Projekt z MOW – dokumentacja końcowa

Tomasz Bocheński, Mariusz Słapek

Styczeń 2019

1 Temat projektu

Naiwny klasyfikator bayesowski do klasyfikacji tekstu w binarnej i częstościowej reprezentacji *bag of words* (warianty *Bernoulli NB* i *multinomial NB*). Porównanie ze standardowym naiwnym klasyfikatorem bayesowskim dostępnym w R.

2 Interpretacja projektu

W ramach projektu zaimplementowane zostaną w języku R dwa naiwne klasyfikatory bayesowskie: *Bernoulli NB* oraz *Multinomial NB*. Zostaną one udostępnione jako paczka w języku R. Ponadto zostaną znalezione trzy zbiory danych, na których przeprowadzone zostaną eksperymenty. Przygotowane zostaną skrypty do przetwarzania znalezionych zbiorów w taki sposób, aby każdy dokument ze zbioru reprezentowany był w postaci *bag of words* – zarówno w wersji binarnej jak i częstościowej. Oprócz tego porównane zostaną wyniki otrzymane dla zaimplementowanych klasyfikatorów z wynikami naiwnego klasyfikatora bayesowskiego dostępnego w R.

3 Opis algorytmu

Wśród wielu zadań uczenia maszynowego jednym z najpowszechniejszych jest zadanie klasyfikacji. Rozróżniamy klasyfikację binarną, gdzie wybieramy jedną z dwóch klas oraz klasyfikację wieloklasową. W naszym przypadku będziemy starali się pokazać oba rodzaje wykorzystując przy tym naiwny klasyfikator bayesowski. Należy on do metod probabilistycznych, które wykorzystują wnioskowanie probabilistyczne oparte na twierdzeniu Bayesa. Dla dowolnej hipotezy $h \in H$ oraz zbioru danych D (który wpływa na ocenę prawdopodobieństwa hipotezy) możemy wyznaczyć prawdopodobieństwo *a priori - P(h)*. Jest to prawdopodobieństwo wyznaczone przed realizacją doświadczenia losowego (nie uwzględniamy danych). Uwzględniając te dane wyznaczamy prawdopodobieństwo *a posteriori - P(h—D)*. Wówczas twierdzenie Bayesa przedstawia się następująco:

$$P(h|D) = \frac{P(h) * P(D|h)}{P(D)} \tag{1}$$

gdzie:
P(D) - prawdopodobieństwo danych,
P(D—h) - prawdopodobieństwo danych przy założeniu poprawności hipotezy.

W wielu przypadkach prawdopodobieństwo danych może zostać pominięte, gdyż nie wpływa ono na wynik (nie zależy ono od hipotezy).

Wówczas definicję odpowiedniej hipotezy może zapisywać w następujący sposób:

$$h(x^*) = arg \max_{c \in D} P(c(x) = d | a_1(x) = a_1(x^*) \wedge a_2(x) = a_2(x^*) \wedge ... \wedge a_n(x) = a_n(x^*)) \tag{2}$$

gdzie:
x* - pewien przykład.

Wybieramy kategorię najbardziej prawdopodobną dla przykładu o znanych wartościach atrybutów.

Naiwny klasyfikator Bayesa zakłada, że warunkowe prawdopodobieństwa dla zmiennych niezależnych są wzajemnie niezależne. Inaczej mówiąc, obecność każdego atrybutu nie jest związana z obecnością każdego innego atrybutu. Ze względu na tą cechę klasyfikator jest nazywany naiwnym, gdyż w wielu przypadkach założenie to jest nieprawdziwe.

Po uwzględnieniu ostatniego założenia postać hipotezy jest następująca:

$$h(x^*) = arg \max_{c \in D} P(c(x) = d) * \prod_{i=0}^{n} P(a_i(x) = a_i(x^*)|c(x) = d) \tag{3}$$

Może zdarzyć się, iż w zbiorze trenującym nie wystąpi wartość pewnego atrybutu w ramach pewnej kategorii. Może to prowadzić do błędnych wniosków, iż wystąpienie przykładu w całej dziedzinie jest niemożliwe (trudno oczekiwać, że w zbiorze trenującym wystąpią wszystkie wartości). W celu uniknięcia zerowych prawdopodobieństw zastosujemy wygładzanie Laplace'a.

3.1 Bernoulli NB

W algorytmie *Bernoulli NB* atrybuty są traktowane jako niezależne wartości binarne. Dany atrybut opisuje wystąpienie albo niewystąpienie danego słowa ze słownika (jedynki i zera są interpretowane jako *słowa występujące w tekście* i *słowa nie występujące w tekście*). W algorytmie tym ignorowana jest liczba wystąpień danego słowa. Liczba atrybutów danego tekstu będzie taka sama jak liczba słów w słowniku.

