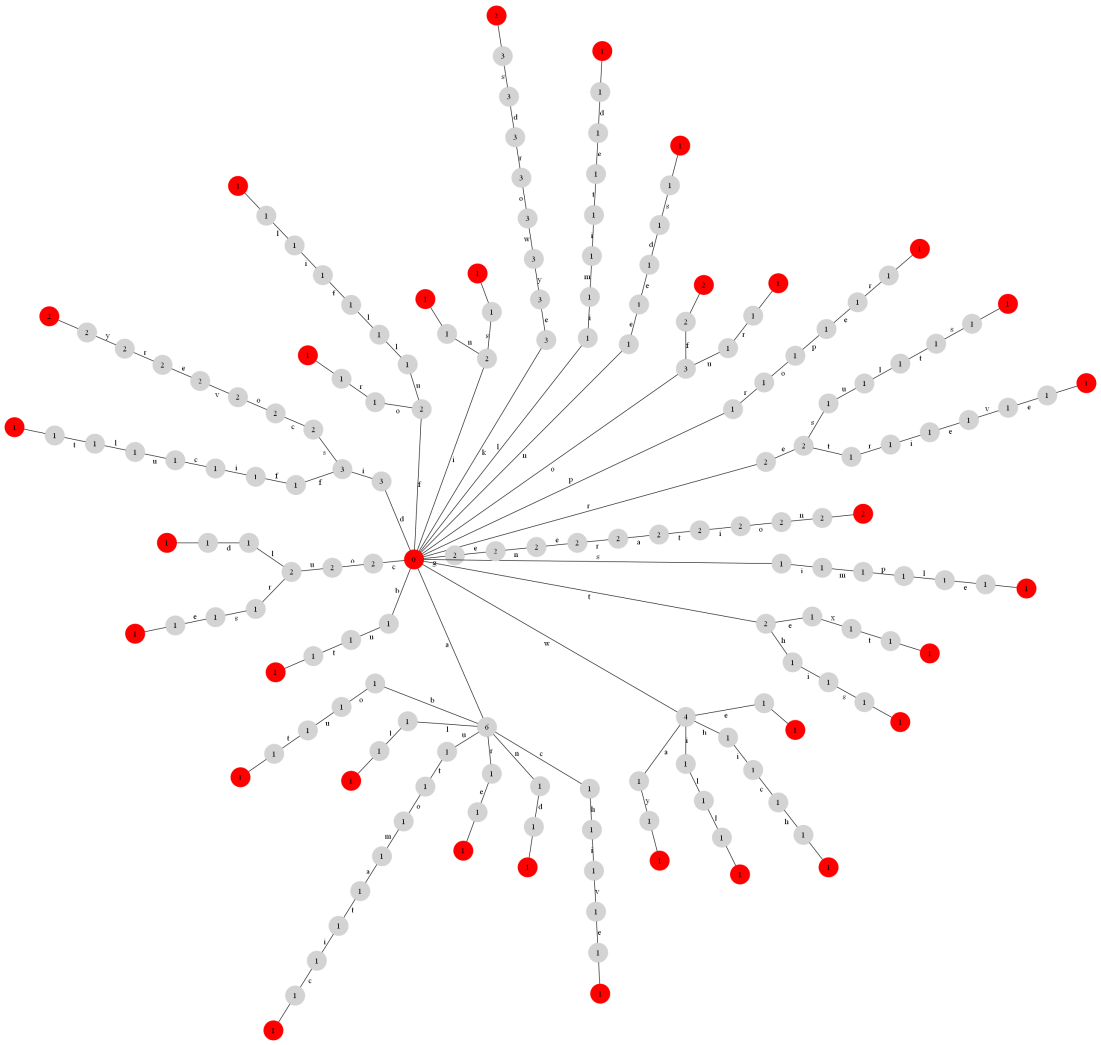
[2]Trzeba tu zauważyć, że każda modyfikacja repozytorium modyfikuje również aktualną stop-listę.

[3]W rozważanym przykładzie. Szczegółowe wyniki analizy dokumentów naukowych podane są w dodatkach.

$WHILE\ c = przeczytajZnak(T)$
$IF\ (\exists v \in wierzchołki(G)(p,v) \in krawędzie(G) \wedge etykieta(p,v) = c)$
$licznik(v) \Leftarrow licznik(v) + 1$
$ELSE$
$v \Leftarrow nowyWierzchołek()$
$licznik(v) \Leftarrow 1$
$wierzchołki(G) \Leftarrow wierzchołki(G) \cup \{v\}$
$etykieta(p,v) \Leftarrow c$
$krawędzie(G) \Leftarrow krawędzie(G) \cup \{(p,v)\}$
$END\ IF$
$IF(c \in WC)$
$p \Leftarrow g$
$ELSE$
$p \Leftarrow v$
$END\ IF$
$END\ WHILE$



Rysunek 1.6: (a) Drzewo skompresowane wygenerowane dla tekstu "this word is very very important", (b) Drzewo skompresowane wygenerowane dla tekstu "this word is very very important impossible"



Rysunek 1.7: Drzewo wygenerowane dla tekstu „This is simple text about keywords generation discovery. Of course all keywords are difficult for automatic generation of discovery but in limited way we could achieve results which will fulfil our needs and retrieve proper keywords."

W zbudowanym drzewie prefiksowym tekstu każdy liść odpowiada słowu występującemu w tym tekście. Etykieta liścia określa ilość wystąpień tego słowa.

Można zauważyć, że istnieją wierzchołki, do których wchodzi i z których wychodzi tylko jedna krawędź. Ciągi takich wierzchołków dają się bardzo łatwo kompresować poprzez zastąpienie listy wierzchołków jednym. Kompresja taka może być wykonana już podczas budowy drzewa prefiksowego, co implikuje duże oszczędności na złożoności pamięciowej algorytmu. Przykładowe stany prostego grafu skompresowanego widać na rysunku 1.6.

Idea algorytmu polega na tym, że początkowo do drzewa dodawane są gałęzie reprezentujące całe słowa. Jeżeli któryś prefiks kolejnego słowa pokrywa się z prefiksem słowa już dodanego do drzewa, następuje modyfikacja gałęzi poprzez podzielenie dotychczasowej krawędzi na dwie nowe i wstawienie pomiędzy nie nowego wierzchołka a następnie dodanie nowej krawędzi wychodzącej z niego, reprezentującej pozostały fragment nowego słowa.

Podczas działania algorytmu można jednocześnie generować drzewo prefiksowe całego zbioru tekstów, za pomocą którego zostaną wyznaczone słowa nieistotne dla tego zbioru.

Algorytm 3 ***Zbuduj drzewo prefiksowe*** N ***dla zbioru tekstów*** T

Niech N będzie pustym drzewem
WHILE $i < length(T)$
Zbuduj drzewo G dla tekstu $T[i]$
Dodaj drzewo G do drzewa N
$i := i + 1$
END WHILE

Operacja dodawania drzew

Operacja dodawania drzew sprowadza się do stworzenia nowego drzewa, zawierającego gałęzie z obu składników dodawania. Dla każdego fragmentu gałęzi drzewa, identyfikowanego przez scieżkę etykiet w kolejności od korzenia, jeśli fragment ten istnieje w obu drzewach, składnikach dodawania, wynikowe drzewo również będzie zawierało daną ścieżkę, przy czym etykiety jej wierzchołków będą sumą etykiet składowych. Jeśli fragment gałęzi, istnieje tylko w jednym z drzew, składników dodawania, wówczas jest dołączany do wynikowego drzewa w formie niezmienionej.

Algorytm 4 ***Dodaj drzewo prefiksowe tekstu*** T ***do drzewa prefiksowego tekstu*** S

Dane G będące drzewem prefiksowym tekstu T
Dane F będące drzewem prefiksowym tekstu S
FOR $\forall v \in G$
IF $\exists v' \in F$, *takie, że* $sciezka_F(korzen_F, v') = sciezka_G(korzen_G, v)$
$etykieta_F(v') = etykieta_F(v') + etykieta_G(v)$
ELSE
Znajdź $u \in G$ *takie, że* $\{sciezka_G(korzen_G, u), sciezka_G(u, v)\} = sciezka_G(v)$ *i* $\exists u' \in S$, *takie, że* $sciezka_F(korzen_F, u') = sciezka_G(korzen_G, u)$
dodaj v do F
dodaj krawędź pomiędzy u' *a v w F*
END IF

END FOR Zwróć F.

Drzewo N, podczas analizy kolejnych dokumentów jest wciąż rozbudowywane, zlicza wystąpienia wszystkich słów we wszystkich dokumentach. Drzewo to jest przechowywane w pamięci i w przypadku rozszerzenia repozytorium o nowy dokument modyfikowane za pomocą drzewa tego dokumentu. W każdej chwili możliwe jest wyznaczenie zgodnie z przytoczonym wcześniej ograniczeniem stop-listy słów, a więc również usunięcie z drzew dokumentów tych słów:

Algorytm 5 ***Usuń z drzewa prefiksowego tekstu*** T ***słowa nieistotne (na podstawie drzewa*** N

Niech G będzie drzewem tekstu T

Usuń z drzewa G wszystkie gałęzie, które występują w drzewie N i zgodnie z wcześniejszymi rozważaniami należą do zbioru $T^s_{>avg'}$*, czyli takie, które reprezentują słowo należące do zrbioru* $T^s_{>avg'}$*, czyli spełniające kryterium ilości wystąpień liścia gałęzi.*

$i := i + 1$

END WHILE

1.2.3 Miara różnic (ze względu na częstość słów) pomiędzy dokumentami

Na podstawie obliczonych częstości słów dla dokumentów zdefiniowano miarę różnic (ze względu na częstość słów) pomiędzy tymi dokumentami. Jest to miara różnic podobna do metryki taksówkowej, w której poszczególnymi wymiarami są kolejne słowa zawarte w dokumentach.

Definicja 2 *Metryka taksówkowa, zwana również metryką Manhattan, jest metryką na płaszczyźnie Euklidesowej definiowaną przez*

$$g((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$$

dla wszystkich punktów $P_1(x_1, y_1)$ *i* $P_2(x_2, y_2)$*.*

Definicja 3 *Miara różnic (ze względu na częstość słów) pomiędzy dokumentami*

$$P_1(x_1, x_2, ..., x_n), P_2(y_1, y_2, ..., y_n)$$

zdefiniowana jest jako

$$\hat{d}_s((x_1, x_2, ..., x_n), (y_1, y_2, ..., y_n)) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_n - y_n|$$

gdzie x_i*,* y_i *to odpowiednio ilości wystąpień słowa o numerze i w dokumentach* P_1 *i* P_2*.*

Każde słowo występujące w obliczeniach jest traktowane jako kolejny wymiar przestrzeni, w której znajdują się dokumenty. Przestrzeń ta jest (ze względu na nieskończoną ilość możliwych kombinacji liter) przestrzenią *nieskończenie* wymiarową. Jednakże należy tu zwrócić uwagę, że jest to uwaga czysto teoretyczna, gdyż żaden ze znanych języków nie zawiera więcej niż 5 milionów. Drugie wydanie *Oxford English Dictionary* zawiera 600 000 słów, co razem z terminami specjalistycznymi daje około 1 000 000 słów.[4]. Istnieją języki, które zawierają więcej słów, jednak ograniczenie rzędu 2 000 000 powinno wyczerpać wszystkie możliwości w dowolnym języku, 1 000 000 wystarczy dla języka angielskiego.

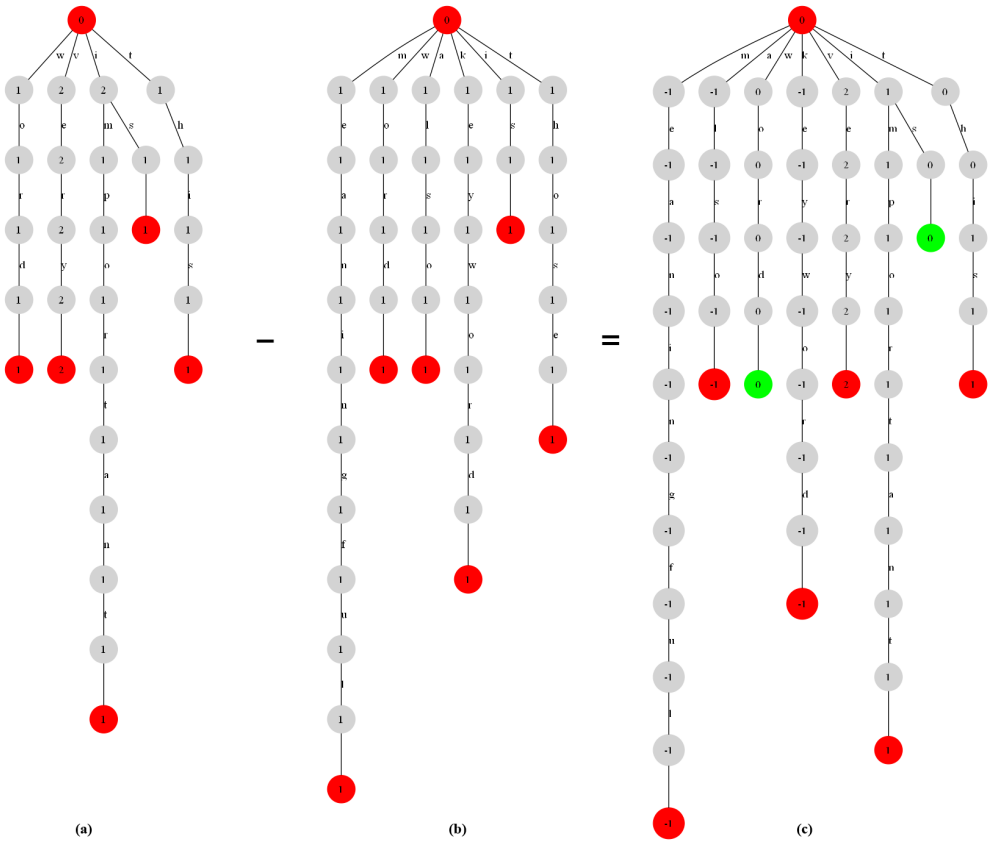
Twierdzenie 1 *Funkcja* $\hat{d}_s(-,-)$ *określona na zbiorze* $W \times W \to \mathbb{R}$ *gdzie* W *to zbiór wszystkich dokumentów spełnia następujące własności:*

1. $\hat{d}_s(A,B) \geq 0$, $\forall_{A,B \in W}$
2. $\hat{d}_s(A,A) = 0$, $\forall_{A \in W}$
3. $\hat{d}_s(A,B) = \hat{d}_s(B,A)$, $\forall_{A,B \in W}$

Dowód wynika z definicji i jest oczywisty. Zamiast porównywania wszystkich słów pojawiających się w tekstach, w zaprezentowanej w tej pracy metodzie, pod uwagę brane są tylko słowa nie uznane za nieistotne. Ważnym jest zatem fakt przechowywania w zasobach kompletnych drzew dokumentów i podczas dokonania porównania, uwzględnienia tylko aktualnie istotnych słów.

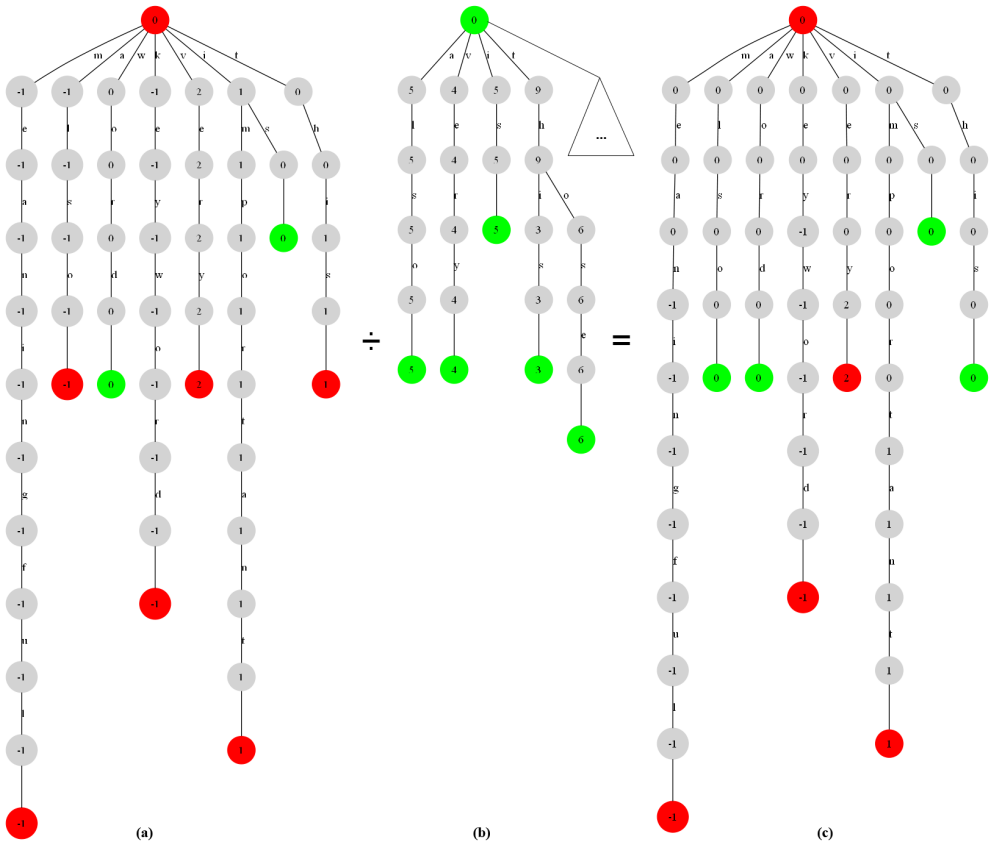
Zakładając, że drzewa słów w dokumentach X i Y (oznaczmy jako $t(X)$ i $t(Y)$) oraz drzewo słów nieistotnych (oznaczmy je jako $t(LSN)$) zostały już wyznaczone można przejść do prostego wyznaczenia miary różnic pomiędzy tymi dokumentami ze względu na częstość słów. Należy zauważyć, że każde słowo reprezentowane jest przez jeden liść w drzewie. Na początek należy wyznaczyć różnice wag pomiędzy odpowiadającymi sobie liśćmi z drzew $t(X)$ i $t(Y)$. Należy tu zwrócić uwagę, że nie każde słowo musi występować w każdym dokumencie, a co za tym idzie nie każda krawędź z drzewa $t(X)$ posiada odpowiadającą krawędź w drzewie $t(Y)$. Algorytm jako wejście otrzymuje więc dwa drzewa $t(X)$ i $t(Y)$, zwraca zaś drzewo $t(X-Y)$, które zawiera wszystkie gałęzie z obu drzew z wagami będącymi różnicą wag tych gałęzi. Jeśli gałąź nie istnieje w jednym z drzew, należy przyjąć, że ma ona w tym drzewie wagę 0.

[4] http://hypertextbook.com/facts/2001/JohnnyLing.shtml, http://www.wolframalpha.com/



Rysunek 1.8: (a) Drzewo wygenerowane dla tekstu ”this word is very very important”, (b) Drzewo wygenerowane dla tekstu ”those keyword is also meaningful word”, (c) Różnica drzew a) i b) uwzględniająca brakujące gałęzie.

Kolejnym etapem jest usunięcie z wynikowego drzewa słów nieistotnych. Ta czynność wykonywana jest dopiero w tym momencie, aby zmniejszyć liczbę potrzebnych obliczeń, oraz dlatego, żeby w każdym momencie wyznaczania miary różnic ze względu na częstość słów, lista słów nieistotnych dla obu testowanych dokumentów była identyczna. Usunięcia dokonuje się przeglądając w głąb aktualne drzewo $t(X - Y)$ i zerując te wierzchołki, dla których istnieją w drzewie słów nieistotnych $t(LSN)$ liście im odpowiadające, i których waga w tym drzewie jest większa od wyznaczonej granicy (avg', określona w 1.2.2). Tak otrzymane drzewo oznaczmy przez $t(X - Y)'$.



Rysunek 1.9: (a) Różnica drzew 1.8a) i 1.8b) uwzględniająca brakujące gałęzie, (b) Fragment drzewa słów nieistotnych, (c) Drzewo z wyzerowanymi wagami gałęzi zidentyfikowanych jako nieistotne.

Na koniec pozostaje jedynie przejrzeć kolejne liście reprezentujące istotne słowa w drzewie $t(X - Y)'$ i dodać wartości absolutne ich wystąpień. Rezultatem jest miara różnic tych dokumentów ze względu na częstość słów. Podany algorytm jest najprostszym sposobem porównywania dokumentów. Dzięki niemu możemy uzyskać informacje, czy dokumenty zawierały podobną ilość podobnych słów. Dzięki usuwaniu słów nieistotnych, nie mają one wpływu na miarę różnic pomiędzy dokumentami $\hat{d}_s$.

Niestety różna długość dokumentów i (co za tym idzie) ilość wystąpień tego samego słowa w różnych dokumentach, powoduje, że miara różnic pomiędzy nimi będzie wzrastać, dlatego docelowo należy zmodyfikować algorytm tak, aby zamiast ilości wystąpień słowa w tekście wykorzystywał względną ilość tych wystąpień.

Definicja 4 *Niech* $S_x = \sum_{i=1..n} x_i$ *a* $S_y = \sum_{i=1..n} y_i$. *Znormalizowana miara różnic (ze względu na częstość słów) pomiędzy dokumentami* $P_1(x_1, x_2, ..., x_n)$ *i*

$P_2(y_1, y_2, ..., y_n)$ zdefiniowana jest jako

$$d_s((x_1, x_2, ..., x_n), (y_1, y_2, ..., y_n)) = |\frac{x_1}{S_x} - \frac{y_1}{S_y}| + |\frac{x_2}{S_x} - \frac{y_2}{S_y}| + ... + |\frac{x_n}{S_x} - \frac{y_n}{S_y}|$$

gdzie x_i, y_i to odpowiednio ilości wystąpień słowa o numerze i w dokumentach P_1 i P_2.

Każde słowo występujące w obliczeniach jest traktowane jako kolejny wymiar przestrzeni, w której znajdują się dokumenty.

Twierdzenie 2 *Funkcja $d_s(-,-)$ określona na zbiorze $W \times W \to \mathbb{R}$ gdzie W to zbiór wszystkich dokumentów spełnia następujące własności:*

1. $d_s(A, B) \geq 0$, $\forall_{A,B \in W}$
2. $d_s(A, A) = 0$, $\forall_{A \in W}$
3. $d_s(A, B) = d_s(B, A)$, $\forall_{A,B \in W}$

Dowód

1. Dowód wynika bezpośrednio z definicji funkcji d_s, w której żaden ze składników sumy nie może przyjmować wartości ujemnych.
2. $d_s(A, A) = \sum_{i=1..n} |\frac{a_i}{S_a} - \frac{a_i}{S_a}| = \sum 0 = 0$
3. $d_s(A, B) = \sum_{i=1..n} |\frac{a_i}{S_a} - \frac{b_i}{S_b}| = \sum_{i=1..n} |-(-\frac{a_i}{S_a} + \frac{b_i}{S_b})| = \sum_{i=1..n} |-(\frac{b_i}{S_b} - \frac{a_i}{S_a})| = \sum_{i=1..n} |\frac{b_i}{S_b} - \frac{a_i}{S_a}| = d_s(B, A)$

Tak zdefiniowana Miara różnic nie jest wrażliwa na różne rozmiary dokumentów, tylko na rzeczywistą względną ilość wystąpień poszczególnych słów w dokumentach.

1.3 Metoda częstości n-gramów

Drugą cechą istotną podczas porównywania dokumentów jest struktura tekstu i jego konstrukcja. Składają się na nie wystąpienia nowych linii, znaki interpunkcyjne, długości wierszy, czy miejsca występowania tabel. Rozwijanych jest wiele metod porównywania strukturalnego podobieństwa dokumentów. Zadanie to jest ułatwione, o ile dokument posiada wyraźną metastrukturę (jak w przypadku stron internetowych - Cruz *et al.* (1998), gdzie wszelkie właściwości tekstu są wyraźnie oznaczone poprzez odpowiednie znaczniki). Gorzej, jeśli dostęp do

metadanych jest ograniczony, lub takie metadane nie istnieją. Wówczas analizie podlegać może jedynie czysty tekst.

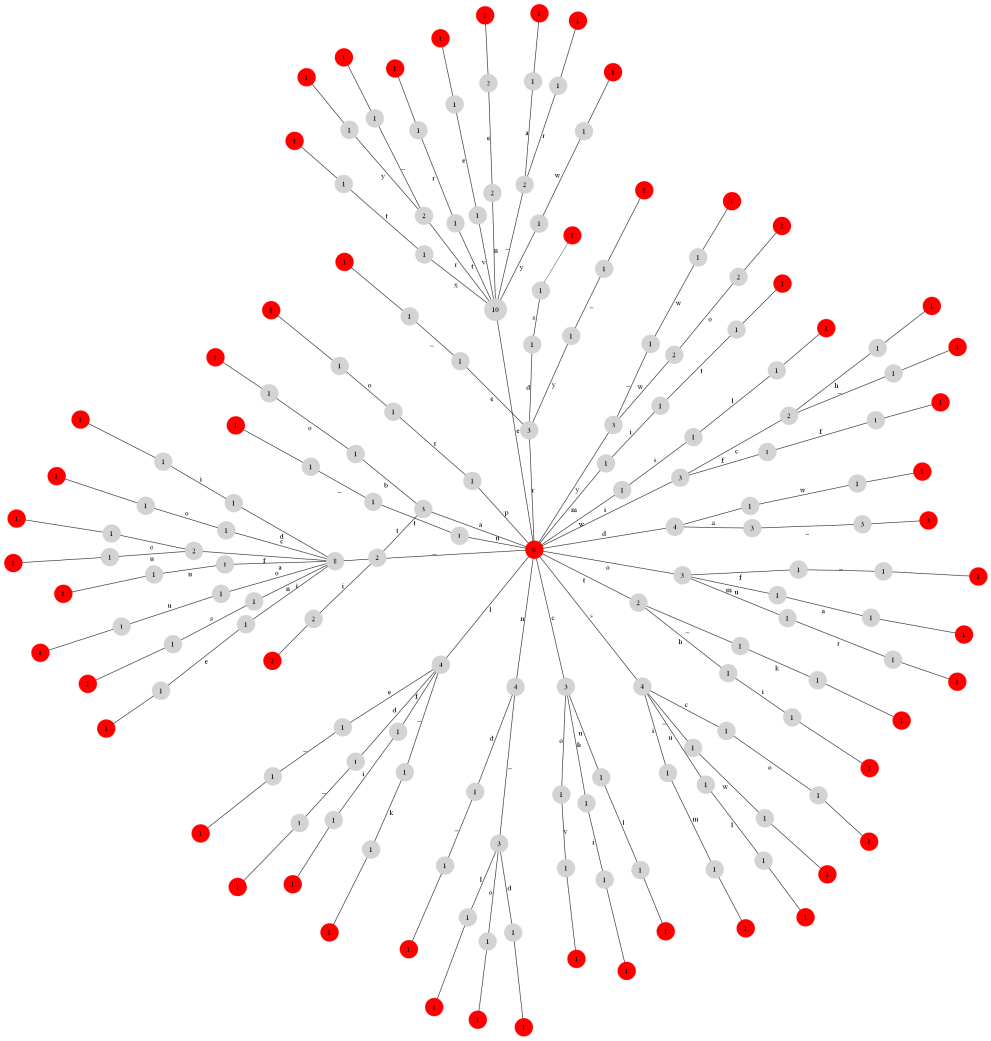
W tym przypadku analizowany jest tekst pełny, nie zmodyfikowany poprzez usuwanie lub zamienianie jakichkolwiek znaków, ponieważ mogą one wpływać na strukturę tekstu i opisywać jego formatowanie. Jednocześnie są tu również brane pod uwagę wszelkie słowa uznane za nieistotne podczas analizy częstości słów. Dzięki temu analizie podlega cała struktura tekstu, a co za tym idzie algorytm jest bardziej uczulony na wygląd i kształt dokumentu. Wybrana w ramach niniejszych rozważań metoda analizy strukturalnej tekstu opiera się o częstości n-gramów.

Definicja 5 *n-gramem tekstu T jest każdy ciągły podciąg znaków $t \in T$ o długości n.*

W celu szybkiej analizy n-gramów, został zmodyfikowany przedstawiony wyżej algorytm budujący drzewo słów dla dokumentu. Różnica polega na tym, że do drzewa dodawane są wszystkie ciągi n-literowe, za każdym razem nową krawędź rozpoczynające w korzeniu drzewa. W tej pracy ograniczono się do badania ciągów 3-literowych, które jak stwierdzono w testach dobrze opisują strukturę tekstu.

Algorytm 6 ***Zbuduj drzewo n-gramów G tekstu T***

$g \Leftarrow nowyWierzchołek()$
$G \Leftarrow \{g\}$
$p \Leftarrow g$
$i \Leftarrow 0$
WHILE$(i < 3)$
$ustawWskaźnikCzytaniaTekstuNaPozycji(i, T)$
$r \Leftarrow 0$
WHILE$(c = przeczytajZnak(T))$
$r \Leftarrow r + 1$
IF $(\exists v \in wierzchołki(G)(p, v) \in krawędzie(G) \wedge etykieta(p, v) = c)$
$licznik(v) \Leftarrow licznik(v) + 1$
ELSE
$v \Leftarrow nowyWierzchołek()$
$licznik(v) \Leftarrow 1$
$wierzchołki(G) \Leftarrow wierzchołki(G) \cup \{v\}$
$etykieta(p, v) \Leftarrow c$
$krawędzie(G) \Leftarrow krawędzie(G) \cup \{(p, v)\}$
END IF
IF$(r = 3)$
$p \Leftarrow g$
$r \Leftarrow 0$



Rysunek 1.10: Drzewo 3-gramów wygenerowane dla tekstu „This is simple text about keywords generation discovery. Of course all keywords are difficult for automatic generation of discovery but in limited way we could achieve results which will fulfil our needs and retrieve proper keywords.”

ELSE
$p \Leftarrow v$
END IF
END WHILE
$i \Leftarrow i + 1$
END WHILE

1.3.1 Miara różnic (ze względu na częstość n-gramów) między dokumentami

Definicja 6 *Miara różnic (ze względu na częstość n-gramów) pomiędzy dokumentami* $P_1(x_1, x_2, ..., x_n)$ *i* $P_2(y_1, y_2, ..., y_n)$ *zdefiniowano jako*

$$d'_n((x_1, x_2, ..., x_n), (y_1, y_2, ..., y_n)) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_n - y_n|$$

gdzie x_i, y_i *to odpowiednio ilości wystąpień n-gramu o numerze i w dokumentach* P_1 *i* P_2.

Podobnie jak w przypadku słów kluczowych lepszym wyborem jest tutaj obliczenie względnej ilości wystąpień n-gramów. Definicja wykorzystywana w ramach tej pracy ma więc postać:

Definicja 7 *Niech* S_x *oznacza ilość n-gramów w tekście* P_1 *a* S_y *- ilość n-gramów w tekście* P_2. *Znormalizowana miara różnic (ze względu na częstość n-gramów) pomiędzy dokumentami* $P_1(x_1, x_2, ..., x_n)$ *i* $P_2(y_1, y_2, ..., y_n)$ *zdefiniowana jest jako*

$$d_n((x_1, x_2, ..., x_n), (y_1, y_2, ..., y_n)) = |\frac{x_1}{S_x} - \frac{y_1}{S_y}| + |\frac{x_2}{S_x} - \frac{y_2}{S_y}| + ... + |\frac{x_n}{S_x} - \frac{y_n}{S_y}|$$

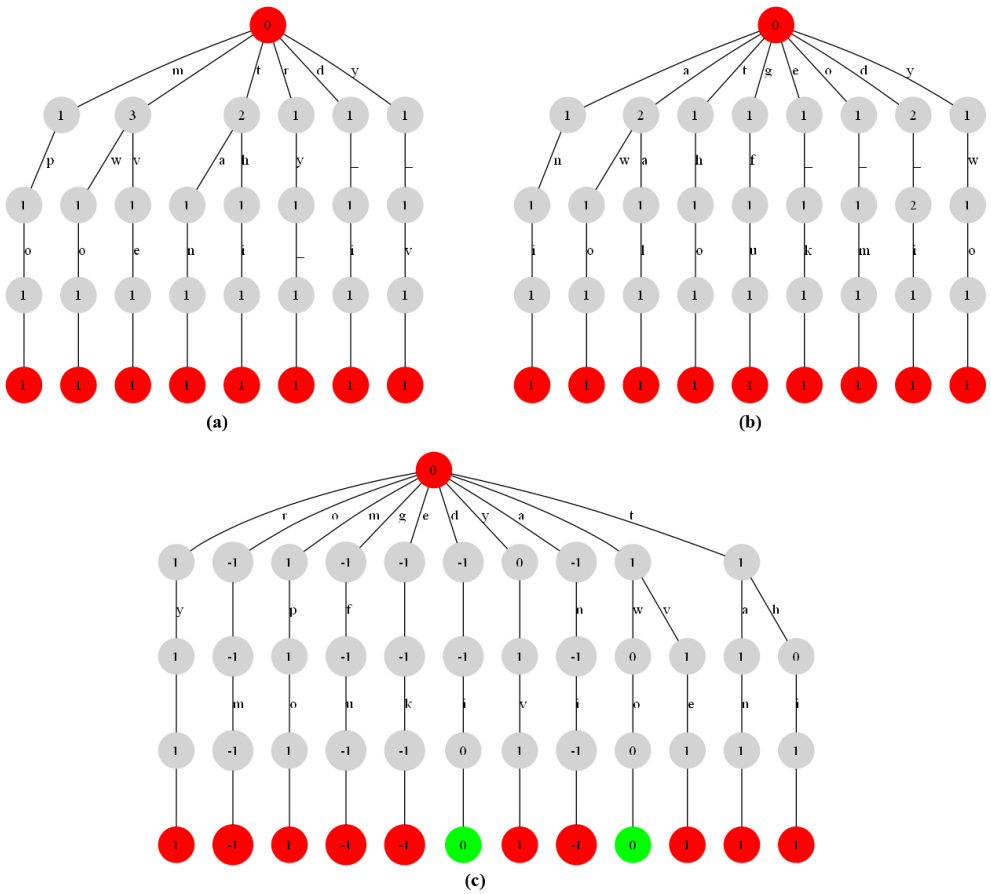
gdzie x_i, y_i *to odpowiednio ilości wystąpień n-gramu o numerze i w dokumentach* P_1 *i* P_2.

Twierdzenie 3 *Funkcja* $d_n(-,-)$ *określona na zbiorze* $W \times W \to \mathbb{R}$ *gdzie* W *to zbiór wszystkich dokumentów spełnia następujące własności:*

1. $d_n(A, B) \geq 0$, $\forall_{A,B \in W}$
2. $d_n(A, A) = 0$, $\forall_{A \in W}$
3. $d_n(A, B) = d_n(B, A)$, $\forall_{A,B \in W}$

Dowód wynika bezpośrednio z definicji funkcji d_n i jest trywialny.

Mając wygenerowane drzewa n-gramów dla tekstów X i Y (oznaczone przez $tn(X)$ i $tn(Y)$) łatwo można wygenerować drzewo $tn(X - Y)$, które będzie zawierało wszystkie gałęzie z obu drzew, przy czym wierzchołki będą etykietowane różnicami wag odpowiadających im wierzchołków z tych drzew. Przykładowe drzewa pokazane są na rysunku 1.11. Podobnie jak w poprzednim przypadku należy przyjąć, że nieistniejące krawędzie mają wagi równe 0.



Rysunek 1.11: (a) Drzewo 3-gramów wygenerowane dla tekstu , (b) Drzewo 3-gramów wygenerowane dla tekstu , Różnica drzew a) i b) uwzględniająca brakujące gałęzie.

Każdy z liści tego drzewa odpowiada jednemu n-gramowi występującemu w tekście, więc również tutaj, aby wyznaczyć miarę różnic (ze względu na częstość n-gramów) pomiędzy dokumentami należy przejrzeć drzewo $tn(X - Y)$ w poszukiwaniu liści o wagach niezerowych i zsumować bezwzględne wartości ich etykiet.

1.4 Złożoność Kołmogorowa

Ostatnią metodą użytą do porównywania dokumentów jest złożoność Kołmogorowa tekstów pisanych. Okazuje się, że pomimo analizowania dokumentu z szerszej perspektywy (nie są rozważane elementy składowe, a tekst jako całość), metoda ta jest w stanie wyznaczyć różnice pomiędzy tekstami reprezentującymi takie kategorie jak poezja, proza, program komputerowy, tekst matematyczny, co z założeniem wykorzystania ustalonych prototypów danych badane było w Pieńkowska (2005).

Formalna definicja złożoności Kołmogorowa (Li & Vitanyi (1997)) wykorzystuje pojęcie maszyny Turinga. Formalne właściwości złożności Kołmogorova analizowane były w Gács & Körner (1973) a wykorzystaniem jej do porównywania tekstów zajmowano się w Pieńkowska (2005).

Definicja 8 *Złożność Kołmogorowa słowa x, oznaczana jako K(x) jest zdefiniowna jako długość najkrótszego programu działającego na maszynie Turinga, który zwraca słowo x.*

Definicja 9 *Złożoność Kołmogorowa tekstu X (oznaczana jako $K(X)$) jest długością najkrótszego skompresowanego pliku binarnego $X*$, z którego może zostać zrekonstruowany oryginalny tekst.*

Zasada łańcuchowej złożoności Kołmogorowa (udowodniona w wersji oparrej o pojęcie maszyny Turinga w Li & Vitányi (1997)) mówi, że najkrótszy program, zwracający X i Y jest dłuższy o co najwyżej logarytmiczny czynnik od programu, zwracającego X oraz programu zwracającego Y, zakładając znajomość X.

Uwaga 2 ***Zasada łańcuchowa złożoności Kołmogorowa.***

$$K(X,Y) = K(X) + K(Y|X) + O(log(K(X,Y)))$$

W dalszych rozważaniach złożoność ta zostaje zastąpiona przez definicję złożoności kompresji (za Cilibrasi & Vitányi (2005). Do testów praktycznych wykorzystano kompresję ZIP dostępną w standardowej dystrybucji Javy w paczce `java.util.zip`

Definicja 10 *Kompresor jest koderem, który mapuje dokument X na $\{0,1\}*$, tworząc kod prefiksowy $X*$, z którego możliwe jest odtworzenie tekstu X. Przez K_c oznaczamy funkcję dziającą ze zbioru dokumentów W na zbiór liczb naturalnych, przyporządkowując dokumentom długość ich skompresowanej wersji. K_c nazwiemy złożonością kompresji.*

$$K_c : W \to \mathbb{N}$$

$$K_c(X) = |X*|$$

Definicja 11 *Kompresor K_c jest normalny, jeśli spełnia z dokładnością do $O(logn)$, gdzie n jest długością najdłuższego dokumentu w zbiorze W, następujące własności.*

1. $K_c(XX) = K_c(X)$
2. $K_c(\lambda) = 0$, *gdzie* λ *to ciąg pusty*
3. $K_c(XY) \geq K_c(X)$
4. $K_c(XY) = K_c(YX)$
5. $K_c(XY) + K_c(Z) \leq K_c(XZ) + K_c(YZ)$

Możemy założyć, że wykorzystywana w testach kompresja ZIP spełnia te warunki.

Definicja 12 *Na tej podstawie definiuje się warunkową złożoność kompresji jako:*

$$K_c(Y|X) = K_c(XY) - K_x(Y)$$

1.4.1 Miara różnic (ze względu na złożoność Kołmogorova) pomiędzy dokumentami

Opierając się na złożoności kompresji tekstu ($K_c()$), będącej substytutem złożoności Kołmogorowa ($K()$) może zostać zdefiniowany następny typ miary różnic pomiędzy dokumentami. Definicja ta opiera się na spostrzeżeniu, że dwa podobne pliki skompresują się razem lepiej niż dwa różne.

