Rozpoznawanie języka tekstu na podstawie częstotliwości liter

Krzysztof Kutt, Michał Nowak

***Abstrakt*—W artykule przedstawiono problem rozpoznawania języka zadanego tekstu i propozycje jego rozwiązania. Autorzy zdecydowali się na wykorzystanie informacji o częstotliwości występowania liter. Przygotowano program wykorzystujący sieć neuronową i zestaw danych uczących. W trakcie testów osiągnięto dobrą skuteczność. Zaproponowano możliwości dalszego rozwoju projektu.**

***Słowa kluczowe*—sieć neuronowa, język naturalny, częstotliwość liter.**

I. WPROWADZENIE

SZTUCZNE sieci neuronowe to bardzo uproszczone modele ludzkiego mózgu. Każda z sieci składa się z setek, albo tysięcy sztucznych neuronów, stworzonych na wzór naturalnego neuronu. Sztuczny neuron zbiera sygnały wejścia (poprzez 'dendryty'), dokonuje na nich odpowiedniej transformacji i zwraca odpowiednią wartość na wyjściu ('akson'). W zależności od budowy sieci (układu neuronów) oraz od wykorzystywanej przez neurony funkcji aktywacji, wyróżniane są różne rodzaje sieci [3].

A. Rodzaje sieci

Najważniejsze typy sieci neuronowych to sieci jednokierunkowe, sieci rekurencyjne i sieci Kohonena [8].

1) Sieci jednokierunkowe: W tych sieciach neurony są ułożone warstwowo. Sygnały przechodzą od wejścia do wyjścia sieci, przez wszystkie jej warstwy, bez nawrotów (bez sprzężenia zwrotnego). Rozwiązują dosyć szeroką klasę problemów.

2) Sieci rekurencyjne: Sieci ze sprzężeniem zwrotnym. Połączenia między neuronami stanowią graf z cyklami. Stosowane są np do rozwiązywania problemów minimalizacji, przykładowo: problemu komiwojażera.

3) Sieci Kohonena: Sieć, która dopasowuje swoją strukturę przestrzenną do zbioru danych (mapa). Uczona bez nauczyciela.

B. Zastosowania sieci

Sztuczne sieci neuronowe, ze względu na fakt podawania jedynie przybliżonych wyników, nie nadają się do obliczania dokładnych wartości. Za to, dzięki 'umiejętności' uczenia się, nadają się do rozwiązywania problemów słabo określonych, do których nie jesteśmy w stanie stworzyć algorytmów. Sieć w trakcie procesu uczenia się może wykryć zależności, których nie jesteśmy w stanie dostrzec.

Przykładowymi problemami, do których sieci neuronowe są odpowiednimi narzędziami, są: rozpoznawanie pisma, przetwarzanie obrazu (np w poszukiwaniu ukrtych jednostek wojskowych, czy podejrzanych pakunków na lotniskach), prognozowanie cen.

K. Kutt i M. Nowak, Katedra Automatyki, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki, Akademia Górniczo-Hutnicza, Kraków, Polska, e-mail: kkutt@student.agh.edu.pl, rial@student.agh.edu.pl

C. Rozpoznawanie języka zadanego tekstu

W niniejszym artykule przedstawiony zostaje inny, ciekawy problem: rozpoznawanie języka zadanego tekstu.

Wg projektu "Ethnologue", na świecie istnieje prawie 7.5 tysiąca języków [4]. Narzędzie, które byłoby w stanie rozpoznać z jakim językiem mamy doczynienia, byłoby bardzo użyteczne - dzięki niemu, można np ustalić jakiego tłumacza potrzebujemy. Wizją przyszłości jest stworzenie automatycznego tłumacza, który najpierw rozpoznaje mowę naszego rozmówcy, następnie rozpoznaje język, tłumaczy go, a na samym końcu syntezuje mowę, byśmy mogli w naszym ojczystym języku usłyszeć słowa rozmówcy.

