

*Słowa kluczowe: klasyfikacja tekstu, Internet,
Open Source, Java.*

Tomasz ŁUKASZEWICZ*
Łukasz SZNUK⁺

System monitorowania opinii o produktach w Internecie

W pracy opisano system klasyfikowania opinii o produktach i doświadczenia zdobyte przy jego budowie. System powstał w języku Java, z użyciem kilku bibliotek o dostępnym kodzie źródłowym (Open Source). Interfejs użytkownika jest zrealizowany poprzez strony WWW. Opinie dzielone są na pozytywne i negatywne za pomocą statystycznego algorytmu TFIDF. Wyniki działania programu są satysfakcjonujące.

1. Wprowadzenie

W pracy przedstawiamy zrealizowany przez nas system klasyfikacji opinii o produktach. System nasz z założenia jest ogólny, ale zdecydowaliśmy się na analizę informacji dotyczących telefonów komórkowych, w Internecie bowiem można znaleźć wiele stron poświęconych dyskusjom na ich temat. Dodatkowo, dyskusje dotyczące różnych aparatów, nawet różnych producentów, są w sensie naszych potrzeb na tyle do siebie podobne, że można z nich korzystać nie zwracając uwagi na te różnice, co pozwala łatwo stworzyć zbiór danych do uczenia systemu o wielkości umożliwiającej skorzystanie ze statystycznych algorytmów klasyfikacji.

Pracom nad systemem przyświecały dwa główne cele. Po pierwsze, chcieliśmy sprawdzić, na ile skomplikowane jest stworzenie systemu pozwalającego na testowanie algorytmów klasyfikacji opinii korzystając z dostępnych komponentów Open Source. Po drugie, mieliśmy na celu stwierdzenie, czy statystyczne algorytmy klasyfikacji – z ewentualnymi poprawkami – nadają się do klasyfikowania opinii w języku polskim.

* NuTech Solutions Polska Sp. z o.o., ul. Żołą 35, Warszawa <tomasz@lukaszewicz.net>

⁺ Instytut Informatyki, Uniwersytet Warszawski, ul. Banacha 2, Warszawa <luke@mimuw.edu.pl>

Poniżej opisujemy nasze doświadczenia związane z tworzeniem systemu oraz wyniki przeprowadzonych przez nas testów.

2. Klasyfikacja opinii

Często, gdy zachodzi potrzeba podjęcia decyzji o zakupie jakiegoś produktu, zbiera się informacje na jego temat. Obecnie popularne jest wyszukiwanie takich informacji w Internecie. Można w nim znaleźć nie tylko klasyczne testy napisane przez profesjonalistów, ale też krótkie recenzje pisane przez użytkowników danego produktu. Ilość takich informacji może być przytłaczająca, czytanie ich jest raczej nużące, a dobrze byłoby wiedzieć, ile z nich zawiera pozytywną, a ile negatywną opinię o danym przedmiocie. My zainteresowaliśmy się telefonami komórkowymi – na rynku dostępnych jest wiele modeli, a o każdym można znaleźć bardzo wiele opinii. O ile nie ma raczej sensu automatycznie klasyfikować dogłębnych, profesjonalnych recenzji – ich liczba jest stosunkowo niewielka, a na dodatek recenzenci opisują tak wiele cech, z których każda może mieć inną ocenę, że nawet inteligentnemu człowiekowi trudno jest stwierdzić czy opinia jest pozytywna czy negatywna – to opinie wyrażane przez internautów są dobrym tematem badań. Większość takich „recenzji” jest stosunkowo krótka i zawiera zdecydowane stanowisko – „tak, aparat jest świetny”, albo „nie, nie nadaje się do niczego”. W dyskusjach nie ma zbyt wielu uwag merytorycznych, zwłaszcza wśród opinii negatywnych, zazwyczaj wyrażanych przez osoby będące zwolennikami innego modelu telefonu. Te cechy przemawiają za automatycznym klasyfikowaniem opinii.

