

Łukasz BORYCKI*
Przemysław SÓLDACKI

AUTOMATYCZNA KLASYFIKACJA TEKSTÓW

W artykule tym przedstawione są doświadczenia nabyte podczas tworzenia aplikacji służącej do automatycznej klasyfikacji dokumentów opartego na języku Java. Przedstawione są wyniki z zastosowaniem stemmingu angielskiego i polskiego. Testy i przykłady przedstawiają poprawę wyników działania klasyfikatora. Na uwagę zasługują też wyniki testów pruningu (usuwania słów występujących najczęściej lub najrzadziej). Pozwalają one na obserwację, w jaki sposób i w których słowach trzymana jest informacja w klasyfikatorze, co pozwala na usunięcie słów niepotrzebnych (co znacznie skróca czas klasyfikacji) oraz takich, które wprowadzają błąd do klasyfikacji, pogarszając jej właściwości. Stworzona architektura pozwala na porównywanie poszczególnych modułów systemu. Architektura wykorzystuje tzw. ścieżkę przetwarzania, pozwalającą na dowolny preprocessing wejściowego strumienia słów. Dzięki temu istnieje możliwość klasyfikowania dokumentów takich jak HTML i XML (usuwanie znaczników), stemmingu, i innych prostych transformacji, jak usuwanie dużych liter

1. WSTĘP

Automatyczne grupowanie i klasyfikacja dokumentów tekstowych wywodzą się z technik eksploracji danych, czyli grupowania i klasyfikowania rekordów z baz danych. Obie te metody okazują się być bardzo przydatne w analizie bardzo dużych zbiorów danych. Rosnące zainteresowanie technikami automatycznej analizy tekstu i dokumentów tekstowych zaowocowało dostosowaniem metod eksploracji danych do przetwarzania tekstu. Na wstępie warto wyraźnie zaznaczyć różnice pomiędzy klasyfikacją/kategoryzacją (ang. *document classification* lub *document categorization*) a grupowaniem (ang. *document*

* Instytut Informatyki, Politechnika Warszawska, Ul. Nowowiejska 15/19, 00-665 Warszawa,
(e-mail: lborycki@infovide.pl, psoldack@datacom.pl)

clustering). Celem klasyfikacji jest przyporządkowanie dokumenty do jednej lub kilku z uprzednio zdefiniowanych klas. Klasy te zwykle definiowane są nie wprost, lecz poprzez zbiór trenujący, to jest grupę dokumentów już odpowiednio zaklasyfikowanych ręcznie, np. przez ekspertów. W większości przypadków klasy nie są zagnieżdżane, natomiast przyjmuje się, iż jeden dokument może należeć do więcej niż jednej klasy [5].

Problemem pokrewnym do klasyfikacji jest grupowanie dokumentów. W tym przypadku system nie posiada wejściowej wiedzy, w postaci już zaklasyfikowanych dokumentów, czy też samych klas, zaś jego zadaniem jest takie pogrupowanie dokumentów, by dokumenty należące do jednej klasy były do siebie możliwie podobne i jednocześnie różniły się znacząco od dokumentów należących do innych klas [4].