Do zadania tego można podejść na wiele sposobów [1].

1) Alfabet: Pierwszą przesłanką może być wykorzystywany alfabet. Wiele języków posiada charakterystyczne dla siebie znaki diakrytyczne, jak np język polski i 'ogonki'.

2) Charakterystyczne wyrazy: W przypadku takich samych alfabetów, decyzję o języku można podjąć, zwracając uwagę na charakterystyczne dla danego języka wyrazy, czy końcówki. Duże zestawienie takich prostych reguł znajduje się w serwisie Wikipedia [5].

3) Częstotliwość liter: Można również ustalić częstotliwość występowania poszczególnych liter. Już krótkie spojrzenie na listę zestawów do gry w Scrabble w różnych językach [7] (ilość i wartość danej litery zależy od częstotliwości jej występowania w danym języku), pozwala na stwierdzenie, że częstotliwości te są różne w różnych językach. Nie jest jednak możliwe proste określenie reguł rozpoznawania języka na tej podstawie. Dlatego idealnym narzędziem do operowania na takich danych jest sieć neuronowa.

D. Rozpoznawanie języka na podstawie częstotliwości liter

Niniejszy artykuł przedstawia próbę rozpoznawania języka na podstawie częstotliwości liter. Celem uproszczenia budowy programu, zrezygnowano z analizy alfabetu i zdecydowano się na analizowanie tylko i wyłącznie częstotliwości 26 liter alfabetu łacińskiego.

1) Liczba wejść i wyjść: Sieć składa się z 26 wejść: na każdym podawana jest częstotliwość występowania odpowiedniej litery alfabetu łacińskiego w danym tekście. Liczba wyjść zależna jest od założonej ilości rozpoznawanych języków. W przygotowanej aplikacji założono obsługę 11 języków.

2) Budowa sieci: Ze względu na charakter zadania, czyli potrzebę wykrycia odpowiednich zależności w zestawie 26 sygnałów wejściowych, zdecydowano się na wykorzystanie sieci jednokierunkowej, złożonej z trzech warstw (dodatkowa warstwa celem usprawnienia procesu uczenia sieci). Nie istnieje żaden algorytm wspomagający podejmowanie decyzji o ilości neuronów w warstwie ukrytej [2]. Decyzja ta jest zawsze podejmowana arbitralnie i następnie weryfikowana w czasie testów. Należy tylko pamiętać, aby nie była ani zbyt duża, ani zbyt mała. Zdecydowano się przyjąć, że będzie się ona składała z 10 neuronów.

3) Funkcja aktywacji: Każdy neuron wykorzystuje sigmoidalną funkcję aktywacji. Pozwala ona na lepsze różnicowanie zbliżonych sygnałów niż funkcja liniowa. W rozważanym problemie, pozwala na lepsze różnicowanie zbliżonych częstotliwości występowania liter.

4) Uczenie sieci: Do uczenia sieci wykorzystano algorytm Resilient Propagation. Jest to zmodyfikowana wersja podstawowego algorytmu uczenia z nauczycielem sieci wielowarstwowych jednokierunkowych, czyli algorytmu propagacji wstecznej. Zdecydowano się na ten algorytm, ponieważ jest on najbardziej efektywnym algorytmem uczenia dla prostych sieci neuronowych, takich jak nasza [6].

II. Specyfikacja wymagań

PODCZAS analizy problemu zidentyfikowano dwa najważniejsze przypadki użycia aplikacji: wczytanie zbioru uczącego i trenowanie sieci, oraz wczytanie zbioru testowego i egzaminowanie sieci.

Zdecydowano się również na obsługę dodatkowych przypadków użycia, które ułatwiają korzystanie z aplikacji: modyfikację zbioru uczącego / testowego (czyszczenie, dodawanie plików) i jego zapisywanie / wczytywanie do / z pliku, modyfikację zbioru obsługiwanych języków, zapis / odczyt wyuczonej sieci neuronowej do / z pliku.

