Warszawa, 11.04.2022

Klasyfikacja tekstowa wiadomości poczty elektronicznej - filtrowanie e-maili SPAM

Wojciech Rokicki, Jakub Kowalczuk

1. Szczegółowa interpretacja tematu projektu

Tematem naszego projektu jest przeanalizowanie wpływu różnych metod reprezentacji wektorowej tekstu oraz szeregu modeli na wyniki klasyfikacji treść wiadomości poczty elektronicznej. Celem treningu jest stworzenie filtru wiadomości SPAM. Konkretyzacja zadania wymaga:

- wyboru algorytmów klasyfikacji,
- wskazania parametrów algorytmów klasyfikacji wymagających strojenia,
- ustalenia wykorzystywanych technik wektorowej reprezentacji tekstu (co najmniej dwie różne reprezentacje),
- określenia zakresu przygotowania danych (np. modyfikacji rozkładu klas, losowania prób),
- ustalenia kryteriów lub algorytmu selekcji atrybutów (słów),
- ustalenia procedur i kryteriów oceny jakości modeli (z uwzględnieniem rozkładu oraz, tam gdzie to uzasadnione, kosztów pomyłek).

W poniższych rozdziałach przedstawione są szczegółowe założenia projektu. Opisują one konkretne techniki, algorytmy wraz z ich modyfikowanymi parametrami oraz narzędzia przewidziane do realizacji analizy.

2. Charakterystyka zbioru danych

Zbiór danych zawiera sumarycznie 6047 wiadomości e-mail w języku angielskim, gdzie 31% stanowi spam. Składa się on z trzech części:

- “spam” 500 + 1397 = 1897 wiadomości spam
- “easy_ham” 2500 + 250 = 2740 wiadomości pożądanych
- ”hard_ham” 250 widomości pożądanych posiadających cechy spam’u.

W ramach niniejszego projektu powyższy zbiór zostanie podzielony odpowiednio na zbiory: trenujący i testowy.

2.1. Wstępne przetwarzanie danych

2.1.1. Oczyszczanie danych

Dane dostarczone do budowanie oraz testowania modeli muszą zostać oczyszczone z nadmiarowych informacji. Z punktu widzenia niniejszego projektu istotnymi danymi jest treść oraz tytuł wiadomości e-mail. W pierwszej kolejności zostaną usunięte nagłówki związane z protokołami komunikacji sieciowej. Następnie z ciała samej wiadomości zostaną wyczyszczone znaczniki HTML stosowane w komunikacji e-mail.

2.1.2. Reprezentacja wektorowa

Wyłuskane w procesie oczyszczania dane w celu dalszego przetwarzania zostaną przekonwertowane do postaci wektorowej. Do celów badawczo-porównawczych zostaną zastosowane dwa modele reprezentacji: unigramy(*ang. "bag of words"*) oraz bigramy.
Modele oparte zarówno o unigramy jak i bigramy stanowią częstotliwościową reprezentację tekstu. Różnicę między modelami stanowi sposób podziału tekstu. W przypadku unigramów zliczamy wystąpienie pojedynczych słów, natomiast w przypadku bigramów tworzymy pary słów występujących bezpośrednio po sobie. W obydwu modelach znaki interpunkcyjne będą pomijane, a zapis słów będzie sprowadzany do małych liter. Do wektoryzacji wykorzystany zostanie pakiet języka R ***text2vec***.

Dodatkowo w celu poprawy jakości wektorów utworzona zostanie stop lista zawierająca najczęściej występujące słowa i pozwalająca je ignorować przy dalszej analizie. Wynika to z faktu, iż najczęściej pojawiającymi się słowami są np: "and", "i", "me", które nie niosą istotnej informacji o treści.

2.1.3. Podział danych

Do wstępnego strojenia modeli zdecydowano się utworzyć zbiór testowy i trenujący w następujących proporcjach:

- Trenujący: 80%, Testowy: 20%
- Trenujący: 67%, Testowy: 33%
- Trenujący: 50%, Testowy: 50%

W każdym zbiorze zostanie zachowany taki sam stosunek wiadomości spam do wiadomości istotnych jak w zbiorze źródłowym (31% spam).