Kroki algorytmu:

1. Zdefiniowanie słownika V - wszystkie teksty zostaną w ścisły sposób przefiltrowane. Z każdego tekstu zostaną usunięte końcówki charakterystyczne dla języka angielskiego oraz tzw. *stopwordy*. Ponadto usunięte zostaną liczby, symbole oraz interpunkcja. Liczba słów w słowniku będzie definiowała rozmiar wektora atrybutów.

2. Zostaną policzone następujące wartości w zbiorze trenującym:

   (a) całkowita liczbę dokumentów - N,

   (b) liczba dokumentów w danej klasie C = k, gdzie $k = 0, 1..., K$ - N_k,

   (c) liczba dokumentów z klasy C = k zawierającą słowo w_t z każdej klasy i dla każdego słowa w słowniku - $n_k(w_t)$.

3. Zostanie obliczone prawdopodobieństwo P(w_t—C=k) wykorzystując równanie:

$$P(w_t|C = k) = \frac{n_k(w_t)}{N_k} \tag{4}$$

4. Zostanie obliczone prawdopodobieństwo *a priori P(C=k)* wykorzystując równanie:

$$P(C = k) = \frac{N_k}{N} \tag{5}$$

Aby sklasyfikować dokument D_j obliczymy prawdopodobieństwo a posteriori dla każdej klasy wykorzystując równanie:

$$P(C|D) = P(C) * \prod_{t=1}^{|V|} b_{jn} * P(w_t|C) + (1 - b_{jn}) * (1 - P(w_t|C)) \tag{6}$$

gdzie:
b_{jn} - n-ty element j-tego wektora tekstu.

3.2 Multinomial NB

W algorytmie *multinomial NB* dokument jest reprezentowany przez zbiór słów występujących w tekście. Kolejność występowania słów w tym algorytmie nie ma znaczenia. Bierze się pod uwagę jakie słowa w nim występują i jest zliczana liczba poszczególnych słów. W problemie tym stawiamy pytanie: jak często dane słowo występuje w tekście.

Kroki algorytmu:

1. Zdefiniowanie słownika V - wszystkie teksty zostaną w ścisły sposób przefiltrowane. Z każdego tekstu zostaną usunięte końcówki charakterystyczne dla języka angielskiego oraz tzw. *stopwordy*. Ponadto usunięte zostaną liczby, symbole oraz interpunkcja. Liczba słów w słowniku będzie definiowała rozmiar wektora atrybutów.

2. Zostaną policzone następujące wartości w zbiorze trenującym:
   (a) całkowita liczbę dokumentów - N,
   (b) liczba dokumentów w danej klasie C = k, gdzie $k = 0, 1..., K$ - N_k,
   (c) częstotliwość występowania słowa w_t w dokumencie D_t obliczona dla każdego słowa w_t dla słownika V.
3. Zostanie obliczone prawdopodobieństwo P(w_t—C=k) wykorzystując równanie:

$$P(w_t|C = k) = \frac{\sum_{i=1}^{N} x_{it} * z_{ik}}{\sum_{s=1}^{|V|} x_{is} * z_{ik}} \tag{7}$$

   gdzie:
   x_{in} - liczba wystąpień n-tego elementu i-tego wektora tekstu
   z_{ik} - zmienna, która równa się 1, gdy słowo i-te ze słownika znajduje się w tekście k-tym
4. Zostanie obliczone prawdopodobieństwo *a priori P(C=k)* wykorzystując równanie:

$$P(C = k) = \frac{N_k}{N} \tag{8}$$

Aby sklasyfikować tekst D_j obliczymy prawdopodobieństwo a posteriori dla każdej klasy ze wzoru:

$$P(C|D_j) \sim P(C) * \prod_{t=1}^{|V|} P(w_t|C)^{x_{in}} \tag{9}$$

Przedstawiony powyżej sposób wyznaczania prawdopodobieństwa jest jednak wrażliwy na błędy numeryczne. Z tego względu w ostatecznej implementacji klasyfikatora wykorzystamy wersję logarytmiczną, która jest stabilniejsza numerycznie.

$$arg\max_k \; log(P(C)) \; + \sum_t x_{in} \, log(P(w_t|C)) \tag{10}$$

4 Zbiory danych

W tej sekcji opisane zostaną trzy zbiory danych, na których zdecydowaliśmy się prowadzić eksperymenty. Wszystkie zbiory posiadają podobną strukturę - w katalogu głównym znajdują się podkatalogi, które reprezentują daną klasę. W każdym podkatalogu znajdują się pliki tekstowe należące do danej klasy.