Definicja 13 *Miara różnic (ze względu na złożoność Kołmogorowa) pomiędzy dokumentami P_1 i P_2 zdefiniowana jest jako*

$$d_k(P_1, P_2) = \left| \frac{K_c(P_1|P_2) - K_c(P_2|P_1)}{K_c(P_1P_2)} \right|$$

Twierdzenie 4 *Funkcja $d_k(-,-)$ określona na zbiorze $W \times W \to \mathbb{R}$ gdzie W to zbiór wszystkich dokumentów spełnia następujące własności:*

1. $d_kA, B) \geq 0$, $\forall_{A,B \in W}$
2. $d_k(A, A) = 0$, $\forall_{A \in W}$
3. $d_k(A, B) = d_k(B, A)$, $\forall_{A,B \in W}$

Dowód

1. Przy założeniu, że teksty są długości ≥ 0, wszystkie składniki definicji spełniają ten warunek.

2.
$$d_k(A,A) = \left|\frac{K_c(A|A) - K_c(A|A)}{K_c(AA)}\right| \tag{1.1}$$
$$= \left|\frac{0}{K_c(AA)}\right| \tag{1.2}$$
$$= 0 \tag{1.3}$$

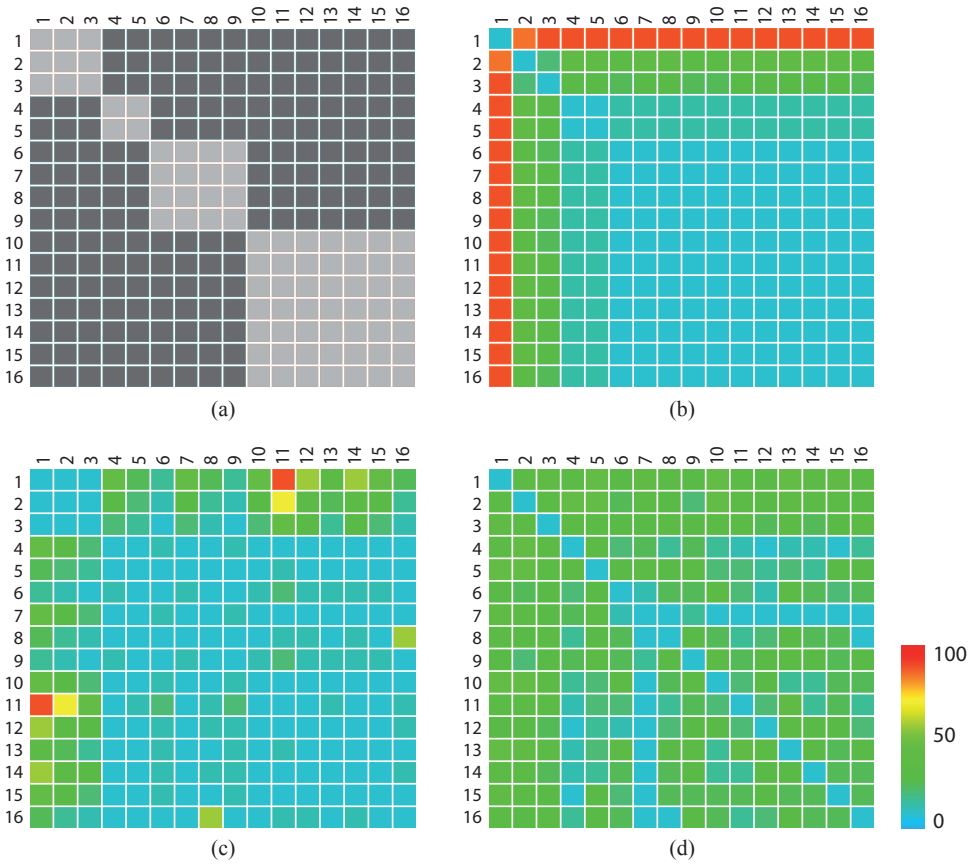
3.
$$d_k(A,B) = \left|\frac{K_c(A|B) - K_c(B|A)}{K_c(AB)}\right| \tag{1.4}$$
$$= \left|-\frac{K_c(B|A) - K_c(A|B)}{K_c(BA)}\right| \tag{1.5}$$
$$= \left|\frac{K_c(B|A) - K_c(A|B)}{K_c(AB)}\right| \tag{1.6}$$
$$= d_k(B,A) \tag{1.7}$$

1.5 Cechy, na które zwracają uwagę poszczególne miary różnic dokumentów.

Aby przeanalizować różnice wykrywane przez poszczególne algorytmy, wykonano testy na przykładowych zbiorach losowych dokumentów. Każdy z typów miary różnic wskazuje na inne różnice pomiędzy tekstami, co widać na ilustracji 1.12, na której przedstawione są wyniki reprezentacyjnego testu dla 16 przykładowych dokumentów, zawierających:

1. 3 teksty naukowe dotyczące różnych zagadnień (nr 1-3):
   - „*A comparison of event models for naive Bayes*"
   - „*Diameter of the World Wide Web*"
   - „*Explicit construction of linear sized tolerant networks*"
2. 2 różne kody źródłowe w języku Java (nr 4-5),
3. 4 fragmenty powieści o miłości (nr 6-9),

4. 6 wierszy o miłości różnych autorów (nr 10-16).



Rysunek 1.12: (a) kolor jasnoszary oznacza miejsca gdzie miary różnic powinny być bliższe, (b-d) Miary różnic wyznaczone za pomocą złożoności Kołmogorowa (b), za pomocą zliczania n-gramów (c) i za pomocą zliczania słów (d). Miary różnic oznaczone są kolorami od niebieskiego (najbliższe sobie) do czerwonego (najdalsze).

Zliczanie słów. Na wykresie miar różnic 1.12 (d) widać, że te wyznaczone za pomocą zliczania słów pokazują różnice pomiędzy poszczególnymi reprezentantami kolejnych klas dokumentów. I tak widać, że:

- elementy grupy 1 pozostają dość odległe od siebie i od wszystkich pozostałych tekstów;
- dokumenty z grupy 2 pozostają również dość odległe od pozostałych reprezentantów;

- dość blisko siebie występują wszyscy reprezentanci grupy 3, jednocześnie będąc bardziej odległymi we wszystkich przypadkach od tekstów naukowych i kodów źródłowych oraz w większości przypadków od reprezentantów wierszy;

- teksty z grupy 4 są w większości przypadków różne od siebie - skutek różnego doboru słów i abstrakcji tekstu.

Powyższe wyniki świadczą o tym, że można rozróżnić częściowo typy dokumentów już na tym etapie.

Zliczanie n-gramów. Zliczanie n-gramów powinno wskazać na różnice w budowie strukturalnej dokumentów. Na 1.12 (c) widać, że:

- teksty z grupy 1 są podobne do siebie, więc podobieństwa w ich strukturze są tu widoczne;

- kody źródłowe z grupy 2 są bardzo podobne do siebie i do wierszy - wpływ ma na to duża ilość nowych linii i mała ilość tekstu w wierszach;

- fragmenty powieści z grupy 3 są podobne, aczkolwiek widoczne są różnice między ich budową;

- wiersze również wykazują się podobieństwem, jednak z widocznymi drobnymi różnicami.

Prezentowany algorytm potrafi więc wskazać na różnice w budowie tekstów.

Złożoność Kołmogorowa. Zgodnie z wykonanymi w Frank *et al.* (2000) badaniami kategoryzacja tekstów z użyciem modeli kompresji, można rozróżnić poezję od prozy czy kodu źródłowego. Na 1.12 (b) widać, że:

- teksty naukowe (1) są bardzo odległe od pozostałych dokumentów, ale duże odległości występują również wśród nich;

- podobnie sprawa się ma z kodami źródłowymi (2) które różnią się od pozostałych elementów;

- w tym przypadku fragmenty powieści o miłości (3) i wierszy miłosnych (4) znalazły się w pobliżu siebie.

Połączenie metod. Analiza powyższych wyników pokazuje, że nie można ograniczać się do analizy tylko jednej z miar różnic. Dobre wyznaczenie powiązań pomiędzy dokumentami wymaga rozważenia jednocześnie wielu miar różnic, które porównują dokumenty pod względem różnych właściwości. Jednocześnie widać, że każda z przedstawionych powyżej miar różnic jest wrażliwa na szczególnego typu cechy tekstów. Na podstawie tych spostrzeżeń w ramach niniejszej pracy sformułowano tezę, że do dobrego oddzielenia od siebie różnych dokumentów, należy wykorzystać wszystkie przedstawione typy miar różnic. Za nową miarę różnic można przyjąć:

Definicja 14 *Miara różnic d pomiędzy dokumentami P_1 i P_2 zdefiniowana jest jako*

$$d(P_1, P_2) = max(\left\{\hat{d_s}(P_1, P_2), \hat{d_k}(P_1, P_2), \hat{d_n}(P_1, P_2)\right\}).$$

gdzie przez $\hat{d_-}$ oznaczono znormalizowane odpowiednie miary różnic d_-. W testach wykazano, że przyjmując taki typ miary różnic można uzyskać satysfakcjonujące wyniki w odróżnianiu dokumentów. Dzięki temu wszystkie różnice mogą zostać wykryte i różne dokumenty mogą zostać oddzielone od siebie. Jednocześnie pokazano że te trzy przedstawione typy miary różnic w zupełności wystarczą do odróżniania dokumentów od siebie. Testy wykonano na zbiorach dokumentów takich jak w sekcji 1.5

Twierdzenie 5 *Funkcja $d(-,-)$ określona na zbiorze $W \times W \to \mathbb{R}$ gdzie W to zbiór wszystkich dokumentów spełnia następujące własności:*

1. $d(A, B) \geq 0$, $\forall_{A,B \in W}$,
2. $d(A, A) = 0$, $\forall_{A \in W}$,
3. $d(A, B) = d(B, A)$, $\forall_{A,B \in W}$.

Dowód wynika bezpośrednio z definicji.

Idea ta, traktująca jako podobne dokumenty, które są podobne jednocześnie ze względu na wszystkie trzy typy miar odległości, jest szczegółowo omówiona w kolejnym rozdziale, który poświęcony został wykorzystaniu przedstawionych miar różnic w procesie podziału dokumentów na spójne tematycznie kategorie. Również tam pokazane jest, że dokumenty podobne w tym sensie, są podobne również semantycznie.

Rozdział 2

Kategoryzacja tekstów

W tym rozdziale przedstawiono algorytmy kategoryzacji danych opierające się o sieci neuronowe Kohonena. W ramach niniejszej pracy wykorzystano te algorytmy do kategoryzacji dokumentów naukowych w oparciu o zaprezentowane w rozdziale 1 trzy typy miar różnic i pokazano, że wykorzystanie ich wszystkich gwarantuje dobrą kategoryzację. W ramach tej rozprawy zaproponowano również modyfikację sieci neuronowych Kohonena, wprowadzając dynamiczną możliwość zmiany ich struktury poprzez usuwanie zbędnych węzłów i zastąpienie standardowego sąsiedztwa węzłów przez uznanie za sąsiadów węzłów o prototypach kategorii najbliższych względem prototypu węzła rozpatrywanego, oraz wprowadzono modyfikację kroku zmiany prototypów węzłów poprzez zastąpienie wyznaczania uogólnionej mediany algorytmem aproksymacyjnym opartym o ideę algorytmów genetycznych.

2.1 Wprowadzenie

Biorąc pod uwagę rosnącą liczbę dostępnych informacji, ważnym problemem staje się kategoryzacja tekstów pisanych (Frank *et al.* (2000)). Przez kategoryzację należy rozumieć przydzielanie konkretnych tekstów do konkretnych kategorii. Wyróżnia się dwa podejścia do klasyfikacji danych:

- uczenie nadzorowane - wcześniej sklasyfikowane, dobrze reprezentujące swoje kategorie dane są używane jako dane treningowe algorytmu;
- uczenie nienadzorowane - kategorie są wynikiem działania algorytmu.

Do klasyfikacji tekstów można wykorzystać wiele różnorodnych metod. Popularnym przykładem są naiwne klasyfikatory Bayesa, będące klasyfikatorami probabilistycznymi opartymi na nierealnych założeniach o niezależności predykatów,

aczkolwiek dających sensowne rezultaty (McCallum & Nigam (1998)). Sztandarowym przykładem klasyfikatorów Bayesa jest filtrowanie poczty i oznaczanie odpowiednich listów jako *spam*.

Innym przykładem jest przydzielanie kolejnych tekstów do kategorii na podstawie wyznaczonej wstępnie miary różnic i jakiegoś ustalonego, bardzo małego ϵ:

Algorytm 7 ***Prosta kategoryzacja***

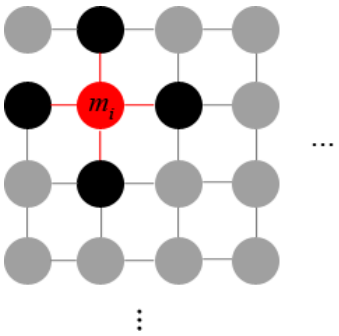
Jeśli istnieje kategoria, której prototyp jest odległy od aktualnie testowanego dokumentu o mniej niż ϵ, dodaj ten dokument do tej kategorii, w przeciwnym wypadku stwórz nową kategorię i uznaj testowany dokument za jej prototyp.

Taki algorytm jest szybki, ale nie wyznacza optymalnych kategorii i jego skuteczność nie jest duża. Wynika to z faktu, że kategorie nie są wyznaczane dynamicznie i nie mogą zostać w żadnym momencie zweryfikowane.

W dalszych rozważaniach za podstawowy algorytm kategoryzacji przyjęto algorytm samoorganizujących się sieci Kohonena dla elementów symbolicznych (Cilibrasi & Vitányi (2005)), który w ramach niniejszej pracy dostosowano do potrzeb kategoryzacji dokumentów, po raz pierwszy wykorzystując jako bazę zdefiniowane w poprzednim rozdziale miar różnic.

2.2 Sieci neuronowe Kohonena

Sieć Kohonena jest popularną nazwą dla sieci samoorganizujących się (Self-Organizing Maps (SOM)).



Rysunek 2.1: Przykładowa kwadratowa sieć Kohonena z zaznaczonym wybranym węzłem (kolor czerwony) i jego sąsiadami (kolor czarny).

Zgodnie z informacjami zawartymi w Kohonen & Honkela (2007), sieć Kohonena jest pewnym uporządkowanym zbiorem modeli danych (m_i) definiujących

mapowanie ze zbioru danych $W = w_1, ..., w_n$. Każdy z węzłów sieci (k_i) skojarzony jest z modelem danych (m_i). Każdy element ze zbioru danych w_k jest mapowany na ten węzeł k_i, którego model m_i jest mu najbliższy (zwykle w sensie odległości w pewnej metryce. Zbiór modeli danych $M = m_1, ..., m_m$ jest wyznaczany przez algorytm SOM.

Model sieci Kohonena został stworzony na podstawie prac dotyczących sieci neuronowych (Kohonen (1984)). Oryginalnie został użyty do wizualizacji rozkładu wektorów takich, jak na przykład zbiory uporządkowane atrybutów statystycznych. Okazuje się jednak, że mapowanie typu SOM może zostać zastosowane dla dowolnych danych, dla których można zdefiniować miary różnic. Przykład takiej procedury dla danych niewektorowych pokazał sam Kohonen w Kohonen & Somervuo (2002).

2.2.1 Model matematyczny

Poniżej przedstawiony został ogólny model matematyczny sieci SOM, później uszczegółowiony w celu kategoryzacji tekstów.

Model ogólny

Rozważmy zbiór danych X w postaci n-wymiarowych wektorów:

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$

oraz sekwencję modeli $m_i(t)$:

$$m_i(t) = \begin{bmatrix} m_{i1}(t) \\ m_{i2}(t) \\ \vdots \\ m_{in}(t) \end{bmatrix}.$$

Sekwencja ta jest zdefiniowana jako proces, w którym wartość $m_i(t+1)$ jest obliczana iteracyjnie na podstawie wartości poprzedniej $m_i(t)$ i kolejnych elementów $x(t)$ ze zbioru danych X:

$$m_i(t+1) = m_i(t) + \alpha(t)h_{ci}(t)[x(t) - m_i(t)].$$

W powyższym równaniu $\alpha(t)$ oznacza skalar definiujący wielkość korekty, którego wartość zmniejsza się z krokiem t. Indeks i oznacza model, który jest przetwarzany. Indeks c jest indeksem modelu, który jest najbliższy (ma najmniejszą

odległość) elementowi $x(t)$. Współczynnik $h_{ci}(t)$ jest nazywany funkcją sąsiedztwa i jest równy 1 wtedy gdy $i = c$, a jego wartość maleje wraz ze wzrostem odległości pomiędzy modelami m_i i m_c w siatce sieci.

Szczegółowy przegląd teorii sieci SOM znajduje się w Fort (2006). Autorzy prezentują tam osiągnięte wyniki i udowodnione twierdzenia dotyczące sieci Kohonena.

Algorytm 8 ***Scenariusz uczenia się***

W każdym kroku czasu t:

1. *Obserwuj aktualny stan sieci $m(t)$.*
2. *Wybierz akcję a_t identyfikowaną przez wielkość korekty (α_t).*
3. *Wykonaj akcję a_t.*
4. *Obserwuj następny stan sieci $m(t+1)$.*
5. *Ucz się na podstawie trójki $\langle m(t), a_t, m(t+1)\rangle$.*

Prezentowany sposób uczenia się zakłada ciągłą znajomość stanu sieci. Według definicji każdy stan sieci zależy od stanu poprzedniego całej sieci. W praktyce stan każdego elementu sieci będzie zależał od jego stanu poprzedniego i stanu tylko jego sąsiadów.

Model szczegółowy

Modelem danych w zaproponowanym w tej pracy sposobie kategoryzacji tekstów są wektory n-wymiarowe postaci:

$$x(t) = \begin{bmatrix} x^1(t) \\ x^2(t) \\ \vdots \\ x^n(t) \end{bmatrix}$$

spełniające warunek, że:

$$x^i(t) = \left\{ \begin{array}{c} 1 \\ 0 \end{array} \right\},$$

gdzie $x^i(t)$ przyjmuje wartość 1 jeśli dokument T_i należy do rozważanego modelu w kroku czasu t i 0 w przeciwnym wypadku. Na tej podstawie model wszystkich n badanych tekstów ma w każdym kroku czasu t postać:

$$x_{00}(t) = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

W każdym kroku czasu t konstruowany jest nowy model sieci $m(t)$ spełniający następujące warunki:

$$m(t) = \begin{bmatrix} m_{11}(t) & m_{21}(t) & \dots & m_{j1}(t) \\ m_{12}(t) & m_{21}(t) & \dots & m_{j2}(t) \\ \vdots & \vdots & \ddots & \vdots \\ m_{1j}(t) & m_{2j}(t) & \dots & m_{jj}(t) \end{bmatrix},$$

takie, że

$$m_{kl}(t) = \begin{bmatrix} x_{kl}^1(t) \\ x_{kl}^2(t) \\ \vdots \\ x_{kl}^n(t) \end{bmatrix}$$

oraz

$$\sum_{k,l=1\dots j} m_{kl}(t) = \sum_{k,l=1\dots j} \begin{bmatrix} x_{kl}^1(t) \\ x_{kl}^2(t) \\ \vdots \\ x_{kl}^n(t) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = x_{00}(t).$$

W kroku pierwszym ($t = 1$) wektory $m_{kl}(1)$ dobierane są losowo. W każdym kolejnym ($t > 1$) stan sieci zależny jest o stanu w kroku poprzednim:

$$m_{kl}(t) = m_{kl}(t-1) + m_{kl}^T(t-1) \times (M \times D_{kl}) \times A_{kl},$$

gdzie D_{kl} jest $n \times n$ zerojedynkową macierzą definiującą sąsiedztwo składników modelu m w stosunku do składnika m_{kl} a A_{kl} jest $n \times n$ macierzą przejścia definiującą operacje modyfikacji modelu m, a:

$$M = \begin{bmatrix} m_{11}(t) \\ m_{12}(t) \\ \vdots \\ m_{1j}(t) \\ m_{21}(t) \\ \vdots \\ m_{2j}(t) \\ \vdots \\ m_{jj}(t) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{M}_{n \times n}.$$

W każdym kroku czasu algorytm wyznacza więc dla każdego $m_{kl}(t)$ macierz przejścia A_{kl} i oblicza jego wartość na jej podstawie.

2.2.2 Algorytm

Zgodnie z Kohonen & Somervuo (2002) użyty w rozważaniach zbiór danych $T = \{t_1, ..., t_n\}$ składa się z n tekstów. Na zbiorze T wyznaczone zostały trzy typy miar różnic: oparte o statystyczne słowa kluczowe (ozn. $d_s(-,-)$), częstości n-gramów (ozn. $d_n(-,-)$) oraz miara różnic Kołmogorowa (ozn. $d_k(-,-)$).

W niniejszej pracy przyjęto, że sieć neuronowa składa się z $m \times m$ węzłów, gdzie $m = \lfloor \sqrt[4]{n} \rfloor$, zatem algorytm potrafi podzielić zbiór dokumentów na m^2 określonych automatycznie kategorii. Mamy tu do czynienia z uczeniem nienadzorowanym. Poniższy algorytm zakłada, że nie została przeprowadzona żadna częściowa kategoryzacja tekstów.

Algorytm 9 ***Samoorganizująca się sieć Kohonena***

1. *Stwórz sieć neuronową o wielkości* $m \times m$*,gdzie* $m = \lfloor \sqrt[4]{n} \rfloor$ *a* n *oznacza ilość analizowanych tekstów. Każdy węzeł* w_i *posiada czterech sąsiadów* $(n(i))$*. Do każdego węzła* w_i *przyporządkowany jest prototyp* p_i *oraz lista dokumentów* $l(w_i)$*.*

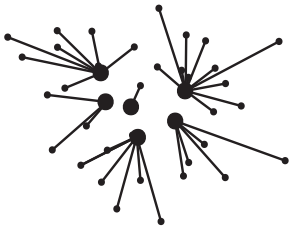
2. *Dla każdego węzła* w_i *wybierz z* T *losowy tekst i uznaj go za prototyp kategorii* $(p_i = w_r)$*.*

3. *Dla każdego tekstu* t_i *wybierz najbliższy mu (według zadanej miary różnic) prototyp* (p_c) *w sieci i dodaj go do listy dokumentów tego prototypu* $(l(w_c) = l(w_c) \cup \{t_i\})$

4. *Dla każdego węzła* w_i *wyznacz uogólnioną medianę dla wszystkich dokumentów z tego węzła* $(l(w_i))$ *i jego sąsiadów* $(l(n_i))$*, a następnie zastąp nią prototyp tego węzła. Uogólniona mediana jest zdefiniowana jako element* $m \in T$ *minimalizujący funkcję:*

$$\sum_{t_s \in l(w_i) \cup l(n_i)} d^2(t_s, m)$$

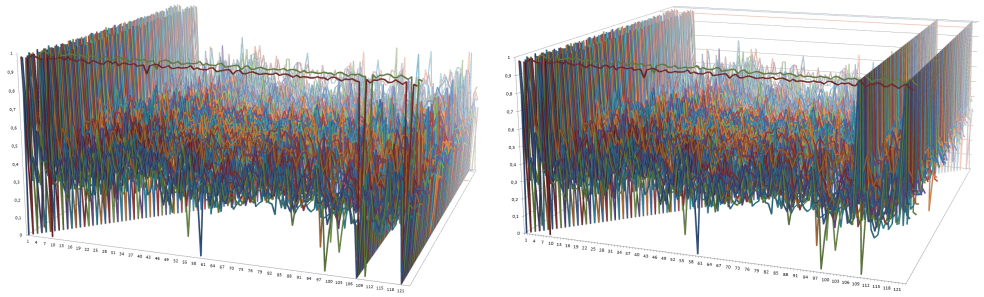
5. *Powtarzaj krok 3 i 4 do momentu ustabilizowania się sieci.*



Rysunek 2.2: Rzut na płaszczyznę przestrzeni miary różnic tekstów. Poszczególne teksty są reprezentowane jako punkty. Punkty większe reprezentują prototypy kategorii. Każdy dokument połączony jest linią z prototypem kategorii w której się znajduje.

W powyższym algorytmie dla oznaczenia miar różnic między tekstami użyty został symbol d. W kolejnych wykonaniach algorytmu odległość ta została zastąpiona odpowiednio przez d_s, d_n i d_k.

Aby spełnić założenia algorytmu, wykorzystuje się tu miary znormalizowane, które przyjmują wartości od 0 do 2. Ponieważ w przypadku błędów odczytu plików mogą powstać błędy, takie dokumenty uznaje się za najbardziej odległe i przyjmuje, że ich odległość od pozostałych dokumentów wynosi 2.



(a) (b)

Rysunek 2.3: Przykładowa macierz miar różnic wyznaczonych pomiędzy dokumentami (a) przed korektą i (b) po korekcie błędnych obliczeń.

Rezultatem wykonania algorytmu jest przyporządkowanie każdego z dokumentów do trzech kategorii. W poszczególnych kategoryzacjach dokumenty zostaną podzielone ze względu na:

- zawartość słów (d_s),
- strukturę tekstu (d_n),
- typ tekstu(d_k).

Zaproponowana w niniejszej pracy ilość węzłów sieci równa m^2 (gdzie $m = \lfloor \sqrt[4]{n} \rfloor$) została wyznaczona na podstawie serii testów. Ilość ta jest wystarczająca, aby podział na kategorie był skuteczny. W większości przypadków jest również zbyt duża - wówczas niektóre z węzłów posidają jedynie prototypy.

2.2.3 Zbieżność sieci

Ważnym zagadnieniem jest zbieżność sieci neuronowej Kohonena do stanu stabilnego. Dowody zbieżności sieci w przypadkach jednowymiarowych i wielowymiarowych można znaleźć w Rojas (1996). W dowodach tych przyjęte jest, że waga sieci dla kolejnych stanów zachowuje porządek monotoniczny.

Niech $p_{ij}(t)$ oznacza prototyp, czyli wybrany dokument z $m_{ij}(t)$ ($m_{ij}(t)[p_{ij}(t)] = t$). Przez $w_{ij}(t)$ oznaczmy wagę składnika m_{ij} zdefiniowaną jako suma kwadratów miar różnic pomiędzy prototypem a wszystkimi dokumentami zawartymi w m_{ij}.

$$w_{ij}(t) = \sum_{k=1..n} d^2(p_{ij}(t), k) m_{ij}(t)[k]$$

Przez wagę całej sieci rozumiemy

$$w(t) = \sum_{ij} w_{ij}(t)$$

Zgodnie z założeniami algorytmu w każdym kroku czasu t waga całej sieci $w(t)$ jest mniejsza lub równa wadze sieci w kroku $t-1$.

$$w(t) \leq w(t-1)$$

Jednocześnie $\forall_t w(t) > 0$

To spostrzeżenie pozwala sprowadzić problem do wymiaru rozważanego w Rojas (1996), co zapewnia jego zbieżność.

2.2.4 Uzupełnianie kategoryzacji

Zajmijmy się sytuacją, w której system musi uzupełnić zbiór wyników, w związku z dodaniem do repozytorium nowego dokumentu. W ogólności znów zajmiemy się jednym rodzajem miary różnic oznaczanym d. Załóżmy więc, że $T = \{t_1, ..., t_n\}$, $k < n$ i $T_k = \{t_1, ..., t_k\}$ jest zbiorem dokumentów podzielonych na kategorie. Przez $p(t_k)$ oznaczmy prototyp, do którego przydzielony był dokument t_k w ostatniej iteracji kategoryzacji. Załóżmy, że $P = \{p_1, ..., p_s\}$ i $\forall_{t \in T_k} \exists p \in P, p(t) = p$. Wiadomo również, że $|P| \leq \lfloor \sqrt[4]{n} \rfloor = m$.

Algorytm 10 ***Samoorganizująca się sieć Kohonena: poprawianie kategoryzacji***

1. *Tworzymy sieć neuronową o wielkości* $m \times m$.

2. *Dla pierwszych węzłów indeksowanych* $i = 1, ..., |P|$ *ustalamy prototypy na* $p_i = P(i)$. *Dla pozostałych* w_i *indeksowanych* $i = |P| + 1, ..., m^2$ *wybieramy z* T *losowy tekst* $w_r, w_r \notin P$, $(p_i = w_r)$.

3. *Dla każdego tekstu* t_i *wybieramy najbliższy mu (według zadanej miary różnic) prototyp* (p_c) *w sieci i dodajemy go do listy dokumentów tego prototypu* $(l(w_c) = l(w_c) \cup \{t_i\})$

4. *Dla każdego węzła* w_i *wyliczamy uogólnioną medianę dla wszystkich dokumentów z tego węzła i jego sąsiadów, a następnie zastępujemy nią prototyp tego węzła. Uogólniona mediana jest zdefiniowana jako element* $m \in T$ *minimalizujący funkcję:*

$$\sum_{t_s \in l_i \cup l(n(i))} d^2(t_s, m)$$

5. *Powtarzaj krok 3 i 4 do momentu ustabilizowania się sieci.*

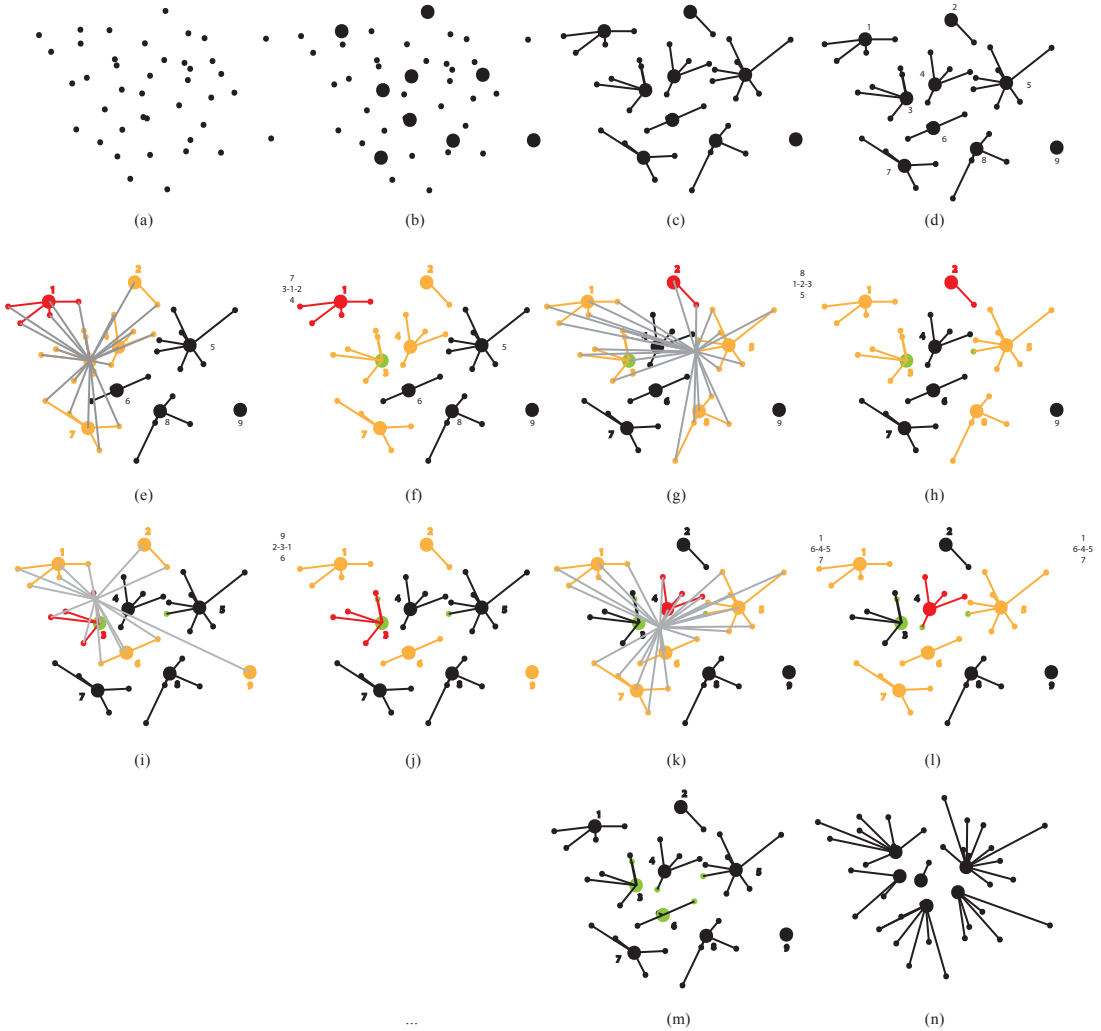
Wizualizację przebiegu algorytmu widać na obrazku 2.4.

Przedstawiona w tej pracy modyfikacja algorytmu umożliwia dynamiczną analizę nowych danych, przy założeniu ciągłego działania sieci. Należy tu również zwrócić uwagę na potrzebę dodawania dodatkowych węzłów do sieci, w przypadku, gdy sieć jest już skonstruowana i dodanie do analizy nowego dokumentu powoduje przekroczenie ustalonej ilości danych wejściowych (przypomnijmy: ilość węzłów ustalona została na m^2, gdzie $m = \lfloor \sqrt[4]{n} \rfloor$), tzn: $\lfloor \sqrt[4]{n} \rfloor > \left\lfloor \sqrt[4]{n-1} \right\rfloor$. Problem pojawia się w przypadku korzystania z sieci kwadratowej, gdyż w celu zachowania tej struktury dodanie jednego elementu wiązałoby się często z potrzebą dodania całego wiersza i kolumny nowych węzłów.

Należy tu odpowiedzieć na pytania

- czy wstawić dodatkowe węzły
- gdzie wstawić dodatkowe węzły.

Problem ten rozwiązuje rezygnacja ze sztywnej struktury sieci neuronowej.



Rysunek 2.4: Wizualizacja przebiegu algorytmu: (a) wszystkie dokumenty; (b) wybranie reprezentantów; (c) przypisanie dokumentów do kategorii; (d) przypisanie prototypów i dokumentów do węzłów (egik) wybranie 5 sąsiadujących węzłów i wyznaczenie mediany; (fhjlm) ustawienie mediany jako prototypu kategorii; (n) przypisanie wszystkich dokumentów do nowych prototypów.

2.3 Sieci dynamiczne

W ramach niniejszej rozprawy zaproponowana została modyfikacja sieci Kohonena poprzez uzupełnienie jej o możliwość dynamicznej zmiany struktury już podczas wykonywania algorytmu. Jednocześnie zaproponowany został inny sposób wybierania sąsiednich węzłów, bardziej zbliżony do naturalnych powiązań nerwowych. Rozwiązuje to problem miejsca dodawania dodatkowych węzłów sieci.

Podczas kategoryzacji danych za pomocą przedstawionego algorytmu zakłada się, że każdy z węzłów sieci posiada w każdej chwili ustalony prototyp. Założenie to implikuje często sytuację, w której część węzłów nie posiada żadnego elementu poza prototypem, a co za tym idzie prototyp ten tworzy sam w sobie kategorię, która nie jest zwykle odzwierciedleniem realnej odrębności dokumentu.

Rozwiązanie tego problemu przynosi idea sieci dynamicznych.

2.3.1 Sieci zmieniające swoją strukturę

W Jankowski (2003) opisane jest wiele modeli sieci neuronowych, których struktura modyfikowana jest w trakcie pracy. W poszczególnych modelach sieci struktura ta może być albo zmniejszana, albo powiększana. Zwykle dzieje się to na podstawie danych, które są aktualnie przetwarzane. Modyfikacje struktury również mogą przebiegać dwojako:

- poprzez usuwanie lub dodawanie konkretnych połączeń pomiędzy neuronami;
- poprzez usuwanie lub dodawanie konkretnych neuronów.

Powstaje więc podstawowe pytanie:

- co:
  - usuwać;
  - dodawać;
- czy / kiedy:
  - usuwać;
  - dodawać?

Zdefiniowanych jest kilka modeli ontogenicznych sieci neuronowych:

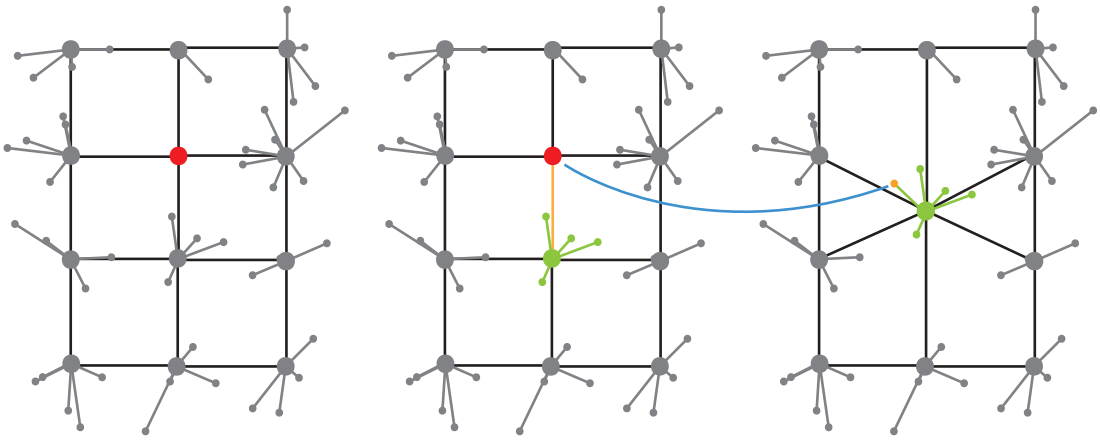
- powiększające - umożliwiające dodawanie nowych połączeń i neuronów,
- zmniejszające - umożliwiające usuwanie zbędnych neuronów i połączeń,
- powiększająco-zmniejszające - posiadające obie powyższe cechy,
- komitety modeli - czyli zbiory wielu modeli (z których każdy wyznacza jakiś sposób modyfikacji sieci) i kontroler (który decyduje którego z rozwiązań użyć).

Istnieje wiele algorytmów, które pomagają podjąć decyzję o modyfikacji struktury sieci neuronowej. Mogą to być:

- metody wyznaczania współczynników istotności (przydatności),
- metody regulacyjne (w tym rozpad wag, eliminacja wag i.t.d.) przedstawione między innymi przez Hintona w (Hinton (1987)),
- metody statystyczne (pokazane w Finnoff & Zimmermann (1994) i w Finnoff *et al.* (1993), metody rozrostu sieci analizowane w Albert & Barabási (2002)).

2.3.2 Przypadek kategoryzacji tekstów

W rozważanym w niniejszej pracy problemie, wystarczy zająć się ograniczeniem ilości neuronów. Dla konkretnego neuronu bądź połączenia obliczany jest jego współczynnik przydatności, reprezentujący jego wartość w dalszym działaniu sieci. Na podstawie wartości współczynnika zapada decyzja, co zrobić z dalszym elementem. Po obliczeniu tych współczynników dla wszystkich neuronów można wyznaczyć te, dla których współczynnik jest znacznie niższy niż dla innych i te neurony usunąć.



Rysunek 2.5: Wizualizacja modyfikacji sieci. Kolorem czerwonym oznaczono węzeł, który zostanie usunięty. Kolorem pomarańczowym oznaczono nowe miejsce prototypu usuniętego węzła.

W przypadku kategoryzacji tekstów każdy neuron odpowiada pewnej części zbioru danych - pewnemu obszarowi przestrzeni przyporządkowując elementy tej przestrzeni do określonej klasy. W niniejszej pracy zaproponowano rozwiązanie polegające na wyznaczeniu dla każdego neuronu współczynnika istotności s_i

opierającego się na informacjach o tym przyporządkowaniu:

$$s_i = \frac{|w_i|}{|T|},$$

gdzie $|w_i|$ oznacza ilość skategoryzowanych przez neuron w_i danych a $|T|$ moc zbioru danych uczących, czyli w tym przypadku ilość klasyfikowanych tekstów. Te neurony, dla których współczynnik jest najbliższy zeru mogą zostać usunięte z sieci.