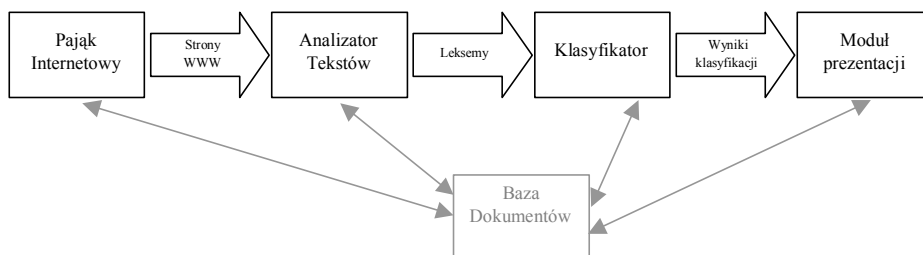
Oprócz klasyfikowania opinii o produktach podobny system może być stosowany m.in. do sprawdzania, czy przychodzący pocztą elektroniczną list zawiera informacje handlowe i czy, w związku z tym, należy go dostarczyć czy odrzucić. Oczywistym zastosowaniem jest też opisane dla języka angielskiego ocenianie recenzji filmów, np. w pracy [2].

Do klasyfikacji wymowy tekstu można podejść na kilka sposobów. Popularna metoda stosowana dla języka angielskiego polega na zastosowaniu technik przetwarzania języka naturalnego do oddzielnego klasyfikowania wymowy (pozytywnej/negatywnej) pojedynczych zdań, a następnie wnioskowania o wymowie całego tekstu na podstawie tych ocen (m.in. w [3]). Nie skorzystaliśmy z tego podejścia, ze względu na różnice między językiem angielskim a polskim i bardzo duży nakład pracy wymagany do implementacji takiego systemu (nie znaleźliśmy żadnych bibliotek Open Source operujących na języku, które mogłyby ułatwić nam pracę). Zdecydowaliśmy się więc na ocenę tekstu polegającą na jego klasyfikacji metodami statystycznymi. Taka metoda była już z powodzeniem stosowana dla języka angielskiego (w [2]), a jej szczegóły przedstawiamy w punkcie 4.

3. Architektura systemu

Ze względów praktycznych do implementacji systemu wybraliśmy język Java. Dzięki dużej liczbie dostępnych bibliotek umożliwiło to szybkie i stosunkowo proste stworzenie systemu. Kolejna cecha Javy, przenośność, pozwala uruchamiać system na wielu różnych platformach. Interfejs systemu zrealizowany jest poprzez strony WWW, co pozwala na zdalny dostęp do niego.

System składa się z czterech głównych modułów: pająka internetowego, zbierającego teksty do analizy, analizatora tekstów przygotowującego zebrane opinie do klasyfikacji, klasyfikatora przydzielającego opinie do właściwej grupy i modułu prezentacji wyników. Dokumenty przechowywane są po prostu jako oddzielne pliki w systemie; do przetwarzania większej liczby dokumentów można skorzystać z bazy danych. Na poniższej ilustracji przedstawiamy architekturę systemu.



Rys. 1. Architektura systemu
Fig. 1. System Architecture

Pająk internetowy i moduł prezentacji nie są, z punktu widzenia naszej pracy, szczególnie ciekawe, przedstawiamy więc tylko krótki opis. Funkcje pająka ograniczają się do ściągania nowych dokumentów z kilku predefiniowanych serwisów (www.gsmonline.pl, www.telix.pl). Zdecydowaliśmy się na to rozwiązanie, gdyż było ono stosunkowo proste, a do naszych testów w zupełności wystarczało. Pająk został zrealizowany z pomocą biblioteki WebSPHINX, co pozwoliło łatwo zaprogramować wyszukiwanie nowych recenzji i przekazywanie ich do analizatora. Działa on asynchronicznie, w ustalonych odstępach czasu.

