BIULETYN INSTYTUTU AUTOMATYKI I ROBOTYKI
NR 33, 2013

# Efektywna implementacja folksonomii

**ADAM SOBANIEC, BOHDAN MACUKOW**

Wydział Matematyki i Nauk Informacyjnych, Politechnika Warszawska,
ul. Koszykowa 75, 00-662 Warszawa
sobanieca@gmail.com, b.macukow@mini.pw.edu.pl

STRESZCZENIE: Artykuł opisuje efektywną metodę implementacji folksonomii, powszechnie wykorzystywanej metody klasyfikacji treści w wielu współczesnych serwisach internetowych. W założeniu implementacja ma umożliwiać automatyczną klasyfikację treści bez konieczności znacznych nakładów pracy ze strony administratorów i moderatorów serwisów internetowych. Całość opiera się na obserwacji, że dzięki użyciu słów kluczowych do oznaczenia treści, każdy element można zamienić na odpowiadający wektor binarny. Następnie tak otrzymany zbiór można poddać procesowi klasyfikacji i na jego podstawie wyznaczyć podobne treści. Przedstawiona w artykule metoda implementacji jest usprawnieniem rozwiązania z wykorzystaniem sieci neuronowej ART-1. Metoda wykorzystująca sieć neuronową ART-1 wymagała zbyt dużej ilości pamięci w przypadku znacznej ilości danych poddanych klasyfikacji. Przedstawione rozwiązanie osiąga znacznie lepsze rezultaty w kwestii wykorzystania pamięci, oraz prędkości obliczeń. Jednocześnie jakość klasyfikacji pozostaje na tym samym poziomie.

SŁOWA KLUCZOWE: folksonomia, tagowanie, klasyfikacja

## 1. Folksonomia we współczesnym internecie

Oblicze internetu uległo znacznej zmianie na początku 21'ego wieku. Medium prezentujące tylko statyczne treści, które rzadko kiedy były aktualizowane, za sprawą tzw. technologii Web 2.0 zyskało nową formę. Dodawanie nowej treści do strony internetowej stało się znacznie łatwiejsze i nie wymagało znajomości zaawansowanych technologii. Największa jednak zmiana dotyczyła interaktywności. Użytkownicy dostali możliwość wpływania na kształt aplikacji internetowej. Zaczęły powstawać aplikacje, które były w większości tworzone przez użytkowników, takie jak np. fora internetowe. Ta zmiana sprawiła, że w internecie zaczęły pojawiać się terabajty informacji, a serwisy internetowe zyskały zupełnie nowe oblicze. Liczba ludzi korzystających

z internetu z roku na rok zaczęła rosnąć, a twórcy aplikacji zaczęli prześcigać się w pomysłach na wykorzystanie potencjału Web 2.0. W efekcie strony internetowe zyskały bardzo wiele nowych funkcjonalności.

Jedną z nich stało się wyświetlanie użytkownikowi podobnych treści. Ma to na celu jak najdłuższe zatrzymanie i zaabsorbowanie internauty co w konsekwencji zwiększa popularność (i wartość) witryny. Jako przykład można przedstawić aplikację do udostępniania wideo - Youtube.com. Kiedy użytkownik zaczyna oglądać konkretny klip zostaje mu dodatkowo wyświetlona lista podobnych materiałów. Dzięki temu wzrasta szansa, że spędzi on więcej czasu na stronie niż pierwotnie zakładał.


**Rys. 1. Przykład wyświetlenia podobnej treści w serwisie Youtube.com**

W przedstawionym powyżej przykładzie widać jak użytkownikowi oglądającemu film reklamujący samochód Honda Civic wyświetlane są podpowiedzi z innymi filmami które mogą go zainteresować. W tym przypadku są to reklamy Hondy Civic na przestrzeni kilkunastu lat. Przykładów wyświetlania podobnych treści użytkownikowi można przytoczyć bardzo wiele. Większość współczesnych serwisów internetowych w których użytkownicy umieszczają treści oferuje taką funkcjonalność. Dotyczy to nie tylko udostępnianych filmów, ale również plików, muzyki, artykułów, zdjęć i wielu innych.