Przedstawiona metoda jest metodą pasywną. Oznacza to, że sprawdzenia przydatności poszczególnych neuronów dokonuje się podczas przerwy działania sieci (w przedstawionym przypadku) lub nawet po zakończeniu jej działania.

2.3.3 Algorytm

Algorytm sieci neuronowej został więc zmodyfikowany tak, aby korzystał z informacji o obliczonym współczynniku istotności węzłów. W każdej iteracji po kroku 4 dodano dodatkowy krok:

Algorytm 11 ***Usuwanie zbędnych węzłów***

5 *Dla każdego węzła w_c oblicz współczynnik istotności $s_c = \frac{|l(w_c)|}{|T|}$. Usuń te węzły, które mają najmniejszy współczynnik s_c. Prototyp p_c dodaj jako dokument do jednego z pozostałych węzłów.*

Usprawnienie to wykorzystuje bardzo prosty model modyfikacji struktury sieci neuronowej, mimo to osiągane dzięki temu rezultaty są bardzo dobre. Przede wszystkim, w związku z mniejszą ilością kolejnych obliczeń, skrócony jest czas potrzebny na przeprowadzenie następnych kroków symulacji.

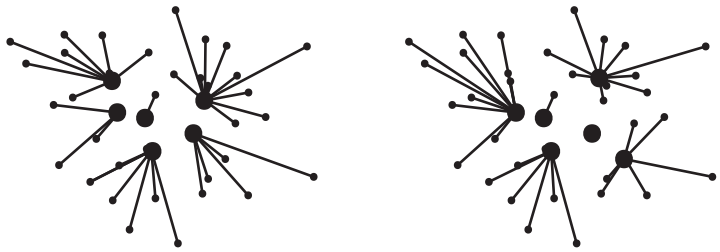
Jednocześnie funkcją sąsiedztwa węzłów jest funkcja miara różnic ich prototypów, tzn. sąsiadami węzła w_i są zawsze 4 węzły najbliższe mu w sensie miary różnic prototypów.

2.4 Algorytmy genetyczne a mediana

Kolejnym problemem, jaki pojawia się podczas kategoryzacji tekstów jest złożoność obliczeniowa potrzebna do wyznaczania mediany (dyskutowana w Ceselli & Righini (2005) i Charikar *et al.* (2002)) dla wszystkich dokumentów skategoryzowanych w poszczególnych neuronach. Nad algorytmami wyznaczającymi medianę mającymi lepszą złożoność obliczeniową pracowali, między innymi, Charikar, Guha, Tardos i Shmoys, którzy pokazali w Charikar *et al.* (2002) algorytm

aproksymacyjny obliczający k-medianę w czasie lepszym niż $0(log(k)log(log(k)))$. Znajdowaniem przybliżeń uogólnionej mediany zajmował się Cheng w Cheng (2008).

W ramach niniejszej pracy zauważono, że ilość obliczeń potrzebnych do przeprowadzenia kroku zamiany prototypu może zostać ograniczona. Okazało się, że do wykonania tego kroku nie jest potrzebna znajomość uogólnionej mediany. Zamiast tego wystarczy wyznaczyć element, który może z powodzeniem przybliżyć medianę ze standardowego algorytmu. Na rysunku 2.6 zilustrowane są podobieństwa wyniku pracy obu algorytmów.



(a) (b)

Rysunek 2.6: (a) Wynik działania algorytmów przybliżającego medianę i (b) obliczającego medianę.

Jak widać wynik pracy zaproponowanego w ramach tej rozprawy rozwiązania (rysunek 2.6), nie korzystającego z uogólnionej mediany, jest zbliżony do wyniku działania algorytmu obliczającego ją w każdym kroku. Wyznaczanie nowego prototypu modyfikowanej kategorii oparto o algorytm składający się z dwóch faz:

- fazy porządkującej sieć (wykorzystującej uproszczone algorytmy genetyczne);
- fazy mieszającej sieć (wykorzystującej przybliżone wyznaczanie mediany).

Zanim jednak algorytm zostanie opisany szczegółowo, należy wspomnieć o podstawach teorii algorytmów genetycznych będących szczególnym przypadkiem algorytmów ewolucyjnych.

2.4.1 O algorytmach ewolucyjnych

Algorytmy ewolucyjne są (zgodnie z informacjami przedstawionymi w Rojas (1996)) jednym ze sposobów aproksymacji najlepszego rozwiązania. Wywodzą się z obserwacji świata i sposobu rozmnażania się organizmów, oraz w szczególności z teorii doboru naturalnego Darwina, mówiącej o tym, że przeżywają organizmy silniejsze i lepiej przystosowane do środowiska. Rozwiązania kandydujące do miana

najlepszego rozwiązania tworzą populację. Środowisko jest zdefiniowane za pomocą specjalnej funkcji. Rozwiązania żyją i ewoluują wykorzystując reprodukcje, mutacje, rekombinacje i selekcje. Każde kolejne pokolenie rozwiązań prezentuje lepsze wyniki. Algorytmy te nadają się do rozwiązywania wszelkiego rodzaju problemów (Bäck (1996)), ponieważ nie są ograniczone żadnymi założeniami dotyczącymi rodzaju środowiska czy rodzajów rozwiązań.

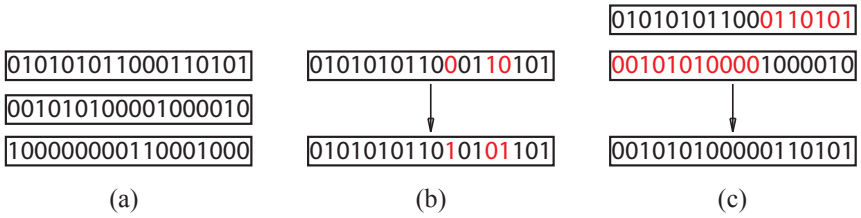
2.4.2 O algorytmach genetycznych

Algorytmy genetyczne są szczególnym przypadkiem algorytmów ewolucyjnych. Rozważane są bardzo często jako uzupełnienie uczenia się ze wzmocnieniem. Podstawowe elementy używane podczas projektowania algorytmów genetycznych to dobrze zdefiniowane środowisko, w którym żyje populacja elementów. Każdy element populacji posiada własny genotyp. Podstawowe operacje, które mogą być wykonane na elementach populacji to:

- **mutacja** - umożliwiająca zmianę genotypu w obrębie pojedynczego elementu,
- **krzyżowanie** - umożliwiające powstawanie nowych elementów potomnych za pomocą wyboru fragmentów genotypu z każdego z elementów macierzystych,
- **selekcja** - powodująca usuwanie elementów gorszych jakościowo, czyli nie przystosowanych do środowiska.

W podstawowej wersji genotyp jest ciągiem długości n składającym się z elementów zbioru $0, 1$. Modyfikacja problemu dopuszcza wykorzystanie jako elementów ciągu liczb z dowolnego zakresu. Daje to większe możliwości podczas projektowania opartych na tym pomyśle algorytmów.

Algorytmy genetyczne są projektowane tak, aby kolejne pokolenia elementów przybliżały coraz lepsze rozwiązanie zadanego problemu. Elementy macierzyste, które najgorzej przybliżają to rozwiązanie, zostają usunięte (giną). W większości podejść populacje rozrastają się do bardzo dużych rozmiarów.



Rysunek 2.7: (a) Elementy populacji w algorytmie genetycznym oraz podstawowe operacje na nich (b) mutacja, (c) krzyżowanie.

2.4.3 Obliczanie przybliżonej mediany

Przedstawiony w niniejszej pracy algorytm kategoryzacji zakłada, że w każdej iteracji wykonujemy następujące kroki:

Algorytm 12 ***Fragment algorytmu kategoryzującego***

3 Dla każdego tekstu t_i wybieramy najbliższy (według zadanej miary różnic) prototyp (p_c) w sieci i dodajemy do listy dokumentów tego prototypu ($l(w_c) = l(w_c) \cup \{t_i\}$)

4 Dla każdego węzła w_i wyliczamy uogólnioną medianę dla wszystkich dokumentów z tego węzła i jego sąsiadów, a następnie zastępujemy nią prototyp tego węzła. Uogólniona mediana jest zdefiniowana jako element $m \in T$ minimalizujący funkcję:

$$\sum_{t_s \in l_i \cup l(n(i))} d^2(t_s, m)$$

Znaczące usprawnienie tego algorytmu możliwe jest poprzez zastąpienia obliczenia mediany, wyznaczeniem jej przybliżenia oraz zastąpienia ponownego przydzielania dokumentów do prototypów poprzez modyfikację istniejących przydziałów. Innymi słowy, zamiast ponownie konstruować całą sieć, można dokonać jej modyfikacji, przenosząc środki ciężkości (prototypy) poszczególnych kategorii i przenosząc najbardziej odległych członków do bardziej pasujących kategorii.

Faza 1: Algorytm genetyczny

Przedstawione rozwiązanie zakłada, że do każdego z węzłów sieci przyporządkowany jest prototyp oraz lista dokumentów. Jednocześnie nie można założyć, że prototyp jest medianą dla tych dokumentów.

Algorytm 13 ***Algorytm genetyczny***

Dla każdego węzła w_c:

1. *Przez $l^s(w_c)$ oznacz listę dokumentów przydzielonych do węzła w_c, posortowaną w kolejności od najbliższego prototypowi węzła (ozn. p_c).*
2. *Testuj po kolei dokumenty z listy $l^s(w_c)$, aż znaleziony zostanie taki, który jest lepszym prototypem (ozn. $l^s(w_c)[k]$).*
3. *Ustaw $l^s(w_c)[k]$ jako prototyp w_c. Prototyp p_c dodaj do listy dokumentów $l^s(w_c) = l^s(w_c) \cup \{p_c\}$.*
4. *Testuj w kolejności od końca dokumenty z listy $l^s(w_c)$. Dla kolejnych dokumentów ($l^s(w_c)[i]$) sprawdź, czy przenieść go do innej kategorii.*

- *Jeśli istnieje taki węzeł w_l, że miara różnic pomiędzy jego prototypem a prototypem węzła w_c jest mniejsza niż podwojona miara różnic dokumentu $l^s(w_c)[i]$ od prototypu p_c i miara różnic tego dokumentu od prototypu p_l jest mniejsza niż jego miara różnic od prototypu p_c:*

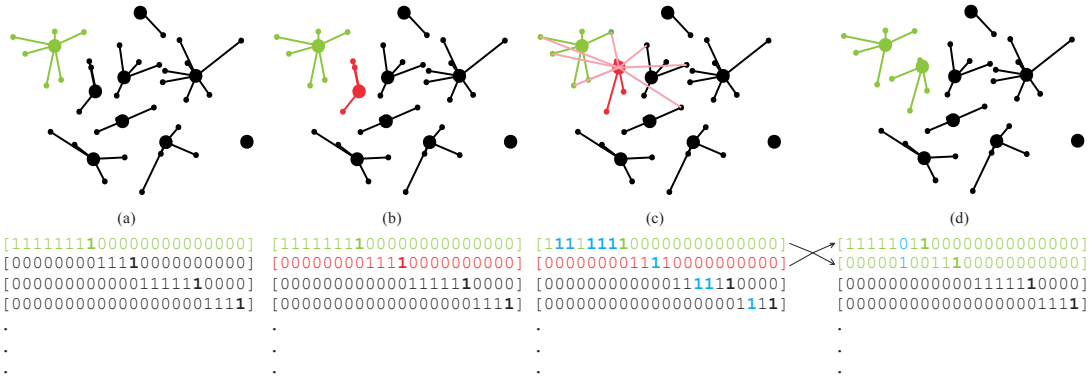
$$\exists_l 2 * d(l^s(w_c)[i], p_c) > d(p_c, p_l) \wedge d(l^s(w_c)[i], p_c) > d(l^s(w_c)[i], p_l),$$

wówczas przenieś dokument z listy $l^s(w_c)$ do listy $l^s(w_l)$:

$$l^s(w_c) = l^s(w_c) \setminus \{l^s(w_c)[i]\},$$
$$l^s(w_l) = l^s(w_l) \cup \{l^s(w_c)[i]\}.$$

Sortowanie elementów może się odbywać w czasie $O(nlogn)$. Sieci sortujące w takim czasie były prezentowane na przykład przez Ajtai *et al.* (1983).

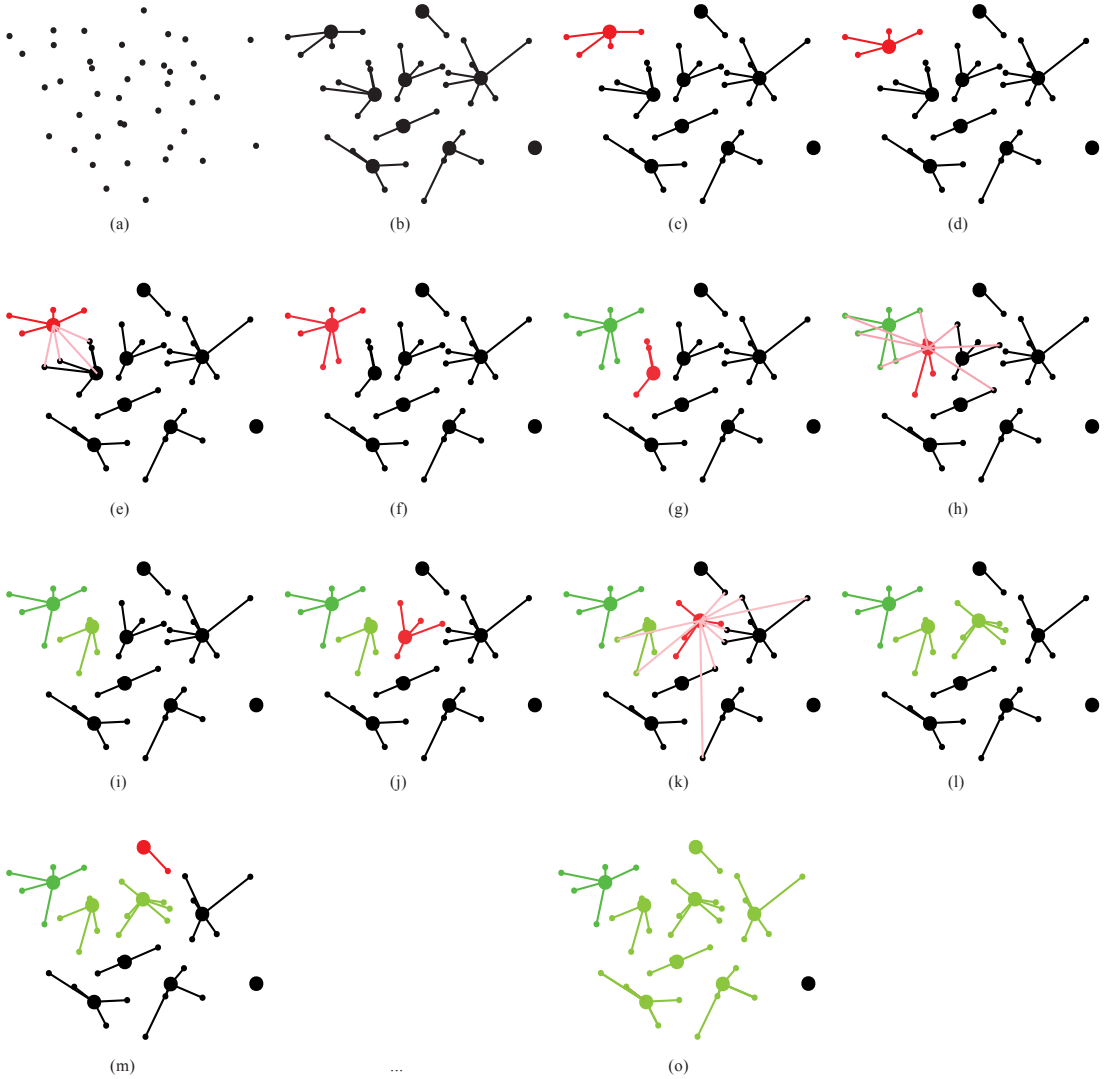
Zaproponowany algorytm można traktować jako algorytm genetyczny (2.8). Środowiskiem jest tutaj sieć. Populację tworzą wszystkie węzły. Genom każdego węzła reprezentuje wybór prototypu oraz listę dokumentów, które są do niego przydzielone. Operację poprawiania prototypu można traktować jako mutację. Operację przenoszenia elementów do sąsiednich węzłów można traktować jako krzyżowanie - w tym przypadku węzeł aktualny i sąsiedni są rodzicami natomiast dwa nowe węzły, powstałe poprzez wymianę fragmentów genomu są ich potomkami. Tutaj należy założyć **silną selekcję**, powodującą śmierć rodziców zaraz po wydaniu na świat potomstwa. Jest to specyficzna własność proponowanego rozwiązania.



Rysunek 2.8: Interpretacja algorytmu jako algorytmu genetycznego.

Faza 2: Algorytm aproksymacyjny

Druga faza zaproponowanego algorytmu przybliża medianę z sąsiednich węzłów. W przeciwieństwie do oryginalnego podejścia, jako sąsiadów traktujemy tutaj najbliższe pod względem miary różnic pomiędzy prototypami węzły.



Rysunek 2.9: Przebieg fazy pierwszej algorytmu: algorytm genetyczny.

Algorytm 14 ***Algorytm aproksymacyjny***

Dla każdego węzła w_c przez w_c^s oznacz listę węzłów posortowaną względem miary różnic do węzła w_c w sensie miary różnic prototypów ($d^*(w_c, w_l) = d(p_c, p_l)$). Jest to dobra miara różnic, zakładając, że każdy z prototypów jest w centrum elementów z danego węzła (gwarantuje to pierwsza faza algorytmu).

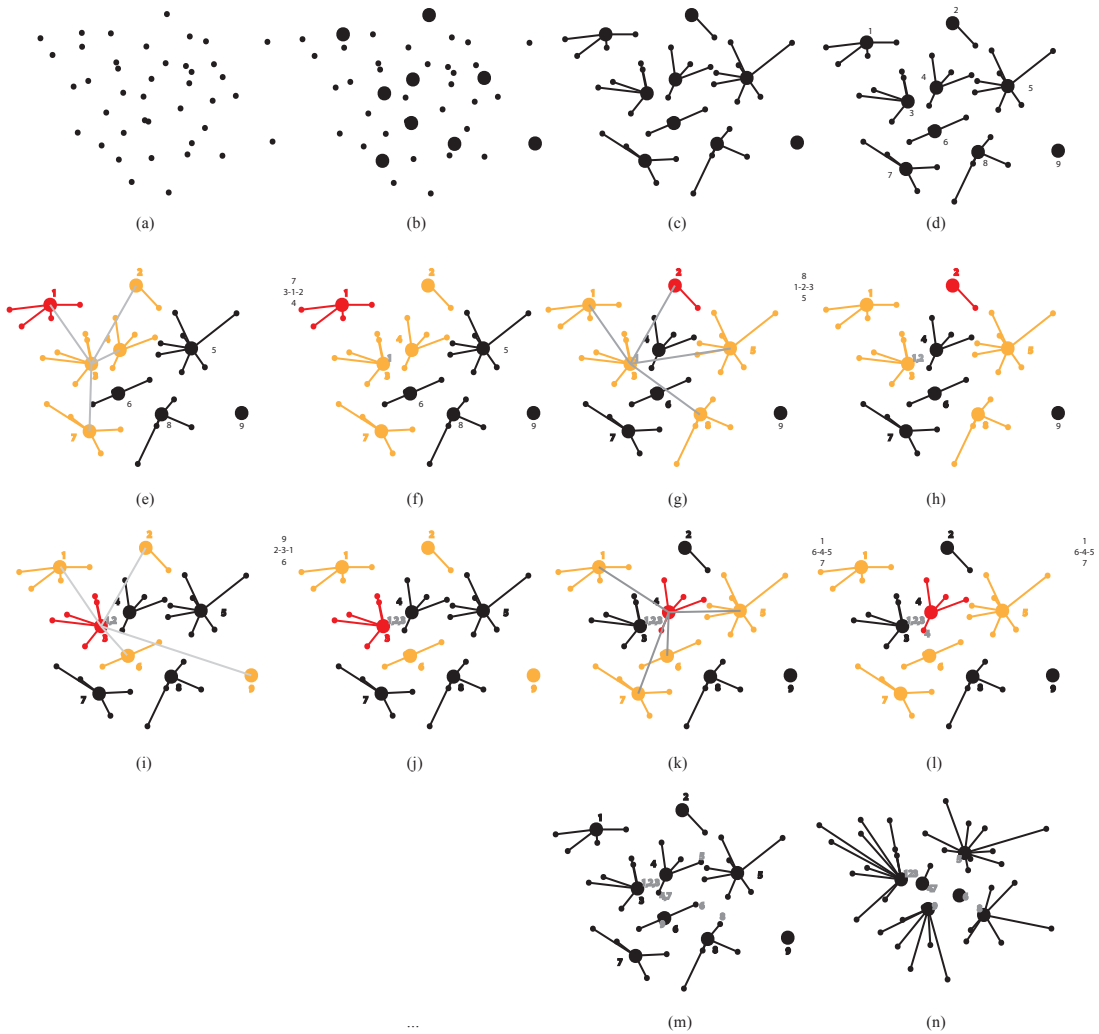
1. *Wśród węzłów $W_c = \{w_c, w_c^s[1], w_c^s[2], w_c^s[3], w_c^s[4]\}$ wyznacz medianę (ozn. w_c^m)w sensie miary różnic d^*.*

2. *Wśród dokumentów znajdujących się na liście węzła w_c^m ($l^s(w_c^m)$) znajdź element, który jako prototyp węzła w_c^m sprawi, że węzeł ten będzie lepszym*

kandydatem na medianę w sensie d^*:

- *Znajdź dokument $l^s(w_c^m)[k]$ taki, że $\sum_{w \in W_c \{w_c^m\}} d(p_w, l^s(w_c^m)[k])$ jest najmniejsza.*
- *Ustaw dokument jako nowy prototyp dla węzła w_c.*

3. *Popraw listy dokumentów przenosząc je do odpowiednich kategorii.*



Rysunek 2.10: Przebieg fazy drugiej algorytmu: algorytm aproksymacyjny.

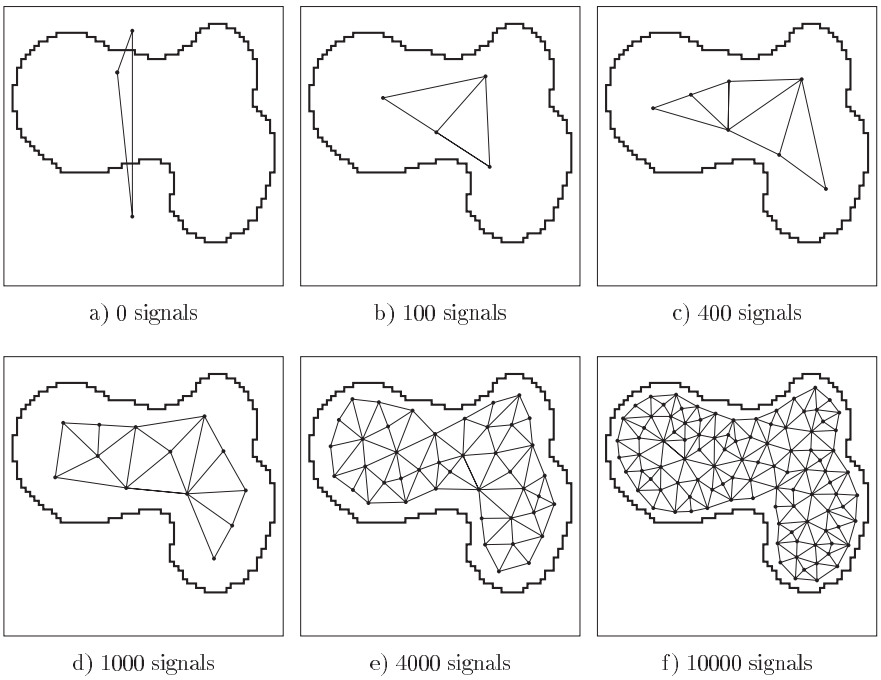
W zaproponowanym w ramach niniejszej pracy algorytmie, powiązania pomiędzy węzłami sieci zostały zastąpione poprzez rzeczywistą bliskość tych elementów. Zachowują się zatem bardziej jak prawdziwe neurony, przekazujące sygnały (w

tym przypadku dokumenty i informacje o nich) do neuronów sąsiednich. Modyfikacja ta przypomina algorytm Gazu Neuronowego zaproponowanego w Fritzke (1993).

Gaz neuronowy

W 1991 roku został zaproponowany algorytm organizacji neuronów, przypominający ruch cząsteczek gazu. Nazwany został Gazem Neuronowym (Neural Gas, NG,). Algorytm cechował swobodny przepływ neuronów, które mogą poruszać się w określonej przestrzeni i z określoną swobodą. Ilość neuronów była dobierana losowa, a kolejne kroki algorytmu polegały na podaniu na wejście wektora cech, sortowania neuronów na podstawie ich odległości od wektora x w porządku rosnącym, a następnie iteracyjną modyfikację ich wag.

Modyfikację tego algorytmu, nazwaną Rosnącym Gazem Neuronowym (Growing Neural Gas, GNG) zaproponowano w Fritzke (1993). Algorytm ten pozwalał na konstrukcję sieci neuronowej (zbioru neuronów) z dowolnymi wagami, kóra później w kolejnych iteracjach algorytmu była poszerzana o kolejne neurony (2.11), tak długo, dopóki nie została osiągnięta maksymalna wielkość sieci, albo inne kryterium zakończenia algorytmu.



Rysunek 2.11: Wizualizacja przebiegu algorytmu GNG, ilustracja pochodzi z Fritzke (1993)

.

Algorytm pozwalał na modyfikownie struktury sieci. Podczas dodawania nowych neuronów, tworzono nowe połączenia pomiędzy odpowiednio dobranymi neuronami, bądź usuwano połączenia już istniejące. W szczególnych przypadkach możliwe było również usunięcie wybranych neuronów z sieci.

Wyniki działania algorytmu

W przeciwieństwie do algorytmów przedstawionych przez Fritzke, nie ma tutaj ustalonej struktury połączeń pomiędzy neuronami - komunikacja między neuronami przebiega na podstawie ich położenia względem siebie, ustalanego w trakcie trwania algorytmu.

Kolejna różnica dotyczy samego poruszania się algorytmu neuronów. Podczas gdy w w/w algorytmie położenie i wagi neuronów są modyfikowane na podstawie kolejnych danych uczących, z uwzględnieniem malejącego współczynnika uczenia się, tak tutaj położenie neuronów jest ściśle związane z położeniem danych w przestrzeni i polega na zmianie agregacji znanych danych.

Przeprowadzone testy pokazują, że tak zmodyfikowany algorytm pozwala na osiągnięcie podobnych wyników w dużo lepszym czasie, mimo tego, że jego pesymistyczna złożoność obliczeniowa jest taka sama jak algorytmu wyjściowego.

2.4.4 Zbieżność sieci dynamicznej

Model matematyczny przedstawionej sieci dynamicznej można traktować identycznie jak model sieci standardowej, z zastrzeżeniem, że węzły połączone reprezentowane są przez jeden z nich. Takie założenie pozwala skorzystać z cech zbieżności sieci standardowej. Podobnie jak w przypadku standardowej sieci Kohonena, możemy tutaj zauważyć fakt, że w każdym kroku czasu t

$$w(t) \leq w(t-1)$$

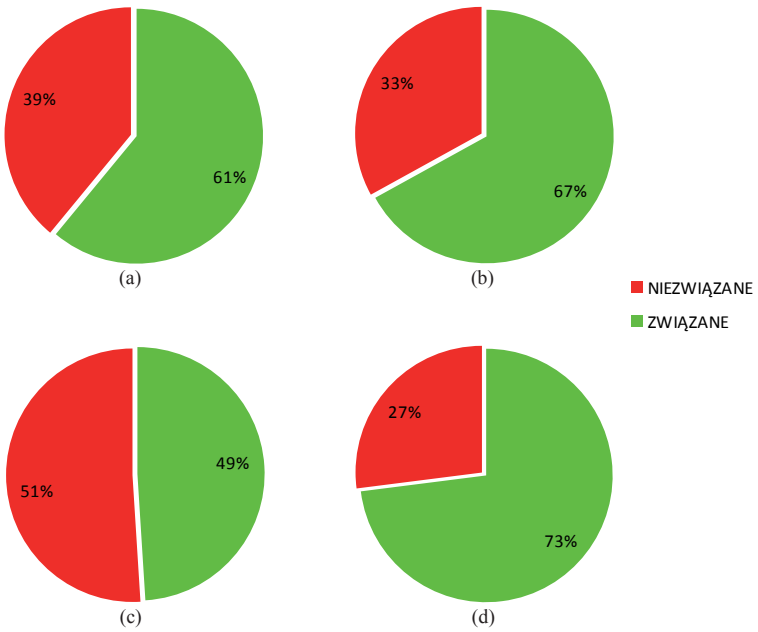
$$w(t) > 0$$

Dlatego sieć ta jest również zbieżna, a jej waga zbiega do wagi granicznej sieci.

2.4.5 Porównanie efektywności algorytmów

Pierwotny algorytm wykorzystujący samoorganizujące się sieci neuronowe Kohonena (SOM) jest mniej efektywny od modyfikacji wykorzystującej algorytmy genetyczne i przybliżanie mediany (GA). Zostało wykonane porównanie wydajności obu algorytmów na przykładowych zbiorach danych (o różnych rozmiarach) i wykazałao, że algorytm GA jest w większości przypadków znacząco szybszy od SOM, dając przy tym podobne wyniki.

2.5 Wyniki kategoryzacji



Rysunek 2.12: Skuteczność kategoryzacji względem (a) d_s, (b) d_n, (c) d_k, (d) d_s,d_n i d_k. Kolorem czerwonym oznaczono średnią ilość dokumentów w tej samej kategorii niezwiązanych z testowanym dokumentem. Kolorem zielonym oznaczono średnią ilość dokumentów w tej samej kategorii związanych z testowanym dokumentem. Wykorzystując wszystkie wyznaczone miary różnic (d) osiągnięto lepszą klasteryzację dokumentów - powstało więcej kategorii, ale każda zawierała bardziej spójny tematycznie zestaw dokumentów

Na testowej próbie około $n = 1000$ losowych dokumentów dokonane zostało obliczenie funkcji miary różnic (w sensie d_s, d_n i d_k). Następnie zostały przeprowadzone trzy kategoryzacje, w wyniku których dokumenty zostały podzielone na kategorie względem każdej z tych miar.

Wyniki, to znaczy sensowność i trafność podziału sprawdzono empirycznie, przeglądając każdą z kategorii i zawarte w niej dokumenty, a następnie oceniając, czy dokument jest związany tematycznie z pozostałymi, czy nie.

- $T = \{t_1, t_2, ..., t_i\}$ - zbiór dokumentów
- $\forall_{t_i,t_j \in T}$ obliczono $d_s(t_i, t_j), d_n(t_i, t_j), d_k(t_i, t_j)$
- Przez $K_d(t)$ oznaczono kategorię dokumentu $t \in T$ w sensie $d \in \{d_s, d_n, d_k\}$. Przez $T(k)$ oznaczono wszystkie dokumenty przyporządkowane do kategorii k.

- Dla każdego $d \in \{d_s, d_n, d_k\}$ dla dokumentów $t_r \in T$, sprawdzono empirycznie, czy $t \in T(K_d(t_r))$ są powiązane z tekstem t_r.

Okazuje się, że każdy typ miar różnic dał nieco inne wyniki, jednak żaden z nich nie okazał się ponadprzeciętnie efektywny. Sprawdzeniu została poddana pewna próba losowa dokumentów.

Wyniki kategoryzacji otrzymanych dla każdej z miar różnic są zgodne z wynikami porównań dokumentów przedstawionymi na wykresie 1.12. Potwierdza to przedstawiony poziom ich skuteczności.

Metoda częstości słów. Ta metoda przyniosła zdecydowanie najmniej wyraźne wyniki. Spośród wszystkich wybranych dokumentów wśród dokumentów będących w tej samej kategorii, średnio 49% było związanych z dokumentem testowym. Znaczący wpływ na wynik miało usunięcie słów nieistotnych (2.12 (a)).

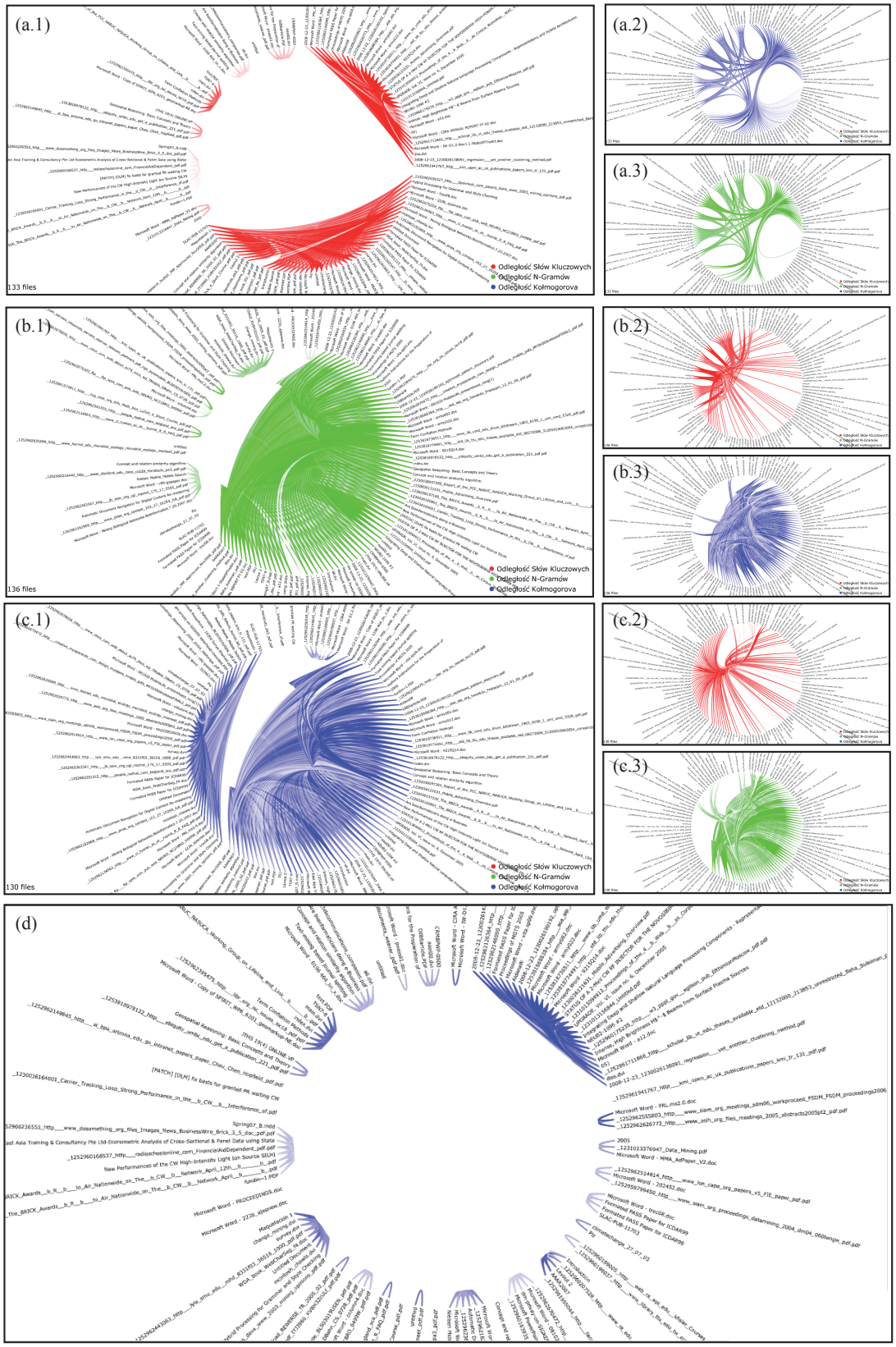
Metoda częstości n-gramów. Ta metoda przyniosła wyniki trochę lepsze. Spośród wszystkich wybranych dokumentów wśród dokumentów będących w tej samej kategorii, średnio 61% było związanych z dokumentem testowym (2.12 (b)).

Złożoność Kołmogorowa. Zgodnie z wykonanymi w Frank *et al.* (2000) badaniami kategoryzacja tekstów z użyciem modeli kompresji, można rozróżnić poezję od prozy czy kodu źródłowego, jednak ucząca się sieć neuronowa z przykładów podanych w przytoczonej pracy jest siecią nadzorowaną i na początku jest trenowana za pomocą danych trenujących.

W rozważanej w niniejszej pracy metodzie sieć uczy się bez nadzoru, mimo to ta metoda przyniosła najwyraźniejsze wyniki. Spośród wszystkich wybranych dokumentów wśród dokumentów będących w tej samej kategorii średnio, 67% było związanych z dokumentem testowym (2.12 (c)).

2.5.1 Porównanie wyników

Na obrazie 2.13 widać wszystkie dokumenty z testowanego zbioru, które zostały porównane za pomocą algorytmu. Kolorami zaznaczono siłę powiązań pomiędzy poszczególnymi dokumentami (im ciemniejszy, tym dokumenty są bliżej siebie, tzn są bardziej podobne). Kolor czerwony odpowiada miarom różnic wyznaczonym na podstawie analizy słów, kolor zielony miarom różnic wyznaczonym na podstawie badania n-gramów, a kolor niebieski na podstawie badania miar różnic Kołmogorowa. Dla każdej z miar różnic (a - d_s, b - d_n, c - d_k) pokazano w jaki sposób wygląda kategoryzacja dla niej (1) oraz jak wyglądają pozostałe miary różnic (2,3) w ujęciu tej kategoryzacji.



Rysunek 2.13: Wyniki kategoryzacji względem wszystkich miar różnic z zaznaczonymi miarami różnic. Im ciemniejszy kolor tym bliższe dokumenty.

2.5.2 Scalanie wyników

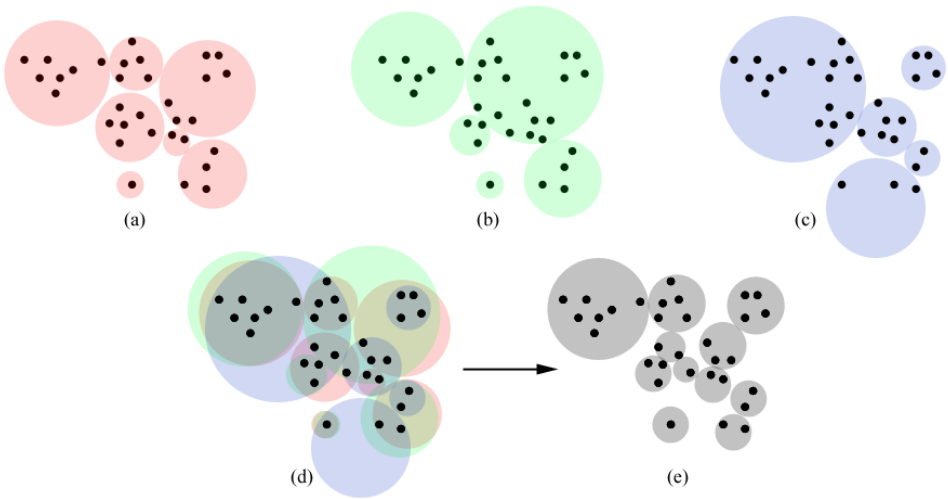
Analiza wyników pokazuje, że niektóre zestawy dokumentów powtarzają się w kategoriach wyznaczonych względem różnych miar różnic. To spostrzeżenie razem z uwagą, że każdy z typów miar różnic zwraca uwagę na inne cechy dokumentów zostało potraktowane jako podstawa do modyfikacji procesu kategoryzacji tak, aby uwzględniał wszystkie trzy typy miar różnic. Kategorie zostały więc zawężone, a ich liczba jednocześnie wzrosła w wyniku uwzględnienia ich części wspólnej (2.14). Oznacza to tyle, że do jednej kategorii z dokumentem t (w sensie miary różnic d) trafią tylko takie dokumenty t', które są razem z nim w kategoriach wyznaczonych względem pozostałych miar różnic.