Moduł prezentacji pozwala wybrać interesujący nas aparat telefoniczny i prezentuje wyniki klasyfikowania dokumentów dotyczących tego aparatu. Pozwala też przejrzeć dokumenty zaklasyfikowane jako pozytywne i negatywne. Do stworzenia modułu prezentacji skorzystaliśmy z technologii JSP i servletów, standardowych w Javie.

Analizator tekstu i moduł klasyfikatora są ze sobą ściśle powiązane, bowiem działania, które należy przeprowadzić na tekście, zależą istotnie od wybranej metody klasyfikacji. Analizator dokonuje wstępnej obróbki zebranych dokumentów. Na wejściu dostaje pojedyncze opinie, a jego wyjściem jest strumień obrobionych leksemów.

Początkowo sądziliśmy, że wystarczy skorzystać z biblioteki Lucene, która służy nam do podziału dokumentów na leksemy i wydajnych operacji na kolejnych leksemach, oraz z lematyzatora (dostępnego w Internecie jako biblioteka do Javy lematyzatora D. Weissa) sprowadzającego słowo do postaci bazowej. Planowaliśmy też usuwanie wszystkich często spotykanych słów (ang. stop-words), co jest standardową techniką używaną przy statystycznym klasyfikowaniu dokumentów.

Okazało się jednak, że ściągane z Internetu dokumenty są obarczone dwiema wadami. Pierwszą z nich jest to, że wiele z nich nie zawiera polskich znaków. Z tym problemem próbowaliśmy sobie poradzić na kilka sposobów. Zmodyfikowanie lematyzatora tak, by akceptował zarówno formy z polskimi literami jak i bez, odrzuciliśmy ze względu na pracochłonność – wymagałoby to zmodyfikowania całego słownika. Kolejna próba polegała na generowaniu wszystkich możliwych podstawień liter polskich za odpowiadające im łacińskie (np. ład przerabiany jest na lad, ład, ład, ład) i wysyłanie wszystkich takich słów do lematyzatora z nadzieją, że któreś z nich będzie mu znane. To rozwiązanie było stosunkowo mało wydajne (sprawdzenie jednego słowa z 3 literami, które można było zastąpić, z których jedną jest „z”, wymaga sprawdzenia 12 słów), a na dodatek problemy techniczne związane z lematyzatorem ograniczyły jego przydatność.

Żadne z tych rozwiązań nie pomagało też rozwiązać drugiego z problemów – wiele z tekstów w Internecie nie stosuje się do reguł polskiej ortografii. Ostatecznie zastosowaliśmy rozwiązanie, które pomagało przezwyciężyć oba problemy. Skorzystaliśmy ze słownika ortograficznego. Każde ze słów, jeszcze przed lematyzacją, jest sprawdzane i ewentualnie poprawiane. Rzecz jasna, gdy wejściem jest „ład”, nie wiadomo czy poprawna forma to „ład”, czy „ład”. Ponieważ jednak słownik działa deterministycznie, wszystkie zapisane bez polskich znaków wystąpienia danego słowa sprowadzone zostaną do tej samej formy. Wybrane słowo może nie być tym właściwym, ale bez zastosowania bardziej złożonych metod, na przykład analizy semantycznej, nie da się nic na to poradzić. Słownik, którego użyliśmy, pochodził z jednego z dostępnych na rynku komercyjnych pakietów biurowych. Udostępniliśmy go za pomocą usługi WWW. Próbowaliśmy skorzystać z dostępnego darmowego programu `ispell`, ale osiągnięte rezultaty nie były zachęcające, za to zastosowany słownik komercyjny spisywał się znakomicie.

Ostatni moduł programu, klasyfikator, jest szczegółowo opisany w kolejnym punkcie.