4.1 C50

Zbiór zawiera listę pięćdziesięciu pisarzy, każdy z pisarzy reprezentowany jest przez sto swoich krótkich dzieł. Źródło: *http://archive.ics.uci.edu*.

Do prowadzenia eksperymentów zdecydowaliśmy się skorzystać z całego zbioru.

4.2 20_newsgroups

Zbiór zawiera dwadzieścia grup tematycznych, każda grupa zawiera tysiąc przykładowych tekstów. Jedną z wad tego zbioru jest fakt, że niektóre pliki są puste – posiadają tylko nagłówki. Dodatkową wadą jest sam fakt nagłówków w plikach, które trzeba odrzucić na etapie preprocessingu. Źródło: *http://www.cs.cmu.edu*.

Do prowadzenia eksperymentów zdecydowaliśmy się wybrać pięć różnych kategorii, przy czym każda z nich reprezentowana była przez pięćset tekstów.

4.3 emotions

Zbiór o nazwie *Large Movie Review Dataset* zawierający recenzje filmów ze strony internetowej *IMDb*. Zbiór ten zawiera dwie klasy (opinie pozytywne oraz opinie negatywne), przy czym każda z klas reprezentowana jest przez dwadzieścia pięć tysięcy przykładów. Źródło: *http://ai.stanford.edu/*.

Do prowadzenia eksperymentów zdecydowaliśmy się wykorzystać po pięć tysięcy przykładów z każdej klasy.

5 Preprocessing danych

Preprocessing danych składał się z dwóch etapów. Pierwszy etap polegał na generowaniu statystyki poszczególnych słów, a wyniki tego etapu zapisywane były do pliku z rozszerzeniem csv. W kolejnym kroku, na podstawie wygenerowanego pliku, jak również samych danych oraz informacji o wielkości słownika, tworzone były reprezentacje tekstów w postaci wektorów *bag of words*. Wektory te (*dataframe*) zapisywane były do plików z rozszerzeniem Rda.

Podczas przetwarzania danych skorzystaliśmy z funkcji udostępnionych w paczce *tm*. Pozwoliło nam to w sprawny sposób przeprowadzić na tekście takie operacje jak:

- usunięcie interpunkcji;
- usunięcie *stopwordów*;
- usunięcie końcówek fleksyjnych ze słów (*stemming*;
- zastąpienie skrótów ich pełnymi nazwami;
- usunięcie wszystkich wyrazów znajdujących się w nawiasach;
- przedstawienie liczb całkowitych za pomocą wyrazów;
- zastąpienie wszystkich dużych liter małymi;
- usunięcie z tekstu liczb.

Tworzenie słownika o wielkości X polegało na wybraniu X najczęściej występujących słów z pliku z rozszerzeniem csv (opisanego wcześniej). Następnie każdy tekst zamieniany był na reprezentację *bag of words* w wersji binarnej oraz częstościowej (wektory). Oznacza to, że jeśli słowo s_i ze słownika nie wystąpiło w tekście, element wektora w_i przyjmował wartość 0. W przypadku gdy słowo s_i wystąpiło w tekście, wektor w_i przyjmował wartość 1 (w wersji binarnej) lub odpowiadał ilości słów s_i w tekście (w wersji częstościowej).

6 Struktura projektu i implementacja

W katalogu *data* znajdują się różnego typu dane: od surowych (*raw*), przez przejściowe (*transitional*), aż po końcowe reprezentacje tekstów *bag of words* (*processed*). W katalogu *src* znajdują się wszystkie kody źródłowe: zarówno skrypty jak i paczka z implementacją naiwnego klasyfikatora bayesowskiego.

Wszystkie skrypty zaimplementowane zostały w języku R. Poniżej znajduje się opis każdego ze skryptów:

- *runWordsCounter.R* - skrypt służący do wyznaczenia liczby poszczególnych słów w danym zbiorze danych. Wynikiem jego działania jest plik z rozszerzeniem csv. Sposób uruchomienia: **Rscript runWordsCounter.R** *dataset*, gdzie *dataset* to nazwa zbioru danych (możliwe nazwy: *C50*, *20_newsgroups*, *emotions*).
- *runBerBOWGenerator.R* - skrypt służący do generowania reprezentacji tekstu *bag of words* w wersji binarnej, o okreslonej wielkości i dla określonego zbioru danych. Sposób uruchomienia: **Rscript runBerBOWGenerator.R** *dataset size flag*, gdzie *dataset* to nazwa zbioru danych (możliwe nazwy: *C50*, *20_newsgroups*, *emotions*), *size* to rozmiar wektora *bag of words*, natomiast *flag* to dodatkowa flaga (powinna wynosić *removeHeaders* dla zbioru danych *20_newsgroups*).
- *runMulBOWGenerator.R* - skrypt służący do generowania reprezentacji tekstu *bag of words* w wersji częstościowej, o okreslonej wielkości i dla określonego zbioru danych. Sposób uruchomienia: **Rscript runMulBOWGenerator.R** *dataset size flag*, gdzie *dataset* to nazwa zbioru danych (możliwe nazwy: *C50*, *20_newsgroups*, *emotions*), *size* to rozmiar wektora *bag of words*, natomiast *flag* to dodatkowa flaga (powinna wynosić *removeHeaders* dla zbioru danych *20_newsgroups*).
- *runBernoulli.R* - skrypt służący do trenowania klasyfikatora *Bernoulli NB* oraz ewaluacji wyników. Trenowane są dwa klasyfikatory: klasyfikator udostępniony już w R (*e1071*) oraz własna implementacja. Wyświetlane są czasy działania oraz dokładność klasyfikacji. Sposób uruchomienia: **Rscript runBernoulli.R** *bagOfWords*, gdzie *bagOfWords* to nazwa pliku wygenerowanego przez skrypt *runBerBOWGenerator.R*.

- *runMultinomial.R* - skrypt służący do trenowania klasyfikatora *Multinomial NB* oraz ewaluacji wyników. Trenowane są dwa klasyfikatory: klasyfikator udostępniony już w R (*e1071*) oraz własna implementacja. Wyświetlane są czasy działania oraz dokładność klasyfikacji. Sposób uruchomienia: **Rscript runMultinomial.R** *bagOfWords*, gdzie *bagOfWords* to nazwa pliku wygenerowanego przez skrypt *runMulBOWGenerator.R*.