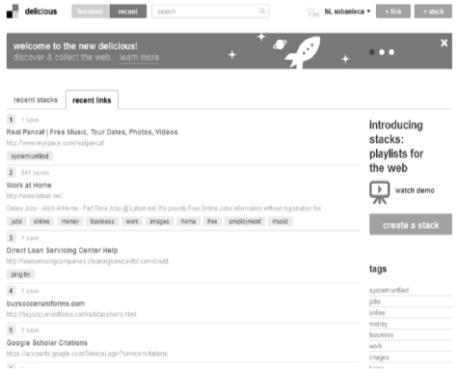
Aby wyświetlić użytkownikowi podobne treści, należy przeprowadzić odpowiedni proces klasyfikacji. Najpopularniejszym przypadkiem klasyfikacji jest klasyfikacja z użyciem słów kluczowych tzw. tagów. Przypisane słowa kluczowe stanowią cechy danego rekordu na podstawie których można odnaleźć

podobne rekordy. Jako przykład niech posłużą poniższe zdjęcia z przypisanymi słowami kluczowymi:


**Rys. 3. Trzy zdjęcia oznaczone słowami kluczowymi (tagami)**

Zdjęcia A i B można oznaczyć jako podobne ponieważ zawierają wspólne słowa kluczowe (w tym przypadku „Kwiat" i „Roślina").

Metoda przypisywania słów kluczowych stała się w internecie bardzo popularna. Jeżeli chodzi o sam proces przypisywania tagów do danej treści to odbywa się on z wykorzystaniem pomocy samych użytkowników. Każdy użytkownik, który wgrywa daną treść, proszony jest o podanie odpowiednich słów kluczowych. Jest to tzw. folksonomia, czyli społeczne tagowanie. Słowo folksonomia powstało z połączenia słów folks (ludzie) i taksonomia (nauka kategoryzacji treści). Jedną z pierwszych aplikacji web 2.0, która wykorzystywała folksonomię do kategoryzacji treści był serwis Delicious. Strona ta umożliwia użytkownikom dodawanie własnych zakładek stron internetowych na serwerze, dzięki czemu mają oni dostęp do nich z każdego miejsca na świecie.




**Rys. 4. Serwis delicious.com był jednym w pierwszych serwisów wykorzystujących folksonomię**

Wadą tej metody kategoryzacji jest podatność na nadużycia i błędy samych użytkowników. Możliwa jest sytuacja w której dwie podobne treści zostaną rozpoznane jako różne gdyż do ich opisania użyto czasowników w innej formie lub z powodu błędów gramatycznych. Z tego też powodu twórcy aplikacji starają się implementować mechanizmy, które minimalizują możliwość wystąpienia takiej sytuacji.

W przypadku folksonomii treści dostępne w serwisie nie są zhierarchizowane. Nie jest budowane żadne drzewo kategorii. Zamiast tego dane są podzielone na grupy treści z podobnymi słowami kluczowymi, które zostały do nich przypisane.

## 2. Transformacja do wektorów binarnych

Każda treść oznaczona słowami kluczowymi może zostać opisana przez odpowiadający wektor binarny. Transformacja odbywa się według następującego algorytmu:

Dane:

$D=\{d_1, d_2, \ldots, d_N\}$ - zbiór wszystkich słów kluczowych

$T=\{t_1, t_2, \ldots, t_M\}$ - słowa kluczowe przypisane do danego zdjęcia

$v=\{v_1, v_2, \ldots, v_N\}$ - wynikowy wektor

```
dla każdego d_i w zbiorze D
jeżeli d_i ∈ T
     ustaw v_i = 1
w przeciwnym razie
     ustaw v_i = 0
```

Załóżmy, że w systemie zostały wprowadzone następujące słowa kluczowe: „**Samochód**”, „**Kot**”, „**Pies**”, „**Ferrari**”, „**Czerwony**”, „**Niebieski**”, „**2012**”

Dla zdjęcia o przypisanych słowach kluczowych „**Ferrari**”, „**Samochód**”, „**2012**”:

**Rys. 6. Zdjęcie wgrywane przez użytkownika.**

Odpowiadający wektor binarny to **[1, 0, 0, 1, 0, 0, 1]**

Otrzymawszy zbiór wektorów binarnych możliwe jest przeprowadzenie klasyfikacji wszystkich treści i w konsekwencji rozpoznanie podobnych elementów.