III. Implementacja

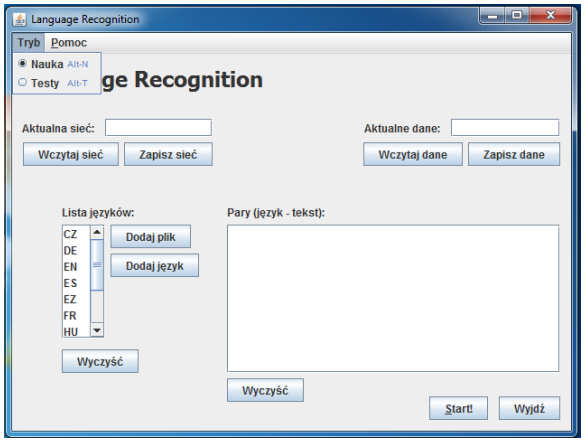
APLIKACJA została napisana w języku Java (wersja 1.6). Wykorzystano środowisko Netbeans IDE oraz bibliotekę Encog Machine Learning Framework.

Program składa się z trzech głównych części: obsługi plików, obsługi sieci neuronowej oraz GUI.

A. Obsługa plików

Aplikacja wykorzystuje trzy rodzaje plików. Wszystkie pliki przechowywane są w folderze data.

1) Pliki z tekstami źródłowymi: Pliki o rozszerzeniu *.txt, zawierają tekst źródłowy. Każdy tekst znajduje się w osobnym pliku. Dla każdego pliku dokonywana jest analiza częstotliwości występowania liter, przy pomocy prostego modułu, stworzonego na potrzeby przedstawianej aplikacji.



Rys. 1. GUI. Widok ogólny

2) Pliki zestawów danych: Plik o rozszerzeniu *.txt, zawierający informacje o zestawie plików z tekstami źródłowymi (do nauki/testu). W pierwszej linii zawiera liczbę N plików tekstowych. W kolejnych N liniach zawiera informacje o każdym z plików w postaci par: JĘZYK ŚCIEŻKA, gdzie JĘZYK to oznaczenie literowe języka, w jakim jest napisany dany tekst, np. PL, EN, FR (może to być dowolne słowo), zaś ŚCIEŻKA to ścieżka dostępu do danego pliku (wymagane jest, by nie posiadała białych znaków).

3) Pliki sieci neuronowych: Pliki binarne o rozszerzeniu *.net, zawierające zserializowany obiekt sieci neuronowej.

B. Obsługa sieci neuronowej

W aplikacji wykorzystywana jest biblioteka Encog Machine Learning Framework. Udostępnia ona własne typy danych, różne rodzaje sieci, funkcji aktywacji i metod uczenia. W stworzonej aplikacji została zaimplementowana obsługa udostępnianych przez bibliotekę narzędzi:

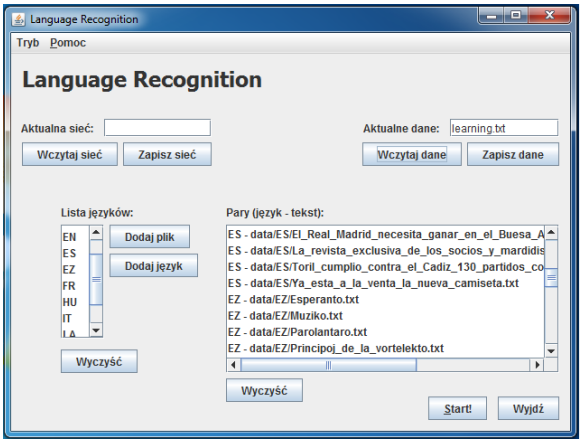
- transformacja informacji o częstotliwościach na typ MLDataSet,
- przygotowanie odpowiedniej sieci,
- uruchomienie procesu uczenia / testowania,
- zebranie informacji zwrotnych.