4. Metoda klasyfikacji

Zastosowana przez nas metoda klasyfikacji to opisywany m.in. w [1] algorytm TFIDF. Uznaliśmy, że implementacja tej metody jest stosunkowo nieskomplikowana, a

z publikacji wynika, że dla wystarczająco dużego zbioru danych wejściowych osiągnane wyniki są zbliżone do wyników uzyskiwanych za pomocą bardziej złożonych algorytmów.

Celem działania algorytmu jest, w naszym przypadku, przypisanie dokumentu do jednej z dwóch kategorii: „pozytywny” lub „negatywny”. Decyzja ta oparta jest na podobieństwie klasyfikowanego dokumentu do dokumentów wcześniej arbitralnie zakwalifikowanych do jednej z tych kategorii. Ścisłej, załóżmy, że początkowo dysponujemy pewnym zbiorem D dokumentów, które są wstępnie zakwalifikowane. Niech ta klasyfikacja będzie funkcją $c: D \rightarrow \{P, N\}$, gdzie P i N oznaczają, odpowiednio, ocenę pozytywną i negatywną. Zadanie polega więc na przyporządkowaniu nowo otrzymanego dokumentu d do P albo N . W tym celu trzeba wyznaczyć, najlepiej jako liczbę, podobieństwo d do dokumentów, dla których wartością c jest P i do tych, dla których wartością jest N .

Tekst dokumentu reprezentowany jest jako wektor słów w formach bazowych, wraz ze współczynnikami ważności tych słów. Waga każdego słowa jest zależna między innymi od liczby wystąpień tego słowa w tekście. Nie jest istotna kolejność wystąpień w oryginalnym tekście, ale słowa w wektorach muszą być uporządkowane tak, by na tej samej pozycji występowało to samo słowo. Przy takiej reprezentacji podobieństwo dwóch dokumentów to podobieństwo wektorów je reprezentujących.

Intuicyjnie, liczba na i -tej pozycji w wektorze, reprezentująca słowo s_i oznacza na ile istotne jest to słowo dla danego dokumentu. Słowo występujące w danym dokumencie stosunkowo często, czyli charakterystyczne dla dokumentu, powinno być ważne, ale jeżeli występuje równie często w wielu innych dokumentach, to nie jest ono wyjątkowe i trzeba jednak zmniejszyć jego wagę. Formalnie, niech $TF(s_i, t)$, od angielskiego Term Frequency, oznacza liczbę wystąpień słowa s_i w tekście t , a $DF(s_i)$, od Document Frequency, oznacza liczbę dokumentów z pewnego ustalonego zbioru D , w których słowo s_i występuje. Wówczas można zdefiniować wartość IDF oznaczającą odwrotność częstości występowania słowa w dokumentach (ang. Inverse Document Frequency) jako:

$$IDF(s_i) = \log\left(\frac{|D|}{DF(s_i)}\right)$$

a wagę słowa s_i w tekście t jako:

$$TFIDF(s_i, t) = TF(s_i, t) * IDF(s_i)$$

Podobieństwo dwóch wektorów łatwo jest wyliczyć jako cosinus kąta pomiędzy nimi, ale potrzebne jest jeszcze określenie, w jaki sposób wyznaczyć wektor dla zbioru dokumentów. W naszej pracy przyjęliśmy, że wektor ten jest sumą wektorów dla wszystkich dokumentów należących do zbioru.

Problemem pojawiającym się przy ocenie wymowy opinii jest to, że nie wszystkie z nich muszą zawierać charakterystyczne słowa. W szczególności, rozważmy zdanie

„Ten telefon nie był dobry”. Słowo „nie” zostanie wyeliminowane na etapie usuwania nieistotnych wyrazów i zdanie to nie będzie się wcale różniło od „Ten telefon był dobry”. Oczywiście, chcielibyśmy, by dokumenty z tymi zdaniami znalazły się w różnych kategoriach. Zastosowane rozwiązanie pochodzi z pracy [2] i polega na propagowaniu słowa „nie” aż do najbliższego końca zdania bądź przecinka. Po modyfikacji zdanie zostanie przekształcone na „Ten telefon niebył niedobry”. To zdanie zawiera wprawdzie niepoprawne słowo – „niebył” – ale w sensie klasyfikacji zupełnie nam to nie przeszkadza. Natomiast zamiast słowa „dobry” pojawiło się słowo „niedobry”, które zapewne będzie charakterystyczne dla negatywnych opinii. Niestety, jak pokażą dalej testy, metoda ta nie dała nam wymiernych korzyści.