- Przez $K_d(t)$ oznaczono kategorię dokumentu $t \in T$ w sensie d z poprzedniego rozdziału. Przez $T(k)$ oznaczono wszystkie dokumenty przyporządkowane do kategorii k.

- Dokument $t' \in T$ należy do kategorii $K_d(t)$ $(t' \in T(K_d(t)))$ wtedy i tylko wtedy gdy $t' \in T(K_{d_n}(t)) \wedge t' \in T(K_{d_k}(t)) \wedge t' \in T(K_{d_s}(t)) \wedge$.

- Dokumenty $t' \in T(K_d(t))$ są dokumentami podobnymi do t.

Okazuje się, że średnio 74% dokumentów w $T(K_d(t))$ jest treściowo i konstrukcyjnie związanych z dokumentem t.



Rysunek 2.14: Schemat ograniczenia wyboru przez wymuszenie tych samych kategorii w sensie (a) d_s , (b) d_n i (c) d_k. Kategorie przecinając się (d) tworzą nowe, mniejsze kategorie (części wspólne) (e).

Na rysunku 2.13 (d) widać, że po połączeniu wszystkich wyników znacząco zwiększyła się liczba kategorii, co implikuje bardziej szczegółowy dobór podobnych dokumentów. Zawężenie kategorii powoduje jednak stratę części informacji, gdyż niektóre z dobrze dopasowanych dokumentów mogą znaleźć się poza finalną kategorią. Jednak poprawa jakości pozostałych wyników skutecznie rekompensuje tę stratę.

Przeprowadzone badania pokazały, że prawdziwa była teza zakładająca wykorzystanie kilku typów miar różnic do wyznaczenia powiązań między dokumentami. Zweryfikowana została również ich efektywność. Ten zaproponowany w niniejszej pracy sposób porównywania dokumentów może być wykorzystany do pomocy w wyznaczeniu ich słów kluczowych.

2.5.3 Testy scalania wyników

Testy scalania wyników zostały przeprowadzone na kilku różnych zbiorach dokumentów. Poniżej przedstawione zostały wyniki jednego z nich, przeprowadzonego na zbiorze 520 dokumentów naukowych z różnych dziedzin[1]. W tabelach zaprezentowane są wyniki wyznaczania dokumentów podobnych do wybranego inicjalizującego wyszukiwanie (*A mutually beneficial integration of data mining and information extraction*), ze względu na miry różnic (a) d_s , (b) d_n i (c) d_k.

Id	Dokument	Odległosć
1	A mutually beneficial integration of data mining and information extraction	0.0
2	Web 2.0 Principals	0.06
3	First Beam Tests of the TTF Injector	0.09
4	Determine similarities in groups of data	0.09
5	11th Annual Data Mining Conference Casesars	0.10
6	Text Mining based journal Splitting	0.10
7	Fuzzy Segmentation of Characters in Web Images Based on Human Colour Perception	0.11
8	Combining Text Mining and Machine Scheduling	0.11
9	Surflet Pair Relation Histograms A statistical 3D	0.12
10	An architecture for interactive musical	0.12

Ciąg dalszy na następnej stronie

[1]Dokumenty zostały wyszukane automatycznie za pomocą wyszukiwarki Google. Otrzymany zbiór około 1000 dokumentów został zweryfikowany empirycznie, po odrzuceniu dokumentów nie noszących znamion dokumentów naukowych, w zbiorze zostało 520 tekstów.

Id	Dokument	Odległosć
11	Planning Sucessful Data Mining Projects	0.12

Tablica 2.1: Dokumenty bliskie inicjalizującemu, względem miary różnic d_k

Id	Dokument	Odległosć
1	A mutually beneficial integration of data mining and information extraction	0.0
2	Text mining for Documents Annotation	0.08
3	11th Annual Data Mining Conference Caesars	0.08
4	Regression - yet another clustering method	0.15
5	Text mining based journal splitting	0.18
6	Histograms A statistical 3D	0.18
7	Construction oft the TELSA	0.18
8	Test Facility Injector	0.18
9	Breakthrough tools for intelligent data analysis	0.18
10	Surflet Pair Relation Histograms A statistical 3D	0.18
11	Breakthrough tools for intelligent data analysis	0.18
12	Text Extraction from Web Images Based on Human Perception and Fuzzy Inference	0.19

Tablica 2.2: Dokumenty bliskie inicjalizującemu, względem miary różnic d_n

Id	Dokument	Odległosć
1	A mutually beneficial integration of data mining and information extraction	0.0
2	Breakthrough tools for intelligent data analysis	0.14
3	Web 2.0 Principals	0.15
4	CWI Research 2006	0.15
5	First Beam Tests of the TTF Injector	0.15
6	Fuzzy Segmentation of Characters in Web Images Based on Human Colour Perception	0.15
7	Optimised Pattern Discovery	0.15

Ciąg dalszy na następnej stronie

Id	Dokument	Odległosć
8	11th Annual Data Mining Conference Casesars	0.15
9	Regression - yet another clustering method	0.15
10	Text Mining for documents annotation	0.16
11	Combining Text Mining and Machine Scheduling	0.16

Tablica 2.3: Dokumenty bliskie inicjalizującemu, względem miary różnic d_s

Id	Dokument	k_s	k_n	k_k
1	**A mutually beneficial integration of data mining and information extraction**	4	0	0
2	**Breakthrough tools for intelligent data analysis**	4	0	0
3	Microsoft Pragmathic path to the semantic web	4	1	2
4	**CWI Research 2006**	4	0	0
5	Public Service Comission	1	1	0
6	**Web 2.0**	4	0	0
7	Brick Awards	1	2	0
8	**First Beam Tests of the TTF**	4	0	0
9	Fuzzy Segmentation of Characters in Web Images Based on Human Colour Perception	4	1	1
10	**Optimised Pattern Discovery**	4	0	0

Tablica 2.4: Dokumenty z uwzględnionymi numerami kategorii (k_s, k_n, k_k), do których zostały przyporządkowane.

Jak pokazują wyniki, filtrowanie listy rezultatów ze względu na kategorie przydzielone dokumentom, ze względu na trzy analizowane typy miar różnic pozwalają na zwiększenie dopasowania tematycznego dokumentów. Mimo widocznego obniżenia miejsca na liście niektórych dokumentów, w rezultacie otrzymujemy listę, która w większości spełnia założone wymagania tematyczne.

2.5.4 Inne metody analizy skupień

Zaproponowana w niniejszej pracy metoda jest metodą z rodziny „*metod k-średnich*", które rozpoczynają od wstępnego podziału elementów na z góry założoną liczbę

kategorii a następnie dążą do poprawienia tych kategorii (MacQueen (1967)). Ogólnie metody analizy skupień dzielą się na:

- metody hierarchiczne,
  - aglomeracyjne, których idea polega na łączeniu początkowo odrębnych elementów w coraz to większe kategorie
  - deaglomeracyjne, których idea polega na podziale jednej kategorii na mniejsze z każdym krokiem kategorie
- metody k-średnich,
- metody analizy rozmytej, które pozwalają na przydział elementów do więcej niż jednej kategorii.

Poniżej przedstawiam wyniki porównania metody analizy skupień zaproponowanej w niniejszej pracy z popularną metodą TFIDF (Term Frequency Inverce Document Frequency), odległością cosinusową.

TFIDF

TFIDF jest metodą wyznaczania wag słów na podstawie częstości ich wystąpień w danym dokumencie (d) i we wszystkich dokumentach (ich zbiór oznaczmy D) w repozytorium. Dla dokumentu d wagę słowa i wyznaczyć można za pomocą wzoru

$$tfidf_d[i] = tf_d(i) \cdot idf(i)$$

gdzie częstość termu i oznaczana $tf_d(i)$ to

$$tf_d(i) = \frac{n_d(i)}{\sum_k n_d(k)}$$

gdzie $n_d(k)$ oznacza ilość wystąpień termu k w dokumencie d.

Dla termu i zdefiniowana jest Odwrotna Częstość Dokumentów oznaczana idf_i

$$idf_i = \frac{\|D\|}{\|\{d, i \in d\}\|}$$

Dla każdych dwóch dokumendów (d, c) definiujemy odległość cosinusową w następujący sposób:

$$dist(c, d) = 1 - cos(tfidf_c, tfidf_d)$$

gdzie $tfidf_c$ oznacza tablicę wag wszystkich termów występujących w dokumentach c i d, a

$$cos(x, y) = \frac{x \cdot y}{(\|x\| \, \|y\|)} x \cdot y = \sum_i x[i]y[i] \, \|x\| = \sqrt{x \cdot x}$$

Porównanie wyników algorytmów

Oba algorytmy zostały przetestowane na zbiorze dokumentów składającym się z 500 elementów związanych z tematyką data mining, dokumentów naukowych związanych z badaniami chorób serca oraz dokumentów naukowych związanych z fizyką i matematyką. Podzbiory te były mniej więcej równoliczne.

Okazało się, że wyniki były bardzo podobne i wśród pierwszych dziesięciu dokumentów wszystkie związane były z dokumentem testowym. Należy tu jednak zwrócić uwagę, że algorytmy mimo analogicznej implementacji, miały różne czasy wykonywania. Algorytm prezentowany w niniejszej pracy wygenerował wyniki kilkakrotnie szybciej niż algorytm TFIDF.

Id	Dokument
1	**A mutually beneficial integration of data mining and information extraction**
2	**Text mining for document annotations and ontology support**
4	**Data Mining for Technical Operation of Telecommunications Companies**
6	**Text Mining for Business Inteligence**
8	**Introduction to data mining**
10	**Using Data Mining methodology for text retrieval**
9	**Towards a Text Mining Methodology Using Frequent Itemsets and Association Rule Extraction**
3	**Knowledge discovery in the Internet**
5	**Discovery of Important Keywords in the Cyberspace**
7	**Text Mining applied to multilingual corpora**

Tablica 2.5: 10 pierwszych dokumentów najbardziej podobnych do przykładowego (*A mutually beneficial integration of data mining and information extraction*, z zaznaczonymi dokumentami ocenionymi empirycznie jako podobne. Testowanie wykonane za pomocą prezentowanych w niniejszej pracy algorytmów)

Id	Dokument

Ciąg dalszy na następnej stronie

Id	Dokument
1	**A mutually beneficial integration of data mining and information extraction**
2	**Text mining for document annotations and ontology support**
3	**Data Mining for Technical Operation of Telecommunications Companies**
4	**Text Mining for Business Inteligence**
5	**Introduction to data mining**
6	**Towards a Text Mining Methodology Using Frequent Itemsets and Association Rule Extraction**
7	**Knowledge discovery in the Internet**
8	**Discovery of Important Keywords in the Cyberspace**
9	**Text Mining applied to multilingual corpora**
10	**Visualization of Navigation Patterns on a Web Site Using Model Based Clustering**

Tablica 2.6: 10 pierwsszych dokumentów najbardziej podobnych do przykładowego (*A mutually beneficial integration of data mining and information extraction*, z zaznaczonymi dokumentami ocenionymi empirycznie jako podobne. Porównywanie wykonane algorytmu TFIDF i odległości cosinusowej)

Rozdział 3

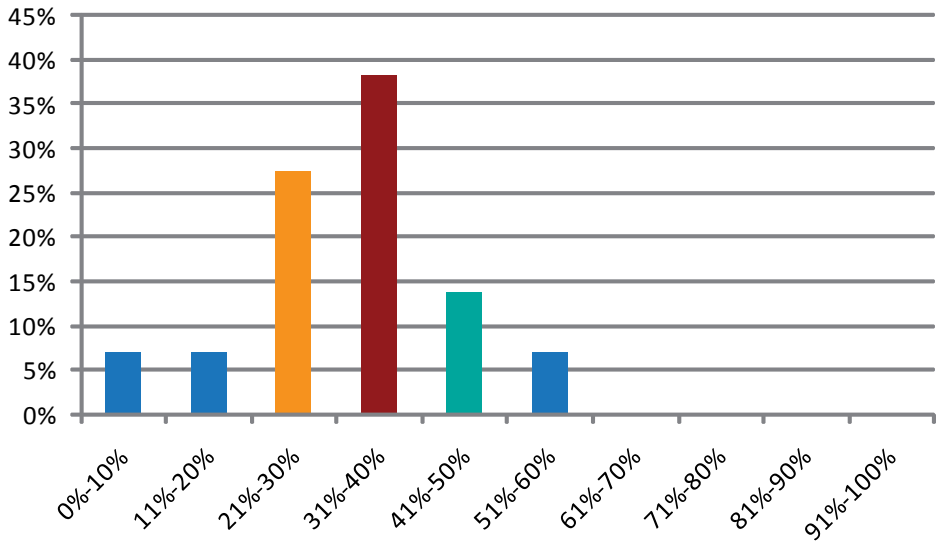
Wyznaczanie słów kluczowych

W tym rozdziale podjęto dyskusję na temat znanych sposobów wyznaczania słów kluczowych. Przedstawiono tutaj zaprojektowany w ramach niniejszej pracy nowy sposób wyznaczania słów kluczowych, wykorzystujący kategoryzację słów w dokumencie, opartą o sieci neuronowe Kohonena. Sieci te zostały wzmocnione poprzez sprawdzanie częściowych wyników kategoryzacji, w oparciu o dane zgromadzone podczas porównywania dokumentów. Dodatkowo został zaproponowany algorytm wyznaczający wstępny ranking słów na podstawie ich położenia w dokumencie. Przedstawione wyniki składają się na nową, oryginalną metodę wyznaczania słów kluczowych w dokumentach naukowych.

3.1 Wprowadzenie

Jak wskazują wyniki z 1.2 wyznaczenie właściwych słów kluczowych w dokumencie nie jest prostym zadaniem. Wyniki osiągnięte za pomocą algorytmu zliczania słów a następnie usuwania z nich słów uznanych za zbędne nie są imponujące (układają się na poziomie skuteczności niższym niż 60%). Wykres 3.1 pokazuje, że w większości przypadków jest to dużo mniej. Podczas testów przeanalizowanych zostało kilkaset dokumentów z różnych dziedzin a następnie sprawdzona została skuteczność metody. Kryterium tej skuteczności było subiektywnym kryterium sprawdzających. Takie podejście ma jeszcze jedną zasadniczą wadę: nie zwraca uwagi na kontekst słów, które wybiera. Co więcej zwykle okazuje się, że (nawet odfiltrowane) najczęściej występujące słowa nie są tymi kluczowymi.

W ramach niniejszej pracy zaproponowano nowe podejście do problemu wyboru słów kluczowych. Wykorzystując sieci neuronowe Kohonena pogrupowano słowa w dokumencie w lokalnie bliskie zbiory. Główną ideą zaprezentowanego algorytmu jest wykorzystanie wyników uzyskanych w rozdziale 2. Opiera się ona na spostrzeżeniu, że wiedza zdobyta przez człowieka w trakcie nauki poprawia



Rysunek 3.1: Wyniki statystycznego wybierania słów kluczowych. Oś X pokazuje skuteczność wyboru słów kluczowych (tzn. względną ilość poprawnie zaproponowanych słów kluczowych), oś Y pokazuje ilość dokumentów, dla których taka skuteczność została osiągnięta.

jego zdolności poznawcze i interpretacyjne i ułatwia dalsze przyswajanie. Zaproponowany algorytm korzysta z wiedzy zgromadzonej podczas analizy innych dokumentów i ich porównywania, poprawiając w ten sposób znacznie jakość wyboru słów kluczowych.

3.2 Idea wyznaczania słów kluczowych

Zaproponowane w niniejszej pracy podejście do wyznaczania słów kluczowych dokumentu wymaga kilku założeń:

- dokument musi być powiązany funkcją miary różnic (w sensie statystyki słów (d_s), statystyki n-gramów (d_n) i złożoności Kołmogorowa (d_k)) z pozostałymi (a przynajmniej częścią) dokumentami z analizowanego zbioru dokumentów;
- dokument powinien być przydzielony do pewnych kategorii wyznaczonych ze względu na te miary różnic .

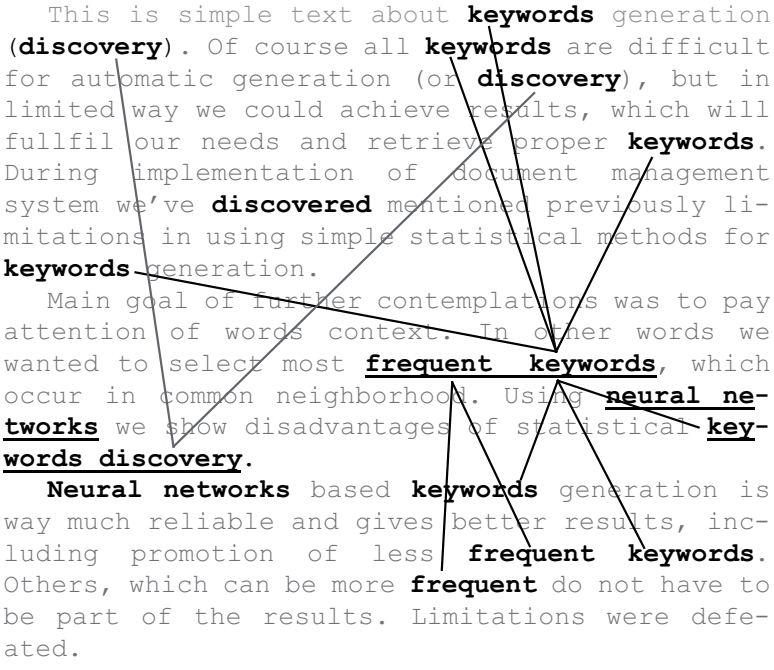
Prezentowany algorytm wyznacza słowa najczęściej występujące w tekście, zwracając uwagę na to, w jakim sąsiedztwie się one znajdują. Sąsiedztwo to ma wpływ

na końcową ocenę przydatności wybranych słów. Dla każdej grupy lokalnie bliskich (znajdujących się w pobliżu siebie, znajdujących się w jednej wynikowej kategorii) wyznacza się jej współczynnik istotności na podstawie którego następnie jest wyznaczana pozycja rankingowa. Na podstawie tego rankingu i częstości występowania słów wyznacza się końcową listę wynikową zawierającą większość słów kluczowych.

Dodatkowo na końcową ocenę każdego słowa mają wpływ miary różnic dokumentu analizowanego do innych dokumentów z repozytorium zawierających to słowo. Przykładowe słowa kluczowe wraz z zaznaczonym położeniem w innych częściach dokumentu pokazane są na ilustracji 3.2. W tabeli 3.1 widać przykładowy wybór słów kluczowych (zaprezentowanych jako para: słowo i ilość jego wystąpień) pogrupowanych w kategorie z obliczonym współczynnikiem istotności.

Tablica 3.1: Przykładowy wybór słów kluczowych wraz z kategoriami, w których się znajdują i rankingami kategorii.

Kategoria	Współczynnik istotności	Słowo	Ilość wystąpień
1	0,94	keywords	8
		discovery	3
		frequent	3
		simple	2
		text	1
		automatic	1
2	0,58	neural	2
		networks	2
		discovered	1
		neighborhood	1
3	0,50	fulfill	1
4	0,41	statistical	2
		words	2
		contemplations	1
		promotion	1
		disadvantages	1
5	0,36	retrieve	1
		reliable	1



Rysunek 3.2: Przykładowy wybór słów kluczowych w dokumencie oraz wskazanie w jakich miejscach dokumentu występują w większych odstępach.

3.3 Model odległości słów w dokumencie

Na początku należy zająć się modelem odległości słów w dokumencie zaproponowanym w ramach niniejszej pracy. Podstawową koncepcją jest zliczenie ilości słów znajdujących się w tekście pomiędzy rozpatrywanymi słowami x i y, albo ilości znaków pomocniczych (białych, przestankowych) znajdujących się pomiędzy tymi słowami. Należy tu zwrócić uwagę, że nie wszystkie znaki pomocnicze powinny być traktowane jednakowo. Wynika to z faktu, że teksty mają strukturę komunikatów. Każdy komunikat to zdanie lub równoważnik zdania. Zdania takie zazwyczaj oddzielone są kropką. Oznacza to, że słowa występujące w kontekście „*słowo przykładowe*" są sobie bliskie, natomiast „*słowo. Przykładowe*" już nie. W ramach niniejszej pracy zaproponowano wprowadzenie dodatkowych wag ($v(c)$) przyporządkowanych znakom pomocniczym c (tabela 3.2).

Na podstawie powyższych założeń zdefiniowano odległość słów w dokumencie. Ponieważ zgodnie z nią słowo x występujące w tekście T na pozycji i jest innym słowem niż słowo występujące w tekście T na pozycji j, $j \neq i$. Należy zatem wprowadzić pojęcie klasy abstrakcji słowa.

Definicja 15 *Niech $|c|_x^y$ oznacza ilość znaków c pomiędzy napisami x i y. Przez P oznaczmy zbiór znaków pomocniczych. Odległość między napisami x i y doku-*

Tablica 3.2: Wagi znaków pomocniczych.

c	$\upsilon(c)$	Powód
":",	2	Słowa związane trochę słabiej
";"	10	Słowa prawdopodobnie ze sobą powiązane
".!?"	30	Prawdopodobnie nie powiązane ze sobą
i pozostałe	1	Słowa związane ze sobą

mencie T (ozn. $\delta(-,-)$) zdefiniowana jest jako suma:

$$\delta(x,y) = \sum_{c \in P} |c|_x^y \, \upsilon(c).$$

Definicja 16 *Przez klasę abstrakcji $[x]$ słowa x występującego w tekście T należy rozumieć zbiór wszystkich napisów x występujących w tekście T:*

$$[x] = \{x, \exists_{i \in \mathbb{N}}\} \, T(i) = x.$$

Przez $T(i)$ należy rozumieć słowo występujące w tekście T na pozycji i. Przez pozycję słowa należy rozumieć tutaj jego odległość od początku tekstu (w sensie δ).

Definicja 17 *Odległość pomiędzy klasami abstrakcji $[x]$ i $[y]$ oznaczono przez δ' i zdefiniowano jako:*

$$\delta'([x],[y]) = min\left\{\delta(T(i),T(j)); i,j; x = T(i) \wedge y = T(j)\right\}.$$

Twierdzenie 6 *Funkcja $\delta'(-,-)$ określona na zbiorze $T' \times T' \to \mathbb{N}$ gdzie T' to zbiór wszystkich wystąpień słów zawartych w tekście T:*

1. $\delta'([x],[y]) \geq 0$, $\forall_{x,y \in T'}$,
2. $\delta'([x],[x]) = 0$, $\forall_{x \in T'}$,
3. $\delta'([x],[y]) = \delta'([y],[x])$, $\forall_{x,y \in T'}$.

Dowód:

1. Wynika bezpośrednio z definicji, ponieważ odległość każdego elementu klasy abstrakcji $[x]$ od każdego elementu klasy abstrakcji $[y]$ musi być większa lub równa zero.

2. Dla każdych dwóch elementów klasy abstrakcji $[x]$ odległość pomiędzy nimi może być większa lub równa zero. Jeżeli weźmiemy pod uwagę jedno wystąpienie, to jego odległość do samego siebie jest równa zero, a zatem, z definicji, również odległość pomiędzy klasami abstrakcji ($\delta'([x],[x])$) jest równa zero.

3.

$$\begin{aligned}\delta'([x],[y]) = & \quad (3.1)\\ =& min\left\{\delta(T(i),T(j)); i,j; x = T(i) \wedge y = T(j)\right\} \quad (3.2)\\ =& min\left\{\delta(T(j),T(i)); i,j; x = T(i) \wedge y = T(j)\right\} \quad (3.3)\\ =& \delta'([y],[x]) \quad (3.4)\end{aligned}$$

Należy tu zwrócić uwagę, że w analizie dokumentu wykorzystać trzeba dokument z usuniętymi słowami zbędnymi wyznaczonymi w rozdziale 1.

3.3.1 Efektywny algorytm wyznaczania odległości słów w dokumencie

Kolejne rozważania dotyczą zaprojektowania efektywnego algorytmu, który wyznacza odległości pomiędzy klasami abstrakcji słów w dokumencie. Algorytm buduje drzewo, którego każda gałąź odpowiada pojedynczej klasie abstrakcji i numeruje te klasy abstrakcji.

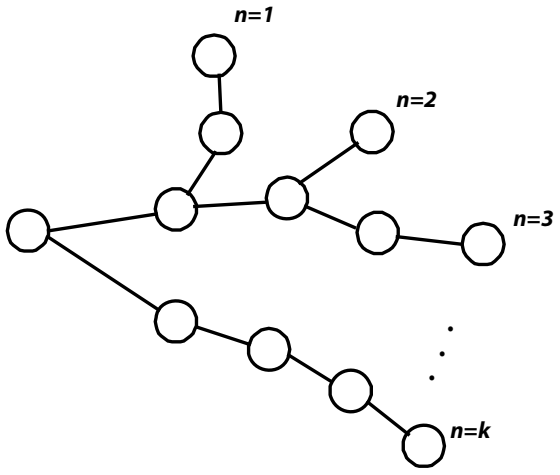
Niech $\widehat{T}^f = \widehat{T}(f)$ będzie tekstem pobranym z dokumentu f, a $\widehat{T}^f(i)$ będzie słowem występującym na pozycji i w tym tekście. Oznaczmy odległość pomiędzy klasami abstrakcji słów A i B w tekście $\widehat{T}^f$ prze $\delta^f(A,B)$.

Algorytm jest analogiczny do algorytmu 6, jednak tutaj za każdym razem gdy osiągany jest w drzewie liść (reprezentujący klasę abstrakcji słowa) uaktualniana jest macierz odległości M (zaprezentowana na obrazku 3.3), która jest macierzą o wymiarach $n \times n$, gdzie $n \in \mathbb{N}$ oznacza liczbę słów przeczytanych do tej pory. Macierz ta jest macierzą dynamiczną, jej rozmiary zmieniają się w trakcie działania algorytmu i zależą od ilości słów przeczytanych przez algorytm.

ξ^f	*1*	*2*	...	k	...	n
1	0			*		
2		0		*		
...			...	*		
k				0*	*	*
...					...	
n						0

Rysunek 3.3: Macierz odległości klas abstrakcji słów.

Wygenerowane drzewo (pokazane na obrazku 3.4) jest wykorzystywane do sprawdzania poprzednich wystąpień aktualnie przeczytanego słowa, przypisywania identyfikatora do klasy abstrakcji tego słowa (oznaczonego jako $\xi^f(j) = \xi(\widehat{T}^f(j))$,$\xi^f(j) \in \mathbf{N}$) i do decydowania czy uaktualnić macierz odległości M czy zmienić jej wielkość poprzez dodanie kolejnego wiersza i kolumny. Obrazek 3.3 ma zaznaczone gwiazdkami pola uaktualniane w pierwszym przypadku. Aby obliczać wartości nowych pól macierzy oraz uaktualniać zawartość już istniejących zaproponowany w niniejszej pracy algorytm używa dodatkowo tablicy z pozycjami ostatnich wystąpień wszystkich przeczytanych słów. Tablica ta jest oznaczona przez $\lambda(\xi^f(j))$



Rysunek 3.4: Drzewo słów.

Należy rozważyć dwa przypadki :

1. Załóżmy, że z wejściowego tekstu zostało przeczytanych $j-1$ słów, a słowo

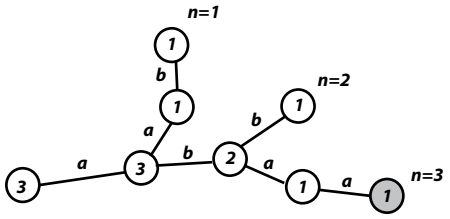
z pozycją j jest przeczytane po raz pierwszy. Wówczas:

$$\forall_{i<j} \widehat{T}^f(j) \neq \widehat{T}^f(i) \tag{3.5}$$

$$\lambda(\xi^f(j)) = i \tag{3.6}$$

$$\forall_{k\in\{1,\ldots,\xi^f j\}} \mathbf{M}[k, \xi^f(j)] = |j - \lambda(k)| \tag{3.7}$$

Ten przypadek pokazany jest na obrazkach 3.5 i 3.6, na których przedstawiona jest wizualizacja stanu algorytmu po przeczytaniu pierwszych trzech słów z przykładowego ciągu „aab abb abaa ‖ abb bbba abb bbaa baa bbaa bbbb bbba bbbb". Na obrazku 3.6 jest zaprezentowana tablica ostatnich wystąpień słów.



Rysunek 3.5: Drzewo słów wygenerowane po przeczytaniu pierwszych trzech słów.

	$\lambda(\xi^f)$	1	2	3
	ξ^f	*1*	*2*	*3*
aab	*1*	0	1	2
abb	*2*		0	1
abaa	*3*			0

Rysunek 3.6: Macierz odległości wygenerowana po przeczytaniu pierwszych trzech słów.

2. Załóżmy że $j-1$ słów zostało przeczytanych z tekstu wejściowego a aktualnie czytane słowo j powtórzyło się i ma już zdefiniowaną klasę abstrakcji (tzn. ma przypisany identyfikator). Czyli:

$$\exists_{i<j} \widehat{T}^f(j) = \widehat{T}^f(i) \tag{3.8}$$

Należy wówczas uaktualnić tablicę ostatnich wystąpień poprzez

$$\lambda(\xi^f(j)) = j \tag{3.9}$$

oraz uaktualnić odpowiedni wiersz i kolumnę w macierzy M:

$$\forall_{k\in\{1,\ldots,\xi^f(j)-1\}} M[k,\xi^f(j)] = \Delta(k,\xi^f(j)) \tag{3.10}$$

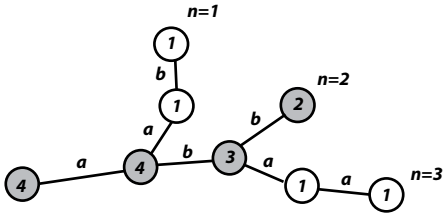
$$\forall_{k\in\{\xi^f(j)+1,\ldots,\widehat{\xi}^f\}} M[\xi^f(j),k] = \Delta(\xi^f(j),k) \tag{3.11}$$

gdzie

$$\widehat{\xi}^f = \max_{i=\{1,\ldots,j\}} \{\xi^f(i)\} \tag{3.12}$$

$$\Delta(k,l) = min(|\lambda(k) - \lambda(l)|, M[k,l]) \tag{3.13}$$

Ten przypadek pokazany jest na obrazkach 3.7 i 3.8, które zawierają wizualizację stanu algorytmu po po przeczytaniu pierwszych czterech słów z przykładowego ciągu „aab abb abaa abb || bbba abb bbaa baa bbaa bbbb bbba bbbb". W tym przypadku słowo „abb" zostało przeczytane dwukrotnie, więc musimy uaktualnić tablicę λ oraz macierz odległości M.



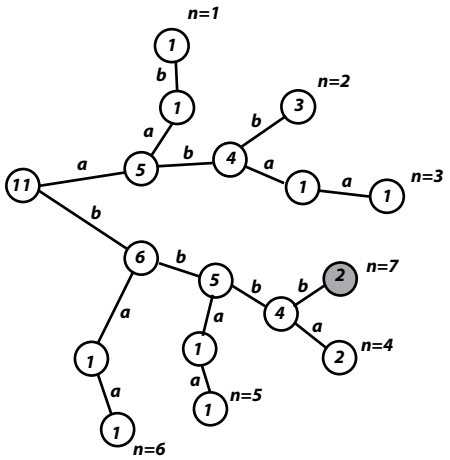
Rysunek 3.7: Drzewo słów wygenerowane po przeczytaniu pierwszych czterech słów.

	$\lambda(\xi^f)$	1	**4**	3
	ξ^f	*1*	*2*	*3*
aab	*1*	0	1	**2**
abb	*2*		0	**1**
abaa	*3*			0

Rysunek 3.8: Macierz odległości wygenerowana po przeczytaniu pierwszych czterech słów.

Pogrubione elementy oznaczają wartości, które mogą być wyznaczone za każdym razem z użyciem tablicy λ, więc nie muszą być zapamiętywane w macierzy. W wyniku tej obserwacji możemy pominąć zapisywanie odległości w macierzy odległości, wypełniając tylko istotne wartości na specjalnych listach (obrazek 3.11),

zawierających tylko te elementy, które nie mogą być łatwo wyznaczone za pomocą tabeli λ. Dzięki temu możemy osiągnąć dużą oszczędność pamięciową. Dla uproszczenia rozważań w dalszej części pracy będziemy dalej się posługiwać macierzą odległości. Końcowy wynik działania algorytmu na przykładowym tekście „aab abb abaa abb bbba abb bbaa baa bbaa bbbb bbba bbbb ‖") widać na obrazkach 3.9 i 3.10.



Rysunek 3.9: Drzewo słów wygenerowane po przeczytaniu wszystkich słów z przykładowego tekstu.

	$\lambda(\xi^f)$	1	6	3	11	9	8	12
	ξ^f	*1*	*2*	*3*	*4*	*5*	*6*	*7*
aab	*1*	0	1	**2**	4	6	**7**	9
abb	*2*		0	1	1	1	**2**	4
abaa	*3*			0	2	4	**5**	7
bbba	*4*				0	**2**	**3**	**1**
bbaa	*5*					0	**1**	1
baa	*6*						0	2
bbbb	*7*							0

Rysunek 3.10: Macierz odległości wygenerowana po przeczytaniu wszystkich słów z przykładowego tekstu.

$\lambda(\xi^f)$	$\lambda(1)$	$\lambda(2)$	...	$\lambda(\widehat{\xi}^f)$
ξ^f	*1*	*2*	...	$\widehat{\xi}^f$
	$\delta^f(1,u_{1,1})$	$\delta^f(1,u_{2,1})$	...	$\delta^f(1,u_{\widehat{\xi}^f,1})$
	$\delta^f(1,u_{1,2})$	$\delta^f(1,u_{2,2})$	...	$\delta^f(1,u_{\widehat{\xi}^f,2})$
	...	...	...	...
	$\delta^f(1,u_{1,k_1})$	$\delta^f(1,u_{2,k_2})$	...	$\delta^f(1,u_{\widehat{\xi}^f,k_{\widehat{\xi}^f}})$

Rysunek 3.11: Schemat tablicy odległości.

Zaproponowany algorytm gwarantuje, że macierz zawiera odległości pomiędzy klasami abstrakcji słów.

$$\delta^f(A,B) = M[\xi(A),\xi(B)] \tag{3.14}$$

Definicja 18 *Przez słowa lokalnie bliskie należy rozumieć słowa, których klasy abstrakcji są sobie bliskie.*

3.4 Kategoryzacja słów w oparciu o sieci neuronowe Kohonena

Podstawą dalszych rozważań jest spostrzeżenie, że słowa lokalnie bliskie są ze sobą związane tematycznie. Dlatego łącząc takie słowa w grupy można zwiększyć znaczenie tych, które w grupie występują rzadziej. Sytuacja taka występuje na przykład, gdy tekst naukowy opisuje jakieś pojęcie złożone, a następnie dla uproszczenia korzysta tylko z jednego członu tego pojęcia.

3.4.1 Model matematyczny

W tym przypadku model matematyczny jest analogiczny do modelu wykorzystanego podczas kategoryzacji dokumentów. Zgodnie z przedstawionym wcześniej algorytmem, wszystkie różne słowa znalezione w tekście zostały ponumerowane liczbami od 1 do n. Dlatego modelem danych są tu wektory n-wymiarowe postaci

$$x(t) = \begin{bmatrix} x^1(t) \\ x^2(t) \\ \vdots \\ x^n(t) \end{bmatrix}$$

spełniające warunek, że

$$x^i(t) = \begin{cases} 1 \\ 0 \end{cases}$$

$x^i(t)$ przyjmuje wartość 1 jeśli słowo o numerze i (oznaczmy je s_i) należy do rozważanego modelu (który reprezentuje kategorię) w kroku czasu t i 0 w przeciwnym wypadku. Na tej podstawie model zawierający wszystkie n badanych słów ma w każdym kroku czasu t postać

$$x_{00}(t) = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

W każdym kroku czasu t wykonujemy wybraną akcję zmieniając stan sieci. Stan sieci w kroku czasu t $m(t)$ spełnia następujące warunki:

$$m(t) = \begin{bmatrix} m_{11}(t) & m_{21}(t) & \dots & m_{j1}(t) \\ m_{12}(t) & m_{21}(t) & \dots & m_{j2}(t) \\ \vdots & \vdots & \ddots & \vdots \\ m_{1j}(t) & m_{2j}(t) & \dots & m_{jj}(t) \end{bmatrix}$$

takie że

$$m_{kl}(t) = \begin{bmatrix} x^1_{kl}(t) \\ x^2_{kl}(t) \\ \vdots \\ x^n_{kl}(t) \end{bmatrix}$$

oraz

$$\sum_{k,l=1\dots j} m_{kl}(t) = \sum_{k,l=1\dots j} \begin{bmatrix} x^1_{kl}(t) \\ x^2_{kl}(t) \\ \vdots \\ x^n_{kl}(t) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = x_{00}(t)$$

W kroku pierwszym ($t = 1$) wektory $m_{kl}(1)$ dobierane są losowo. W każdym kolejnym ($t > 1$) stan sieci zależny jest od stanu w kroku poprzednim i wykonanej akcji.

$$m_{kl}(t) = m_{kl}(t-1) + m^T_{kl}(t-1) \times (M \times D_{kl}) \times A_{kl}$$

gdzie D_{kl} jest $n \times n$ zero-jedynkową macierzą definiującą sąsiedztwo składników modelu m w stosunku do składnika m_{kl} a A_{kl} jest $n \times n$ macierzą przejścia

definiującą funkcję zmiany wartości a:

$$M = \begin{bmatrix} m_{11}(t) \\ m_{12}(t) \\ \vdots \\ m_{1j}(t) \\ m_{21}(t) \\ \vdots \\ m_{2j}(t) \\ \vdots \\ m_{jj}(t) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{M}_{n \times n}$$

Modelem matematycznym jest tu więc trójka $M = \langle X, A, \tau \rangle$, gdzie X to zbiór stanów ($\forall_t m(t) \in X$), A to zbiór akcji, τ to funkcja zmiany stanów zdefiniowana jako:

$$\tau(m) = \tau([m_{kl}]_{k,l=1..j}) = [\tau'(m_{kl})]_{k,l=1..j}$$

gdzie

$$\tau'(m_{kl}) = m_{kl} + m_{kl}^T \times (X \times D_{kl}) \times A_{kl}$$

Zbiór wszystkich możliwych stanów sieci X zawiera p^n różnych elementów, gdzie $p = j^2$ to ilość kategorii a n to ilość zidentyfikowanych słów.

Strategia działania sieci polega na zminimalizowaniu sumy:

$$\sum_{k,l} \left(\sum_{A,B | x_{kl}^{\xi(A)}=1 \wedge x_{kl}^{\xi(B)}=1} d^f(A,B)) \right)$$

Zbieżność

Zbieżność modelu zapewniona jest podobnie jak w przypadku kategoryzacji tekstów, poprzez sprowadzenie problemu do przypadku jednowymiarowego rozważanego w Rojas (1996).

3.4.2 Algorytm

Zaproponowana w tej pracy procedura kategoryzacji słów odbywa się podobnie jak w przypadku kategoryzacji tekstów i składa się z 5 kroków.