C. GUI

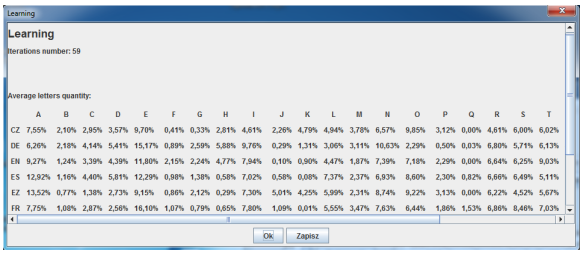
Graficzny interfejs użytkownika, stworzony przy użyciu biblioteki Swing, umożliwia dostęp do wszystkich funkcjonalności aplikacji (rys. 1).

Składa się z 5 głównych obszarów:

- rozwijane menu wyboru trybu (Nauka/Testy),
- wczytywanie / zapisywanie pliku sieci neuronowej; podczas zapisu nie jest zapamiętywana lista obsługiwanych języków - podczas wczytywania sieci należy zadbać o to, aby lista języków była taka sama jak podczas uczenia - w przeciwnym wypadku test sieci może zwracać błędne wyniki,
- wczytywanie / zapisywanie pliku zestawu danych; po załadowaniu lista plików i ich języków pojawia się w polu poniżej (rys. 2),



Rys. 2. GUI. Widok po załadowaniu zestawu danych



Rys. 3. Okienko z wynikami. Uczenie

- lista języków (możliwość dodania kolejnego / wyczyszczenia listy),
- lista aktualnie załadowanych plików i ich języków (możliwość dodania kolejnego / wyczyszczenia listy); aby dodać kolejny plik należy wpierw zaznaczyć odpowiedni język na liście, a następnie wybrać opcję Dodaj plik.

Po zakończeniu procesu uczenia / testowania sieci, pojawia się okienko z wynikami:

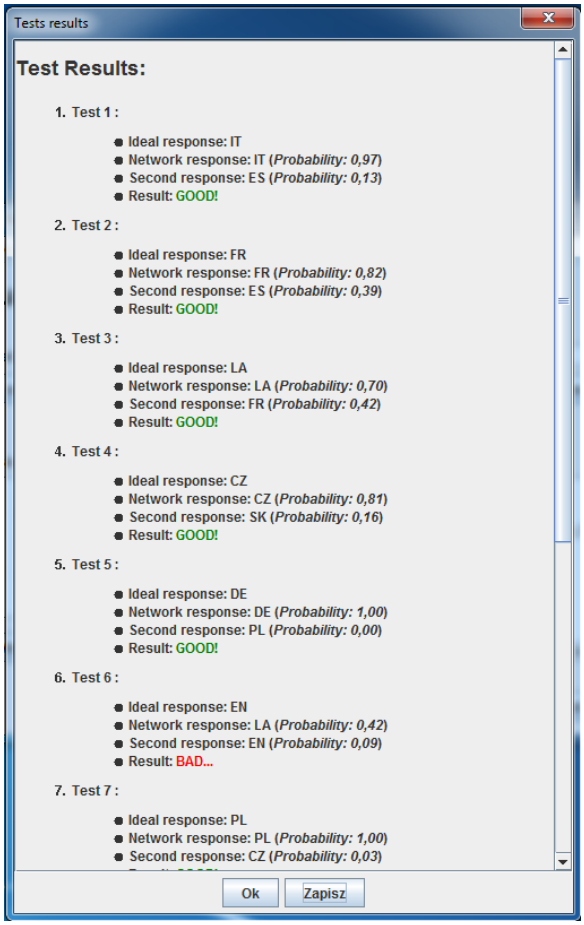
- uczenie: informacja o ilości iteracji oraz tabela ze średnimi częstotliwościami liter dla każdego języka w zadanym zestawie uczącym (rys. 3),
- testowanie: informacja o każdym przeprowadzonym teście: prawidłowa odpowiedź, dwie najbardziej prawdopodobne odpowiedzi udzielone przez sieć oraz ich prawdopodobieństwo, informacja o wyniku testu (GOOD / BAD); na końcu wyświetlana jest informacja o obliczonej skuteczności sieci (rys. 4).