3. Opis algorytmów, które będą wykorzystane do badań

Podczas eksperymentów wykorzystane będą algorytmy pochodzące z paczek ***caret***, ***e1071***, ***randomForest***. Testowane modele oraz ich modyfikowane parametry zostały wymienione poniżej:

3.1. Naiwny klasyfikator Bayesa

Algorytm ten bazuje na Twierdzeniu Bayesa:

$$\mathrm{P}(A \mid B) = \frac{\mathrm{P}(B \mid A)\,\mathrm{P}(A)}{\mathrm{P}(B)},$$

gdzie A to zdarzenie przewidywane pod warunkiem wystąpienia zdarzenia B. Model jest zwany naiwnym, ponieważ zakłada niezależność zdarzeń warunkowych. Dzięki temu jesteśmy w stanie zastosować poniższe równanie:

$$P(y|x_1,...,x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

gdzie y to klasa predykcji, a x'y to wartości atrybutów. Dla każdego przykładu mianownik jest stały, dlatego możemy go pominąć i wstrzyknąć odpowiednią proporcję. Otrzymujemy równanie:

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

Aby znaleźć predykowaną klasę szukamy takiej klasy, dla której jest największe prawdopodobieństwo:

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

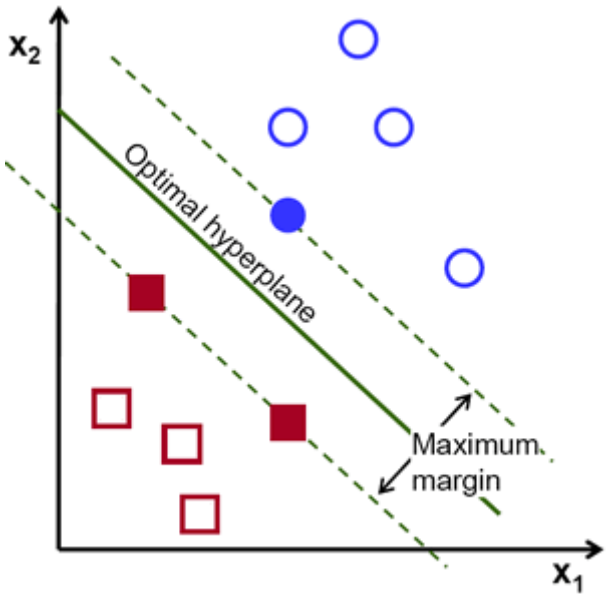
Czasami może zabraknąć przykładów dla niektórych klas. Skutkuje to zerowym prawdopodobieństwem dla danej klasy. W celu uniknięcia takiej sytuacji stosuje się wygładzanie Laplace'a:

$$P(w'|positive) = \frac{\text{number of reviews with w' and y = positive } + \ \alpha}{\text{N } + \ \alpha * K}$$

gdzie w' to zestaw wartości atrybutów, dla danej klasy positive, alfa to parametr wygładzania, K to liczba atrybutów, N to liczba wszystkich elementów klasy positive.

3.2. SVM (Supported Vector Machine)

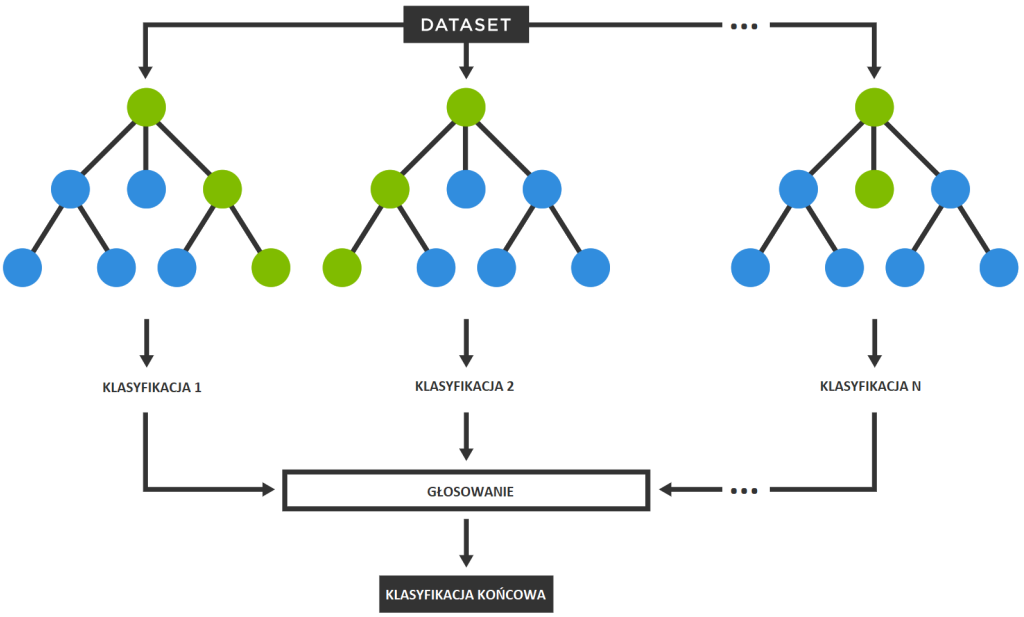
Maszyna wektorów nośnych to algorytm, którego celem jest znalezienie hiperpłaszczyzny, dla której uzyskujemy maksymalny margines (maksymalny dystans między dwoma elementami przeciwnych klas). Do wyznaczenia płaszczyzny SVM wykorzystuje wektory nośne, które zaczepione w najbliższym punkcie hiperpłaszczyzny wskazują na najpodobniejsze elementy różnych klas.