Algorytm 15 ***Kategoryzacja słów***

1. *Stwórz kwadratową sieć o wymiarach $m \times m$, gdzie $m = \lfloor \sqrt[4]{\widehat{\xi}^f} \rfloor$. Prezentowany algorytm może rozróżnić maksymalnie m^2 kategorii. Każdy węzeł (oznaczany jako $\omega_{x,y}$) jest połączony z czterema sąsiadami i zawiera prototyp słowa (oznaczony jako $\widehat{T}^f_\omega(x,y)$) oraz zbiór słów lokalnie bliskich prototypowi (oznaczony jako $\beta^f_\omega(x,y)$).*

2. *Dla każdego węzła wybierz losowy prototyp kategorii $p \in \{1, 2, ..., \widehat{\xi}^f\}$.*

3. *Dla każdego słowa $k \in \{1, 2, ..., \widehat{\xi}^f\}$ wybierz najbliższy mu prototyp $\widehat{T}^f_\omega(x,y)$ w sieci i dodaj to słowo do jego listy $\beta^f_\omega(x,y)$.*

4. *Dla każdego węzła $\omega_{x,y}$ oblicz uogólnioną medianę ze słów z jego listy $\beta^f_\omega(x,y)$ i list sąsiadów (oznaczoną jako β). Uogólniona mediana jest zdefiniowana jako element A minimalizujący funkcję:*

$$\Sigma_{B \in \beta} \delta^{f2}(A, B) \tag{3.15}$$

   ustaw $\widehat{T}^f_\omega(x,y) = A$

5. *Powtarzaj krok 4 dopóki sieć nie ustabilizuje się. Stabilność sieci jest osiągnięta kiedy w dwóch kolejnych iteracjach wszystkie listy słów pozostaną niezmienione (ich położenie względem poszczególnych węzłów może się zmienić).*

Taki algorytm dokonuje podziału zbioru wszystkich słów znalezionych w dokumencie na rozłączne podzbiory, zawierające słowa lokalnie bliskie. Słowa z takich zbiorów są ze sobą związane.

Podczas kategoryzacji tekstów słowa zostają podzielone na kategorie. Każdej kategorii (zbiorowi słów) można przydzielić pewną liczbę rankingową danej kategorii, wpływającą na ogólne znaczenie (a co za tym idzie pozycję w końcowym wyborze) słów znajdujących się w tej kategorii. Testy pokazały, że jednym z dobrych oszacowań takiej liczby jest

$$\rho_{x,y} = \frac{\Sigma_{w \in \beta^f_\omega(x,y)} |[w]|}{|\beta^f_\omega(x,y)|} \tag{3.16}$$

Teraz kategorie należy posortować zstępująco ze względu na tę liczbę rankingową, a następnie z każdej kategorii wybrać proporcjonalną do ilości zawartych w niej słów, grupę kandydatów na słowa kluczowe. Takie postępowanie otwiera przed słowami rzadko występującymi w tekście możliwość pojawienia się na liście

wynikowej. Dla przykładu: jeżeli celem badania jest wyznaczenie 20 propozycji słów kluczowych, a dokument został podzielony na 5 równolicznych kategorii, to z każdej z kategorii należy wziąć 4 najlepsze wyniki. Mimo tego, że niektóre z wyników mogą zostać usunięte z finalnej listy, to i tak trafność dopasowania jest większa niż w przypadku wybrania 20 najczęściej występujących w dokumencie słów. Widać to na wykresie 3.12, na którym przedstawiono wyniki.

3.5 Kategoryzacja słów w oparciu o sieci neuronowe Kohonena ze wzmocnieniem

Kolejnym etapem niniejszej pracy było poprawienie działania sieci, poprzez pomoc w wyborze prototypów słów w poszczególnych iteracjach. Celem jest zmniejszenie prawdopodobieństwa, że prototypem zostanie słowo nieistotne dla rozważań.

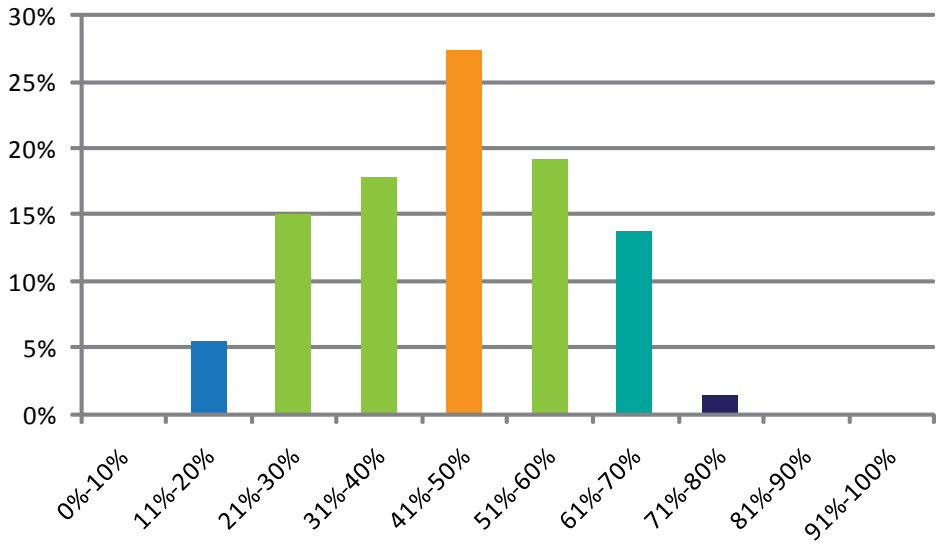
Przedstawione tu rozwiązanie jest oryginalnym rozwiązaniem zaproponowanym w ramach niniejszej pracy i opiera się na manipulację odległościami pomiędzy słowami, w taki sposób, aby słowa niekluczowe były bardziej odległe od pozostałych słów w dokumencie, co implikuje zmniejszenie prawdopodobieństwa wyboru ich jako kolejnych prototypów kategorii. Zaproponowane w niniejszej pracy nowe rozwiązanie korzysta z idei uczenia się ze wzmocnieniem, które dokonywane jest na podstawie wiedzy zgromadzonej podczas analizy innych dokumentów.

3.5.1 Uczenie się ze wzmocnieniem

Podstawową ideą uczenia się ze wzmocnieniem są interakcje algorytmu (ucznia) ze środowiskiem, w którym realizuje zadanie uczenia się. Na podstawie aktualnego stanu środowiska uczeń podejmuje decyzje co do kolejnych kroków. Poniżej przedstawiono zaprezentowany w Cichosz (2000) scenariusz uczenia się ze wzmocnieniem, który jest ogólnym zarysem tego procesu.

Algorytm 16 ***Scenariusz uczenia się ze wzmocnieniem***
W każdym kroku czasu t:

1. *obserwuj aktualny stan x_t;*
2. *wybierz akcję a_t do wykonania w stanie x_t;*
3. *wykonaj akcję a_t;*
4. *obserwuj wzmocnienie r_t i następny stan x_t;*
5. *ucz się na podstawie doświadczenia $< x_t, a_t, r_t, x_{t+1} >$.*



Rysunek 3.12: Wyniki wybierania słów kluczowych opartego o sieć Kohonena. Oś X pokazuje skuteczność wyboru słów kluczowych (tzn. względną ilość poprawnie zaproponowanych słów kluczowych), oś Y pokazuje ilość dokumentów, dla których taka skuteczność została osiągnięta.

Poszczególne implementacje uczenia się ze wzmocnieniem opierają się zwykle na dostosowanej do własnych potrzeb wersji podanego modelu.

Po wykonaniu każdej akcji uczeń otrzymuje nagrodę (na przykład rzeczywistoliczbową), przyznawaną przez tak zwanego krytyka. Tak więc sam uczeń nie może na nagrody wpływać. Z implementacyjnego punktu widzenia krytyk może być częścią środowiska, w którym pracuje uczeń lub samym środowiskiem. Może jednak on być również częścią ucznia. Z logicznego punktu widzenia lepiej rozważać krytyka jako niezwiązany z uczniem fragment środowiska.

Uczenie się ze wzmocnieniem ma więc specyficzne właściwości:

1. dane trenujące mają charakter wartościujący,
2. określają cel zadania, zamiast sposobu jego osiągnięcia,
3. wykonywanie zadania odbywa się jednocześnie z uczeniem się.

3.5.2 Ogólny model matematyczny

Modelem matematycznym uczenia się ze wzmocnieniem jest proces decyzyjny Markowa (MDP - Markov Decision Process).

$$MDP = \langle X, A, \rho, \delta \rangle$$

gdzie X to skończony zbiór stanów, A to skończony zbiór akcji, ρ to funkcja nagrody, δ to funkcja zmiany stanów.

Dla każdej pary $\langle x_t, a_t \rangle \in X \times A$ nagrodę otrzymaną po wykonaniu akcji a_t w stanie x_t oznaczamy jako $\rho(x_t, a_t) = r_t$ (ρ jest zmienną losową). Przez $\delta(a_t, x_t) = x_{t+1}$ oznaczamy następny stan po wykonaniu akcji a_t w stanie x_t.

Należy zwrócić uwagę, że ważnym elementem jest tutaj własność, że ρ i δ zależą tylko od aktualnego stanu i akcji, natomiast nie ma na nie wpływu historia procesu decyzyjnego. Własność ta znana jest jako *własność Markowa.*

Definicja 19 *Własność Markowa: ρ i δ nie zależą od historii.*

Od ucznia oczekuje się nauczenia strategii, czyli odwzorowania stanów na akcje, które mają być na nich wykonane. Cel ten określany jest pośrednio przez funkcję wzmocnienia, ponieważ definiuje ona kryterium jakości, które ma zostać zmaksymalizowane przez uczoną strategię. Zwykle maksymalizuje się sumę:

$$\sum_{t=0}^{\infty} \gamma^t r_t,$$

gdzie $\gamma \in [0, 1]$ jest współczynnikiem regulującym względną wartość nagród. Aby ułatwić posługiwanie się funkcjami przejścia i wzmocnienia, definiuje się:

$$R(x, a) = E[\rho(x, a)]$$

oraz

$$P_{xy}(a) = Pr(\delta(x, a) = y).$$

W procesie decyzyjnym Markowa można zdefiniować pojęcie strategii i funkcji wartości.

Definicja 20 *Strategią nazywa się dowolną funkcję działającą ze zbioru stanów (X) na zbiór akcji (A)*

$$\pi : X \to A.$$

Definicja 21 *Funkcja wartości ze względu na strategię π jest dla każdego stanu $x \in X$ określona następująco:*

$$V^{\pi}(x) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t | x_0 = x].$$

Funkcja wartości przyporządkowuje każdemu stanowi oczekiwaną wartość sumy przyszłych nagród, jakie są otrzymane przez ucznia rozpoczynającego uczenie w stanie x.

Definicja 22 *Funkcja wartości akcji ze względu na strategię π jest dla każdej pary stan-akcja $< x, a > \in X \times A$ określona następująco:*

$$Q^{\pi}(x,a) = E_{\pi}[\rho(x,a) + \sum_{t=1}^{\infty} \gamma^t r_t | x_0 = x, a_0 = a].$$

Funkcja wartości akcji przyporządkowuje każdej parze stan-akcja oczekiwaną wartość sumy przyszłych nagród, jakie będą otrzymane przez ucznia rozpoczynającego uczenie w stanie x poprzez wykonanie akcji a.

Symbol E_{π} oznacza tu wartość oczekiwaną przy założeniu użycia strategii π. W powyższych definicjach użyta jest strategia niezmieniająca się w czasie, jednak większość implementacji stosuje strategie modyfikowane podczas uczenia się. Zwykle również używane strategie są niedeterministyczne, to znaczy, że wykonywana akcja zależy od strategii probabilistycznie.

Definicja 23 *Strategia π' jest lepsza od strategii π (ozn. $\pi' > \pi$) jeżeli*

$$\forall_{x \in X} V^{\pi'}(x) \geq V^{\pi}(x) \wedge \exists_{x_0 \in X} V^{\pi'}(x_0) > V^{\pi}(x_0).$$

Definicja 24 *Strategia π' jest optymalna jeżeli nie istnieje strategia od niej lepsza*

$$!\exists_{\pi} \pi > \pi',$$

to znaczy, że strategią optymalną jest każda strategia maksymalizująca wartość każdego stanu.

Twierdzenie 7 *Dla dowolnego procesu decyzyjnego Markowa istnieje przynajmniej jedna strategia optymalna, która jest stacjonarna i deterministyczna.*

Powyższe twierdzenie jest zagwarantowane przez teorię programowania dynamicznego zainicjowaną przez Bellmana (Bellman (1957)) w latach 50. Wynika z niej, że każdej strategii optymalnej odpowiada ta sama optymalna funkcja wartości i optymalna funkcja wartości akcji. Pozwala to na wyznaczenie dowolnej z tych funkcji, przy założeniu znajomości $R(x,a)$ i $P_{xy}(a)$ dla każdych $x, y \in X$ i $a \in A$.

Równania Bellmana, na których opierają się metody programowania dynamicznego i które są postawą algorytmów obliczania funkcji V^{π} Q^{π}, mają postać:

$$V^{\pi}(x) = R(x, \pi(x)) + \gamma \sum_{y} P_{xy}(\pi(x)) V^{\pi}(y)$$

,

$$Q^{\pi}(x,a) = R(x,a) + \gamma \sum_{y} P_{xy}(a) Q^{\pi}(y, \pi(y))$$

.

Uczenie się na podstawie tych równań może być wykonane za pomocą:

1. **uczenia pasywnego**, które wykorzystuje informacje o przeprowadzonych próbach i ich wynikach, i na ich podstawie wyznacza funkcje wartości V^π;

2. **uczenia pasywnego metodą TD-learning** (Cichosz & Mulawka (1995)), które podczas prób tworzy ciąg funkcji wartości V_n^π, który zbiega do V^π ($V_n^\pi \to V^\pi$);

3. **uczenia aktywnego metodą Q-learning**, które podczas prób tworzy ciąg funkcji wartości akcji Q_n^π, który zbiega do Q^π ($Q_n^\pi \to Q^\pi$).

Przy wyznaczonej funkcji wartości V^π lub funkcji wartości akcji Q^π możemy zająć się wyborem strategii, która powinna preferować akcje, dla których wartości tych funkcji są możliwie jak największe.

Przykład labirynt

Jednym ze sztandarowych przykładów uczenia się ze wzmocnieniem jest opisany między innymi w Aycinena & Brunskill (n.d.) przykład labiryntu (rys 3.13), w którym agent musi pokonać odpowiednią drogę. Agent rozpoczyna drogę na polu zielonym. W każdym kroku może poruszyć się w dowolnym kierunku. Koniec drogi to pole czerwone.

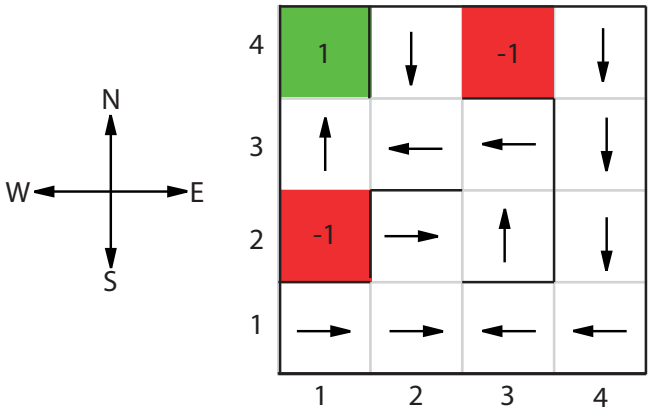
3.5.3 Wykorzystanie nauki ze wzmocnieniem w wyznaczaniu słów kluczowych dokumentu

Jednym z najpopularniejszych algorytmów uczenia się, które są wykorzystywane także do znajdowania słów kluczowych w tekstach są algorytmy wykorzystujące metodę różnic czasowych (TD - temporal differences (Cichosz (1994)), których idea polega na wykorzystaniu różnic pomiędzy aktualnie dokonywanymi decyzjami, a nie pomiędzy decyzjami a końcowym wynikiem, jak jest to w klasycznym algorytmie uczenia się ze wzmocnieniem. Paweł Wawrzyński (Wawrzyński (2005)) udowadnia, że aproksymacja stochastyczna wykorzystywana w większości algorytmów uczenia się ze wzmocnieniem może zostać z powodzeniem zastąpiona poprzez estymację parametrów wzmocnienia, opierającą się na całej historii interakcji między agentem a środowiskiem.

W proponowanych w niniejszej pracy rozwiązaniach wykorzystano te idee do poprawiania wyników działania sieci:

- na bieżąco - bez znajomości wyników działania sieci,

- opierając wzmocnienie na wynikach osiągniętych podczas całej dotychczasowej pracy systemu (w szczególności wynikach analizy innych dokumentów).



Rysunek 3.13: Środowisko labiryntu. Osiągnięcie przez agenta czerwonego pola oznacza wzmocnienie negatywne, zielonego pozytywne.

Opracowane w ramach niniejszej pracy metody wzmocnienia powodują, że poziom poprawności wyników algorytmu wzrasta znacząco w stosunku do klasycznej kategoryzacji przez sieć SOM.

Odległość pomiędzy słowami

Aby zastosować wzmocnienie w projektowanym algorytmie należy zmodyfikować definicję odległości pomiędzy dwoma klasami abstrakcji słów. Należy uwzględnić w niej dodatkowo wagę każdego ze słów.

Definicja 25 *Wagą słowa x jest funkcja $\omega : T' \to (0, 2)$.*

Definicja 26 *Odległość pomiędzy klasami abstrakcji $[x]$ i $[y]$ oznaczamy przez δ'' i definiujemy jako*

$$\delta''([x],[y]) = min\left\{\delta(T(i),T(j)); i,j; x = T(i) \wedge y = T(j)\right\} * \frac{2}{\omega(x)+\omega(y)}$$

Twierdzenie 8 *Funkcja $\delta''(-,-)$ określona na zbiorze $T' \times T' \to \mathbb{N}$ gdzie T' to zbiór wszystkich wystąpień słów zawartych w tekście T:*

1. $\delta''([x],[y]) \geq 0$, $\forall_{x,y \in T'}$
2. $\delta''([x],[x]) = 0$, $\forall_{x \in T'}$
3. $\delta''([x],[y]) = \delta''([y],[x])$, $\forall_{x,y \in T'}$

Dowód

1. Wynika bezpośrednio z definicji, ponieważ odległość każdego elementu klasy abstrakcji $[x]$ od każdego elementu klasy abstrakcji $[y]$ musi być większa lub równa zero.

2. Dla każdych dwóch elementów klasy abstrakcji $[x]$ odległość pomiędzy nimi może być większa lub równa zero. Jeżeli weźmiemy pod uwagę jedno wystąpienie, to jego odległość do samego siebie jest równa zero, a zatem, z definicji, również odległość pomiędzy klasami abstrakcji ($\delta''([x],[x])$) jest równa zero.

3.

$$\delta''([x],[y]) = \tag{3.17}$$

$$= min\left\{\delta(T(i),T(j)); i,j; x = T(i) \wedge y = T(j)\right\} * \frac{2}{\omega(x)+\omega(y)} \tag{3.18}$$

$$= min\left\{\delta(T(j),T(i)); i,j; x = T(i) \wedge y = T(j)\right\} * \frac{2}{\omega(y)+\omega(x)} \tag{3.19}$$

$$= \delta''([y],[x]) \tag{3.20}$$

Matematyczny opis algorytmu

Modelem matematycznym wykorzystanym w uczeniu się bez wzmocnienia z użyciem sieci Kohonena jest trójka

$$M = \langle X, A, \tau \rangle$$

gdzie X to zbiór stanów ($\forall_t m(t) \in X$), A to zbiór akcji a τ to funkcja zmiany stanów. Elementy te zdefiniowane są tak samo jak w przypadku uczenia bez wzmocnienia.

Rozważmy proces decyzyjny Markowa

$$MDP = \langle X, A, \rho, \tau \rangle$$

rozszerzający model M o funkcję nagrody $\rho : X \mapsto [0,1]^n$. Funkcja ta zdefiniowana jest z wykorzystaniem funkcji miary różnic pomiędzy dokumentami. Wzmocnienie w definiowanym przypadku jest dokonywane w każdym kroku algorytmu. Niech $c_d(-) : N \times N$ będzie funkcją przyporządkowującą słowu o podanym numerze ilość jego wystąpień w dokumencie o numerze d. Niech $\epsilon(-) : N \mapsto 0, 1^n$ gdzie n oznacza ilość zgromadzonych dokumentów będzie funkcją przyporządkowującą numerowi słowa, wektor zawierający na k-tej pozycji 1 jeśli słowo i

występuje w dokumencie o numerze k, a 0 w przeciwnym wypadku. Niech $\bar{d}(-) : N \mapsto [0,1]^n$ będzie funkcją przyporządkowującą numerowi aktualnie przeglądanego dokumentu wektor znormalizowanych (należących do przedziału $[0,1]$) funkcja miary różnic od niego do każdego ze zgromadzonych dokumentów:

$$\bar{d}(i) = \begin{bmatrix} d(i,1) \\ d(i,2) \\ \vdots \\ d(i,n) \end{bmatrix}$$

wówczas przez $\rho''(-) : N \mapsto [0,1]$ oznacza się funkcję zdefiniowaną w następujący sposób:

$$\rho''(i) = \frac{\epsilon(i)^T \times \bar{d}(k)}{\epsilon(i)^T \times [1]}$$

$$\rho'_{kl} = \frac{\sum_{s=1..n} x^n_{kl} c_d(x^n_{kl})}{\sum_{s=1..n} x^n_{kl}}$$

$$\rho = [\rho'_{kl}]_{k,l=1..j}$$

Strategia modyfikowana jest na podstawie aktualnego stanu i funkcji nagrody, tak aby w kolejnym kroku wykonana została akcja minimalizująca podaną wcześniej sumę.

Implementacja algorytmu

Początkowo każde ze słów ma przypisaną wagę równą 1.

$$\omega(x) = 1 \forall_{x \in T'}$$

W tym przypadku cały algorytm przebiega identycznie jak poprzednio, ponieważ

$$\omega(x) = 1 \forall_{x \in T'} \Rightarrow \delta'' = \delta'$$

Różnice pojawiają się w momencie, gdy wagi poszczególnych słów są modyfikowane. Należy tu zauważyć, że w każdej iteracji algorytmu wszystkie słowa podzielone są zawsze na kategorie. Po każdej iteracji może więc zostać wygenerowana tymczasowa lista kandydatów na słowa kluczowe. Słowa te powinny być sprawdzone pod kątem trafności a następnie ich wagi powinny zostać zmodyfikowane.

Poprawiona procedura składa się z 6 kroków:

Algorytm 17 ***Algorytm kategoryzacji słów ze wzmocnieniem***

1. *Stwórz kwadratową sieć o wymiarach $m \times m$, gdzie $m = \lfloor \sqrt[4]{\widehat{\xi}^f} \rfloor$. Prezentowany algorytm może rozróżnić m^2 kategorii. Każdy węzeł (oznaczany jako $\omega_{x,y}$) jest połączony z czterema sąsiadami i zawiera prototyp słowa (oznaczony jako $\widehat{T}^f_\omega(x,y)$) oraz zbiór słów lokalnie bliskich prototypowi (oznaczony jako $\beta^f_\omega(x,y)$).*

2. *Dla każdego węzła wybierz losowy prototyp kategorii $p \in \{1, 2, ..., \widehat{\xi}^f\}$.*

3. *Dla każdego słowa $k \in \{1, 2, ..., \widehat{\xi}^f\}$ wybierz najbliższy mu prototyp $\widehat{T}^f_\omega(x,y)$ w sieci i dodaj to słowo do jego listy $\beta^f_\omega(x,y)$.*

4. *Dla każdego węzła $\omega_{x,y}$ stwórz listę testową (oznaczaną przez l_t) zawierającą jego prototyp $\widehat{T}^f_\omega(x,y)$ i w najbliższych mu słów z listy przyporządkowanej do tego prototypu:*

   (a) *Dla każdego ze słów $w \in l_t$ pobierz z bazy dokumentów n dokumentów, zawierających to słowo i oblicz średnią znormalizowaną miarę różnic od nich do aktualnie analizowanego dokumentu (oznaczona przez δ''').*

   (b) *Zmodyfikuj wagę słowa w według następującego wzoru:*

$$\omega(w) = \begin{cases} \frac{\omega(w)}{2} & \delta''' \leq \frac{1}{3} \\ \omega(w) & \frac{1}{3} \leq \delta''' \leq \frac{2}{3} \\ \omega(w) + \frac{(2-\omega(w))}{2} & \delta''' \geq \frac{2}{3} \end{cases}$$

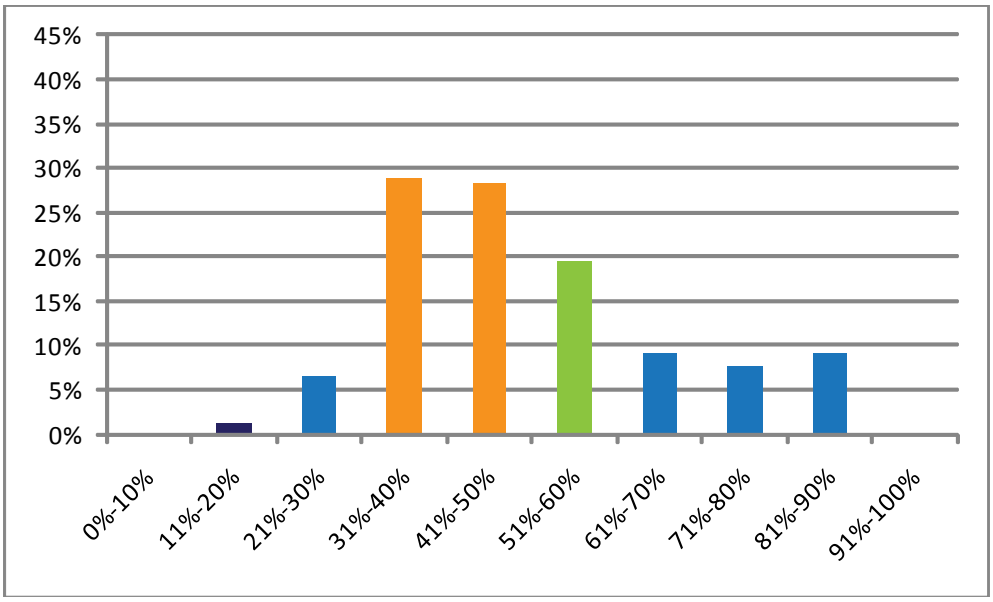
5. *Dla każdego węzła $\omega_{x,y}$ oblicz uogólnioną medianę ze słów z jego listy $\beta^f_\omega(x,y)$ i list sąsiadów (oznaczoną jako β). Uogólniona mediana jest zdefiniowana jako element A minimalizujący funkcję:*

$$\Sigma_{B \in \beta} \delta^{f2}(A, B) \tag{3.21}$$

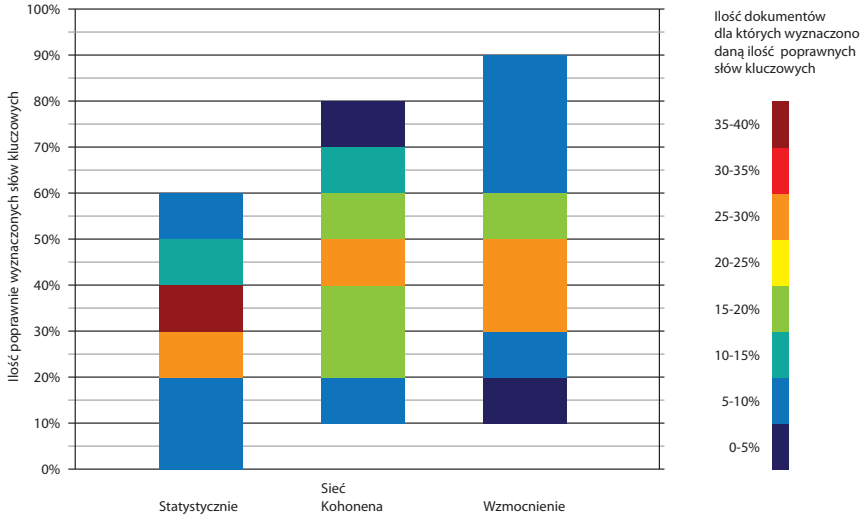
   ustaw $\widehat{T}^f_\omega(x,y) = A$

6. *Powtarzaj krok 4 dopóki sieć nie ustabilizuje się. Stabilność sieci jest osiągnięta kiedy w dwóch kolejncyh iteracjach wszystkie listy słów pozostaną niezmienione (może się zmienić ich położenie względem poszczególnych węzłów.*

Zaprezentowana metoda znacząco poprawia wyniki w stosunku do wersji bez wzmocnienia. Okazuje się, że rezultaty plasują się teraz na poziomie od 40% do 90% skuteczności. Porównanie efektywności przedstawionych metod widoczne jest na wykresie 3.15.



Rysunek 3.14: Wyniki wybierania słów kluczowych opartego o sieć Kohonena ze wzmocnieniem. Oś X pokazuje skuteczność wyboru słów kluczowych (tzn. względną ilość poprawnie zaproponowanych słów kluczowych), oś Y pokazuje ilość dokumentów, dla których taka skuteczność została osiągnięta.



Rysunek 3.15: Porównanie wyników wybierania słów kluczowych trzema przedstawionymi algorytmami. Oś Y oznacza osiągniętą skuteczność a kolor słupka oznacza względną ilość dokumentów, dla których w konkretnym algorytmie (oś X) taka skuteczność została osiągnięta (im ciemniejszy kolor, tym więcej dokumentów w danej grupie).

Specyfika prezentowanego algorytmu sprowadza się do uzależnienia jego skuteczności od ilości dokumentów, które znajdują się w repozytorium systemu. Innymi słowy skuteczność algorytmu dla pojedynczego tekstu zależy od ilości doświadczeń zdobytych dla innych tekstów. Wraz ze wzrostem liczby dokumentów rośnie jego efektywność, ponieważ mniej prawdopodobne staje się uzyskanie pasującego dokumentu na podstawie nieprawidłowego słowa kluczowego. Jednocześnie ważną rolę odgrywa zróżnicowanie tematyczne danych bazowych. Można tu znaleźć analogię do mózgu człowieka - im większy zakres wiedzy człowiek jest w stanie opanować, tym lepiej może poruszać się w jej wybranym fragmencie.

Zróżnicowanie dokumentów w repozytorium osiągnąć można na kilka sposobów:

- udostępnienie systemu wielu użytkownikom,
- automatyczne poszerzanie biblioteki dokumentów na podstawie danych znalezionych w sieci internet.

Zbieżność sieci

Zbieżność sieci może zostać wykazana podobnie jak w przypadku wersji bez wzmocnienia, gdzie każdy kolejny krok czasu powodował zmniejszenie zawsze dodatniej ogólnej wagi sieci.

3.6 Kategoryzacja słów z wykorzystaniem dynamicznych sieci neuronowych i algorytmów genetycznych

Podobnie jak w przypadku kategoryzacji tekstów, algorytm kategoryzacji Kohonena może zostać zastąpiony poprzez algorytm wykorzystujący sieci dynamiczne i algorytmy genetyczne. Wyniki algorytmu są analogiczne do wyników uzyskanych w poprzednim rozdziale.

Podobnie jest ze zbieżnością algorytmu, która została udowodniona w poprzednim rozdziale.

3.7 Kategoryzacja słów z wykorzystaniem rankingu słów

Jednym z założeń prezentowanego wyżej algorytmu jest fakt, że wszystkie słowa mają początkowo wagę 1. Okazuje się jednak, że początkową wagę słów można

określić dużo dokładniej. Owocuje to lepszym i bardziej szczegółowym dopasowaniem słów kluczowych w końcowej fazie algorytmu. W ramach niniejszej pracy zostało zaproponowane rozwiązanie korzystające z idei przyświecającej twórcom wyszukiwarki Google podczas tworzenia algorytmu PageRank (Avrachenkov & Litvak (2004)).

Zaproponowane w ramach niniejszej pracy rozwiązanie wyznacza wstępne wartościowanie słów podlegających później procesowi kategoryzacji, korzystając z łańcuchów Markowa oraz zależności pomiędzy słowami w tekście .

Łańcuchy Markowa

W celu wprowadzenia łańcuchów Markowa, należy podać za Bremaud (2001) i Haggstrom (2002) kilka definicji.

Definicja 27 *Ciąg zmiennych losowych $(X_n)_{n=0,...}$ o wartościach w przeliczalnym zbiorze S (przestrzeni stanów) nazywamy łańcuchem Markowa wtedy i tylko wtedy, gdy dla każdego n naturalnego i każdego ciągu $s_0, s_1, ..., s_n \in S$ mamy*

$$P(X_n = s_n | X_{n-1} = s_{n-1}, ..., X_0 = s_0) = P(X_n = s_n | X_{n-1} = s_{n-1})$$

jeśli tylko $P(X_{n-1} = s_{n-1}, ..., X_0 = s_0) > 0$, to znaczy stan układu w momencie n, czyli X_n zależy tylko od stanu układu w momencie $n-1$ czyli X_{n-1}.

Definicja 28 *Własność Markowa: macierz $P = [p_{ij}]_{i,j \in S}$ nazywamy macierzą przejścia na S, gdy wszystkie jej wyrazy są nieujemne, a ponadto suma każdego wiersza wynosi 1.*

$$p_{ij} \geq 0, \sum_{k \in S} p_{ik} = 1$$

Definicja 29 *Zmienna losowa X_0 jest nazywana stanem początkowym a rozkład jej prawdopodobieństwa $v(i) = P(X_0 = i)$, rozkładem początkowym.*

Twierdzenie 9 *Rozkład łańcucha Markowa zależy od rozkładu początkowego i macierzy przejścia.*

Większość stosowanych łańcuchów Markowa można opisać za pomocą twierdzenia:

Twierdzenie 10 *Niech $(Z_n)_{n \geq 1}$ będzie ciągiem losowych zmiennych o wartościach w pewnej przestrzeni F. Niech E będzie przestrzenią przeliczalną i $f : E \times F \to E$ będzie pewną funkcją. Niech X_0 będzie losową zmienną z wartościami w E, niezależną od Z_n. Równanie rekurencyjne*

$$X_{n+1} = f(X_n, Z_{n+1})$$

definiuje łańcuch Markowa.

Dowód twierdzenia można znaleźć w Bremaud (2001).

3.7.1 Założenia Google PageRank

Jak zostało pokazane w Page *et al.* (1999) PageRank jest przełomową technologią zmieniającą podejście do wyszukiwania materiałów w sieci[1]. Opiera się na ocenie ważności strony.

PageRank każdej strony wyznaczany jest na podstawie ilości odnośników, które na tę stronę wskazują. Są one interpretowane jak głosy na tę stronę. Tak więc zakładając, że na stronach $\{X_1, X_2, ..., X_n\}$ znajdują się odnośniki wskazujące na stronę A, oraz $\|X_i\|$ oznacza ilość wszystkich odnośników na stronie X_i, $\|X_i\|_A$ oznacza ilość wszystkich odnośników na stronie X_i wskazujących na stronę A oraz PR_{X_i} oznacza PageRank strony X_i a N to ilość wszystkich stron, wówczas PageRank strony A będzie równy

$$PR_A = \frac{1-d}{N} + d(PR_{X_1}\frac{\|X_1\|_A}{\|X_1\|} + ... + PR_{X_n}\frac{\|X_n\|_A}{\|X_n\|})$$

Przez d należy rozumieć prawdopodobieństwo wejścia na pojedynczą stronę przez użytkownika. Wielkość tę ustalono empirycznie na 0.85.

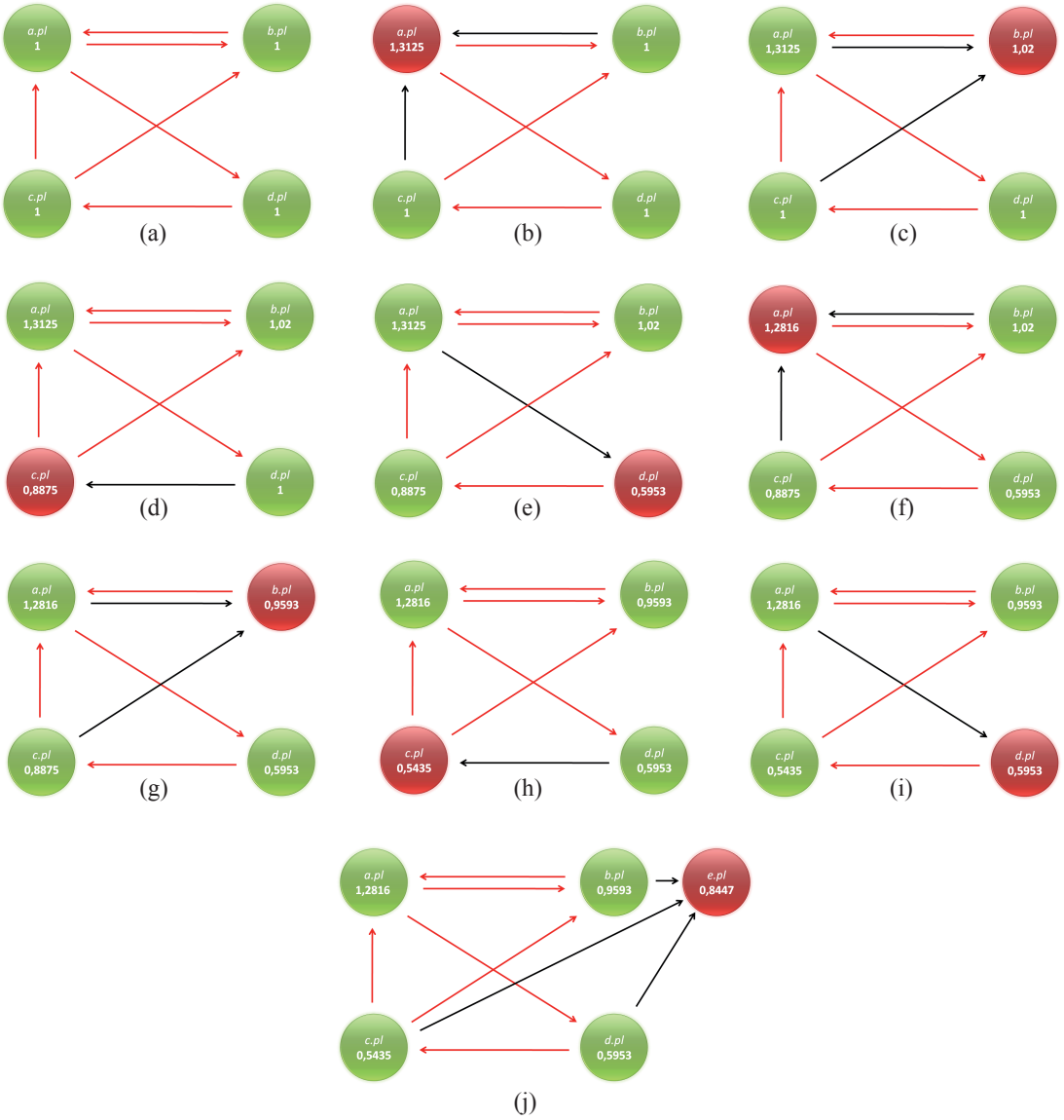
Jak wynika z powyższego równania, nie każdy głos jest tak samo ważny. Ważniejsze są głosy stron, które same mają większy PageRank.

W celu określenia PageRank dla każdej z istniejących stron, poszukuje się rozwiązania równania zawierającego ponad 500 milionów zmiennych.