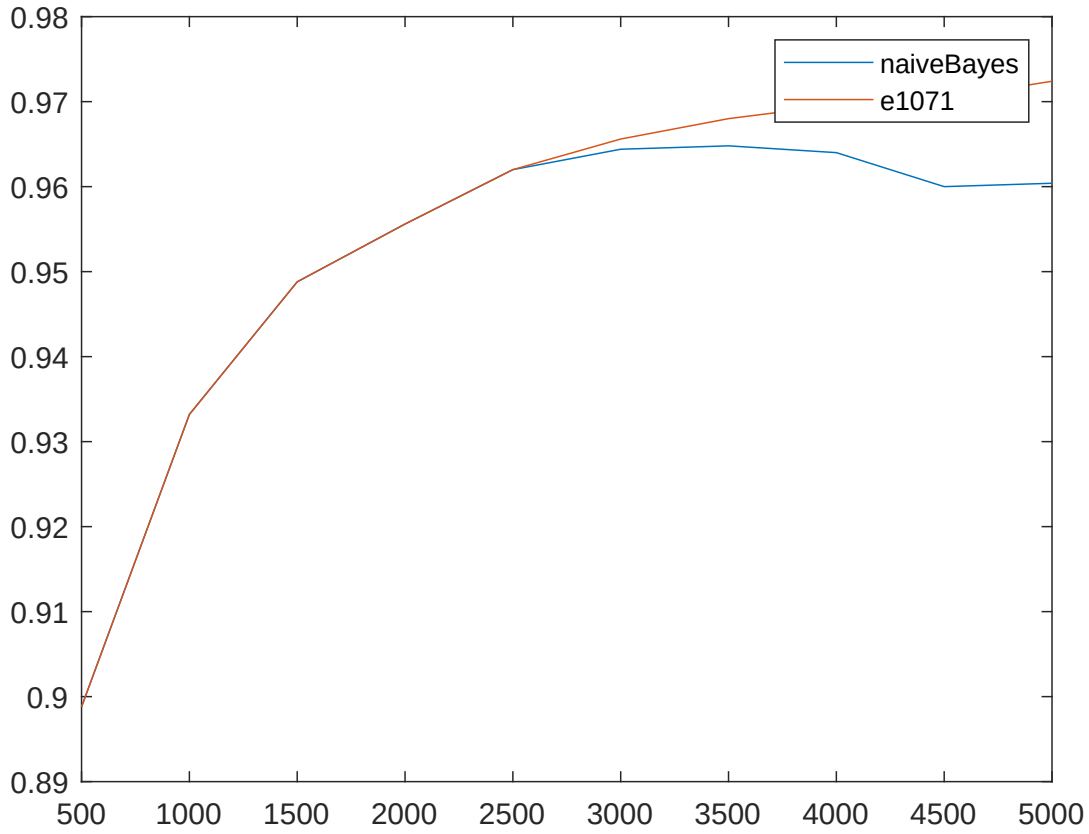
7 Wyniki eksperymentów

7.1 Zbiór danych C50

Poniższa tabela przedstawia porównanie dwóch klasyfikatorów w wersji Bernoulli'ego. Kolumna druga i trzecia opisują wyniki klasyfikatora przygotowanego przez nas, natomiast kolumna czwarta i piąta – klasyfikatora z pakietu języka R. Pierwsza kolumna opisuje rozmiar wektora słów.

#słów	dokładność (własny)	czas (własny)	dokładność (e1071)	czas (e1071)
500	0.8988	0.83	0.8988	23.80
1000	0.9332	1.61	0.9332	3.32
1500	0.9488	2.11	0.9488	4.06
2000	0.9556	3.70	0.9556	1.50
2500	0.962	3.69	0.962	1.81
3000	0.9644	5.36	0.9656	2.29
3500	0.9648	6.02	0.968	2.71
4000	0.964	7.56	0.9696	3.07
4500	0.96	8.74	0.9704	3.52
5000	0.9604	8.36	0.9724	3.96

Poniższy wykres prezentuje zależność dokładności klasyfikacji od rozmiaru wektora słów w wersji Bernoulli'ego.

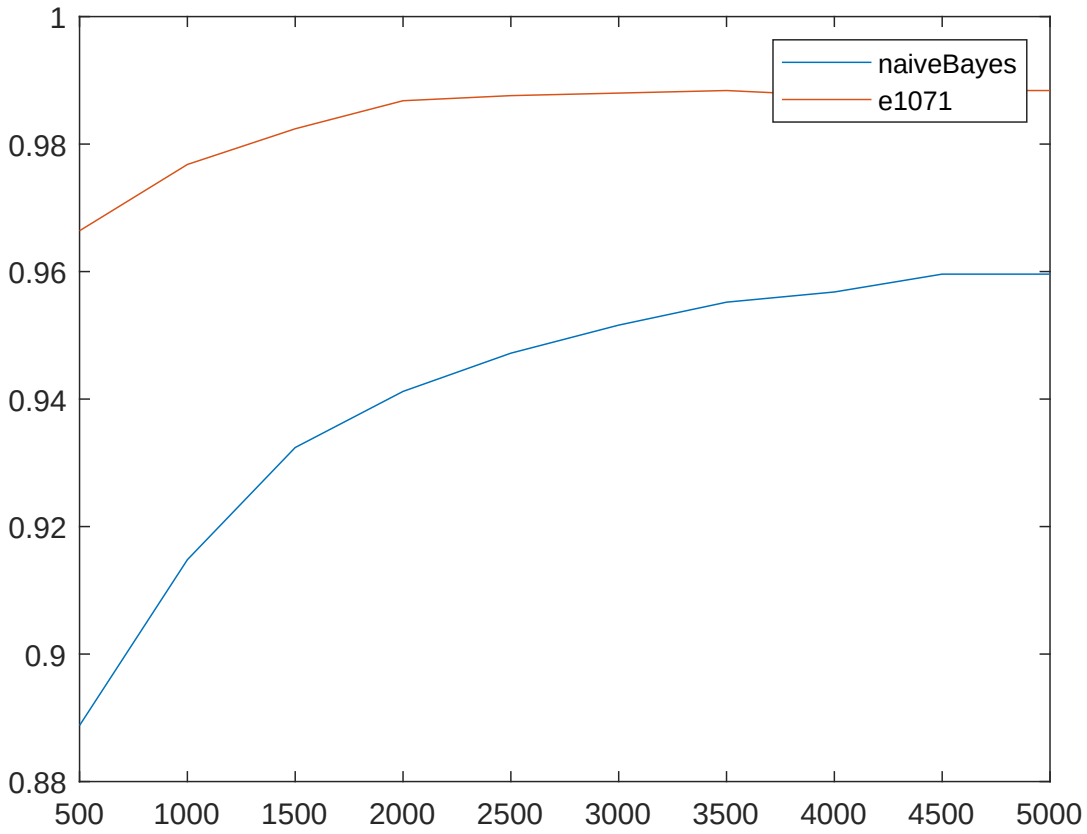


Rysunek 1: Wykres zależności dokładności klasyfikacji od rozmiaru wektora słów w wersji Bernoulli'ego

Poniższa tabela przedstawia porównanie dwóch klasyfikatorów w wersji Mulinomial'ego. Kolumna druga i trzecia opisują wyniki klasyfikatora przygotowanego przez nas, natomiast kolumna czwarta i piąta – klasyfikatora z pakietu języka R. Pierwsza kolumna opisuje rozmiar wektora słów.