## 3. Algorytm klasyfikacji

Główna idea algorytmu opiera się na podobnych założeniach jak klasyfikacja z wykorzystaniem sieci neuronowej ART-1 [1]. Ponieważ rozwiązanie oparte na sieci ART-1, w przypadku większej ilości danych poddanych klasyfikacji (ponad 100 tys.) wymagało zbyt dużej ilości pamięci zastosowanie go we współczesnych serwisach internetowych było niemożliwe.

Prezentowany algorytm został pozbawiony tej wady. Zostało to osiągnięte przez wyeliminowanie połączeń między warstwą wejściową, a wyjściową sieci [2], oraz modyfikację algorytmu uczenia. Cała modyfikacja nie wpłynęła negatywnie na jakość klasyfikacji.

Podczas pracy algorytmu tworzony jest zbiór klastrów z odpowiadającymi im prototypami, a następnie każdy wektor binarny zostaje przypisany do odpowiedniego klastra bazując na tzw. prototypie. Różnica polega na tym, że zamiast konstruować sieć neuronową z połączeniami między warstwą wejściową a wyjściową, konstruowany jest jednowarstwowy zbiór klastrów, gdzie każdy klaster posiada tzw. prototyp na podstawie którego obliczana jest odległość od wektora wejściowego.

Algorytm można opisać przy użyciu poniższego pseudokodu:

Dane:

$V = \{v_1, v_2, \ldots, v_N\}$ - zbiór wektorów binarnych

$C = \{c_1, c_2, \ldots, c_M\}$ - zbiór klastrów (na początku pusty)

```
dla każdego v_i w zbiorze V
    dla każdego c_i w zbiorze C
        weź p - prototyp c_i
        jeżeli dist(v_i, p) ≥ ρ
            przypisz v_i do c_i
            uaktualnij prototyp p = p ∧ v_i
    jeżeli nie znaleziono odpowiadającego klastra
        utwórz nowy klaster c_j
        ustaw jego prototyp p = v_i
        dodaj c_j do zbioru C
        przypisz v_i do c_j
```

Funkcja $dist(v_i, v_j)$ jest to funkcja zwracająca liczbę wspólnych „1” między dwoma wektorami. Wartość $\rho$ definiuje próg powyżej którego wektory są uznane za podobne. Do zadowalających wyników wystarcza ustawienie tej wartości na „2”, czyli przykładowo, dwa zdjęcia zostaną oznaczone jako

podobne tylko wtedy gdy będą miały wspólne przynajmniej dwa słowa kluczowe.

W celu przetestowania jakości tego rozwiązania zostało pobranych 10 000 zdjęć z serwisu Flickr.com. Ze względu na brak ograniczeń dla słów kluczowych, zdjęcia były oznaczone w sposób dość chaotyczny. Jednak pobrana próbka zawierała wystarczającą ilość podobnych zdjęć (oznaczonych podobnymi słowami kluczowymi), aby zweryfikować poprawność klasyfikacji. W rezultacie wyniki prezentowały się następująco:

**Tab. 1. Wyniki działania algorytmu**

| | |
|---|---|
| Liczba zdjęć: | 10 000 |
| Czas klasyfikacji: | 5 995 ms |
| Wykorzystanie pamięci: | 485 kB |
| Liczba klastrów: | 3184 |
| Klastry z więcej niż 1 zdjęciem | 1719 |
| Klastry z dokładnie jednym zdjęciem | 1465 |

Wszystkie obliczenia zostały wykonane na komputerze wyposażonym w procesor Core 2 Duo P8700 2,53 GHz. Wyniki klasyfikacji można obejrzeć na stronie http://folksonomizer.sobaniec.com.

Po przeprowadzeniu klasyfikacji następuje zapisanie wyników w bazie danych, tzn. zapisanie która fotografia trafiła do którego klastra i dzięki temu w momencie wyświetlania odpowiedniej treści użytkownikowi możliwe jest wykonanie jednego prostego zapytania, aby pozyskać rekordy oznaczone jako podobne. Ma to olbrzymie znaczenie zwłaszcza w najbardziej obciążonych serwisach internetowych, gdzie każde połączenie z bazą danych jest bardzo kosztowne.