Przedstawione poniżej wyniki testów systemu wykazują, że zastosowanie podejścia statystycznego do klasyfikacji tego rodzaju opinii jest akceptowalne.

5. Wyniki Testów

Aby dokument był łatwy do zaklasyfikowania przy pomocy opisywanego systemu, powinien zawierać dokładnie jedną opinię dotyczącą pewnej firmy lub produktu. Cały jego tekst powinien być tą opinią; najlepiej, żeby nie zawierał żadnych dodatkowych informacji ponad sformułowania wartościujące. Autorzy zdecydowali się zastosować w eksperymentach dokumenty zawierające opinie telefonów komórkowych, pozyskane z serwisów poświęconych telefonii gsm – Telix (www.telix.pl) i GSM Online (www.gsmonline.pl). Wymienione serwisy zawierają krótkie i spójne opinie dotyczące telefonów komórkowych. Opinie te są zamieszczane przez użytkowników serwisu na tzw. tablicach dyskusyjnych poświęconych poszczególnym modelom telefonów, jedna za drugą. Na potrzeby eksperymentu odfiltrowano wypowiedzi nie zawierające ocen oraz wypowiedzi o niejasnym wydźwięku. W wyniku tych przygotowań otrzymano ostatecznie 328 opinii, w tym 145 negatywnych i 183 pozytywne.

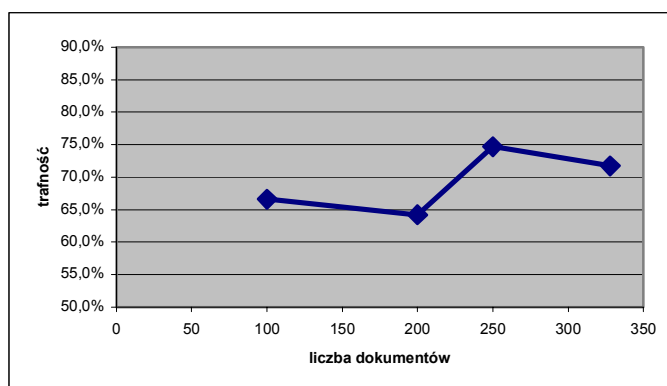
Pewnym mankamentem jest długość tych opinii, wynosząca przeciętnie 207 bajtów, co odpowiada przeciętnie 2.5 linii tekstu. Tak krótkie opinie są trudne do klasyfikacji ze względu na to, że wyuczony na nich klasyfikator będzie posiadał ubogi słownik. Jednakże zaletą tak przygotowanych danych wejściowych jest ich spójność i zdecydowany wydźwięk (pozytywny lub negatywny). Proporcja liczby dokumentów negatywnych do liczby dokumentów pozytywnych odpowiada w przybliżeniu tendencji zauważalnej w obu serwisach, w których dominują jednak dokumenty pozytywne.