#słów	dokładność (własny)	czas (własny)	dokładność (e1071)	czas (e1071)
500	0.8888	36.68	0.9664	18.28
1000	0.9148	8.32	0.9768	3.66
1500	0.9324	11.88	0.9824	4.03
2000	0.9412	16.66	0.9868	1.45
2500	0.9472	19.94	0.9876	1.71
3000	0.9516	26.76	0.988	2.52
3500	0.9552	30.89	0.9884	2.57
4000	0.9568	35.14	0.9876	2.90
4500	0.9596	42.27	0.9884	3.40
5000	0.9596	49.09	0.9884	3.71

Poniższy wykres prezentuje zależność dokładności klasyfikacji od rozmiaru wektora słów w wersji Multinomial'ego.



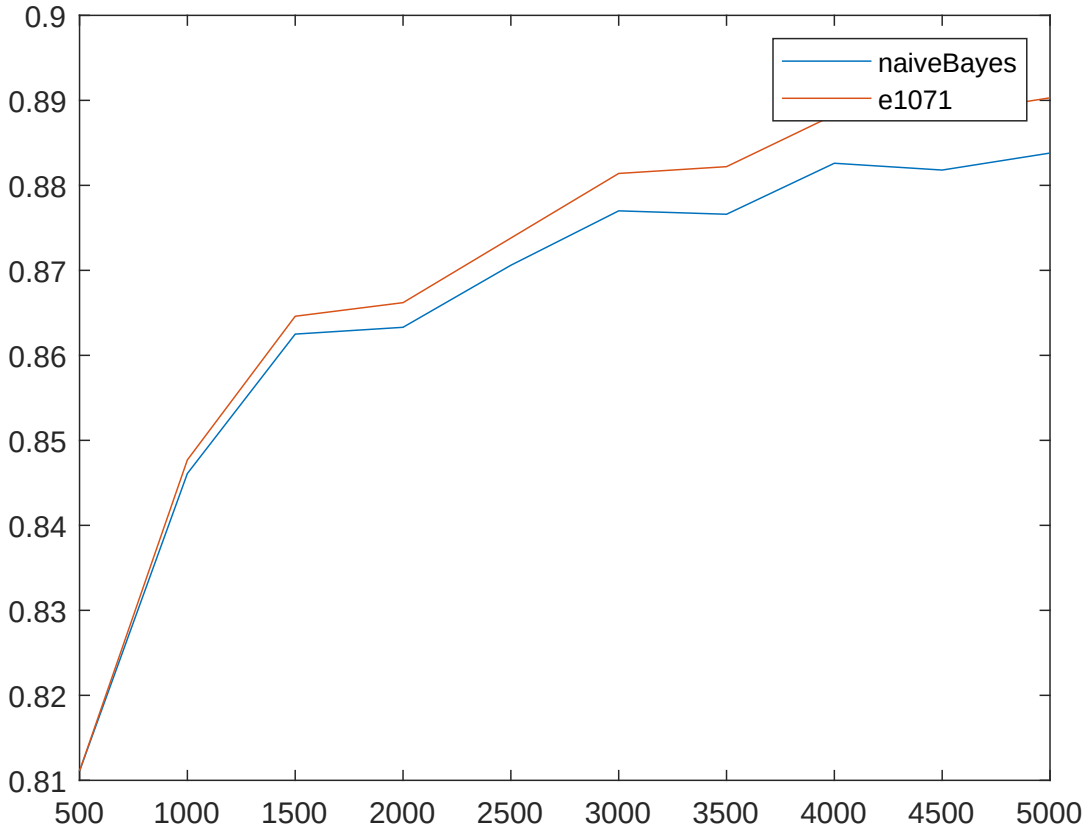
Rysunek 2: Wykres zależności dokładności klasyfikacji od rozmiaru wektora słów w wersji Multinomial'ego

7.2 Zbiór danych 20_newsgroups

Poniższa tabela przedstawia porównanie dwóch klasyfikatorów w wersji Bernoulli'ego. Kolumna druga i trzecia opisują wyniki klasyfikatora przygotowanego przez nas, natomiast kolumna czwarta i piąta – klasyfikatora z pakietu języka R. Pierwsza kolumna opisuje rozmiar wektora słów.