## 4. Porównanie z algorytmem ART-1

Aby wykazać jak znaczne usprawnienie zostało wprowadzone, przeprowadzono symulację klasyfikacji z wykorzystaniem sieci neuronowej ART-1 zaimplementowanej według B.Moore [2] na tym samym zbiorze testowym.

**Tab. 2. Wyniki działania algorytmu ART-1**

| | |
|---|---|
| Liczba zdjęć: | 10 000 |
| Czas klasyfikacji: | 26 minut |
| Wykorzystanie pamięci: | 116 MB |
| Liczba klastrów: | 3235 |
| Klastry z więcej niż 1 | 1805 |

| | |
|---|---|
| zdjęciem<br>Klastry z dokładnie jednym zdjęciem | 1430 |

Obliczenia zostały wykonane na tym samym komputerze z tym samym procesorem. Różnica w czasie wykonania, oraz wykorzystaniu pamięci była na tyle znaczna, że kwestia większej ilości klastrów z więcej niż jednym zdjęciem nie miała żadnego znaczenia.

## 5. Wydajność algorytmu dla dużych zbiorów danych

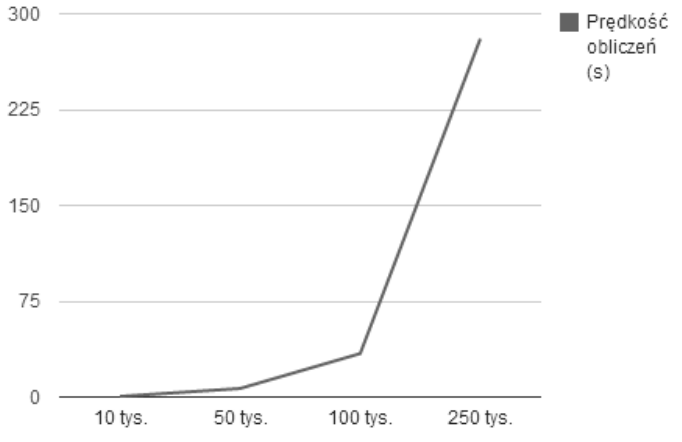
Porównanie z algorytmem ART-1 dało nadzieję na znaczną poprawę wydajności klasyfikacji w przypadku większej ilości danych wejściowych. W celu określenia jak zachowuje się algorytm, zostały przeprowadzone testy klasyfikacji dla odpowiednio: 10 tys., 50 tys., 100 tys. i 250 tys. zdjęć. Zbiór testowy został wygenerowany losowo w taki sposób, aby zawierał zdjęcia ze wspólnymi słowami kluczowymi. Otrzymane wyniki w prędkości działania prezentowały się następująco:

**Tab. 3. Czas działania dla różnego rozmiaru danych wejściowych**

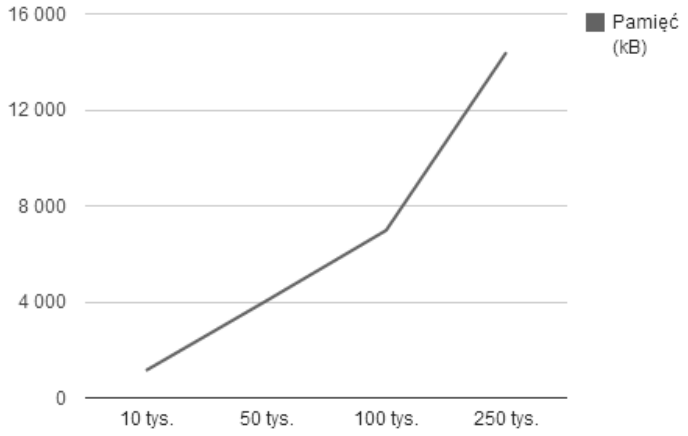
| | |
|---|---|
| 10 tys. zdjęć | 468 ms |
| 50 tys. zdjęć | 6648 ms |
| 100 tys. zdjęć | 33986 ms |
| 250 tys. zdjęć | 280421 ms |

**Tab. 4. Zużycie pamięci dla różnego rozmiaru danych wejściowych**

| | |
|---|---|
| 10 tys. zdjęć | 1138 kB |
| 50 tys. zdjęć | 4035 kB |
| 100 tys. zdjęć | 6980 kB |
| 250 tys. zdjęć | 14397 kB |