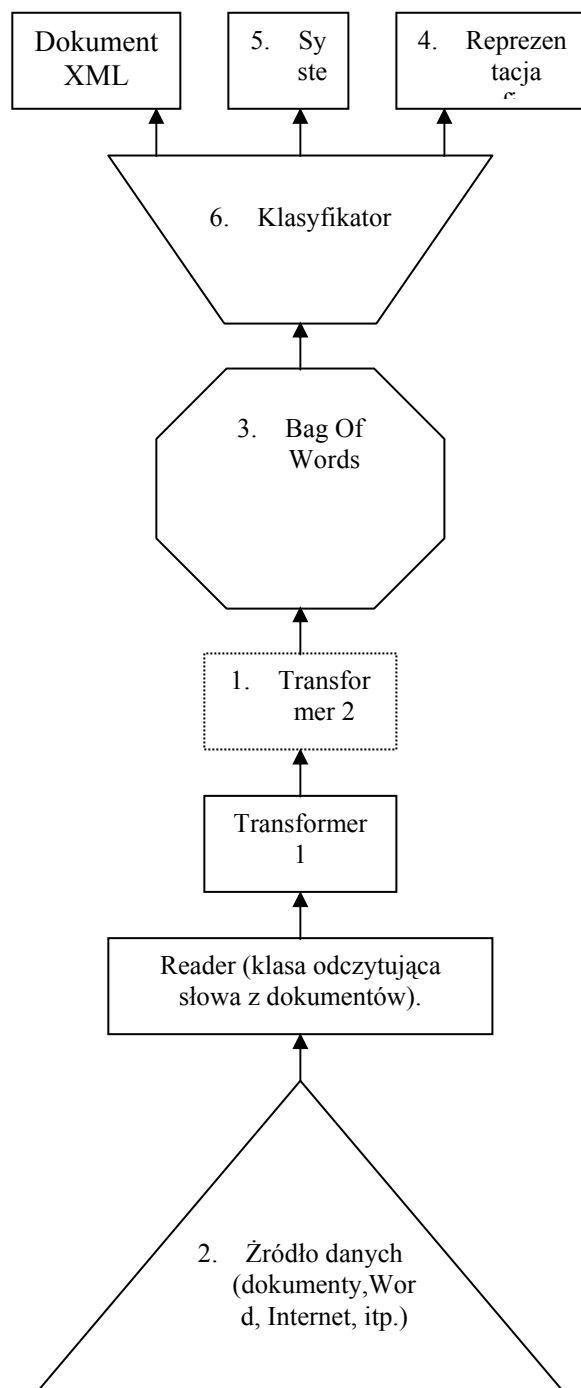
2. KONCEPCJA

Celem było stworzenie prototypowej aplikacji służącej do automatycznej klasyfikacji dokumentów, także polskich. Nie chodziło o stworzenie aplikacji z interfejsem graficznym umożliwiającą użytkownikowi klasyfikację dokumentów, ale jądra, umożliwiającego proste rozszerzanie funkcjonalności o klasyfikację dokumentów XML, HTML, stemmingu w różnych językach, a nawet zmianę sposobu przechowywania słów w systemie, co pozwoliło na badanie efektywności różnych sposobów przetwarzania danych wejściowych.

3. OGÓLNA ARCHITEKTURA

Jednym z założeń projektu, było stworzenie systemu umożliwiającego łatwe rozszerzanie o nowe źródła danych, nowe sposoby przechowywania informacji, ale także pozwalającego na dostęp nie tylko użytkownikowi poprzez interfejs graficzny, ale i innym aplikacjom. W tym celu opracowany został zestaw interfejsów odpowiadających poszczególnym częściom systemu, oraz wyraźny podział funkcjonalności pomiędzy poszczególne warstwy.

1.1. SCHEMAT APLIKACJI



Rys. 1. Schemat aplikacji.

1.2. OPIS POSZCZEGÓLNYCH KOMPONENTÓW.

3.1.1. Źródło danych.

Źródłem danych mogą być dokumenty przechowywane w dowolny sposób i w dowolnym miejscu. Zadaniem użytkownika, lub aplikacji korzystającej z usług klasyfikatora jest zapewnienie zestawu dokumentów przeznaczonych do nauki klasyfikatora, oraz wskazanie dokumentów przeznaczonych do klasyfikacji (nie muszą być z tego samego źródła).

1.2.1. Reader.

Reader jest klasą odczytującą dokument i przekazującą do dalszego przetwarzania kolejne słowa dokumentu. W zależności od implementacji, może to być może to być dokument tekstowy, plik xml, html, adres w internecie, itp.

1.2.2. Transformer.

Klasa służąca do transformacji słowa otrzymanego od Readera, lub poprzedniego transformera. Może to być implementacja stoplisty, stemmera, usuwanie znaczników, dużych liter itp. Transformery mogą być ustawiane jeden za drugim, w ustalonej kolejności, tworząc ścieżkę przetwarzania.

1.2.3. Bag Of Words.

Klasa przechowująca słowa, oraz informacje o nich, czyli w jakich dokumentach występują i ile razy. BOW oblicza też wagi poszczególnych słów w dokumencie, według odpowiedniego algorytmu.