W algorytmie może zostać zaimplementowane jądro, którego celem jest radzenie sobie z nieliniowością oraz wielowymiarowością danych. Jest to funkcja, która przekształcając dane (zwiększając wymiar) umożliwia liniową separowalność.

3.3. Las Losowy

Algorytm lasu losowego działa w oparciu o klasyfikację podjęta na podstawie k klasyfikatorów drzewa decyzyjnego. Decyzja o finalnej klasyfikacji podejmowana jest na podstawie głosowanie z wyników dla poszczególnych drzew.



Do budowy każdego z klasyfikatorów w lesie ze zbioru trenującego losowane są ze zwracaniem przykłady o liczności N. W czasie dostrajania parametrów modelu ustalana jest ilość klasyfikatorów k oraz parametry drzew decyzyjnych.

4. Miary jakości i procedury oceny modeli

W celu oceny jakości poszczególnych modeli badanych ramach eksperymentów, stosowana będzie macierz pomyłek. Na jej podstawie jako końcowa ocena modelu wyliczona zostanie miara F1 w postaci:

$$F = \frac{2 \cdot recall \cdot precision}{recall + precision}$$

Gdzie miary pośrednie wyznaczone zostaną na podstawie poniższych wzorów:

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}$$

W przypadku niniejszego projektu przyjęto następujące wartość:

- TP(*"true positive"*) - poprawnie sklasyfikowana wiadomość jako spam
- FP("false positive") - wiadomość pożądana uznana za spam
- FN("false negative") - spam uznany jako wiadomość pożądana

W czasie dostrajanie parametrów modeli wartość miary F1 będzie maksymalizowana.

5. Plan badań

Do wstępnych eksperymentów prowadzących do wystrojenia modeli użyjemy jednego stałego podziału na zbiór trenujący 67% i testujący 33%. Zapewni on obiektywność i porównywalność wyników.

Wszystkie poniższe eksperymenty zostaną przeprowadzone dla obydwu zaproponowanych reprezentacji. Pozwoli to porównać ich skuteczność dla poszczególnych algorytmów.

5.1. Cel poszczególnych eksperymentów

Celem eksperymentów będzie zbadanie wpływu zmiany parametrów algorytmów na ich skuteczność działania. Poniżej zostały wypunktowane modele wraz z ich modyfikowanymi parametrami:

- Model: e1071::naiveBayes
  Modyfikowane parametry:
  - threshold - używany przy wygładzaniu Laplace'a, określa wartość niezerowej wartości prawdopodobieństwa
  - eps - określa zakres w którym stosujemy wygładzanie Laplace'a
- Model: e1071::svm
  Modyfikowane parametry:
  - jądro - funkcje wewnętrzne przekształcające dane
  - tolerancja - kryterium stopu treningu
- Model: randomForest::randomForest
  Modyfikowane parametry:
  - ntree - liczba drzew
  - mtry - liczba losowanych atrybutów branych pod uwagę przy podziale w węźle
  - nodesize - minimalna liczba liści
  - maxnodes - maksymalna liczba liści dla każdego drzewa w lesie

Wybór parametrów został podyktowany specyfiką implementacji algorytmów zawartych w paczkach języka R.

W celu obiektywnej oceny poszczególnych algorytmów i ich modyfikowanych zestawów parametrów zastosowana zostanie k-krotna walidacja krzyżowa.