#słów	dokładność (własny)	czas (własny)	dokładność (e1071)	czas (e1071)
500	0.8111	0.30	0.8111	21.69
1000	0.8461	0.6989	0.8477	42.91
1500	0.8625	1.14	0.8646	51.61
2000	0.8633	1.3	0.8662	1.53
2500	0.8706	2.23	0.8738	1.70
3000	0.8770	2.24	0.8814	2.04
3500	0.8766	2.93	0.8822	2.59
4000	0.8826	4.89	0.8883	2.98
4500	0.8818	4.05	0.8883	3.62
5000	0.8838	5.01	0.8903	3.93

Poniższy wykres prezentuje zależność dokładności klasyfikacji od rozmiaru wektora słów w wersji Bernoulli'ego.

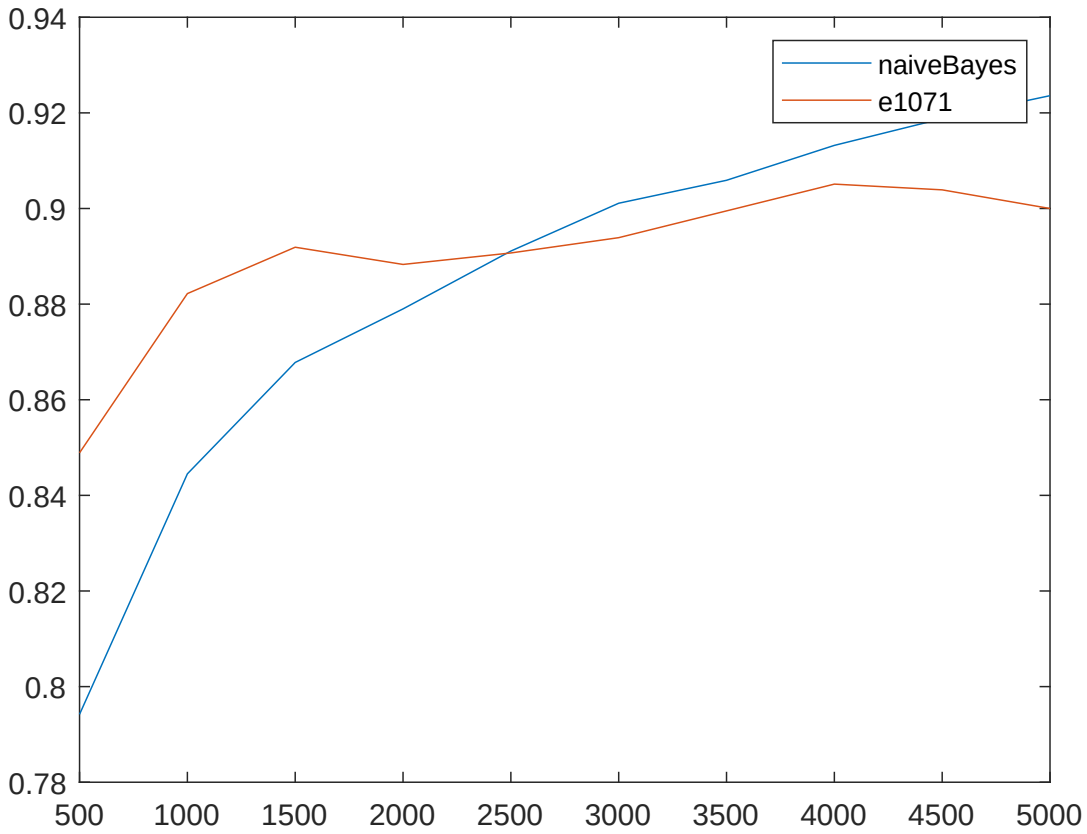


Rysunek 3: Wykres zależności dokładności klasyfikacji od rozmiaru wektora słów w wersji Bernoulli'ego

Poniższa tabela przedstawia porównanie dwóch klasyfikatorów w wersji Mulinomial'ego. Kolumna druga i trzecia opisują wyniki klasyfikatora przygotowanego przez nas, natomiast kolumna czwarta i piąta – klasyfikatora z pakietu języka R. Pierwsza kolumna opisuje rozmiar wektora słów.

#słów	dokładność (własny)	czas (własny)	dokładność (e1071)	czas (e1071)
500	0.7942	0.79	0.8489	21.80
1000	0.8445	1.47	0.8822	4.77
1500	0.8678	2.15	0.8919	1.03
2000	0.8790	2.90	0.8883	1.68
2500	0.8911	4.13	0.8907	1.86
3000	0.9011	3.70	0.8939	2.22
3500	0.9059	5.21	0.8995	2.47
4000	0.9132	7.92	0.9051	2.98
4500	0.9188	8.37	0.9039	3.81
5000	0.9236	10.22	0.9000	4.80

Poniższy wykres prezentuje zależność dokładności klasyfikacji od rozmiaru wektora słów w wersji Multinomial'ego.