1.2.4. Klasyfikator.

Klasyfikator jest klasą, która ma za zadanie złożyć ścieżkę przetwarzania, załadować dokumenty do klasyfikatora, a następnie klasyfikować wyznaczone dokumenty. W zależności od implementacji może to być klasyfikator opierający się na plikach, adresach internetowych, itp. Oraz prezentujący wyniki w pożądanym sposób.

2. UWAGI IMPLEMENTACYJNE

W ramach projektu stworzony został zestaw interfejsów oraz przykładowa implementacja, na której przeprowadzone zostały testy. Aplikacja stworzona została w języku JAVA, dającym obecnie najwięcej możliwości zastosowań, oraz przenośność niedostępną dla innych platform.

Najważniejszym problemem, który należy rozwiązać podczas tworzenia klasyfikatora, jest implementacja BOW'a. Ma on za zadanie przechowywać dużą liczbę słów (rzędu kilkudziesięciu tysięcy słów), pozwalając jednocześnie na szybki dostęp do ich właściwości. W tym przypadku przechowywanie za pomocą list byłoby czasowo nieefektywne, ponieważ aby dostać się do któregoś z słów, należałoby przeszukiwać w najgorszym przypadku całą listę. W tej implementacji obiekty przechowywane są za

pomocą obiektów Javy *HashMap*, implementujących mapy mieszające. Nacisk położony został na szybkość działania aplikacji, kosztem wymagań dotyczących pamięci.

2.1. BOW

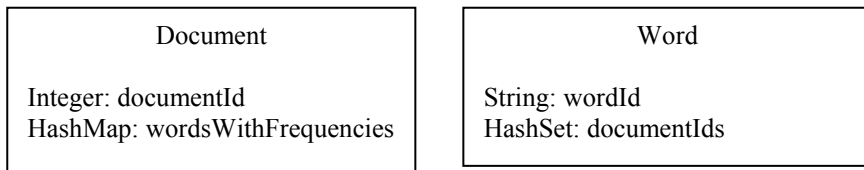
Użyty w tym projekcie algorytm klasyfikacji wymaga następujących wielkości, aby poprawnie sklasyfikować dokument:

- liczba wystąpień danego słowa w dokumencie,
- liczba dokumentów, w których występuje dane słowo,
- licznosc zbioru dokumentów,
- liczba wystąpień danego słowa w zbiorze dokumentów.

Biorąc pod uwagę najprostszą implementację, czyli przechowywanie dokumentów jako listy słów z przypisanymi do nich licznosciami, to aby obliczyć wagę dowolnego słowa, należałoby przeszukać dokument, aby pobrać liczbę wystąpień słowa w dokumencie, następnie przeszukać wszystkie dokumenty, aby sprawdzić, czy występuje w nich to słowo.

Tak więc celem było stworzenie takiej implementacji, w której wszystkie te wielkości byłyby dostępne od razu, bez przeszukiwania list.

Stworzone zostały dwie klasy pomocnicze *Document* i *Word*, przechowujące informacje charakterystyczne dla dokumentu (identyfikator dokumentu, oraz pary: słowo z danego dokumentu z liczbą jego wystąpień w tym dokumencie) i dla słowa (identyfikator słowa i identyfikatory dokumentów, w których to słowo występuje). Jak widać, ta sama informacja przechowywana jest w kilku miejscach, co nie jest optymalne pod względem zajętości pamięci, ale pozwala na bardzo szybki dostęp do potrzebnej informacji.



Jak widać dla danego obiektu natychmiast dostępne są potrzebne wielkości, jak liczba dokumentów, w których dane słowo występuje, czy liczba wystąpień słowa w dokumencie.

Obiekty te należy teraz przechować w osobnych listach pozwalających na natychmiastowy dostęp do poszczególnych elementów.

BOW przechowuje obiekty w dwóch mapach mieszających:

- identyfikator – *Document*, przechowująca dane o dokumentach załadowanych do BOW'a podczas procesu uczenia się
- identyfikator- *Word*, przechowująca słowa.

