27.01.2020

Dokumentacja projektu

E-mail spam filter

wykonali: Marcin Mozolewski i Marcin Różański

1 Treść zadania

Napisać program, który korzystając z uczenia Bayesowskiego (Naive Bayes) będzie w stanie zaklasyfikować wiadomości e-mail jako SPAM lub nie. Zbiór danych do użycia: Enron-Spam in pre-processed form - http://www2.aueb.gr/users/ion/data/enron-spam/

2 Pełen opis funkcjonalny

2.1 Rozwiązanie

Po przefiltrowaniu danych i podzieleniu ich na dane treningowe i dane testowe używamy uczenia Bayesowskiego do określenia czy wiadomość jest spamem czy nie. Przy uczeniu zakładamy że słowa w treści występują niezależnie od siebie, a wynik klasyfikacji zależy od częstości pojawiania się poszczególnych słów zawartych w e-mailu w wiadomościach z dostępnej bazy bezpośrednio sklasyfikowanych jako spam lub nie. Słowa, które nie występują w danych treningowych nie są brane pod uwagę przy wyznaczaniu wyniku.

2.2 Funkcja zwracająca wynik

$$F(E) = log\frac{P(S)}{P(H)} + \sum_{i=0}^{n} log\frac{P(w_i|S)}{P(w_i|H)}$$

F(E) > 0 to wiadomość to spam F(E) <= 0 to wiadomość nie jest spamem

gdzie:

- P(S) - ogólne prawdopodobieństwo, że wiadomość jest spamem
- P(H) - ogólne prawdopodobieństwo, że wiadomość nie jest spamem
- P(wi|S) - prawdopodobieństwo, że wiadomość zawierająca słowo w jest spamem
- P(wi|H) - prawdopodobieństwo, że wiadomość zawierająca słowo w nie jest spamem

2.3 Filtracja danych

Do uczenia Bayesowskiego wykorzystamy tylko słowo zawarte w e-mailu. W tym celu należy:

- Usunąć liczby
- Usunąć samotne znaki specjalne
- Usunąć białe znaki
- Usunąć znaki specjalne z końca słów
- Zamienić duże litery w słowie na małe

3 Struktura programu

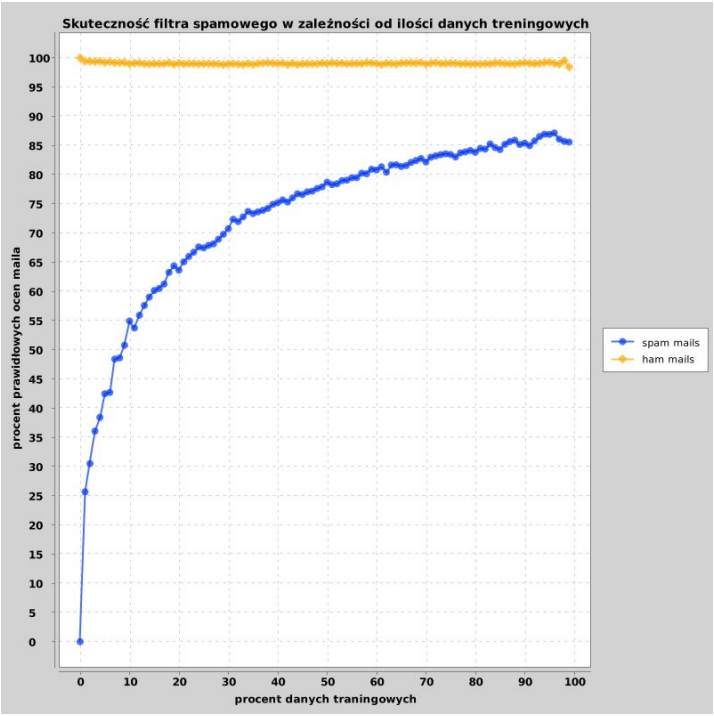
3.1 Podział odpowiedzialności

- Marcin Mozolewski: Wprowadzanie danych wejściowych, implementacja filtracji danych, testy jednostkowe, przykładowe użycie programu
- Marcin Różański: Implementacja algorytmu, projekt filtracji danych, walidacja krzyżowa

3.2 Podział aplikacji

W aplikacji można wyróżnić część odpowiedzialną za interfejs filtru, za pomocą której można dodawać nowe dane i sprawdzać czy email jest spamem. Dane są przetrzymywane w strukturze drzewiastej dla przyśpieszanie wyszukiwania słów. Druga część aplikacji to algorytm Bayesowski wykorzystywany do obliczania prawdopodobieństw dla słów i całych maili. Przy bardzo dużej ilości słów prawdopodobieństwo staje się bardzo małe, dlatego w całym algorytmie zastosowano logarytm.

4 Przeprowadzone testy i wnioski



4.1 Testy

Wykres przedstawia procent dobrych ocen maila pod kątem bycia spamem. W przypadku braku danych filtr ocenia wszystkie maile jako “ham”, a wraz z uczeniem lepiej ocenia spam. “Ham” wraz z uczeniem pozostaje na stałym poziomie ~98.5% prawidłowych ocen.

Podczas testów średni czas oceny maila pod kątem bycia spamem wyniósł mniej niż 0.5 ms na próbie 34 516 maili.

Zastosowanie walidacji krzyżowej z parametrem równym pięć pozwoliło uzyskać średnią stratę równą 8.3989%.

Na wejściu danych do programu jest wykorzystywany filtr dzięki któremu udało się zmniejszyć średnią stratę o jeden punkt procentowy.

4.2 Wnioski

Skuteczność filtra zwiększa się wraz ze wzrostem stosunku danych treningowych do danych testowych. Logiczne jest więc, że aby polepszyć ogólne wyniki należy poszerzyć bazę posiadanych danych. Powinno to pomóc wyeliminować obecnie występujące przypadki skrajne - niektóre słowa występują tylko w 1 rodzaju wiadomości, co w praktyce z miejsca ją klasyfikuje.

5 Instrukcja wykorzystania klasy SpamFilter

1. Stworzenie instancji SpamFilter przez wywołanie konstruktora
2. Podanie danych treningowych do metody *learn(String, boolean).* Podane zdania są rozdzielane na słowa za pomocą regexa, który można ustawić tak aby najlepiej pasował do podanych danych.
3. Następnym krokiem po podaniu danych treningowych jest wywołanie metody *update(),* która wyliczy prawdopodobieństwa dla wszystkich słów. Bez użycia tej metody instancja będzie wskazywała wadliwe wyniki.
4. Aby sprawdzić czy wiadomość jest spamem należy użyć metody *isSpam(String).* Ponieważ wykorzystywane jest twierdzenie Bayesa więc metody *isSpam(String)* i *isHam(String)* nie muszą zwracać przeciwnych wartości dla tych samych danych wejściowych. Filtracja danych pod kątem spamu powinna odbywać się z wykorzystaniem metody *isSpam(String).*
5. W przypadku chęci nauczenia ponownie instancji na nowych danych należy przed rozpoczęciem nauki wywołać metodę *clear(),* która wyczyści pamięć z wcześniej nauczonych słów.