Przykładowe działanie podstawowego algorytmu PageRank jest pokazane na ilustracji 3.16. Kolejne iteracje algorytmu powodują modyfikacje wag poszczególnych stron. I tak w (a) pokazane są wagi początkowe, (b)-(i) modyfikacje wag, (j) to waga dodanej strony. Kolejne obliczenia to

- (b) $PR_{a.pl} = \frac{(1-0.85)}{4} + 0,85(\frac{1}{1} + \frac{1}{2}) = 1,3125$
- (c) $PR_{b.pl} = \frac{(1-0,85)}{4} + 0,85(\frac{1,3125}{2} + \frac{1}{2}) = 1,02$
- (d) $PR_{c.pl} = \frac{(1-0,85)}{4} + 0,85(\frac{1}{1}) = 0,8875$
- (e) $PR_{d.pl} = \frac{(1-0,85)}{4} + 0,85(\frac{1,3125}{2}) = 0,5953$
- (f) $PR_{a.pl} = \frac{(1-0,85)}{4} + 0,85(\frac{1,02}{1} + \frac{0,8875}{2}) = 1,2816$
- (g) $PR_{b.pl} = \frac{(1-0,85)}{4} + 0,85(\frac{1,2816}{2} + \frac{0,8875}{2}) = 0,9593$
- (h) $PR_{c.pl} = \frac{(1-0,85)}{4} + 0,85(\frac{0,5953}{1}) = 0,5435$
- (i) $PR_{d.pl} = \frac{(1-0,85)}{4} + 0,85(\frac{1,2816}{2}) = 0,5821$
- (j) $PR_{e.pl} = \frac{(1-0,85)}{5} + 0,85(\frac{0,9593}{2} + \frac{0,5435}{3} + \frac{0,5954}{2}) = 0,8447$

[1] Technologia ta jest opatentowana przez Google



Rysunek 3.16: Przykładowe działanie algorytmu PageRank (a-i) dla czterech stron i (j) dla strony piątej.

Łańcuchy Markowa w PageRank

Kompletny opis podstawowego algorytmu PageRank przedstawiony jest w Avrachenkov & Litvak (2004). W algorytmie tym konstruowana jest macierz przejścia $P = [p_{ij}]_{i,j \in N}$, gdzie p_{ij} jest prawdopodobieństwem z jakim może nastąpić przejście ze strony i na stronę j. Prawdopodobieństwo p_{ij} jest równe ilorazowi ilości odnośników na stronie i wskazujących na stronę j i ilości wszystkich odnośników na stronie i lub 0, jeśli nie istnieją odnośniki ze strony i do strony j.

Na tej podstawie definiuje się macierz $P' = cP + (1-c)(1/n)E$, gdzie E jest macierzą o wszystkich współczynnikach równych 1, n oznacza ilość wszystkich stron internetowych, a c to prawdopodobieństwo świadomego kliknięcia w konkretny odnośnik na stronie. Ustala się, że prawdopodobieństwo przypadkowego wejścia na stronę wynosi $(1-c)$. Poprzez przypadkowe wejście na stronę można rozumieć przypadkowe kliknięcie w odnośnik, kliknięcie przycisku wstecz, błędne wpisanie adresu itp.

Przyjmuje się również, że prawdopodobieństwo otworzenia dowolnej pierwszej strony (w kroku zerowym) wynosi $P(X_o = s_i) = p_i(0) = \frac{1}{n}$ dla $i = 1, ..., n$.

Rozkład zmiennej X_k jest wyznaczany za pomocą równania macierzowego

$$[p_1(k-1), p_2(k-1), ..., p_n(k-1)] \begin{bmatrix} p'_{11} & p'_{12} & \dots & p'_{1n} \\ p'_{21} & p'_{22} & \dots & p'_{2n} \\ \vdots & \vdots & & \vdots \\ p'_{n1} & p'_{n2} & \dots & p'_{nn} \end{bmatrix} = [p_1(k), p_2(k), ..., p_n(k)]$$

Ciąg $X_0, X_1, ..., X_n, ...$ spełnia założenia definicji łańcucha Markowa a macierz P' spełnia założenia macierzy przejścia.

Zgodnie z założeniami ciąg (X_n) zbiega do rozkładu stacjonarnego reprezentowanego przez wektor π

$$\pi P' = \pi$$

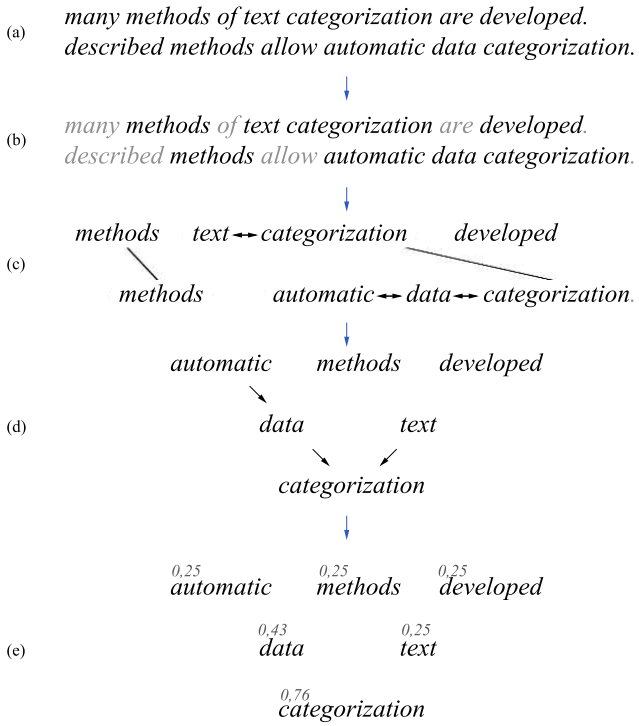
$$\pi[1] = [1]$$

Elementy tego wektora określają prawdopodobieństwo wejścia użytkownika na poszczególne strony internetowe i są wykorzystywane jako wartości PageRank dla tych stron. PageRank determinuje kolejność wyników wyszukiwania Google.

3.7.2 Założenia rankingu słów

W przeciwieństwie do zbioru stron internetowych, w którym można łatwo odnaleźć powiązania między elementami, w przypadku analizy pojedynczego dokumentu, nie jest możliwe łatwe wyznaczenie powiązań między słowami. Zaproponowana w ramach niniejszej pracy idea rankingu słów zakłada, że wykorzystane

zostaną naturalne powiązania pomiędzy nimi, na przykład występowanie słów obok siebie. Jest to podejście o tyle wygodne, że jeżeli słowa występują obok siebie, to zwykle są powiązane treściowo.



Rysunek 3.17: Schemat tworzenia początkowego rankingu słów dla przykładowego tekstu: (a) tekst, (b) usunięcie słów nieistotnych, (c) wyznaczenie powiązań (strzałki) pomiędzy słowami i klas abstrakcji słów (kreski), (d) stworzenie grafu słów, (e) wyznaczenie rankingu.

Wyznaczenie takich powiązań ułatwia fakt, że dokonana została wstępna analiza tekstu i wybrane zostały słowa, które mogą być uznane za nieistotne (ilustracja 3.17 (b) i tabela 3.3). Dzięki temu dokonana została wstępna klasteryzacja dokumentu. Dodatkowy podział jest możliwy poprzez oznaczenie jako zbędne wszelkich znaków interpunkcyjnych. Wszystkie pozostałe grupy słów, które nie są oddzielone słowami zbędnymi i znakami interpunkcyjnymi mogą zostać uznane za słowa potencjalnie kluczowe. Kolejnym krokiem (3.17 (c)) jest określenie zależności pomiędzy słowami, poprzez budowę grafu skierowanego $G = (V, E)$ (V - zbiór wierzchołków, $E = V \times V \times [0, 1]$), w którym każda z krawędzi oznacza zależność i jednocześnie każda zależność może być wartościowana wagą z zakresu $[0, 1]$. Testy pokazały, że najlepszym oszacowaniem zależności jest nadanie wagi 1 wszystkim krawędziom pomiędzy każdym słowem a jego następnikiem oraz

0.3 pomiędzy każdym słowem, a jego poprzednikiem. Podobnie jak w wcześniej przedstawionych algorytmach obliczania odległości pomiędzy słowami, każde wystąpienie słowa oznacza tę samą klasę abstrakcji słowa. Przykładowy wybór powiązań widać na ilustracji 3.17 (d). Następnym krokiem 3.17 (e) jest wyznaczenie wstępnych wag dla każdego słowa $\omega(-)$ zgodnie z rankingiem.

Posiadając wyznaczone w ten sposób wstępne wagi słów można przejść do właściwego algorytmu przedstawionego w poprzednim podrozdziale i dokonać podziału słów kandydujących na kategorie.

Powiązania słów
an ontology, **ontology driven**, **driven similarity**, similarity algorithm, algorithm tech, tech report, report kmi, **kmi maria**, **maria vargas**, **vargas vera**, vera and, and enrico, **enrico motta**, motta an, an ontology, **ontology driven**, driven similarity, **driven similarity**, similarity algorithm, knowledge media, media institute, **institute kmi**, kmi the, the open, open university, university walton, **walton hall**, **hall milton**, **milton keynes**,**keynes mk**, **mk aa**, **aa united**, **aa united**, **united kingdom**, kingdom m, m.vargas, vargas vera, vera open, open.ac, ac.uk, **uk abstract**, abstract.this, this paper, paper presents, presents our, our similarity, similarity algorithm, algorithm between, between relations, relations in, in a, a user, user query, query written, written in, in fol, fol first, first order, order logic, logic and, and ontological, **ontological relations**, relations.our, our similarity, similarity algorithm, algorithm takes, takes two, two graphs, graphs and, and produces, produces a, a mapping, mapping between, between elements, elements of, of the, the two, two graphs, graphs i.e, i.e.graphs, **graphs associated**, graphs associated, associated to, to the, the query, query a, a subsection, subsection of, of ontology, **ontology relevant**

Tablica 3.3: Przykładowy wybór powiązań między słowami. Kolejne pary to kolejne pary słów znalezione w dokumencie. Pary pogrubione zawierają słowa, między którymi zostały wykryte zależności.

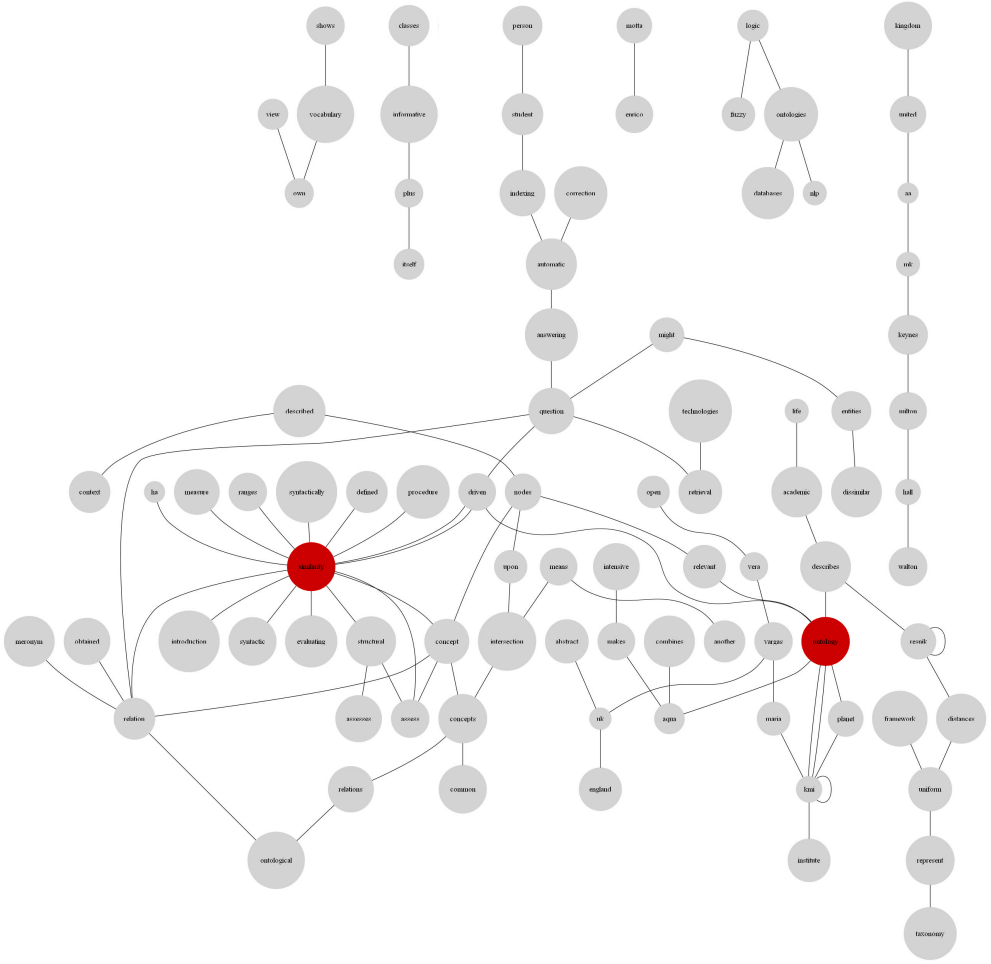
Waga $\omega(y)$ dla każdego słowa może zostać wyznaczona zgodnie ze wzorem:

$$\forall_y \omega(y) = \frac{0.15}{n} + 0.85(\sum_{v \in V} \frac{\omega(v,y)}{n_v})$$

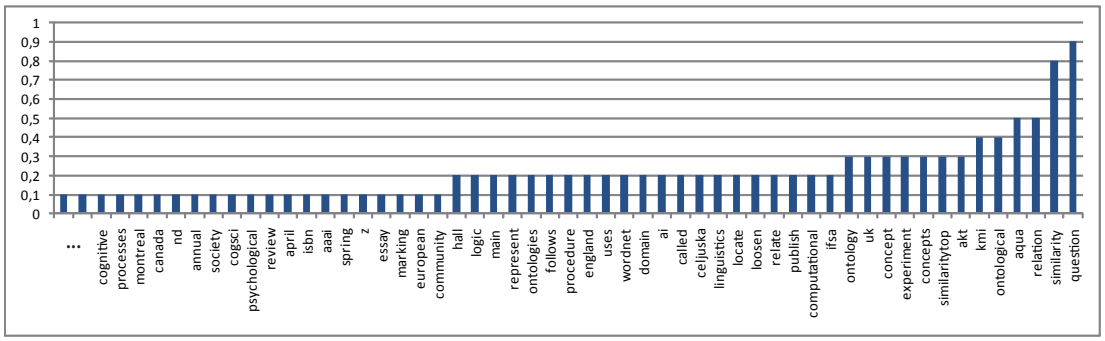
gdzie n oznacza ilość słów znalezionych w dokumencie, $\omega(v,w)$ oznacza wagę krawędzi z v do w a n_v oznacza sumę wag wszystkich krawędzi wychodzących z v. Zgodnie z powyższym waga każdego słowa zależy od wagi słów będących w jego sąsiedztwie.

Powyższe czynności należy wykonywać iteracyjnie. W każdej iteracji wszystkie wagi słów zbiegać będą do rozkładu stacjonarnego stabilizującego układ.

Na ilustracji 3.18 widać graf słów skonstruowany dla pewnego tekstu a na ilustracji 3.19 fragment rankingu słów w tym tekście.



Rysunek 3.18: Fragment grafu słów powiązanych dla przykładowego tekstu.



Rysunek 3.19: Ranking słów wygenerowany za pomocą przedstawionego algorytmu dla przykładowego tekstu.

3.7.3 Podstawy teoretyczne algorytmu

Każdy krok algorytmu, polegający na obliczeniu dla każdego słowa jego wagi może zostać przedstawiony w formie macierzowej. Niech $P \in M(n \times n) = [p_{ij}]_{i,j=1,\ldots,n}$ będzie macierzą kwadratową, gdzie n oznacza ilość słów w dokumencie. Niech $p_{ij} = \frac{\omega(i,j)}{\sum_{k=1..n} \omega(i,k)}$, zakładając, że $p_{ik} = 0$ jeśli słowo i nie występowało bezpośrednio przed słowem j, ani odwrotnie. Niech W_i będzie wektorem wag wszystkich słów w tekście. Oraz niech początkowo każda waga będzie równa 1.

Na tej podstawie zdefiniowana została macierz przejścia $P' = [p'_{ij}] \in M(n \times n)$

$$P' = p\frac{1}{n}\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix} + (1-p)P$$

Macierz przejścia P' spełnia warunki definicji 28.

$$\sum_{j=1..n} p_{ij} = \sum_{j=1..n} \frac{\omega(i,j)}{\sum_{k=1..n}\omega(i,k)} = \frac{\sum_{j=1..n}\omega(i,j)}{\sum_{k=1..n}\omega(i,k)} = 1 \tag{3.22}$$

$$\sum_{j=1..n} p'_{ij} = \sum_{j=1..n} p\frac{1}{n} + (1-p)p_{ij} \tag{3.23}$$

$$= \sum_{j=1..n} p\frac{1}{n} + \sum_{j=1..n}(1-p)\frac{\omega(i,j)}{\sum_{k=1..n}\omega(i,k)} \tag{3.24}$$

$$= p\sum_{j=1..n}\frac{1}{n} + (1-p)\sum_{j=1..n}\frac{\omega(i,j)}{\sum_{k=1..n}\omega(i,k)} \tag{3.25}$$

$$= p + (1-p)\sum_{j=1..n} p_{ij} = p + 1 - p = 1 \tag{3.26}$$

Twierdzenie 11 *Niech $W_i = P'W_{i=1}$. Niech $f : R^n \times M_{n\times n}(R) \to R^n$. Niech $(Z_k) = P'\forall_{k\in N}$ oraz $W_0 = E$. Wówczas łańcuch $W_{i+1} = f(W_i, Z_{i+1})$ definiuje łańcuch Markowa.*

Przedstawione warunki spełniają założenia twierdzenia 10, co implikuje dowód powyższego twierdzenia.

Początkowe wagi słów (stan początkowy łańcucha) został przyjęty jako:

$$W_0 = \begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix}$$

Wówczas każdy krok algorytmu ma postać

$$W_i = P'W_{i-1}$$

i każdy kolejny stan

$$W_i = \begin{bmatrix} \omega_i(v_1) \\ \omega_i(v_2) \\ \vdots \\ \omega_i(v_n) \end{bmatrix}$$

zbiega do rozkładu stacjonarnego W', takiego że

$$W' = P'W'$$

Pozostaje pokazać, że rozkład stacjonarny W' istnieje i że ciąg zmiennych W_n do niego zbiega.

$$W' = lim_{n=0..\infty} W_n$$

Dowód wynika z udowodnionego w Grinstead & Snell (1997) fundamentalnego twierdzenia dla łańcuchów Markowa.

Definicja 30 *Łańcuch Markowa jest nazywany regularnym jeżeli wszystkie elementy pewnej potęgi macierzy przejścia są nieujemne.*

Twierdzenie 12 *Niech P będzie macierzą przejścia dla regularnego łańcucha Markowa. Wówczas ciąg macierzy P^i zbiega wraz z $i \to \infty$ do macierzy granicznej M ze wszystkimi wierszami będącymi wektorami m o tych samych współrzędnych. Jednocześnie m jest wektorem prawdopodobieństwa, to znaczy jego współrzędne są większe od 0 i sumują się do 1.*

Teraz wystarczy zauważyć że macierz P' definiuje regularny łańcuch Markowa.

$$W_i = P'W_{i-1} = P'^i W_0$$

Łańcuch W_i spełnia warunki definicji. Jednocześnie P'^n zbiega do macierzy granicznej M. Biorąc dowolny wiersz m z macierzy M otrzymamy wektor $W' = m$, taki, że $W' = P'W'$, co wynika bezpośrednio z twierdzenia:

Twierdzenie 13 *Niech P będzie macierzą przejścia dla regularnego łańcucha Markowa. Oraz niech*

$$W = lim_{n\to\infty} P^n$$

Niech w będzie wierszem W a c będzie kolumną składającą się z samych 1. Wówczas

1. *$wP = w$ i każdy wektor v taki, że $vP = v$ jest wektorem w pomnożonym przez skalar,*
2. *$Pc = c$ i każda kolumna x taka, że $Px = x$ jest kolumną c pomnożoną przez skalar.*

Pierwszy dowód tego twierdzenia został przedstawiony przez Doeblina (Doeblin (1933)), który sformułował je następująco:

Twierdzenie 14 *Niech P będzie macierzą przejścia dla regularnego łańcucha Markowa. Niech w będzie wektorem stałym tego łańcucha. Wówczas dla dowolnego początkowego wektora prawdopodobieństwa u, $uP^n \to w$ gdy $n \to \infty$.*

Na podstawie powyższych twierdzeń, można wnioskować, że $P'^n W_0 \to W'$ wraz z $n \to \infty$, c.n.d.

3.7.4 Algorytm wyznaczania wag słów

Algorytm 18 ***Algorytm wyznaczania wag słów***

1. *Stwórz macierz $P' \in M(n \times n)$ taką, że $p'_{ij} = 0$ dla każdego i, j*
2. *Dla każdego słowa v występującego w tekście oznacz je numerem $\hat{v} \in \{1, ..., n\}$, zgodnie z wcześniejszą definicją $\hat{v} = \lambda(\xi(v))$.*
3. *Dla każdych dwóch słów v, w występujących w tekście obok siebie*
   (a) *Jeśli v i w są rozdzielone znakami interpunkcyjnymi, przejdź do następnej pary słów,*
   (b) *Jeśli v lub w zostało oznaczone jako słowo nieistotne, przejdź do następnej pary słów,*
   (c) *W przeciwnym wypadku dodaj do grafu G krawędź $k = (v, w, 1)$ i $k' = (w, v, 0.25)$*
      - $p'_{\hat{v}\hat{w}} = p'_{\hat{v}\hat{w}} + 1$
      - $p'_{\hat{w}\hat{v}} = p'_{\hat{w}\hat{v}} + 0.25$
4. *Stwórz macierz $P' \in M(n \times n)$ taką, że $p_{ij} = \frac{p'_{ij}}{\sum_{j=1,..n} p'_{ij}}$*
5. *Stwórz macierz przejścia P'' taką, że*

$$P'' = p\frac{1}{n}\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix} + (1-p)P$$

6. *Niech* $W_0 = \vec{1}$.

7. *Powtarzaj, aż dla pewnego* i *i pewnego ustalonego* ϵ *spełniony będzie warunek* $|w_i - w_{i-1}| < \epsilon$

$$W_i = P'' W_{i-1}$$

Ponieważ zbieżność $P''^n W_0 \to W'$ postępuje w tempie geometrycznym, można przyjąć za W' pewne W_k takie, że $|W_k - W_{k-1}| < \bar{\epsilon}$ dla pewnego małego $\epsilon > 0$ tzn.:

$$|W_k - W_{k-1}| = \left| \begin{bmatrix} \omega_k(v_1) \\ \omega_k(v_2) \\ \vdots \\ \omega_k(v_n) \end{bmatrix} - \begin{bmatrix} \omega_{k-1}(v_1) \\ \omega_{k-1}(v_2) \\ \vdots \\ \omega_{k-1}(v_n) \end{bmatrix} \right| = \begin{bmatrix} |\omega_k(v_1) - \omega_{k-1}(v_1)| \\ |\omega_k(v_2) - \omega_{k-1}(v_2)| \\ \vdots \\ |\omega_k(v_n) - \omega_{k-1}(v_2)| \end{bmatrix}$$

oraz:

$$\forall_{s=1..n} \, |\omega_k(v_s) - \omega_{k-1}(v_s)| < \epsilon$$

Algorytm 19 ***Algorytm kategoryzacji***

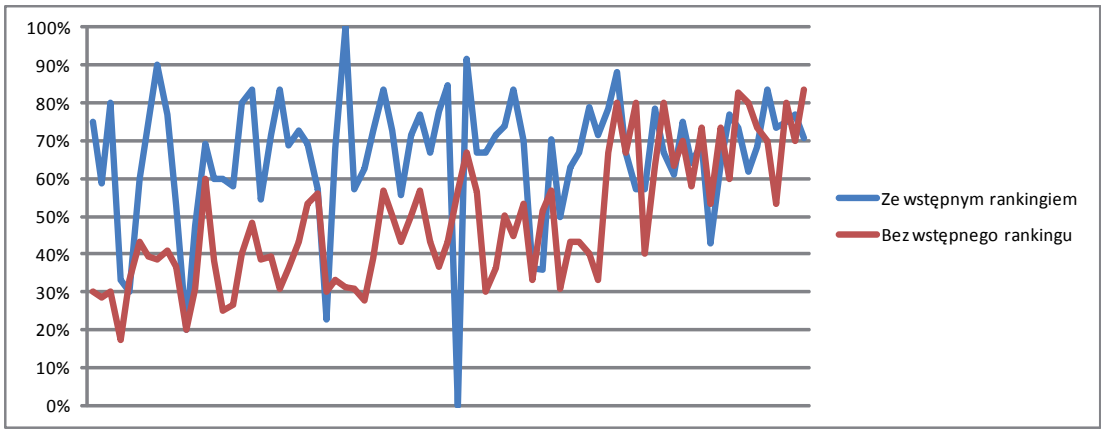
1. Do każdego słowa przypisz jego wagę $\omega(w) = W_i[\hat{w}]$
2. Wykonuj kroki algorytmu kategoryzacji słów 17 aż zostanie osiągnięta stabilność sieci.
3. Wybierz słowa kluczowe zgodnie z zasadami przyjętymi w 17.

3.7.5 Rezultaty

Algorytm został przetestowany na takim samym zbiorze danych jak poprzednie. Okazuje się, że jego efektywność znacząco wzrosła, co jest widoczne na wykresie 3.20 prezentującym porównanie kategoryzacji bez wstępnego wartościowania wag słów i z tym wartościowaniem). Jak widać w większości przypadków ten sam tekst analizowany z użyciem wstępnego wartościowania wag dał znacząco lepsze wyniki - więcej spośród zaproponowanych słów kluczowych było tymi słowami w rzeczywistości, co zostało sprawdzone empirycznie. Należy tu zwrócić uwagę, że algorytm nie wyznaczył większej ilości słów kluczowych, natomiast znacząco zawęził zbiór słów, które nie były kluczowe. Przykładowe różnice w wyborze słów kluczowych widać w tabeli 3.4.

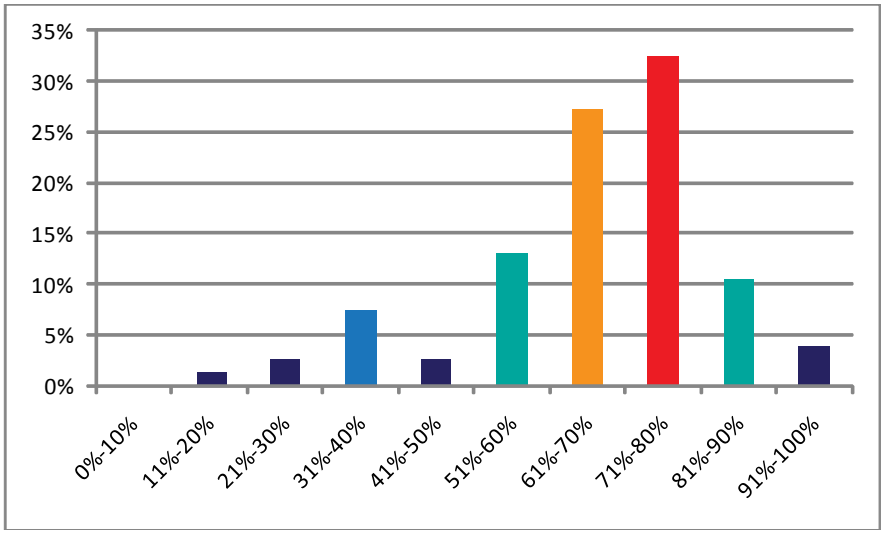
Bez wstępnego rankingu	Ze wstępnym rankingiem
component, **analyses**, deep, configuration, interface, described, ne, **generated**, qa, **linguistic**, names, standard, item, xsl, items, **extraction**, **grammar**, hpsg, np, id, named, efficiency, evaluation, **lexical**, attributes, **semantics**, elements, **german**, infl, main	**parsing**, **german**, **sentence**, **annotation**, id, **architecture**, **xsl**, **grammar**, **parser**, **rmrs**, **linguistic**, component, pos, shallow, nlp, whiteboard, deep, string, hybrid, en, **structures**, thesis

Tablica 3.4: Przykładowy wybór słów kluczowych z wykorzystaniem algorytmu wyznaczającego wstępny ranking słów. Pogrubione słowa oznaczają rzeczywiste (wybrane empirycznie) słowa kluczowe znajdujące się wśród wyników zaproponowanych przez algorytmy.

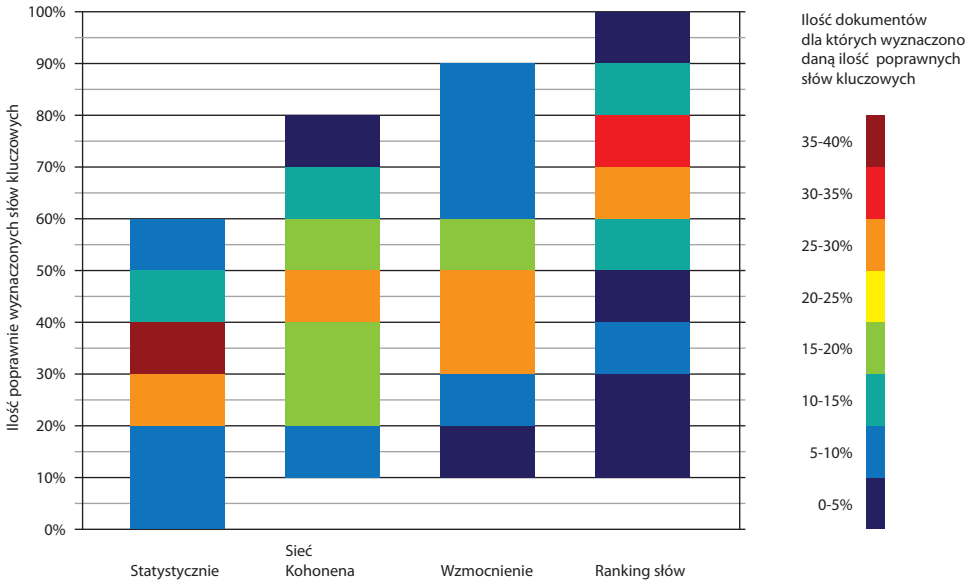


Rysunek 3.20: Porównanie efektywności wyznaczania słów kluczowych poprzez kategoryzację ze wzmocnieniem (kolor czerwony) i tego samego algorytmu poprzedzonego generowaniem wstępnego rankingu słów (kolor niebieski). Oś X oznacza kolejne dokumenty z testowanego zbioru, oś Y skuteczność sprawdzanych algorytmów.

Na ilustracji 3.22 widać porównanie wszystkich przedstawionych w tej pracy podejść do wyznaczania słów kluczowych. Poprawiając algorytmy kategoryzacyjne, wykorzystując algorytmy genetyczne oraz aproksymacyjne oraz wyznaczając wstępne rankingi słów zależne od ich położenia względem siebie, udało się uzyskać wyniki pozwalające na dobre przybliżenie treści dokumentu pisanego w języku naturalnym poprzez trafny wybór słów opisujących jego zawartość.



Rysunek 3.21: Efektywność algorytmu kategoryzacji poprzedzonego wstępnym obliczaniem rankingu słów. Oś X pokazuje skuteczność wyboru słów kluczowych (tzn. względną ilość poprawnie zaproponowanych słów kluczowych), oś Y pokazuje ilość dokumentów, dla których taka skuteczność została osiągnięta.



Rysunek 3.22: Porównanie efektywności wszystkich przedstawionych algorytmów. Oś Y oznacza osiągniętą skuteczność a kolor słupka oznacza względną ilość dokumentów, dla których w konkretnym algorytmie (oś X) taka skuteczność została osiągnięta (im ciemniejszy kolor, tym więcej dokumentów w danej grupie).

Przeprowadzone testy pokazały, że najlepszym algorytmem wyznaczania słów

kluczowych jest algorytm dwufazowy oparty o Łańcuchy Markowa wyznaczające wstępny ranking słów a następnie algorytmy kategoryzacyjne oparte o sieci neuronowe ze wzmocnieniem. Algorytm ten jest oryginalnym wynikiem osiągniętym w ramach niniejszej pracy.

3.8 Zależność wyników od ilości dokumentów zgromadzonych w repozytorium

Zależoność poprawności wyników od ilości dokumentów zgromadzonych w repozytorium wynika z następujących spostrzeżeń:

- Większa liczba dokumentów przekłada się na lepszą statystykę słów nieistotnych,
- Test dla wzmocnienia może lepiej przetestować poprawność testowanego słowa.

Przeprowadzone testy pokazują, że poprawność wyboru słów kluczowych zależy od ilości dokumentów w repozytorium.

Tabela 3.5 prezentuje wyniki wyznaczania słów kluczowych dla przykładowych dokumentów, podczas pracy z różnej wielkości repozytoriami danych

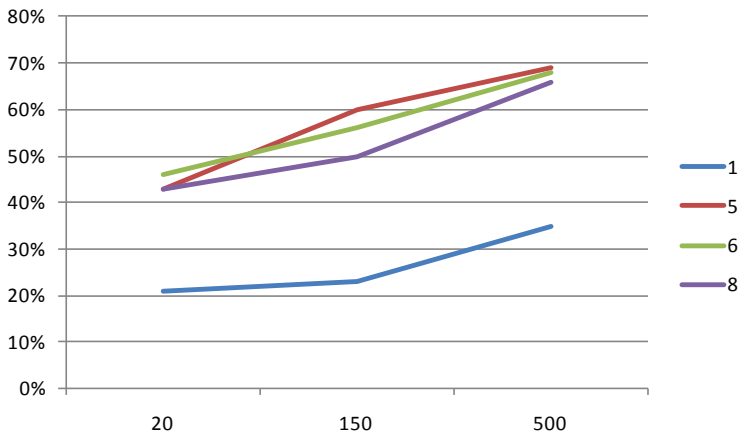
1. 20 dokumentów
2. 200 dokumentów
3. 500 dokumentów

Id	Dokument	20		200		500	
1	A mutually beneficial integration of data mining and information extraction	ax, server, slot, extracted, rapier, artificial, intelligence, fillers, pp, austin, tx, this, predict, improving	21%	ax, slot, mining, extracted, server, intelligence, recall, discotex, kdd, rapier, different, software, tx	23%	additional, examples, recall, job, rapier, database, extracted, artificial, extraction, edu, mutually, typically, much	35%
5	Data mining for technical operation of companies...	quantization, controllers, mining, data, for, study, cg, case, mostly, companies, technical, operation, telecomunications, on, algorithms, remarks	43%	mining, cg, quantization, of, real, data, for, technical, operation, telecomunications	60%	company, calls, cell, traffic, team, warsaw, business, cg, attributes, computation, engineering, quite, remarks	69%

Ciąg dalszy na następnej stronie

Id	Dokument	20		200		500	
6	Discovery of important keywords in the cyberspace	discovery, phrase, keywords, entropy, seamen, optimized, arikawa, optimization, merge, text, data, important, from	46%	discovery, minimization, phrase, keywords, optimized, cyberspace, hiroki, texts, optimization, of, strike, seamen, text, data, framework	56%	optimized, target, keywords, called, proximity, discovery, fujino, experiments, transaction, contain, iranian, trarily, partic, computing, arbi	68%
8	Knowledge Discovery in the internet	query, word, this, hypertext, latent, classical, growing, qi, semantic, hyperlink, retrieval, nodes, internet, unfortunately	42%	query, retrieval, hypertext, hyperlink, nodes, classical	50%	hyperlink, links, ranking, matrix, almost, jaguar	66%

Tablica 3.5: Słowa kluczowe genrowane dla przykładowych dokumentów zawartych w repozytoriach zawierających20, 200 i 500 dokumentów. Przy każdej grupie słów kluczowych umieszczona jest ilość poprawnie wybranych słów



Rysunek 3.23: Poprawa jakości generowanych wyników wyznaczania słów kluczowych dla przykładowego zbioru tekstów w miarę zwiększania się ilości dokumentów w repozytorium.

Jak widać wraz ze wzrostem liczby dokumentów w repozytorium jakość wyboru słów kluczowych zwiększa się (ilustruje to wykres 3.23).

3.8.1 Inne metody wyznaczania słów kluczowych

W ostatnim czasie rozwijanych jest wiele metod wyznaczania słów kluczowych w dokumentach. Są to między innymi metody opierające się na wyznaczaniu wag słów zgodnie z algorytmem TFIDF oraz na „Słowo-sieci" (ang. „*WordNet*"[2]) (jak zaproponowany w Mihalcea (2005) i Sinha & Mihalcea (2007)) w celu określenia znaczenia poszczególnych słów i na tej podstawie wybrania słów kluczowych w dokumencie. O Oczywiście tutaj znowu można rozróżnić algorytmy nadzorowane i nienadzorowane. Algorytmy takie są rozwijane i badane między innymi w ramach projektu „Graph-based_NLP" [3]. Algorytm wyznaczający słowa kluczowe został nazwny przez tworzącą go grupę *TextRank*. Do porównania wybrano ten algorytm, gdyż ma on wysoką skuteczność wyznaczania słów kluczowych, a jednocześnie opiera się na podobnej, wywodzącej się z PageRank idei. Jak się okazało zaproponowany w ramach tej pracy algorytm może z powodzeniem konkurować z innymi czołowymi algorytmami.

Założenia algorytmu TextRank

Algorytm prezentowany przez wspomnianą grupę opiera się na następujących założeniach:

1. Słowa są ważone za pomocą „Słowo-sieci" i na jej podstawie filtrowane.
2. Do słów przydzielane są etykiety (pojedyńczemu słowu może być przydzielona więcej niż jedna etykieta).
3. Iteracyjny algorytm wykorzystujący losowe poruszanie się po grafie poprawia wagi etykiet.
4. Jako słowa kluczowe wybrane są najwyżej punktowane słowa.

Porównanie

Tabela 3.6 prezentuje przykładowe wyniki wyznaczania słów kluczowych dla przykładowych dokumentów, dla algorytmów zaproponowanych w ramach niniejszej pracy (*WordRank*) i algorytmu *TextRank*. Jak pokazują testy, algorytm WordRank pokazuje mniej propozycji słów kluczowych, jednak procentowy udział tych poprawnie wyznaczonych jest większy niż w przypadku algorytmu TextRank. Wynika to z faktu, że w zgromadzonym repozytorium wiele dokumentów związanych było z tą samą tematyką i część słów występujących tu powszechnie została

[2] http://wordnet.princeton.edu/
[3] http://lit.csci.unt.edu/index.php/Graph-based_NLP

odrzucona. Z drugiej strony te same słowa zostały uznane za kluczowe przez algorytm TextRank, gdyż ten działa w kontekście szerszego zbioru danych. Takie zachowanie WordRank jest właściwe i dobrze sprawdza się, przy analizie danych w kontekście zamkniętego repozytorium.

Id	Dokument	WordRank		TextRank	
1	A mutually beneficial integration of data mining and information extraction	additional, examples, recall, job, rapier, database, extracted, artificial, extraction, edu, mutually, typically, much	35%	rules, recall, filler, database, ie, proceedings, artificial, discotex, intelligence, slots, precision, kdd, information, example, document, prediction, f, measure, system, data, figure, jobs, seventeenth, national, conference, unlabeled, examples, rapier, paper, corpus, text, results, mining, information, extraction, austin, tx, training, form, knowledge, performance, experience, application	38%

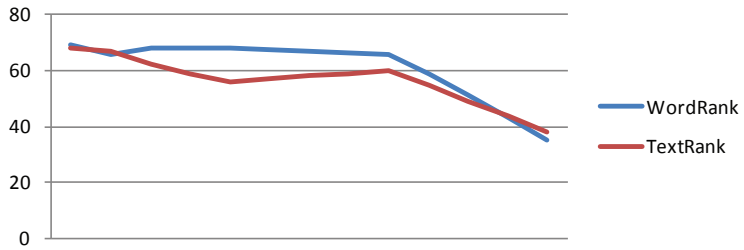
Ciąg dalszy na następnej stronie

Id	Dokument	WordRank		TextRank	
5	Data mining for technical operation of companies...	company, calls, cell, traffic, team, warsaw, business, cg, attributes, computation, engineering, quite, remarks	69%	cells, data , mining, company, problems, rule, network, errors, project, section, information, second , calls, methods, other , hand, association , rules, analysis, team, pixels, areas, results, base , stations, experiments, amount, seconds, behavior, set, marketing, approach, research, knowledge, experts, number, parameters, time, event, calls, multiple, regression, faulty , cell, era , experts, warsaw , university, attribute , valuerange, knowledge , discovery, algorithms, processes, subscribers, activity, quality, opportunity, technology, example, clustering, landuse	25%

Ciąg dalszy na następnej stronie

Id	Dokument	WordRank		TextRank	
6	Discovery of important keywords in the cyberspace	optimized, target, keywords, called, proximity, discovery, fujino, experiments, transaction, contain, iranian, trarily, partic, computing, arbi	68%	patterns, phrases, table, target, document, articles, set, web, pages, optimal, pattern, class, keywords, user, data, optimized, pattern, discovery, number, algorithm, base, sets, strike, time, union, collection, b, experiments, problem, t, honda, large, text, database original, target, phrase, pattern, text, mining, information, entropy, paper, proxim, prelude, arimura, control, pages, text, c, reuter, frequent, pattern, problems, priority, areas, discovery, science	56%
8	Knowledge Discovery in the internet	hyperlink, links, ranking, matrix, almost, jaguar	66%	page, documents, links, examples, methods, user, query, internet, information, words, set, groups, system, number, algorithms, form, hyperlinks, node, www, web, graph, webpage, index, vector, lsa	60%

Tablica 3.6: Porównanie algorytmów generowania słów kluczowych zaprezentowanych w niniejszej pracy oraz algorytmu TextRank



Rysunek 3.24: Różnice między wynikami algorytmów WordRank i TextRank.