3. ALGORYTMY

3.1. PRZYGOTOWANIE DANYCH

Przed wprowadzeniem dokumentów do BOW'a należy je odpowiednio przygotować, aby uzyskać jak najlepsze wyniki. Polega to nie tylko na tym, aby np. usunąć znaczniki z tekstów HTML, można też dokonać operacji które będą miały bardzo duży wpływ na skuteczność klasyfikacji:

- pruning
- stemming

Pruning polega na usuwaniu z dokumentu słów, które nie przenoszą informacji, a nawet wprowadzają błąd. Mogą to być słowa występujące bardzo rzadko, a więc nie mające znaczenia, lub słowa występujące najczęściej, a więc wprowadzające „szum informacyjny”.

Stemming polega na transformacji słowa do postaci bazowej w celu wyeliminowania fleksyjności języka.

Są to najprostsze sposoby przygotowujące dokumenty do klasyfikacji, ale odpowiednio zastosowane dają bardzo dobre wyniki. Otwiera się tu także szerokie pole do popisu dla różnych eksperymentów mających za zadanie zwiększenie znaczenia słów najbardziej charakterystycznych dla danej kategorii na przykład użycie tezaury, który pozwoli pogrupować słowa o podobnym znaczeniu.

3.2. ALGORYTM KLASYFIKACJI

Do klasyfikacji użyty został algorytm Rocchio z reprezentacją przestrzenną oraz skalowanie TFIDF (TF – *term frequency*, IDF – *inverse document frequency*) obliczający wagi słów w oparciu o liczbę ich wystąpień [1],[2],[3].

W tym podejściu każdy dokument reprezentowany jest przez wektor, składający się z wag słów występujących w tym dokumencie. Podobne dokumenty będą miały podobne wektory.

Wagi słowa w w dokumencie d obliczana jest w następujący sposób:

$$waga = TF(w, d) * \log\left(\frac{|D|}{DF(w)}\right)$$

$TF(w, d)$ – liczba wystąpień słowa w w dokumencie d ,

$|D|$ - liczba dokumentów załadowanych do BOW'a,

$DF(w)$ – liczba dokumentów, w których to słowo wystąpiło.

W ten sposób obliczana jest waga każdego słowa w dokumencie. Uczenie się klasyfikatora polega na obliczeniu wektorów odpowiadających danej kategorii dokumentów. Wektor taki jest sumą wszystkich wektorów dokumentów należących do danej kategorii. Klasyfikacja nowego dokumentu polega na znalezieniu kategorii, której wektor jest najbardziej podobny do wektora tego dokumentu, czyli cosinus pomiędzy tymi wektorami będzie najmniejszy.

4. WYNIKI TESTÓW

Klasyfikator został przetestowany na typowych anglojęzycznych zbiorach testowych – zbiór Reuters i Newsgroups oraz dla polskich tekstów, zbiór dokumentów pobranych z portalu Onet. Testowane były czas przetwarzania, skuteczność, oraz wpływ liczby dokumentów i pruningu na skuteczność. Wszystkie czasy zostały podane w milisekundach.

4.1. REUTERS

Zbiór dokumentów Reuters, ten test został przeprowadzony bez żadnych usprawnień:

```
pruning: most frequent: 0 words, least frequent: less than 0
occurrences
```

```
documents loaded: 8066
```

```
words loaded: 26350
```

```
load duration: 1402
```

```
calculate duration: 60126
```

```
classification duration: 0
```

```
documents classified: 128
```

```
classification accuracy: 63 documents - 49.21875 percent
```

Reuters jest to zbiór bardzo krótkich notatek prasowych, składających się często z kilkudziesięciu słów, tak więc w tak dużej ilości słów, trudno było znaleźć wyrażenie odrębną kategorię. Wyniki są znacznie lepsze po zastosowaniu pruningu i zmniejszeniu liczby kategorii:

```
pruning: most frequent: 20 words, least frequent: less than 0
occurrences
```

```
documents loaded: 1843
```

```
words loaded: 12707
```

```
load duration: 952
```

```
calculate duration: 11687
```

```
classification duration: 0
```

```
documents classified: 25
```

```
classification accuracy: 24 documents - 96.0 percent
```

4.2. NEWSGROUPS

Ten zbiór też jest typowym zbiorem testowym, lecz dokumenty w nim zawarte są dłuższe:

```
pruning: most frequent: 3 words, least frequent: less than 0
occurrences
```

```
documents loaded: 1870
```

```
words loaded: 69601
```

```
load duration: 60177
```

```
calculate duration: 4757
```

```
classification duration: 20
```

```
documents classified: 130
```

```
classification accuracy: 110 documents - 84.61539 percent
```

4.3. TEKSTY POLSKIE

Teksty polskie mogą powodować większe trudności w klasyfikacji ze względu na fleksyjność języka i mnogość form odmiany wyrazów. Sugeruje to potrzebę stosowania stemmingu.