**Rys. 7. Czas działania algorytmu dla różnego rozmiaru danych wejściowych**




**Rys. 8. Zużycie pamięci dla różnego rozmiaru danych wejściowych**

Zużycie pamięci w przypadku przedstawionego algorytmu wzrasta liniowo i utrzymuje się na akceptowalnym poziomie. Natomiast w przypadku wzrostu ilości danych czas potrzebny na wykonanie wszystkich obliczeń wzrasta wykładniczo. W związku z tym w przypadku większych serwisów internetowych niemożliwe staje się wykorzystanie algorytmu do przeprowadzenia obliczeń w czasie rzeczywistym (np. gdy użytkownik zaczyna przeglądać daną treść – zdjęcie, artykuł itp.). W przypadku implementacji w rzeczywistych systemach, konieczne byłoby stworzenie oddzielnej instancji aplikacji, która przeprowadzałaby klasyfikację „w tle" i zapisywała jej wyniki do odpowiedniej tabeli w bazie danych.

## 6. Podsumowanie

Celem niniejszego artykułu było przedstawienie algorytmu, który znalazłby zastosowanie w wielu współczesnych serwisach internetowych, niezależnie od ilości danych jakie poddane miałyby być klasyfikacji. Przedstawione rozwiązanie wykazało znaczne usprawnienie względem rozwiązania opartego na standardowej sieci neuronowej ART-1. Obliczenia były wykonywane znacznie szybciej, a zużycie pamięci znacznie zmalało.

Jednocześnie jakość klasyfikacji pozostała na tym samym, wysokim, poziomie. Otrzymane wyniki wskazały jednak, że w przypadku, gdy aplikacja internetowa posiada znaczną ilość danych (rzędu milionów, miliardów wierszy), złożoność algorytmu utrudnia jego wykorzystanie. Rozwiązaniem mogłoby być zaprojektowanie wersji algorytmu opartej na obliczeniach równoległych. Przykładowo, implementacja z wykorzystaniem karty graficznej (CUDA) powinna umożliwić osiągniecie znacznie większej prędkości działania poprzez rozdzielenie obliczeń pomiędzy kolejne ko-procesory.

## Literatura

[1] SOBANIEC Adam, MACUKOW Bohdan, *Folksonomy implementation based on the ART-1 neural network*, FedCSIS 2012.

[2] B. Moore. "ART and pattern clustering", In: *Proceedings of the 1988 Connectionist Models Summer School*, pp. 174-183, 1988.

[3] G.A. Carpenter, S. Grossberg "Adaptive Resonance Theory", Cambridge, MIT Press, 2003.

[4] F. Hao, S. Zhong "ECKDF: Extended Conceptual Knowledge Discovery in Folksonomy", ICCP Proceedings, 2010.

[5] H. Kawakubo, Y. Akima, K. Yanai "Automatic Construction of A Folksonomy-based Visual Ontology", IEEE International Symposium on Multimedia, 2010.

[6] L. Massey "On the quality of ART1 text clustering", Elsevier Science Ltd., 2003.

[7] L. Massey "Determination of Clustering Tendency With ART Neural Networks", Intl. Conf. on Recent Advances in Soft Computing, 2002.

[8] I. Peters "Folksonomies: Indexing and retrieval in Web 2.0", Saur K.G. Verlag GmbH, 2009.

[9] J. Pipes „Tagging and Folksonomy Schema Design for Scalability and Performance", MySQL, Inc., 2006.

# Effective folksonomy implementation

ABSTRACT: This paper describes effective method of folksonomy implementation. Folksonomy is widely used classification method in many modern web sites. Presented implementation is supposed to allow automatic content classification without need to perform additional work from web administrators and moderators. Basing on the keywords, there may be many content types classified. Not only plain text, but also photographs, movies and music. Each content may be transformed into corresponding binary vector. Having set of such vectors, one may perform the classification process. Presented method is an improvement over ART-1 based classification, since ART-1 method occupied too much memory when classifying large data sets. Presented improved decreased the amount of time needed to perform calculations while it didn't decrease the quality of whole classification.

KEYWORDS: folksonomy, tagging, classification