Rozdział 4

Podsumowanie

W epoce łatwego dostępu do powszechnie obecnej informacji, odbiorcy skazani są na kontakt z dużymi ilościami danych. W większości przypadków, wśród wielu informacji bezwartościowych dla odbiorcy, znajdują się nieliczne posiadające wartość. Tyczy się to szczególnie wyników zwracanych przez wszelkiego rodzaju wyszukiwarki. Odbiorca musi więc przebrnąć przez gąszcz danych zanim, znajdzie to czego potrzebuje. W szczególności dotyczy to wszelkiego rodzaju tekstów naukowych, których znaczne ilości można znaleźć w internecie. Wybranie tych, które dotyczą żądanej dziedziny jest wysoce pracochłonne.

Rośnie więc zapotrzebowanie na metody pozwalające zminimalizować potrzebę aktywnego filtrowania danych przez odbiorcę. Szczególnie istotne są tu dane zawarte w bazach nie posiadających ściśle zdefiniowanej struktury, czyli na przykład teksty w języku naturalnym. Analiza takich tekstów jest dziedziną rozwijającą się bardzo dynamicznie. Jest uznawana za najważniejszy obszar *data miningu*. Dlatego powstaje wiele metod wykrywania istotnych danych w tekstach oraz wiele metod porównywania tekstów ze sobą.

W ramach niniejszej pracy, w oparciu o znane modele matematyczne, zaprojektowano nowe metody wyszukiwania słów kluczowych w dokumentach naukowych i porównywania dokumentów a ich efektywność została zweryfikowana. Specyfika tych metod zapewnia szerokie pole ich zastosowań, włączając w to na przykład analizę dokumentów medycznych.

4.1 Nowe metody porównywania tekstów

Wynikiem niniejszej pracy jest nowa metoda porównywania dokumentów naukowych i ich kategoryzacji, bazująca na łatwo wykrywalnych zależnościach (odległościach bazujących na częstości słów, n-gramów i złożoności Kołmogorowa) pomiędzy nimi oraz na dynamicznych sieciach neuronowych typu Kohonena. Metoda

ta pozwala efektywnie wyznaczyć zbiory dokumentów, które są podobne do siebie treściowo i strukturalnie. W ramach niniejszej pracy zmodyfikowane zostały również algorytmy sieci typu Kohonena, w celu zapewnienia lepszej wydajności i skalowalności. Wyniki działania zaproponowanej metody zostały potwierdzone w testach przeprowadzonych na przykładowych zbiorach dokumentów naukowych.

4.2 Nowe metody wyznaczania słów kluczowych w tekstach

W ramach niniejszej pracy opracowana została nowa metoda pozwalająca na wyznaczanie słów kluczowych w dokumentach tekstowych. Opiera się ona na zaproponowanej w tej pracy metodzie obliczania odległości pomiędzy klasami abstrakcji słów. Odległość ta jest wykorzystywana do przeprowadzenia podziału słów na kategorie za pomocą sieci neuronowych typu Kohonena. Sieci te zostały rozszerzone w tej pracy o możliwość wzmacniania wyniku za pomocą wartościowania słów opartego o wiedzę zdobytą podczas porównywania tekstów. Słowa są również wstępnie wartościowane z wykorzystaniem łańcuchów Markowa opierających się na prostym założeniu o powiązaniach słów w tekście. Uzyskane kategorie są wartościowane i na ich podstawie wyznaczana jest wynikowa lista słów kluczowych dla danego tekstu. Testy przeprowadzone na zbiorach dokumentów naukowych pokazały, że zaproponowana metoda osiąga bardzo dobre wyniki.

Kluczowym elementem prezentowanych rozważań było wykorzystanie danych zdobytych podczas analizy i porównywania pozostałych tekstów. Okazuje się, że im więcej wiedzy algorytm zdobył (tzn. im więcej dokumentów przeanalizował), tym dokładniej może wyznaczać słowa kluczowe dla kolejnych dokumentów.

Dzięki tworzeniu grup lokalnie bliskich słów, algorytm potrafi zidentyfikować i umieścić w jednej kategorii wyrażenia kluczowe składające się z więcej niż jednego słowa.

4.3 Uzyskane wyniki

Zaproponowane algorytmy zostały przetestowane na różnego rodzaju danych przedstawionych w formie tekstu pisanego. Wyniki pokazały, że metody kategoryzacji tekstów mogą odróżnić od siebie prozę, poezję, kody źródłowe czy artykuły naukowe. Pokazano również, że potrafią w dobry sposób znaleźć podobieństwa w obrębie jednego typu tekstów. I tak skuteczność wskazania dokumentów naukowych, które są podobne do zadanego, sięgała średnio 75%. Jest to bardzo dobry wynik, zważywszy że testowy system pracował na niejednorodnym zbiorze tekstów.

Dzięki tak dobremu dopasowaniu dokumentów podobnych, również bardzo dobrymi wynikami charakteryzuje się metoda wyznaczania słów kluczowych. Podczas generowania słów, potencjalnie najlepsze słowa sprawdzane były pod kontem występowania w innych, bliskich dokumentach z tej samej kategorii co analizowany i na tej podstawie ich wartość była korygowana. Rozwiązanie to pozwoliło uzyskać wyniki sięgające nawet 100% dobrego dopasowania słów wśród pierwszych 20 proponowanych wyników, przy jednoczesnym ograniczeniu ilości wszystkich proponowanych słów kluczowych. Zapewniło również możliwość wskazania jako słów kluczowych, słów rzadko występujących w danym tekście.

4.4 Wykorzystanie algorytmów

Uzyskane w ramach niniejszej pracy wyniki pozwoliły na skonstruowanie systemu, który na podstawie zadanego zestawu dokumentów naukowych wyszukuje w internecie (korzystając z dostępnych wyszukiwarek) dokumenty potencjalnie podobne do zadanych, a następnie filtruje je, zwracając listę dokumentów, będącą tematycznym uzupełnieniem zestawu zadanego. System znajduje również trafnie słowa kluczowe dla każdego posiadanego dokumentu. Stworzone oprogramowanie jest swego rodzaju systemem pracy grupowej, oferującym repozytorium dokumentów, w obrębie którego użytkownicy mogą przechowywać swoje zbiory dokumentów naukowych, współdzielić je i jednocześnie korzystać z wyników oferowanych przez zaproponowane w ramach niniejszej pracy algorytmy.

4.5 Wyniki

Uzyskane wyniki zostały opublikowane w pracach:

- „*Scientific Documents Management System. Application of Kohonen Neural Networks with Reinforcement in Keywords Extraction*" (Zyglarski & Bała (2009)),
- „*Neural Networks Aided Automatic Keywords Selection*" (Zyglarski & Bała (2010b)),
- „*Web Services Based Scientific Article Manager*" (Zyglarski *et al.* (2008)),
- „*Document Management System based on Neural Networks*" (Zyglarski (2009)).

Przyotowane zostały również prace:

- „*Genetic Algorithms and Dynamic Neural Networks in Data Categorization*" (Zyglarski (2010))[1],

[1]Praca przyjęta do publikacji na SMI 2010.

- „*Keywords Extraction. Selecting Keywords in Natural Language Texts with Markov Chains and Neural Networks*" (Zyglarski & Bała (2010a))[2].

Wyniki badań prezentowane były na konferencjach:

- Information Systems Architecture and Technology (ISAT) 2008,
- Sejmik Młodych Informatyków 2009,
- IC3K Knowledge Management and Information Sharing Conference 2009

oraz będą prezentowane na konferencjach:

- Sejmik Młodych Informatyków 2010.

4.6 Dalsze badania

Przedstawione badania stanowią doskonały punkt wyjścia do dalszego rozwijania metod analizy tekstów, a prezentowane podejście otwiera drogę do szerokiego spektrum zastosowań zaprojektowanych algorytmów.

Wśród zagadnień, które powinny zostać poruszone w dalszych rozważaniach, znajdują się:

- dalsze modyfikacje algorytmów kategoryzacji poprawiające ich wydajność,
- wykorzystanie przedstawionych algorytmów do kategoryzacji i analizy innych typów danych,
- wykorzystanie algorytmów do kategoryzacji danych posiadających zdefiniowaną strukturę, z użyciem innych typów odległości pomiędzy danymi,
- użycie zgromadzonych danych do wygenerowania semantyk (Shadbolt *et al.* (2006)) dla dokumentów znajdujących się w internecie,
- budowa sieci wyszukiwawczej i własnej wyszukiwarki korzystającej z przedstawionych algorytmów do porównywania i kategoryzacji danych,
- usprawnienie przedstawionego algorytmu wyznaczania słów kluczowych poprzez uzupełnienie go o dokładną analizę gramatyczną tekstu.

Kierunek prowadzonych w ramach tej pracy badań pozwala jednakże na niemal nieograniczony wybór dalszego kierunku ich rozwoju, co jest spowodowane ogromną popularnością i zapotrzebowaniem na tego typu rozwiązania.

[2]Praca zgłoszona do publikacji.

Dodatek A

Struktura aplikacji

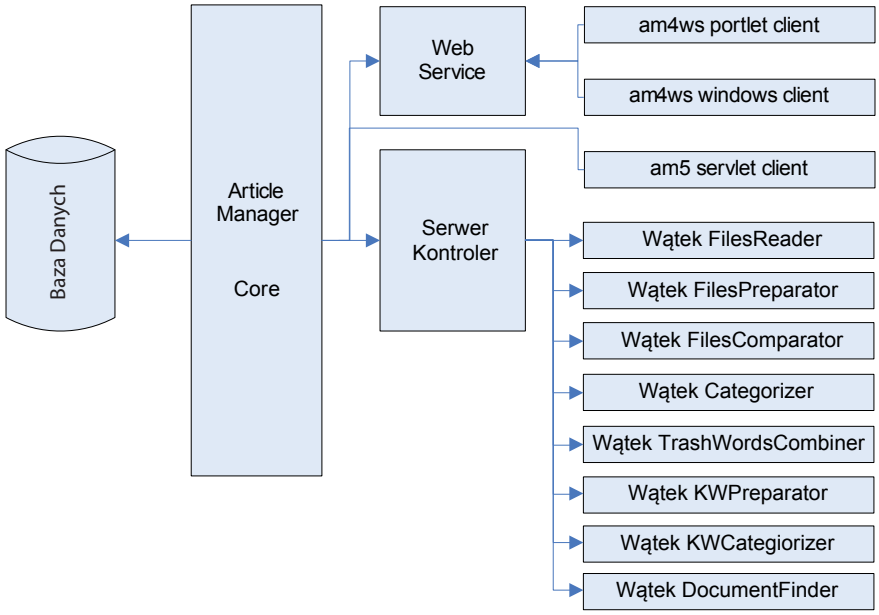
Przedstawione w niniejszej pracy algorytmy zostały wykorzystane do zaprojektowania systemu pracy analizującego dodane do repozytorium dokumenty. System ten nazwano „Article Manager". W ramach projektowania tego oprogramowania wykorzystano Gimbel *et al.* (1999) (o języku Java w *data mining*), Holten (2006)(o sposobach wizualizacji kategorii), McLaughlin (2000) (XML), Eckel (2005) (Java). Oprogramowanie zostało stworzone w nowoczesnych językach oprogramowania:

1. Java (Sun) - serwer i klienci sieciowi;
2. Visual C# (Microsoft) - klient systemu Microsoft Windows.

Główny język programowania (Java) został wybrany ze względu na szeroką gamę dostępnych otwartych dodatków, umożliwiających między innymi pracę z plikami różnego typu, łatwe tworzenie archiwów i kompresję. Jest on również powszechnie stosowany w rozwiązaniach serwerowych i sieciowych, co zaowocowało szeroką gamą dostępnych serwerów aplikacji i usprawniło implementację poszczególnych elementów systemu.

System został podzielony na cztery podsystemy, widoczne na ilustracji A.1:

1. moduł bazodanowy;
2. serwer centralny - składającego się z serwletu komunikacyjnego i webserwisu;
3. serwer analizy dokumentów - podzielony na wiele wątków, które mogą działać jednocześnie na wielu maszynach, dzięki temu przyspieszając znacząco działanie całego projektu;
4. interfejsy klienckie - wśród nich serwlet, portlet i klient natywny.



Rysunek A.1: Struktura systemu.

A.1 Interfejsy użytkownika

Zaprojektowano trzy interfejsy klienckie, które umożliwiają użytkownikowi dostęp do systemu. Dostęp bezpośredni możliwy jest poprzez wbudowany serwlet (Java, specyfikacja JSR-154, działający na serwerze Tomcat 6.0). Dodatkowo wbudowany webserwis (również działający na serwerze Tomcat 6.0) udostępnia funkcjonalności dla klienta instalowanego w dowolnym portalu internetowym (portlet dla portalu Liferay 4, specyfikacja JSR-168) bądź klienta natywnego dla systemu Windows (klient w Visual C# 2005). Dzięki zastosowanej technologii webserwisu, możliwe jest proste dodanie aplikacji klienckich działających w nowych środowiskach.

A.1.1 Dodatkowe interfejsy

W ramach pracy zaprojektowano podstawowe funkcjonalności i interfejsy systemu. Kolejne prace nad interfejsami użytkownik mógłby skupić się na stworzeniu klienta, który w przezroczysty sposób integrowałby się z istniejącymi menadżerami plików (np Windows Explorer, Konqueror) lub wręcz systemem plików i umożliwiałby tym samym jeszcze bardziej intuicyjną obsługę.

A.2 Cykl życia systemu

Ilustracja A.2 przedstawia cykl życia systemu podczas analizy pojedynczego dokumentu dodanego do repozytorium. Jak widać kategoryzacja dokumentów i analiza słów kluczowych dokumentu mogą przebiegać równolegle, przy czym ten drugi proces będzie mógł skorzystać podczas poprawiania wartościowania dokumentów tylko z danych już wyznaczonych przez proces pierwszy.

A.3 Repozytorium

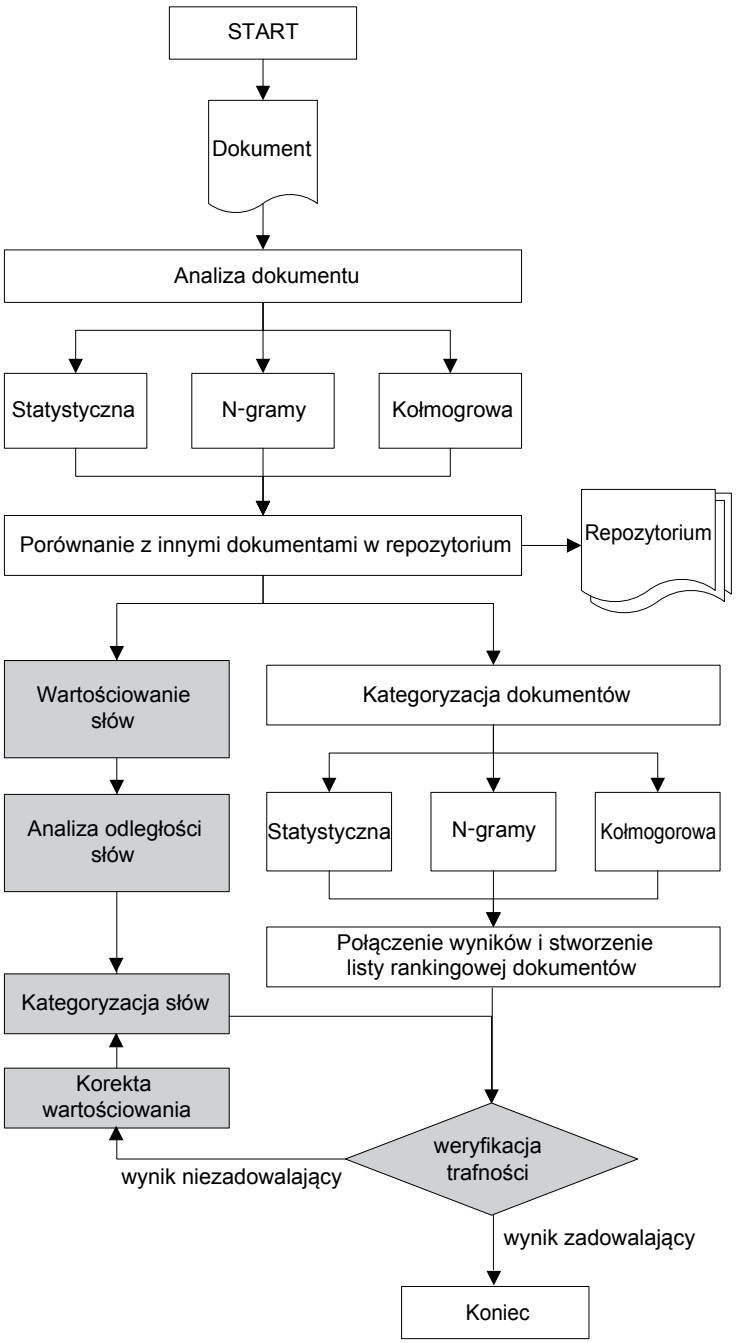
Repozytorium dokumentów zaprojektowano w sposób umożliwiający szczegółowe ustawienia praw dostępu dla poszczególnych użytkowników. Każdy użytkownik posiada własny wirtualny folder, do którego dostęp ma tylko on. Dla każdego dokumentu można ustawić szczegółowe prawa dostępu dla innego użytkownika bądź wszystkich użytkowników. Podczas pracy z dokumentem użytkownik otrzymuje propozycje przejrzenia wszystkich związanych z nim dokumentów z repozytorium z uwzględnieniem informacji o dokumentach niedostępnych i z możliwością zapytania ich właściciela o udostępnienie tych dokumentów do wglądu. Specyficznym typem dokumentów są dokumenty pobrane z internetu. Są one uzyskane z wyników wyszukiwania za pomocą wyszukiwarki Google[1]. Wyniki te są poddawane takiej samej analizie jak w przypadku samodzielnie dodanych do repozytorium dokumentów.

W związku z modularną budową repozytorium możliwe jest włączenie do systemu dodatkowych wtyczek i skorzystanie z wszelkiego rodzaju formatów plików (opracowane zostały metody odczytu plików PDF (korzystające z technologii PDFBox[2]) i plików tekstowych.

A.3.1 Automatyczne pozyskiwanie dokumentów z Google

Wszystkie przytoczone w niniejszej pracy testy opierają się na repozytorium inicjalizującym, zawierającym pewną liczbę inicjalizujących dokumentów, powiększoną o dokumenty pobrane automatycznie z internetu. Pobieranie takie odbywa się na zasadzie zadawania zapytań do google, a następnie pobierania adekwatnych, zwróconych przez tę wyszukiwarkę wyników. Wyszukiwanie odbywa się przy wykorzystaniu Google Web API [3] zostało stworzone w celu uporządkowania dostępu programistów do zasobów tej wyszukiwarki. Dzięki temu przeszukiwania

[1] http://www.google.com
[2] http://www.pdfbox.apache.org
[3] http://www.google.com/apis/



Rysunek A.2: Cykl życia systemu podczas analizy pojedynczego dokumentu.

można dokonać bezpośrednio z kodu programu. W celu zapewnienia szerokiej dostępności Google wykorzystuje technologie SOAP i WSDL , dzięki czemu można używać dowolnego języka programowania. Udostępniony przez firmę developer's kit, zawiera przykłady wykorzystania API w Microsoft Visual Studio oraz Javie. W tym drugim przypadku w paczce jest udostępniania jest również biblioteka **googleapi.jar**[4].

Zapytania składały się z doraźnie wyznaczonych słów kluczowych tekstów zgromadzonych dotychczas w bazie i ograniczone były do wyszukiwania plików **.pdf* w obrębie danych reprezentowanych przez wyszukiwarkę Google Scholar[5]. Kolejnym krokiem było odrzucenie dokumentów, z których nie mógł zostać wyodrębniony czysty tekst. Jakkolwiek w trakcie użytkowania systemu takie filtrowanie wystarczyło, to w celu przeprowadzenia testów prezentowanych w niniejszej pracy dokumenty zostały empirycznie sprawdzone pod kątem bycia dokumentami naukowymi.

A.3.2 Dokumenty naukowe

Dokumenty naukowe zebrane w repozytorium spełniały następujące warunki:

1. Ich dziedziną była informatyka, matematyka, fizyka, bioinformatyka, mechanika.

2. Język wszystkich dokumentów to język angielski.

3. Wszystkie dokumenty były pracami naukowymi publikowanymi w czasopismach.

4. Długość poszczególnych dokumentów wahała się od 6 do 15 stron A4.

A.3.3 Ilość dokumentów w repozytorium

W trakcie testowania algorytmów ilość dokumentów w repozytorium wahała się od kilkunastu do 1000 dokumentów. Ilość ta nie była przekraczana ze względu na potrzebę empirycznego ustalenia słów kluczowych dla tych tekstów i porównania tego wyboru ze słowami generowanymi automatczynie przez tworzone oprogramowanie. Procedura ta wymagała samodzielnej analizy tekstów, co jest operacją wielce czasochłonną. Z tego względu ilość dokumentów została ograniczona do maksymalnego poziomu 1000 dokumentów, a w wielu testach wynosiła jeszcze

[4]Oprogramowanie to nie jest już dostępne dla nowych użytkowników. Odpowiedni klucz oprogramowania oraz licencję uzyskano w roku 2006 podczas tworzenia aplikacji do pracy magisterskiej

[5]http://scholar.google.com/

mniej. Okazało się, że wyniki algorytmów są zadowalające, z uwzględnieniem zależności pomiędzy ich trafnością a rozmiarem repozytorium.

A.4 Baza danych

W pracy wykorzystano system bazodanowy PostgreSQL[6], który po przeprowadzonych testach okazał się najlepiej skalowalnym i wydajnym a dodatkowo otwartym systemem bazodanowym. Zaprojektowana baza danych zawiera 18 tabel związanych z:

1. strukturą repozytorium i dostępem do plików (FILES, FILESPERMISSIONS, VIRTUALFOLDERS, VIRTUALFOLDERPERMISSIONS),
2. analizą powiązań plików (DISTANCES, CATEGORIES, CAT2),
3. analizą słów kluczowych (KEYWORDS, KWS, KEYWORDSWR, FILESDETAILS),
4. elementami społecznościowymi (BOOKMARKS, NOTES, USERS, GROUPS, GROUPMEMBERSHIP),
5. elementami konfiguracyjnymi (SESSIONS, SETTINGS, TYPES).

[6] http://www.postgresql.org/

Dodatek B

Ewaluacja systemu

Zaimplementowany system przetestowano na wielu przykładowych zbiorach dokumentów. Przeprowadzonych zostało wiele reprezentacyjnych testów. W niniejszym dodatku zostały zaprezentowane dwa z nich. Pierwszy z nich dotyczył analizy dokumentów naukowych poświęconych pewnej tematyce.

B.1 Dokumenty naukowe

Zbiór dokumentów inicjujących składał się z dwudziestu krótkich artykułów poświęconych analizie danych, kopalniom danych itp. System został uruchomiony na testowej maszynie z systemem Windows i procesorem Intel Core 2 Duo 2.2 GHz. Wszystkie wątki uruchamiane były w trybie sekwencyjnym.

Testy przeprowadzone w tym dodatku są niezależne od testów przytoczonych w treści pracy. Ze względów objętościowych ograniczono się do analizy małego zbioru danych (mniej niż 200 dokumentów testowych). Testy wykorzystywane w pracy prowadzone były na repozytorium składającym się z niemal tysiąca dokumentów.

B.1.1 Dane wejściowe

Dane wejściowe to zbiór 20 dokumentów, które w większości są związane z tematyką *data miningu*, analizy danych oraz pozyskiwania wiedzy.

Id	Tytuł	Odczytany
2	An ontology driven similarity algorithm	tak
3	Bioinformatics doing e-business	tak
4	A mutually beneficial integration of data mining...	tak
5	Data mining for technical operation of companies...	tak

Ciąg dalszy na następnej stronie

Id	Tytuł	Odczytany
6	Discovery of important keywords in the cyberspace	tak
7	Introduction to data mining	nie
8	Knowledge Discovery in the internet	tak
9	Regression - yet another clustering method	tak
10	Optimised pattern discovery	tak
11	Text mining based journal splitting	tak
12	Text extraction from web images	tak
13	Text mining for document annotations and ontology support	tak
14	Using Data Minig methodolgoy for text retrieval	tak
15	Optimised pattern discovery	tak
160	Dell power edge 700	nie
172	Document clustering	nie
173	Indexing by latent semantic analysys	nie
174	Categorization	tak
177	The Costs and Risk Of Open Source	tak
180	Java neural network gui with joone	tak

Tablica B.1: Artykuły inicjujące kolekcję

B.1.2 Dane pozyskane

Dane pozyskane, to artykuły które zostały pobrane z sieci po analizie istniejących dokumentów i poszerzyły testowaną kolekcję. Wybór ten jest dokonywany na podstawie automatycznie sformułowanych zapytań Google. Zbiór danych pozyskanych po zakończeniu działania algorytmu zawierał 166 dokumentów. Część wyników przedstawiona została w tabeli B.2, w której opuszczone zostały wszelkie dokumenty, które nie mogły być przetwarzane przez system (to znaczy nie było możliwe odczytanie z nich czystego tekstu).

Id	Tytuł
17	Educational Malpractice in the USA Part 2
18	Nanotechnology
24	Mobile Advertising Overview
27	The BRICK Awards
29	Carrier Tracking Loop
38	Integrating Deep and Shallow Natural Language Processing
40	OCRed documents
41	Customer Relationship Management
42	3rd ed Intro to Data Mining

Ciąg dalszy na następnej stronie

Id	Tytuł
43	Breaktrhough Technologies in Inteligent Data Analysis
44	Building profitable customer relationships with data mining
45	Building a Model to Predict Caseworker and Supervisor Turnover...
46	Introduction to Data Mining.pdf
47	A Bayesian Net Inference Tool for Hidden State in Texas Hold'em Poker
48	Data Mining
49	Artificial Neural Networks Based Models for Automatic Performance
50	Because Not All Data Is Flat : IBM's U2 Extended Relational DBMS
52	Using Genetic Algorithms for Data Mining Optimization
53	Resource-Aware Mining with Variable Granularities in Data Streams
54	Civil liberties
56	Global data synchronization: alone is not enough
57	Language Assistance Products Support Mobile Advertising
58	Delivering the web to mobile handsets
59	Critical Mass The Worldwide State of the Mobile Web
61	Saving Data Docs
62	WARM UP TO WORK OUT
64	ECR plasma source for heavy ion beam charge neutralization
66	Intense, high brightness H-beams from surface plasma sources
67	Extra credit project on value flow in Prolog
69	Analysis of Classrooms
70	Programming Assignment: Learning with TAS
71	Econometric Analysis of Cross-Sectional & Panel Data Using STATA
72	VALIDATING THE REGRESSION MODEL
75	Linking Immigrants to Capital & Credit
78	Revised Finding of Ipact in Natural Resource Management
80	LaTeX 4 Short Course pdf
82	Automatic Lexicon Generation Using Only Unannotated Text
83	Intelligent co-operative system in cars for road safety
84	Optimal Portfolio Construction
86	Using a Natural Language Understanding System
87	MultiParadigm Reasoning for Access to Heterogeneous GIS
91	Knowledge management publications

Tablica B.2: Artykuły pozyskane do kolekcji

B.1.3 Kategoryzacja dokumentów

Pomiędzy wszystkimi dokumentami zmierzone zostały miary różnic, a następnie wykonana została kategoryzacja tych dokumentów w oparciu o trzy ich typy.

Id	lista względem d_s	lista względem d_n	lista względem d_k
4	4, 119, 31, 31, 50, 78, 78, 72, 72, 157, 157, 129, 77, 177, 120, 53, 11, 11, 104, 76, 76, 144, 144, 118, 52, 6, 107, 45, 45, 80, 133...	4, 67, 68, 174, 184, 121, 104, 52, 5, 53, 5, 50, 52, 144, 13, 9, 70, 6, 13, 3, 11, 104, 144, 50, 120, 121, 57, 6, 57, 69, 131, 107, 70, 53, 3, 11, 43, 120, 14, 9, 153, 107, 43, 75...	4, 163, 36, 165, 148, 84, 124, 110, 139, 111, 82, 167, 63, 62, 77, 133, 94, 157, 154, 76, 132, 71, 27, 28, 103, 8, 66, 33, 72, 21, 129, 156, 98, 64, 128, 118, 100, 153...
6	6, 78, 31, 31, 77, 144, 129, 52, 177, 120, 50, 72, 11, 119, 45, 116, 104, 47, 53, 157, 107, 4, 118, 106, 76, 133, 94, 136, 82, 80, 163...	6, 67, 68, 174, 184, 11, 120, 10, 15, 11, 120, 10, 15, 53, 98, 121, 12, 126, 128, 53, 9, 153, 121, 72, 66, 3, 104, 9, 3, 104, 64, 69, 156, 12, 126, 128, 106, 33, 153, 31, 5...	6, 163, 36, 165, 148, 84, 124, 110, 139, 111, 82, 167, 133, 94, 157, 76, 62, 63, 77, 103, 8, 154, 21, 132, 71, 27, 28, 100, 118, 45, 48, 171, 66, 14, 72, 33, 129, 101...
8	8, 31, 119, 119, 4, 4, 47, 177, 177, 157, 50, 72, 72, 129, 120, 120, 118, 11, 104, 53, 78, 77, 6, 94, 136, 144, 52, 45, 76, 76, 116...	8, 87, 47, 87, 100, 100, 47, 142, 114, 142, 114, 151, 171, 118, 164, 164, 151, 118, 171, 169, 169, 127, 86, 127, 168, 116, 86, 168, 21, 116, 21, 122, 115, 122, 115, 95, 95	8, 163, 63, 62, 77, 154, 132, 71, 27, 28, 36, 66, 33, 72, 129, 156, 98, 64, 128, 153, 10, 15, 106, 165, 148, 84, 53, 124, 11, 139, 110, 70, 6, 131, 2, 59, 52, 111...
11	11, 120, 31, 177, 78, 52, 72, 144, 53, 50, 77, 129, 6, 118, 118, 104, 4, 4, 47, 45, 45, 119, 107, 133, 106, 94, 76, 157, 136, 136, 116...	11, 67, 68, 174, 184, 120, 120, 121, 121, 98, 12, 126, 128, 12, 126, 128, 72, 33, 64, 53, 64, 66, 53, 10, 15, 3, 31, 6, 153, 98, 9, 9, 10, 15, 72, 69, 78, 69, 156, 6, 153, 58...	11, 163, 36, 165, 148, 84, 124, 110, 139, 111, 82, 167, 133, 94, 157, 76, 63, 103, 62, 77, 8, 21, 154, 118, 100, 71, 132, 45, 48, 27, 171, 28, 14, 101, 168, 66...
177	177, 11, 120, 120, 53, 31, 6, 52, 144, 72, 4, 78, 119, 107, 104, 116, 77, 118, 118, 50, 47, 129, 45, 8, 8, 106, 157, 82, 76, 133, 136...	67, 68, 174, 177, 184, 5, 70, 121, 5, 50, 107, 104, 50, 11, 52, 6, 3, 57, 53, 52, 120, 144, 106, 57, 144, 3, 12, 126, 128, 104, 70, 4, 121, 13, 14, 107, 185, 69, 98, 9, 4, 6...	177, 92, 113, 39, 162, 83, 38, 149, 81, 147, 35, 166, 34, 161, 32, 158, 150, 80, 61, 93, 155, 54, 136, 55, 159, 74, 78, 95, 29, 31, 58, 86, 164, 122, 169, 12...
14	14, 62, 75, 131, 155, 59, 182, 63, 159, 108, 95, 122, 93, 161, 148, 109, 84, 158, 167, 81	14, 67, 68, 174, 184, 13, 13, 4, 5, 5, 4, 52, 57, 50, 57, 52, 144, 50, 104, 144, 121, 70, 6, 3, 119, 104, 2, 70, 107, 91, 11, 120, 121, 53, 2, 3, 6, 9, 91, 43, 119, 43,	14, 13, 163, 36, 165, 148, 84, 124, 110, 139, 63, 62, 111, 82, 77, 167, 133, 154, 94, 157, 76, 71, 132, 27, 28, 103, 66, 33, 72, 129, 156, 8, 98, 64, 21, 128, 153...
2	2, 91, 64, 98, 66, 9, 5, 3, 48, 137, 12, 126, 128, 127, 101, 13, 44, 180, 87, 70, 114, 114, 115, 69, 171, 164, 164, 83, 83, 34, 35, 36	2, 67, 68, 174, 184, 91, 91, 121, 104, 120, 11, 6, 70, 107, 53, 107, 5, 3, 9, 52, 144, 106, 98, 121, 12, 126, 128, 69, 3, 131, 72, 50, 4, 78, 120, 153, 57, 182, 66, 185...	2, 163, 36, 165, 148, 84, 124, 110, 139, 111, 82, 167, 133, 94, 157, 76, 103, 8, 21, 118, 100, 45, 48, 171, 14, 101, 44, 168, 57, 127, 13, 59, 4, 52, 131, 6...
3	3, 64, 98, 66, 12, 126, 128, 127, 13, 9, 70, 69, 87, 44, 101, 48, 137, 5, 115, 2, 91, 164, 114, 171, 171, 180, 34, 34, 35, 36, 83, 83	3, 67, 68, 174, 184, 11, 12, 126, 128, 120, 72, 121, 98, 53, 11, 64, 9, 120, 156, 66, 153, 121, 53, 69, 9, 31, 12, 126, 128, 69, 10, 15, 75, 33, 58, 6, 75, 6, 78, 104, 156, 64...	3, 180, 187, 185, 182, 175, 68, 184
5	5, 98, 64, 64, 69, 12, 126, 128, 12, 126, 128, 66, 48, 137, 44, 70, 70, 3, 2, 91, 13, 127, 127, 87, 101, 101, 115, 115, 9, 171, 171...	5, 67, 68, 174, 184, 121, 3, 6, 70, 50, 11, 53, 52, 120, 50, 9, 104, 3, 52, 121, 6, 144, 104, 70, 107, 53, 4, 144, 4, 57, 9, 12, 126, 128, 13, 11, 153, 120, 13, 57, 69...	5, 92, 113, 39, 162, 83, 38, 149, 81, 147, 35, 166, 34, 161, 32, 158, 80, 150, 61, 93, 155, 136, 159, 54, 55, 74, 95, 78, 29, 31, 86, 58, 164, 122, 169, 126...
9	9, 98, 64, 69, 3, 66, 2, 91, 12, 126, 128, 70, 48, 137, 87, 101, 127, 115, 13, 114, 5, 44, 171, 164, 164, 180, 180, 34, 35, 36, 83	9, 67, 68, 174, 184, 72, 66, 153, 69, 58, 31, 64, 12, 126, 128, 69, 72, 53, 156, 98, 64, 33, 120, 66, 120, 11, 11, 78, 12, 126, 128, 153, 53, 74, 10, 15, 71, 10, 15, 75, 28...	9, 92, 113, 39, 162, 83, 38, 149, 81, 147, 35, 166, 34, 161, 158, 80, 150, 93, 136, 159, 95, 32, 86, 164, 61, 122, 169, 135, 87, 151, 47, 155, 137, 119, 142...

Ciąg dalszy na następnej stronie

Id	lista względem d_s	lista względem d_n	lista względem d_k
12	12, 126, 128, 98, 66, 64, 70, 69, 3, 101, 13, 5, 5, 44, 114, 164, 164, 9, 87, 115, 127, 2, 171, 91, 91, 48, 137, 137, 180, 34, 35, 83, 36	12, 67, 68, 126, 128, 174, 184, 66, 31, 98, 33, 72, 98, 64, 33, 64, 78, 66, 58, 31, 72, 10, 15, 156, 156, 11, 28, 74, 10, 15, 69, 120, 58, 154, 27, 78, 11, 153, 153, 75, 106, 132...	12, 92, 113, 39, 162, 83, 38, 81, 149, 147, 35, 166, 34, 161, 158, 80, 150, 93, 136, 159, 95, 86, 164, 122, 169, 32, 87, 135, 151, 61, 47, 137, 119, 142...
13	13, 64, 98, 66, 3, 69, 12, 126, 128, 87, 87, 127, 70, 5, 44, 48, 137, 101, 91, 2, 9, 164, 171, 115, 180, 35, 35, 114, 34, 83, 83, 36, 36	13, 67, 68, 174, 184, 104, 52, 5, 4, 104, 4, 14, 5, 144, 70, 107, 52, 14, 121, 107, 70, 50, 57, 144, 121, 6, 50, 106, 6, 57, 3, 9, 53, 91, 11, 120, 3, 106, 91, 69, 11, 120, 2, 131...	
180	180, 64, 98, 66, 66, 91, 2, 69, 101, 101, 48, 137, 5, 70, 12, 126, 128, 127, 164, 3, 114, 114, 9, 9, 87, 44, 44, 13, 171, 115, 35, 36...	180, 76, 187, 76, 103, 187, 103, 159, 94, 159, 94, 157, 157, 111, 111, 133, 133, 167, 167	180, 68, 184, 175, 182, 185, 3, 187, 187, 185, 182, 175, 68, 184
10	10, 15, 67, 68, 174, 184, 28, 58, 71, 32, 33, 135, 135, 61, 153, 153, 185, 54, 43, 168, 21, 121, 86, 169, 169, 142, 103, 103, 124...	10, 15, 67, 68, 174, 184, 28, 74, 58, 54, 27, 156, 31, 66, 33, 132, 72, 98, 62, 61, 78, 154, 153, 55, 71, 63, 66, 33, 155, 31, 29, 77, 58, 72, 156, 175, 64, 12, 126, 128, 153...	10, 163, 36, 165, 148, 84, 124, 110, 139, 111, 82, 167, 133, 94, 157, 76, 103, 8, 21, 118, 62, 100, 63, 77, 45, 48, 171, 14, 101, 154, 168, 44, 127...
15	15, 10, 67, 174, 68, 184, 28, 71, 58, 32, 33, 135, 135, 61, 153, 185, 54, 153, 43, 168, 21, 86, 142, 121, 169, 169, 103, 103, 124...	15, 67, 68, 174, 184, 28, 74, 58, 54, 27, 156, 31, 66, 33, 132, 72, 98, 62, 61, 78, 154, 153, 55, 71, 63, 66, 33, 155, 31, 29, 77, 58, 72, 156, 175, 64, 12, 126, 128, 153...	15, 163, 36, 165, 148, 84, 124, 110, 139, 111, 82, 167, 133, 94, 157, 76, 103, 8, 21, 118, 62, 100, 63, 77, 45, 48, 171, 14, 101, 154, 168, 44, 127...
174	10, 15, 21, 28, 32, 33, 38, 43, 54, 58, 61, 67, 68, 71, 86, 103, 110, 111, 121, 124, 135, 139, 142, 147, 153, 162, 165, 168, 169, 174...	174, 2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15, 24, 27, 28, 29, 31, 32, 33, 43, 45, 48, 50, 52, 53, 54, 55, 57, 58, 59, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 74, 75, 77, 78...	