Dokumenty polskie bez stemmingu:

```
pruning: most frequent: 0 words, least frequent: less than 0
occurrences
```

```
documents loaded: 437
words loaded: 23349
```

```
load duration: 701
calculate duration: 6359
classification duration: 0
```

```
documents classified: 20
classification accuracy: 14 documents - 70.0 percent
```

I ze stemmingiem:

```
pruning: most frequent: 0 words, least frequent: less than 0
occurrences
```

```
documents loaded: 437
words loaded: 14878
```

```
load duration: 240
calculate duration: 4116
classification duration: 0
```

```
documents classified: 20
classification accuracy: 17 documents - 85.0 percent
```

Jak widać poprawa jest znacząca.

5. MOŻLIWOŚCI I ZASTOSOWANIA

Celem tego przedsięwzięcia było stworzenie aplikacji pozwalającej przetestować różne rozwiązania dotyczące klasyfikacji dokumentów. W związku z tym architektura aplikacji jest maksymalnie otwarta, stworzony został zestaw interfejsów definiujący jedynie funkcjonalność poszczególnych części klasyfikatora. Implementacja może być dowolna, wykorzystująca różne algorytmy i typy danych, przedstawiona tutaj jest jedynie przykładowa.

Jak wcześniej zostało nadmienione klasyfikator został napisany w języku Java, będącym obecnie jedynym językiem dostępnym na każdej większej platformie. Pozwala to na wykorzystanie go praktycznie w każdym miejscu, w którym klasyfikacja tekstów jest potrzebna. Może to być program pocztowy, sortujący pocztę, lub klasyfikator dokumentów internetowych uruchomiony na serwerze aplikacyjnym.

6. SŁOWNIK

Klasyfikator tekstów – aplikacja umożliwiająca przydzielanie dokumentów do różnych grup na podstawie ich zawartości. Na początku pracy klasyfikator uczy się grup i ich zawartości otrzymując zestaw dokumentów trenujących, wcześniej pogrupowanych.

Bag Of Words (BOW) – “worek słów”, struktura pozwalająca na przechowywanie dużej liczby słów, zachowując ich pochodzenie (z jakiego dokumentu) i licznosc (ile razy wystąpiło w danym dokumencie).

Ścieżka przetwarzania – zestaw klas, które potrafią otrzymane słowo przetworzyć w odpowiedni sposób i przekazać do dalszej transformacji, lub do BOW’a.

Pruning – usuwanie niepotrzebnych słów, mające na celu poprawienie skuteczności klasyfikacji. Można usuwać słowa występujące najczęściej (*most frequent*) i najrzadziej (*least frequent*).

Stemming – transformacja słowa do postaci bazowej (usunięcie czasów, przypadków, liczby) pozwalający na uzyskanie większej ilości informacji z dokumentu, skuteczny szczególnie przy klasyfikacji tekstów polskich.

Stoplista – lista słów, które są odrzucane podczas wczytywania dokumentów.

LITERATURA

- [1] JOACHIMS T., “A probabilistic analysis of the Rocchio algorithm with TDIDF for text categorization”, CMU-CS-98-118
- [2] ROCCHIO J. "Relevance Feedback in Information Retrieval", in The SMART Retrieval System: Experiments in Automatic Document Processing, Chapter 14, pages 313-323, Prentice-Hall Inc.
- [3] SALTON G., "Developments in Automatic Text Retrieval, Science, Vol. 253, pages 974-979.
- [4] SHOLOM W., WHITE B., APTE C., “Lightweight Document Clustering”, IBM T.J. Watson Research Center, 2000
- [5] YANG Y., LIU X., “A re-examination of text categorization methods”, ACM SIGIR Conference on Research and Development in Information Retrieval, New York, 1998