Istnieje możliwość zapisania raportu z wynikami do pliku html, odpowiednio LearnInfo.html i TestResult.html (rys. 5).

IV. Testy

CELEM przetestowania aplikacji, przygotowano zestaw 5 tekstów w 11 językach: czeskim (CZ), niemieckim (DE), angielskim (EN), hiszpańskim (ES), esperanto (EZ), francuskim (FR), węgierskim (HU), włoskim (IT), łacinie (LA), polskim (PL) i słowackim (SK). Dla każdego z języków 4 teksty posłużyły do nauki, a piąty do testowania (nieobecny w zbiorze uczącym).

Osiągnięto skuteczność sieci na poziomie 90% (średnio 1 błąd w zestawie 11 testów).



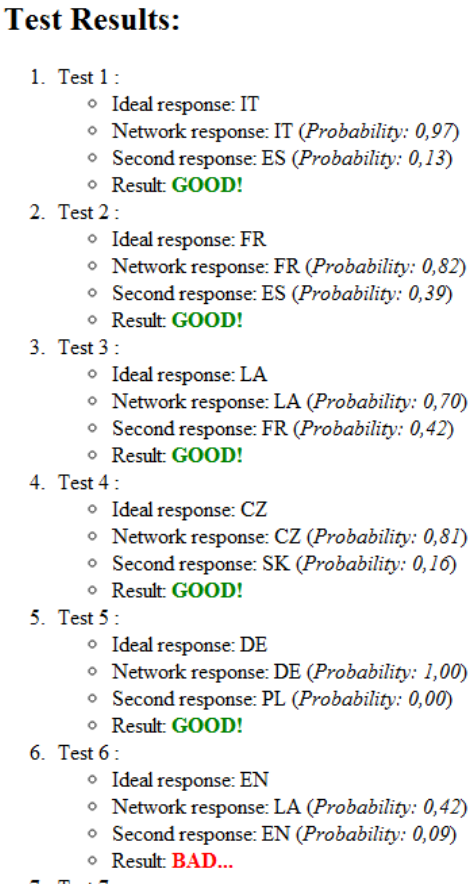
Rys. 4. Okienko z wynikami. Testowanie

W poszczególnych testach osiągano różną pewność odpowiedzi sieci. Jest to ściśle zależne od tego, czy w zestawie występuje język o podobnej częstotliwości liter. Przykładowo:

- języki niemiecki i polski były rozpoznawane z pewnością na poziomie 95-100
- łacina i wywodzące się z niej języki romańskie: hiszpański, włoski, francuski były rozpoznawane z dużo mniejszą skutecznością (30-70%),
- języki czeski i słowacki, uważane za podobne do siebie, były zdecydowanie rozróżniane:
  - test czeski: sieć odpowiedziała, że na 81% jest to język czeski, natomiast język słowacki tylko na 16%;
  - test słowacki: sieć odpowiedziała, że na 90% jest to język słowacki, natomiast język czeski tylko na 1%.

Skuteczność sieci zależy również ściśle od zbioru danych uczących. Powinno się w nim znajdować jak najwięcej tekstów, odpowiedniej długości (im dłuższe tym lepsze). W wykorzystanym zbiorze pojawiały się np teksty piosenek składające się tylko z kilku zwrotek, co mogło okazać się zbyt krótką próbką języka i mogło utrudniać naukę sieci.

Na skuteczność sieci może wpływać również liczba neuronów w warstwie ukrytej, o czym już wcześniej wspomniano. W trakcie testów wypróbowano kilka różnych wartości, jednak nie osiągnięto istotnych różnic. Zdecydowano się pozostawić 10 neuronów w tej warstwie.