Tablica B.3: Elementy kategorii, do których przypisane zostały dokumenty inicjujące. Liczby oznaczają identyfikatory przyporządkowane dokumentom (tablice B.2 i B.1)

B.1.4 Słowa kluczowe

Na podstawie kategoryzacji zostały wyznaczone słowa kluczowe dla poszczególnych dokumentów.

Id	Wybrane słowa	Ilość poprawnych
2	**question**, **experiment**, answering, **similarity**, akt, **ontology**, **relation**, **kmi**, **vargas vera**, **enrico, motta**, concepts, **locate**, situate, properties, **subcategorization**, **wisconsin**	14/20
3	**business**, **bioinformatics**, **processes**, silico, **my, grid**, **discovery**, **bioinformaticians**, doing, enable, runtime, **ebxml**	8/12

Ciąg dalszy na następnej stronie

Id	Wybrane słowa	Ilość poprawnych
4	additional, examples, recall, job, rapier, **database**, **extracted**, **artificial**, **extraction**, edu, **mutually**, typically, much	5/14
5	**company**, **calls**, **cell**, **traffic**, **team**, warsaw, **business, cg**, attributes, **computation**, **engineering**, quite, remarks	20/30
6	**optimized**, **target**, **keywords**, called, **proximity**, **discovery**, **fujino**, **experiments**, **transaction**, contain, iranian, trarily, partic, **computing, arbi**	11/16
8	**hyperlink**, **links**, **ranking**, **matrix**, almost, jaguar	4/6
9	another, **piotr, gawrysiak**, **hrb**, **clustering**, **objects**, yet, **regression**, eg, **cg**, average, create	7/12
10	**background, kyushu, hiroki, hseamen, ichiro, discovery, particular, fujino, characterizing, artificcial,** jun	11/12
11	**journal, splitting, ocr,** takes, **full, proceedings, phrases, journals, find, matches,** match, **xiaofan**, **articles, experiments,** gives, directions	13/17
12	**merging**, connections, **karatzas**, prima, **extraction**, **perception**, significant, **wyszecki, itu, engines**	7/10
13	concept, **term**, concepts, **particular, frequent, intelligence, ontology**, already, **elicit**, jan, paralic, complex	7/13
14	warsaw, **explosion**, piotr, countries, became, special, fi	1/7
15	**discovery, hattacksi, approximate, arbitrary, texts, phrases, target**, background, sun, ultra, **hiroki, shimozono,** unknown	10/14
174	-	0
177	**companies, linux,** had, **open, costs, risks,** best, **company, code, platforms,** forrester, december, locations, **worldwide**	10/15
180	**net, guide, layers, layer, synapse,** te, **monitor, joone, engine,** org, **file**, **object**, last	11/14

Tablica B.4: Słowa kluczowe dla artykułów inicjujących kolekcję. Pogrubione są rzeczywiste, wybrane empirycznie słowa kluczowe.

B.2 Medyczne dokumenty naukowe

Zbiór dokumentów inicjujących składał się z trzynastu artykułów poświęconych medycynie. System został uruchomiony na testowej maszynie z systemem Windows i procesorem Intel Core 2 Duo 2.2 GHz. Wszystkie wątki uruchamiane były w trybie sekwencyjnym.

B.2.1 Dane wejściowe

Dane wejściowe to zbiór 13 dokumentów.

Id	Tytuł	Odczytany
1	Impact of Delay to Angioplasty in Patients With Acute Coronary Syndromes Undergoing Invasive Management	tak
2	Prediction of risk of death and myocardial infarction in the six months after presentation with acute coronary syndrome: prospective multinational observational study (GRACE)	tak
3	Prediction of Mortality After Primary Percutaneous Coronary Intervention for Acute Myocardial Infarction	tak
4	Comparison of the Predictive Value of Four Different Risk Scores for Outcomes of Patients With ST-Elevation Acute Myocardial Infarction Undergoing Primary Percutaneous Coronary Intervention	tak
5	Evaluating the Performance of the Can Rapid Risk Stratification of Unstable Angina Patients Suppress Adverse Outcomes With Early Implementation of the ACC_AHA Guidelines	tak
6	Evaluating the Performance of the Can Rapid Risk Stratification of Unstable Angina Patients Suppress Adverse Outcomes With Early Implementation of the ACC_AHA Guidelines	tak
7	GRACE, TIMI, Zwolle and CADILLAC risk scores — Do they predict 5-year outcomes after ST-elevation myocardial infarction treated invasively?	tak
8	Prediction of risk of death and myocardial infarction in the six months after presentation with acute coronary syndrome: prospective multinational observational study (GRACE)	tak
9	A Validated Prediction Model for All Forms of Acute Coronary Syndrome	tak

Ciąg dalszy na następnej stronie

Id	Tytuł	Odczytany
10	The association of admission heart rate and in-hospital cardiovascular events in patients with non-ST-segment elevation acute coronary syndromes: results from 135 164 patients in the CRUSADE quality improvement initiative	tak
12	The American Journal of Cardiology (www.AJConline.org)	tak
13	Predictive Value of ST Resolution Analysis Performed Immediately Versus at Ninety Minutes After Primary Percutaneous Coronary Intervention	tak
14	Predictors of Outcome in Patients With Acute Coronary Syndromes Without Persistent ST-Segment Elevation	tak
15	Declining In-Hospital Mortality and Increasing Heart Failure Incidence in Elderly Patients With First Myocardial Infarction	tak

Tablica B.5: Artykuły inicjujące kolekcję medyczną.

B.2.2 Dane pozyskane

Dane pozyskane, to artykuły które zostały pobrane z sieci po analizie istniejących dokumentów i poszerzyły testowaną kolekcję. Wybór ten jest dokonywany na podstawie automatycznie sformułowanych zapytań Google. Zbiór danych pozyskanych po zakończeniu działania algorytmu zawierał 166 dokumentów. Część wyników przedstawiona została w tabeli B.2, w której opuszczone zostały wszelkie dokumenty, które nie mogły być przetwarzane przez system (to znaczy nie było możliwe odczytanie z nich czystego tekstu).

Id	Tytuł
17	Guidelines for the Management of Patients With Unstable Angina/Non–ST-Elevation Myocardial Infarction
19	Guideline Update for Coronary Artery Bypass Graft Surgery
20	Guidelines for the Clinical Use of Cardiac Radionuclide Imaging
21	Putting Heart Disease Guidelines Into Practice
22	AACN 2008 NTI & CRITICAL CARE EXPOSITION
23	Prediction of Mortality After Primary Percutaneous Coronary Intervention for Acute Myocardial Infarction
24	Combined prognostic utility of ST-segment recovery and myocardial blush after primary percutaneous coronary intervention in acute myocardial infarction

Ciąg dalszy na następnej stronie

Id	Tytuł
25	Impact of Renal Insufficiency in Patients Undergoing Primary Angioplasty for Acute Myocardial Infarction
26	LEFT VENTRICULAR DILATATION AND NEUROHUMORAL ACTIVATION AS ARRHYTHMOGENIC FACTORS IN MYOCARDIAL INFARCTION
27	Guidelines for the management of arterial hypertension
28	Biochemical markers of acute ischaemia /Percutaneous transluminal coronary angioplasty in acute myocardial infarction
29	Guidelines for the diagnosis and management of syncope (version 2009) The Task Force for the Diagnosis and Management of Syncope of the European Society of Cardiology (ESC)
30	Quality of Care for Acute Coronary Syndrome Patients With Known Atherosclerotic Disease
32	Paclitaxel-Eluting Stents versus Bare-Metal Stents in Acute Myocardial Infarction
33	Particulate Matter Air Pollution and Cardiovascular Disease
34	Study of the problem of Consumer Indebtedness: Statistical Aspects Contract n°: B5-1000/00/000197
35	On speech and language
36	USE OF THE LOW-MOLECULAR-WEIGHT HEPARIN REVIPARIN TO PREVENT DEEP-VEIN THROMBOSIS AFTER LEG INJURY REQUIRING IMMOBILIZATION
37	Clopidogrel versus Other Antiplatelet Agents in the Secondary Prevention of Vascular Events in Adults with Cerebrovascular Disease: Clinical and Cost-Effectiveness Analyses
38	Outcomes of Genetic Testing in Adults with a History of Venous Thromboembolism
39	ANTIPLATELET THERAPY IN MYOCARDIAL INFARCTION AND CORONARY STENT THROMBOSIS
40	Abstracts of the Second International Conference on Women, Heart Disease, and Stroke
41	Increasing Post-Myocardial Infarction Heart Failure Incidence in Elderly Patients
43	Cystatin-C and Mortality in Elderly Persons With Heart Failure
44	The Year in Review of Clinical Cardiac Electrophysiology
45	ACC/AHA/ESC 2006 Guidelines for the Management of Patients With Atrial Fibrillation
46	Acute coronary syndromes

Ciąg dalszy na następnej stronie

Id	Tytuł
47	Results of Systematic Review of Research on Diagnosis and Treatment of Coronary Heart Disease in Women
48	Management of acute myocardial infarction in patients presenting with ST-segment elevation
49	API Expert Consensus Document on Management of Ischemic Heart Disease
50	Medical Guidelines for Airline Travel, 2nd ed.
51	Thoracic Aortic Aneurysm (TAA) Endovascular Stenting And Lumbar Cerebrospinal Fluid (Csf) Drains: To Drain Or Not To Drain
52	CardioVascular Research Foundation(CVRF)

Tablica B.6: Artykuły medyczne pozyskane do kolekcji.

B.2.3 Kategoryzacja dokumentów

Pomiędzy wszystkimi dokumentami zmierzone zostały miary różnic, a następnie wykonana została kategoryzacja tych dokumentów w oparciu o trzy ich typy.

Id	lista względem d_s	lista względem d_n	lista względem d_k
1	1, 42, 36, 4, 14, 25, 12, 16, 28, 3, 43, 8, 45, 2, 23, 30, 13, 24, 34, 15, 9, 7, 41, 5, 6, 11, 10, 32, 49, 46, 33, 26, 52, 35, 17, 38, 44, 50, 48, 19, 51, 20, 21, 27, 47, 40, 22,	2, 1, 4, 14, 9, 11, 10, 16, 25, 13, 7, 3, 30, 23, 24, 8, 45, 36, 42, 5, 6, 41, 15, 12, 28, 32, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	1, 3, 2, 17, 35, 40, 22, 44, 19, 27, 47, 26, 21, 20, 38, 33, 34, 46, 48, 51, 50, 49, 52, 43, 28, 32, 15, 23, 6, 5, 30, 8, 45, 13, 9, 14, 11, 10, 16, 36, 24, 25, 7, 42, 4, 41, 12,
2	8, 2, 45, 12, 16, 24, 13, 9, 42, 34, 36, 32, 30, 1, 28, 14, 4, 25, 52, 23, 6, 5, 3, 50, 46, 51, 7, 43, 11, 10, 15, 41, 49, 48, 38, 33, 26, 35, 21, 40, 47, 22, 20, 17, 19, 27, 44,	1, 45, 8, 2, 9, 4, 16, 7, 24, 25, 13, 42, 11, 10, 14, 41, 3, 12, 23, 30, 36, 5, 6, 15, 28, 32, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	3, 2, 1, 23, 14, 4, 16, 24, 10, 13, 11, 9, 7, 25, 30, 45, 8, 36, 6, 5, 15, 42, 28, 41, 12, 32, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,
3	3, 23, 12, 4, 42, 13, 43, 9, 14, 34, 36, 7, 15, 11, 10, 1, 16, 28, 6, 5, 52, 24, 25, 32, 41, 30, 48, 8, 45, 2, 38, 33, 50, 49, 51, 35, 46, 21, 26, 44, 20, 47, 19, 40, 27, 17, 22,	3, 23, 14, 4, 13, 11, 10, 24, 1, 16, 9, 7, 25, 30, 2, 45, 8, 6, 5, 15, 36, 42, 28, 41, 12, 32, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	2, 3, 1, 35, 40, 44, 19, 47, 21, 20, 38, 33, 34, 46, 48, 51, 50, 49, 52, 43, 32, 15, 6, 5, 30, 8, 45, 13, 23, 14, 9, 10, 11, 36, 16, 24, 25, 28, 7, 42, 4, 41, 12, 26, 27, 22, 17,
4	4, 12, 42, 3, 23, 10, 11, 7, 36, 5, 6, 24, 34, 1, 13, 14, 25, 43, 30, 28, 15, 41, 16, 9, 2, 45, 8, 52, 32, 50, 38, 51, 49, 33, 35, 48, 21, 46, 40, 26, 20, 47, 19, 27, 22, 44, 17,	4, 7, 9, 25, 12, 16, 41, 42, 24, 13, 10, 11, 14, 1, 2, 45, 8, 3, 23, 36, 30, 6, 5, 15, 28, 32, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	4, 42, 36, 7, 25, 41, 11, 10, 16, 14, 24, 9, 12, 1, 45, 8, 13, 30, 5, 6, 15, 3, 23, 32, 28, 43, 52, 49, 50, 51, 48, 46, 34, 33, 38, 20, 21, 26, 47, 27, 44, 19, 22, 35, 40, 17, 2,
5	5, 6, 36, 42, 15, 14, 28, 12, 34, 7, 16, 10, 11, 4, 43, 41, 32, 24, 9, 30, 13, 3, 35, 23, 25, 52, 51, 2, 45, 8, 26, 38, 50, 1, 49, 48, 20, 46, 33, 21, 40, 19, 44, 27, 17, 47, 22,	5, 6, 11, 10, 14, 15, 23, 1, 30, 3, 36, 7, 16, 13, 9, 4, 25, 24, 28, 2, 45, 8, 42, 32, 41, 12, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	5, 1, 3, 23, 30, 15, 45, 8, 13, 6, 32, 24, 9, 16, 10, 11, 14, 36, 28, 25, 7, 42, 4, 43, 52, 49, 41, 12, 50, 51, 48, 46, 34, 33, 38, 20, 26, 21, 47, 27, 19, 44, 22, 35, 40, 17, 2,

Ciąg dalszy na następnej stronie

Id	lista względem d_s	lista względem d_n	lista względem d_k
7	7, 12, 14, 42, 4, 36, 5, 6, 16, 28, 41, 11, 10, 9, 43, 15, 3, 23, 13, 32, 52, 34, 24, 25, 38, 30, 1, 50, 51, 2, 45, 8, 48, 33, 46, 49, 26, 35, 20, 47, 19, 21, 44, 22, 27, 40, 17,	7, 4, 9, 42, 25, 16, 10, 11, 13, 14, 41, 24, 12, 45, 2, 8, 1, 3, 36, 23, 30, 5, 6, 15, 28, 32, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	7, 36, 42, 25, 16, 24, 10, 11, 14, 9, 4, 1, 13, 45, 8, 30, 3, 5, 6, 23, 15, 41, 12, 32, 28, 43, 52, 49, 50, 51, 46, 48, 34, 33, 38, 20, 21, 26, 47, 27, 19, 44, 22, 35, 40, 17, 2,
8	8, 2, 45, 12, 16, 24, 13, 9, 42, 34, 36, 32, 30, 1, 28, 14, 4, 25, 52, 23, 6, 5, 3, 50, 46, 51, 7, 43, 11, 10, 15, 41, 49, 48, 38, 33, 26, 35, 21, 40, 47, 22, 20, 17, 19, 27, 44,	2, 8, 45, 9, 4, 7, 16, 24, 25, 13, 11, 10, 14, 1, 42, 3, 12, 41, 23, 30, 36, 5, 6, 15, 28, 32, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	8, 1, 13, 30, 3, 6, 5, 23, 24, 14, 16, 15, 36, 10, 11, 25, 32, 9, 7, 42, 28, 4, 43, 52, 41, 12, 49, 50, 51, 46, 48, 34, 33, 38, 20, 21, 26, 47, 27, 19, 44, 22, 35, 40, 17, 45, 2,
9	9, 12, 34, 32, 36, 14, 42, 52, 45, 2, 8, 28, 23, 24, 41, 49, 3, 7, 13, 50, 16, 43, 30, 10, 11, 5, 6, 25, 51, 15, 4, 1, 35, 33, 48, 38, 46, 21, 26, 20, 40, 19, 27, 47, 22, 17, 44,	9, 4, 7, 42, 45, 8, 2, 25, 10, 11, 14, 16, 24, 41, 13, 12, 1, 3, 36, 23, 30, 15, 5, 6, 28, 32, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	9, 24, 16, 14, 36, 1, 25, 13, 11, 10, 30, 3, 6, 5, 23, 45, 8, 7, 15, 42, 32, 4, 28, 41, 12, 43, 52, 49, 50, 51, 48, 46, 34, 33, 38, 20, 26, 21, 47, 27, 44, 19, 22, 35, 40, 17, 2,
10	10, 11, 36, 4, 5, 6, 12, 28, 13, 43, 14, 32, 7, 24, 16, 42, 3, 23, 9, 30, 25, 34, 15, 52, 41, 51, 2, 45, 8, 1, 50, 33, 35, 46, 49, 48, 38, 40, 26, 20, 19, 27, 21, 47, 17, 44, 22,	11, 10, 14, 9, 25, 4, 13, 16, 24, 7, 1, 30, 3, 42, 23, 36, 2, 45, 8, 5, 6, 41, 15, 12, 28, 32, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	10, 24, 36, 16, 14, 9, 25, 1, 13, 45, 8, 3, 7, 30, 6, 5, 23, 15, 42, 4, 32, 28, 41, 12, 43, 52, 49, 50, 51, 48, 46, 34, 33, 38, 20, 21, 26, 47, 27, 19, 44, 22, 35, 40, 17, 11, 2,
12	12, 42, 41, 9, 7, 36, 4, 3, 23, 32, 8, 45, 2, 34, 14, 52, 5, 6, 49, 25, 28, 43, 24, 15, 11, 10, 16, 1, 30, 13, 50, 51, 38, 46, 35, 48, 33, 40, 21, 27, 19, 47, 26, 20, 17, 22, 44,	12, 41, 42, 4, 9, 7, 25, 24, 16, 13, 2, 45, 8, 10, 11, 14, 36, 1, 3, 30, 23, 6, 5, 15, 28, 32, 43, 52, 49, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	12, 41, 42, 4, 7, 25, 36, 10, 11, 16, 24, 9, 14, 1, 8, 45, 30, 13, 3, 6, 5, 23, 15, 32, 28, 43, 52, 49, 50, 51, 46, 48, 34, 33, 38, 20, 21, 26, 47, 27, 19, 44, 22, 35, 40, 17, 2,
13	13, 36, 24, 42, 3, 11, 10, 23, 45, 2, 8, 30, 4, 16, 43, 28, 9, 25, 32, 12, 7, 14, 5, 6, 1, 34, 15, 38, 50, 52, 41, 51, 33, 35, 46, 26, 48, 49, 40, 47, 21, 27, 44, 20, 19, 22, 17,	13, 24, 16, 4, 25, 11, 10, 7, 9, 14, 1, 3, 42, 23, 2, 45, 8, 36, 30, 12, 41, 15, 5, 6, 28, 32, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	13, 45, 8, 1, 30, 5, 6, 9, 14, 3, 10, 11, 36, 23, 15, 25, 24, 32, 16, 7, 42, 28, 4, 41, 43, 52, 12, 49, 50, 51, 46, 48, 34, 33, 38, 20, 21, 26, 47, 27, 19, 44, 22, 35, 40, 17, 2,
14	14, 36, 6, 5, 12, 7, 9, 24, 32, 42, 34, 43, 11, 10, 1, 41, 4, 28, 15, 3, 23, 13, 2, 45, 8, 16, 52, 51, 25, 50, 30, 49, 48, 46, 35, 38, 33, 26, 20, 21, 19, 44, 17, 40, 27, 47, 22,	14, 11, 10, 9, 4, 7, 1, 16, 13, 25, 3, 24, 36, 23, 30, 42, 2, 45, 8, 5, 6, 41, 15, 12, 28, 32, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	14, 24, 9, 16, 11, 10, 36, 13, 1, 45, 8, 25, 3, 7, 30, 23, 6, 5, 15, 32, 42, 4, 28, 41, 12, 43, 52, 49, 50, 51, 46, 48, 34, 33, 38, 20, 21, 26, 47, 27, 19, 44, 22, 35, 40, 17, 2,
15	15, 36, 6, 5, 12, 28, 24, 14, 16, 34, 23, 43, 7, 3, 4, 25, 42, 32, 9, 41, 13, 11, 10, 1, 35, 50, 30, 52, 51, 2, 45, 8, 49, 38, 48, 19, 26, 40, 20, 33, 46, 27, 17, 21, 44, 47, 22,	15, 5, 6, 11, 10, 14, 3, 23, 30, 1, 16, 36, 13, 9, 25, 28, 24, 4, 7, 2, 45, 8, 42, 41, 32, 12, 43, 49, 52, 50, 51, 48, 46, 34, 33, 20, 21, 38, 26, 27, 22, 40, 19, 47, 35, 44, 17,	15, 1, 3, 23, 32, 5, 6, 30, 8, 45, 13, 28, 24, 14, 9, 16, 10, 11, 36, 25, 7, 43, 52, 4, 42, 49, 12, 41, 50, 51, 48, 46, 34, 33, 38, 20, 21, 26, 47, 27, 19, 44, 22, 35, 40, 17, 2,

Tablica B.7: Elementy kategorii, do których przypisane zostały dokumenty inicjujące. Liczby oznaczają identyfikatory przyporządkowane dokumentom (tablice B.6 i B.5)

B.2.4 Słowa kluczowe

Na podstawie kategoryzacji zostały wyznaczone słowa kluczowe dla poszczególnych dokumentów.

Id	Wybrane słowa
1	acs, hrs, invasive, nste, timi, iiia, revascularization, presentation, present, composite, ischemia, strategy, ischemic, multivariable, stone, mellitus, carolina, center, royal, dp,
2	full, simplified, model, grace, syndrome, tool, months, sanofi, bristol, aventis, competing, prediction, risk
3	north, integer, baseline, facc, nikolsky, khalil, additional, classes, vessel,
4	score, cadillac, pami, included, stemi, predictive, scores, classification, da, range, performed, statistic,
5	major, bleeding, crusade, syndromes,mail, online, steg, lez, eikelboom, rao, common,
7	grace, zwolle, cadillac, stemi, journal, treated, international, cantly, october, available, recently, accepted, locate,
8	six, months, ardissinod, grace, tool, syndrome, com, continuous, predictors, prediction, pj, variableswithmediansand, bernink, krumholzhm, chenj,
9	syndrome, flather, pieper, phd, eagle, models, model, content, prediction, permissions, timateapatient, graft, jama, registry, ohmanem, current,
10	hr, presenting, admission, crusade, segment, eurheartj, bristol, doi, received, institute,
12	segment, would, reperfusion, likelihood, assessment, although, pretest, computed, cta, vessel, method, hood, complex, eport, relatively, msc,
13	intervention, minutes, resolution, roe, lemos, kaplan, meier, ad, junctive, magnetic, resonance,
14	enrollment, day, lee, segment, boersma, califf, eric, wei, syndromes, wilcox, issn, logistic, simoons, rights,
15	hf, index, cohort, developed, hospitalization, our, occurred, sd, decade, proportion, survivors,

Tablica B.8: Słowa kluczowe dla artykułów inicjujących kolekcję medyczną wyznaczone na podstawie tylko przedstawionego zbioru.

Id	Wybrane słowa
1	acs, nste, hrs, angiography, antithrombotic, timing, presentation, invasive, according, york, foundation, boston, dp,
2	grace, six, uk, fox, prediction,

Ciąg dalszy na następnej stronie

Id	Wybrane słowa
3	angiographic, validation, cadillac, facc, under, pami, variables, prognostic, vessel, final, predictors, reperfusion, incorporated, mass, demographic, er, bl, male, zaret,
4	scores, registry, author, predictive, cadillac, classification, statistic, kg, applied, corresponding, later, israel,
5	crusade, showed, fox, steg, mail, lez, issn, coronario, ndrome, nonis, common,
7	zwolle, timi, grace, ppci, scores, journal, press, cardiogenic, see, international, lvef, further, ha, morrowda,
8	grace, six, kaaf, patientswithacutecoronarysyndrome, afterdischarge, prediction, multinational,
9	grace, jama, com, registry, models, permissions,
10	hr, presenting, crusade, admission, beat, eurheartj, harrington, doi,
12	reperfusion, seg, cta, computed, pretest, intermediate, coronary, noninvasive, vessel, suggest, resolution, larger, gs, ative, msc, accommodate, nested, individual,
13	grines, minutes, em, norfolk, ninety, resolution, delay, saurabh,
14	df, enrollment, uap, multivariable, syndromes, califf, techniques, eric, wilcox, rights, cgi, hungary, budapest,
15	modification, visits, physician, variables, validated,

Tablica B.9: Słowa kluczowe dla artykułów inicjujących kolekcję medyczną wyznaczone na podstawie przedstawionego zbioru rozszerzonego o 200 innych dokumentów o różnej tematyce.

Bibliografia

Ajtai, M., Komlós, J., & Szemerédi, E. 1983. An 0(n log n) sorting network. *Pages 1–9 of: Stoc '83: Proceedings of the fifteenth annual acm symposium on theory of computing*. New York, NY, USA: ACM Press. doi:http://doi.acm.org/10.1145/800061.808726.

Albert, Réka, & Barabási, Albert-László. 2002. Statistical mechanics of complex networks. *Reviews of modern physics*, January, 47–97.

Albert, Réka, Jeong, Hawoong, & Barabási, Albert-László. 1999. Diameter of the world-wide web. *Science*, **401**(Septmeber), 130–131.

Arimura, Hiroki, Abe, Junichiro, Fujino, Ryoichi, Sakamoto, Hiroshi, Shimozono, Shinichi, & Arikawa, Setsuo. 2000. Text data mining: Discovery of important keywords in the cyberspace. *Digital libraries: Research and practice, kyoto international conference on*, **0**, 220. doi:http://doi.ieeecomputersociety.org/10.1109/DLRP.2000.942178.

Avrachenkov, Konstantin, & Litvak, Nelly. 2004. *Decomposition of the Google PageRank and Optimal Linking Strategy*. Research Report RR-5101. INRIA. Dostępne na: http://hal.inria.fr/inria-00071482/PDF/RR-5101.pdf.

Avrachenkov, Konstantin, & Litvak, Nelly. 2004. *Decomposition of the google pagerank and optimal linking strategy*. Dostępne na: HAL:http://hal.inria.fr/inria-00071482/en/;http://hal.inria.fr/docs/00/07/14/82/PDF/RR-5101.pdf.

Aycinena, Meg, & Brunskill, Emma. *Reinforcement learning examples*. Dostępne na: http://www.damas.ift.ulaval.ca/~coursMAS/Complements2K8/rl_examples.pdf.

Bäck, Thomas. 1996. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford, UK: Oxford University Press.

BIBLIOGRAFIA

Barabási, Albert-László, Jeong, Hawoong, Néda, Zoltan, Ravasz, Erzsebet, Schubert, A., & Vicsek, Tamas. 2002. Evolution of the social network of scientific collaborations. *Physica a*, **311**(4), 590–614.

Bellman, Richard. 1957. *Dynamic programming*. Princeton University Press.

Bremaud, Pierre. 2001. *Markov chains: Gibbs fields, monte carlo simulation, and queues*. Corrected edn. Springer-Verlag New York Inc. Dostępne na: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0387985093.

Cavnar, William B., & Trenkle, John M. 1994. N-gram-based text categorization. *Pages 161–175 of: Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*. Dostępne na: citeseer.ist.psu.edu/68861.html.

Ceselli, & Righini. 2005. A branch-and-price algorithm for the capacitated p-median problem. *Networks: Networks: An international journal*, **45**.

Charikar, M., Guha, S., Tardos, E., & Shmoys, D. S. 2002. A constant-factor approximation algorithm for the k-median problem. *Journal of computer and system sciences*, **65**, 129–149.

Cheng, Christine. 2008. The generalized median stable matchings: Finding them is not that easy. *Pages 568–579 of:* Laber, Eduardo Sany, Bornstein, Claudson F., Nogueira, Loana Tito, & Faria, Luerbio (eds), *LATIN 2008: Theoretical informatics, 8th latin american symposium, Búzios, brazil, april 7-11, 2008, proceedings*. Lecture Notes in Computer Science, vol. 4957. Springer. Dostępne na: http://dx.doi.org/10.1007/978-3-540-78773-0_49.

Cherfi, Hacène, Napoli, Amedeo, & Toussaint, Yannick. 2003. Towards a text mining methodology using frequent itemsets and association rule extraction. *Pages 285–294 of:* M. Nadif, A. Napoli, E. SanJuan, A. Sigayret (ed), *Journées d'informatique Messine - JIM'03, Metz, France*INRIA Lorraine, for E. SanJuan.

Cichosz, Paweł. 2000. *Systemy uczące się*. Warszawa: WNT.

Cichosz, Pawel. 1994. *Reinforcement learning algorithms based on the methods of temporal differences*. M.Sc. thesis, Institute of Computer Science, Warsaw University of Technology.

Cichosz, Pawel, & Mulawka, Jan J. 1995. Fast and efficient reinforcement learning with truncated temporal differences. *Pages 99–107 of: Icml*.

Cilibrasi, Rudi, & Vitányi, Paul M. B. 2005. Clustering by compression. *Ieee transactions on information theory*, **51**(4), 1523–1545.

Cruz, Isabel F., Borisov, Slava, Marks, Michael A., & Webb, Timothy R. 1998. Measuring structural similarity among Web documents: Preliminary results. *Lecture notes in computer science*, **1375**, 513–?? Dostępne na: http://link.springer-ny.com/link/service/series/0558/bibs/1375/13750513.htm;http://link.springer-ny.com/link/service/series/0558/papers/1375/13750513.pdf.

Doeblin, W. 1933. Exposé de la théorie des chains simples constantes de markov a un nombre fini d'états. *Rev math union interbalkanique*, **2**, 77–105.

Eckel, Bruce. 2005. *Thinking in java (4th edition)*. Upper Saddle River, NJ, USA: Prentice Hall PTR.

Ferrer i Cancho, Ramon, & Solé, Ricard V. 2001. The small-world of human language. *Proceedings of the royal society of london b*, **268**(1482), 2261–2265.

Finnoff, W., & Zimmermann, H. G. 1994. Detecting structure in small datasets by network fitting under complexity constraints. *Pages 113–131 of: Proceedings of the workshop on computational learning theory and natural learning systems (vol. 2) : intersections between theory and experiment*. Cambridge, MA, USA: MIT Press.

Finnoff, William, Hergert, Ferdinand, & Zimmermann, Hans Georg. 1993. Original contribution: Improving model selection by nonconvergent methods. *Neural netw.*, **6**(6), 771–783. doi:http://dx.doi.org/10.1016/S0893-6080(05)80122-4.

Fort, Jean-Claude. 2006. SOM's mathematics. *Neural networks*, **19**(6-7), 812–816. Dostępne na: http://dx.doi.org/10.1016/j.neunet.2006.05.025.

Fortnow, L. 2004. *Kolmogorov complexity and computational complexity*. Dostępne na: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.4223.

Frank, Eibe, Chui, Chang, & Witten, Ian H. 2000. Text categorization using compression models. *Pages 200–209 of: In proceedings of dcc-00, ieee data compression conference, snowbird, us*. IEEE Computer Society Press.

BIBLIOGRAFIA

Fritzke, Bernd. 1993. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural networks*, **7**, 1441–1460.

Gács, Peter, & Körner, János. 1973. Common information is dar less than mutual information. *Problems of control and information theory*, 119–162.

Gimbel, M., Philippsen, M., Haumacher, B., & Lockemann, P. C. 1999. Java as a basis for parallel data mining in workstation clusters. *Lecture notes in computer science*, **1593**, 884–??

Grinstead, Charles M., & Snell, J. Laurie. 1997. *Introduction to probability*. 2 revised edn. American Mathematical Society. Dostępne na: http://www.dartmouth.edu/~{}chance/teaching_aids/books_articles/probability_book/amsbook.mac.pdf.

Grudin, Jonathan. 1994. Groupware and social dynamics: Eight challenges for developers. *Commun. acm*, **37**(1), 92–105.

Grudin, Jonathan, & Poltrock, Steven. 1996. CSCW, groupware, and workflow: Experiences, state of art, and future trends. *Pages 338–339 of: Proceedings of acm chi 96 conference on human factors in computing systems*. Tutorial 6, vol. 2. Dostępne na: http://www.acm.org/sigchi/chi96/proceedings/tutorial/Grudin/jtg_txt.htm.

Haggstrom, O. 2002. *Finite markov chains and algorithmic applications*. Dostępne na: citeseer.ist.psu.edu/article/haggstrom01finite.html.

Hinton, G E. 1987. Learning translation invariant recognition in massively parallel networks. *Pages 1–13 of: Volume i: Parallel architectures on parle: Parallel architectures and languages europe*. London, UK: Springer-Verlag.

Holten, Danny. 2006. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *Ieee trans. vis. comput. graph*, **12**(5), 741–748. Dostępne na: http://doi.ieeecomputersociety.org/10.1109/TVCG.2006.147.

Jankowski, Norbert. 2003. *Ontogeniczne sieci neuronowe. o sieciach zmieniających swoją strukturę*. Akademicka Oficyna Wydawnicza EXIT.

Kohonen, T. 1984. *Self-organization and associative memory (2nd edition)*. Berlin: Springer-Verlag.

Kohonen, T. 1998. The self-organizing map. *Neurocomputing*, **21**(1-3), 1–6. Dostępne na: http://dx.doi.org/10.1016/S0925-2312(98)00030-7, doi:10.1016/S0925-2312(98)00030-7.

Kohonen, Teuvo, & Honkela, Timo. 2007. Kohonen network. *Scholarpedia*, **2**(1), 1568. Dostępne na: http://www.scholarpedia.org/article/Kohonen_network.

Kohonen, Teuvo, & Somervuo, Panu. 2002. How to make large self-organizing maps for nonvectorial data. *Neural networks*, **15**(8-9), 945–952.

Li, M., & Vitányi, P. 1997. *An introduction to Kolmogorov complexity and its applications*. 2nd edn. Springer.

Li, Ming, & Vitanyi, Paul. 1997. *An introduction to kolmogorov complexity and its applications: Preface to the first edition*.

Lord, Phillip, Wroe, Chris, Stevens, Robert, Goble, Carole, Miles, Simon, Moreau, Luc, Decker, Keith, Payne, Terry, & Papay, Juri. 2003. Semantic and personalised service discovery. Department of Mathematics and Computing Science, Saint Mary's University. Dostępne na: http://eprints.ecs.soton.ac.uk/9455/1/WIKGGI03Semantic.pdf;http://eprints.ecs.soton.ac.uk/9455/.

MacQueen, J. B. 1967. Some methods for classification and analysis of multivariate observations. *Pages 281–297 of:* Cam, L. M. Le, & Neyman, J. (eds), *Proc. of the fifth berkeley symposium on mathematical statistics and probability*, vol. 1. University of California Press.

Mannila, Heikki, Toivonen, Hannu, & Verkamo, A. Inkeri. 1994 (July). Efficient algorithms for discovering association rules. *Pages 181–192 of:* Fayyad, Usama M., & Uthurusamy, Ramasamy (eds), *Aaai workshop on knowledge discovery in databases (kdd-94)*. Dostępne na: ftp://ftp.cs.helsinki.fi/pub/Reports/by_Project/PMDM/Efficient_Algorithms_for_Discovering_Association_Rules.ps.gz.

McCallum, Andrew, & Nigam, K. 1998. A comparison of event models for naive bayes text classification. *In: Proceedings of aaai-98, workshop on learning for text categorization*. Dostępne na: citeseer.nj.nec.com/mccallum98comparison.html.

McLaughlin, Brett. 2000. *Java and XML*. O'Reilly.

Mihalcea, Rada. 2005. unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. *Pages 411–418 of: In hlt/emnlp 2005*.

BIBLIOGRAFIA

Nahm, Un Yong, & Mooney, Raymond J. 2000. A mutually beneficial integration of data mining and information extraction. *Pages 627–632 of: Aaai/iaai*. AAAI Press / The MIT Press.

Page, Lawrence, Brin, Sergey, Motwani, Rajeev, & Winograd, Terry. 1999 (November). *The pagerank citation ranking: Bringing order to the web*. Technical Report 1999-66. Stanford InfoLab. Previous number = SIDL-WP-1999-0120. Dostępne na: http://ilpubs.stanford.edu:8090/422/.

Pieńkowska, Daria. 2005. *Kategoryzacja tekstów pisanych z użyciem samoorganizujących się sieci neuronowych typu kohonenea*. M.Sc. thesis, Uniwersytet Mikołaja Kopernika.

Rajman, Martin, & Besançon, E. 1997. Natural language techniques for text mining applications. *Page 50 of: Ds-7*.

Rojas, Raúl. 1996. *Neural networks: a systematic introduction*. New York, NY, USA: Springer-Verlag New York, Inc.

Rzeczi, Krzysztof, & Mościński, Jacek. 2009. A new method for generation of stop list. *In: In proceedings of International Conrerence of Artifical Inteligence ai-24'2009*.

Shadbolt, Nigel, Berners-Lee, Tim, & Hall, Wendy. 2006. The semantic web revisited. *Ieee intelligent systems*, **21**(3), 96–101. Dostępne na: http://doi.ieeecomputersociety.org/10.1109/MIS.2006.62.

Sinha, Ravi, & Mihalcea, Rada. 2007. Unsupervised Graph-based Word Sense Disambiguation Using Measures of Word Semantic Similarity. *In: Ieee international conference on semantic computing (icsc 2007)*.

Sutton, Richard S. 1996. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Pages 1038–1044 of: Advances in neural information processing systems 8*, vol. 8. Dostępne na: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.4764.

Tan, Ah-hwee. 1999. *Text mining: The state of the art and the challenges*. Dostępne na: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.132.6973;http://www.ntu.edu.sg/home/asahtan/papers/tm_pakdd99.pdf.

Wawrzyński, Paweł. 2005. *Intensive reinforcement learning*. Ph.D. thesis, Wydział Elektroniki and Technik Informacyjnych and Instytut Automatyki and Informatyki Stosowanej.

Zyglarski, Błażej. 2008a. Liferay – przepis na własny portal. *Software Developer's Journal*, **157**, 66—75.

Zyglarski, Błażej. 2008b. Systemy zarządzania treścią na przykładzie portalu liferay. *Page 37 of: Software Development Gigacon'08 - materiały konferencyjne*. Software – Wydawnictwo Sp. zo.o.

Zyglarski, Błażej. 2009. Document management system based on neural networks. *Polish Journal of Environmental Studies*, **18**(3B), 416–420.

Zyglarski, Błażej. 2010. Genetic algorithms and dynamic neural networks in data categorization. *In: Proceedings of SMI 2010*.

Zyglarski, Błażej, & Bała, Piotr. 2009. Scientific documents management system. Application of Kohonen neural networks with reinforcement in keywords extraction. *Pages 55–62 of: In proceedings of IC3K 2009*. INSTICC.

Zyglarski, Błażej, & Bała, Piotr. 2010a. *Keywords extraction. Selecting keywords in natural language texts with Markov chains and neural networks*.

Zyglarski, Błażej, & Bała, Piotr. 2010b. Neural networks aided automatic keywords selection. *Communications in Computer and Information Science*.

Zyglarski, Błażej, Bała, Piotr, & Schreiber, Tomasz. 2008. Web services based scientific article manager. *Pages 205–215 of: Information Systems Architecture and Technology, Web Information Systems: Models, Concepts and Challenges*. Wrocław University of Technology.