Przeprowadzono testy klasyfikatora na 328, 250, 200 i 100 dokumentach. Dla każdej liczby dokumentów przeprowadzono 3 testy i uśredniono wyniki. Każdy pojedynczy test na n dokumentach przebiegał w sposób następujący: z ogólnej liczby 145 negatywnych i 183 pozytywnych dokumentów losowano $n * 37\%$ dokumentów negatywnych i $n * 49\%$ dokumentów pozytywnych. Na tych dokumentach uczono

klasyfikator. Następnie losowano $n * 6.9\%$ dokumentów negatywnych i $n * 7.1\%$ dokumentów pozytywnych, na których testowano klasyfikator (dokumenty testowe były oczywiście rozłączne z dokumentami treningowymi). Wszystkie testy były przeprowadzane zarówno dla klasyfikatora w wersji bez przerzucania negacji, jak i z przerzucaniem negacji. Poniższa tabela przedstawia uzyskane wyniki.

| Liczba dokumentów | Trafność bez przerzucania negacji [%] | Trafność z przerzucaniem negacji [%] | Średnia trafność [%] |
|-------------------|---------------------------------------|--------------------------------------|----------------------|
| 100 | 66.6 | 66.6 | 66.6 |
| 200 | 65.4 | 63.0 | 64.2 |
| 250 | 75.2 | 74.2 | 74.7 |
| 328 | 70.2 | 73.1 | 71.7 |

Z powyższego zestawienia wynika, że przerzucanie negacji nie wywiera wpływu na jakość działania systemu. Na poniższym wykresie zaprezentowano średnią trafność systemu w zależności od liczby dokumentów na wejściu.



Rys. 2. Wyniki testów
Fig. 2. Results of Tests

Wyniki są zadowalające, ale być może byłyby lepsze w przypadku zastosowania bardziej zaawansowanego lematyzatora (użyty lematyzator nie jest w stanie sprowadzić wielu wyrazów, zwłaszcza technicznych, do postaci bazowej) oraz dostarczenia większej liczby i dłuższych dokumentów na wejściu. Pogorszenie wyników przy największej sprawdzanej liczbie dokumentów względem wyników dla nieco mniejszej liczby należy tłumaczyć właściwościami klasyfikowanych opinii. Najwyraźniej do mniejszych zbiorów, które nie zawierały wszystkich dokumentów, losowane były bardziej charakterystyczne teksty.

6. Podsumowanie

Stosunkowo niewielkim nakładem pracy udało się nam stworzyć system służący do oceny opinii o produktach. O ile dostępne są darmowe biblioteki pomocnicze dobrej jakości, służące m.in. do pobierania tekstów z sieci, prezentacji wyników i przetwarzania dokumentów, to mieliśmy problem ze specyficznymi dla języka polskiego modułami, które albo są w fazie rozwoju (lematyzator) albo w ogóle nie są dostępne. Mimo tych problemów, stworzone oprogramowanie osiąga wyniki porównywalne z systemami działającymi dla języka angielskiego (np. opisywanymi w [2]). W związku z tym uważamy, że nasz eksperyment powiódł się.

W ramach dalszych badań należy sprawdzić, jak na jakość działania programu wpłynie zastosowanie komercyjnych bibliotek do przetwarzania wyrażen w języku polskim. Sądzymy, że dostępne produkty mogą być lepszej jakości, niż zastosowane przez nas. Warto też sprawdzić, jak statystyczne algorytmy klasyfikacji zachowują się dla dłuższych tekstów w języku polskim, które mogą dotyczyć wielu tematów.

Literatura

- [1] JOACHIMS T., *A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization*. W: Proceedings of ICML-97, San Francisco, Morgan Kaufmann Publishers, 1997, 143–150.
- [2] PANG B. i Lee L. i Vaithyanathan S., *Thumbs up? Sentiment Classification using Machine Learning Techniques*. W: Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, Association for Computational Linguistics, 2002, 79–86.
- [3] TURNEY P., *Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews*. W: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, ACL, 2002, 417–424

Sentiment Classification of Internet Opinions on Products

We present our experiences with building a document classification system for Polish language. The main functionality of this system is sentiment classification of product reviews found on the Internet. Our program was written in the Java programming language using several Open Source libraries. While we had no problem with finding generic Open Source libraries, Polish-specific modules are still at development stage. The system is available thru the Web interface. Classifier works by using a standard statistical approach, namely the TFIDF algorithm with some changes proposed by Pang et al.