Test Results:

1. Test 1 :
   - Ideal response: IT
   - Network response: IT (*Probability: 0,97*)
   - Second response: ES (*Probability: 0,13*)
   - Result: **GOOD!**
2. Test 2 :
   - Ideal response: FR
   - Network response: FR (*Probability: 0,82*)
   - Second response: ES (*Probability: 0,39*)
   - Result: **GOOD!**
3. Test 3 :
   - Ideal response: LA
   - Network response: LA (*Probability: 0,70*)
   - Second response: FR (*Probability: 0,42*)
   - Result: **GOOD!**
4. Test 4 :
   - Ideal response: CZ
   - Network response: CZ (*Probability: 0,81*)
   - Second response: SK (*Probability: 0,16*)
   - Result: **GOOD!**
5. Test 5 :
   - Ideal response: DE
   - Network response: DE (*Probability: 1,00*)
   - Second response: PL (*Probability: 0,00*)
   - Result: **GOOD!**
6. Test 6 :
   - Ideal response: EN
   - Network response: LA (*Probability: 0,42*)
   - Second response: EN (*Probability: 0,09*)
   - Result: **BAD...**

Rys. 5. Plik TestResult.html

V. Wnioski

Z programistycznego punktu widzenia aplikacja była prosta i implementacja nie stwarzała większych problemów, wymagała przede wszystkim odpowiedniej ilości poświęconego czasu.

Rozważany problem okazał się bardzo ciekawym od strony poznawczej. Zmierzenie się z zadaniem pozwoliło zapoznać się z dostępnymi materiałami na temat technik rozpoznawania języka tekstu. Testowanie większej ilości języków, niż początkowo zakładane trzy, pozwoliło na odkrycie wiedzy, której autorzy wcześniej nie posiadali (m.in. o dużej różnicy pomiędzy językami czeskim i słowackim).

Wyświetlenie tabeli z informacjami o częstotliwości występowania liter w poszczególnych językach pozwoliło na prowadzenie własnych poszukiwań podobieństw i różnic między językami. W ramach rozwoju projektu warto zwizualizować sieć i przedstawić wagi, które są ustalane w procesie uczenia - może to pozwolić na zobaczenie zależności, których nie widać “na pierwszy rzut oka”.

Projekt można również rozwinąć o stosowanie innych technik rozpoznawania języka, np obsługiwać wszystkie dostępne znaki danego alfabetu, a nie tylko litery alfabetu łacińskiego, jak to zrobiono w niniejszym projekcie.

Walory poznawcze z pewnością będzie posiadało rozszerzenie projektu o obsługę kolejnych języków i zwiększenie ilości plików w zbiorze uczącym. Stworzona aplikacja pozwala na dowolne rozszerzanie tych dwóch zbiorów, co zachęca do dalszych eksperymentów.

Literatura cytowana

[1] T. Smykowski, *Jak Google rozpoznaje język tekstu?*, http://polishwords.com.pl/blog/2009/rozpoznawanie-jezyka-tekstu/, (dostęp: 2012-05-31)

[2] R. Tadeusiewicz, *Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami*, Warszawa, Polska: Akademicka Oficyna Wydawnicza PLJ, 1998.

[3] R. Tadeusiewicz, *Sieci neuronowe*, wyd. 2, Warszawa, Polska: Akademicka Oficyna Wydawnicza RM, 1993.

[4] *Ethnologue*, http://www.ethnologue.com/language_index.asp, (dostęp: 2012-05-31)

[5] *Language recognition chart*, http://en.wikipedia.org/wiki/Wikipedia:Language_recognition_chart, (dostęp: 2012-05-31)

[6] *Resilient Propagation*, http://www.heatonresearch.com/wiki/Resilient_Propagation, (dostęp: 2012-05-31)

[7] *Scrabble letter distributions*, http://en.wikipedia.org/wiki/Scrabble_letter_distributions, (dostęp: 2012-05-31)

[8] *Sieć neuronowa*, http://pl.wikipedia.org/wiki/Sieć_neuronowa, (dostęp: 2012-05-31)