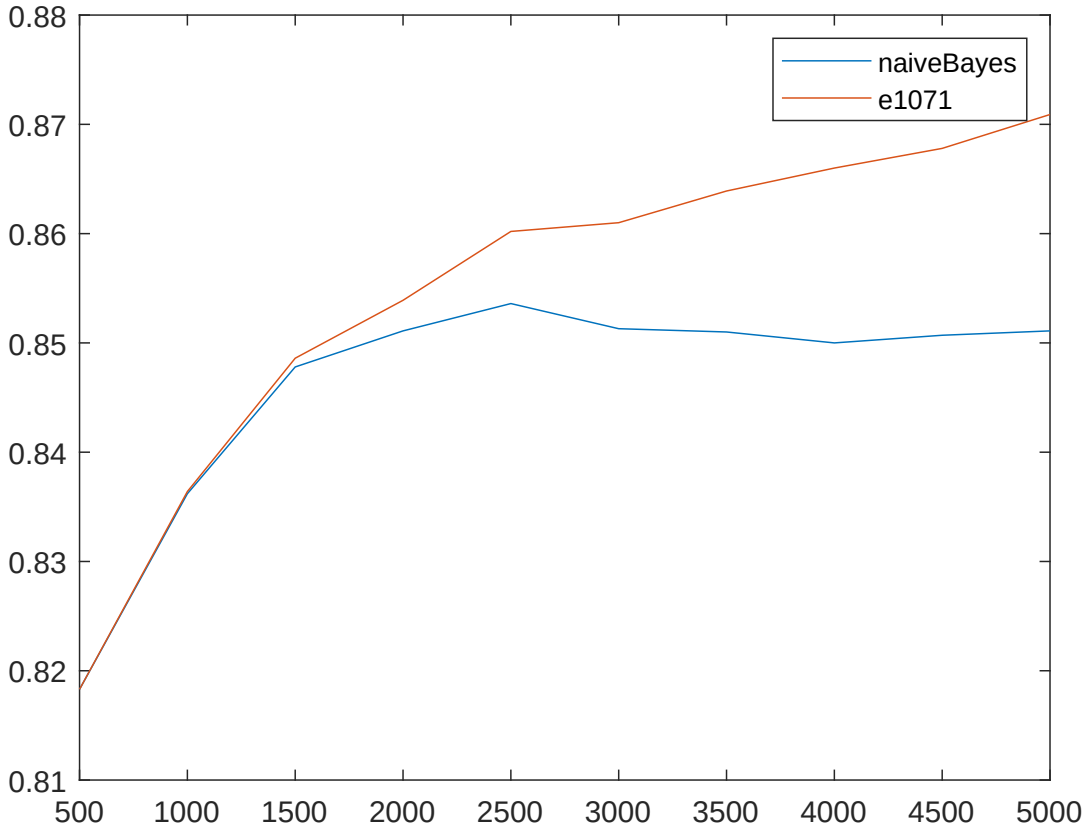
Rysunek 4: Wykres zależności dokładności klasyfikacji od rozmiaru wektora słów w wersji Multinomial'ego

7.3 Zbiór danych emotions

Poniższa tabela przedstawia porównanie dwóch klasyfikatorów w wersji Bernoulli'ego. Kolumna druga i trzecia opisują wyniki klasyfikatora przygotowanego przez nas, natomiast kolumna czwarta i piąta – klasyfikatora z pakietu języka R. Pierwsza kolumna opisuje rozmiar wektora słów.

#słów	dokładność (własny)	czas (własny)	dokładność (e1071)	czas (e1071)
500	0.8183	0.66	0.8183	1.27
1000	0.8362	1.20	0.8364	2.78
1500	0.8478	2.18	0.8486	4.65
2000	0.8511	2.84	0.8539	6.89
2500	0.8536	3.68	0.8602	8.46
3000	0.8513	4.23	0.8610	9.82
3500	0.8510	6.37	0.8639	10.94
4000	0.8500	7.10	0.8660	13.52
4500	0.8503	8.21	0.8683	12.63
5000	0.8511	8.38	0.8709	16.40

Poniższy wykres prezentuje zależność dokładności klasyfikacji od rozmiaru wektora słów w wersji Bernoulli'ego.

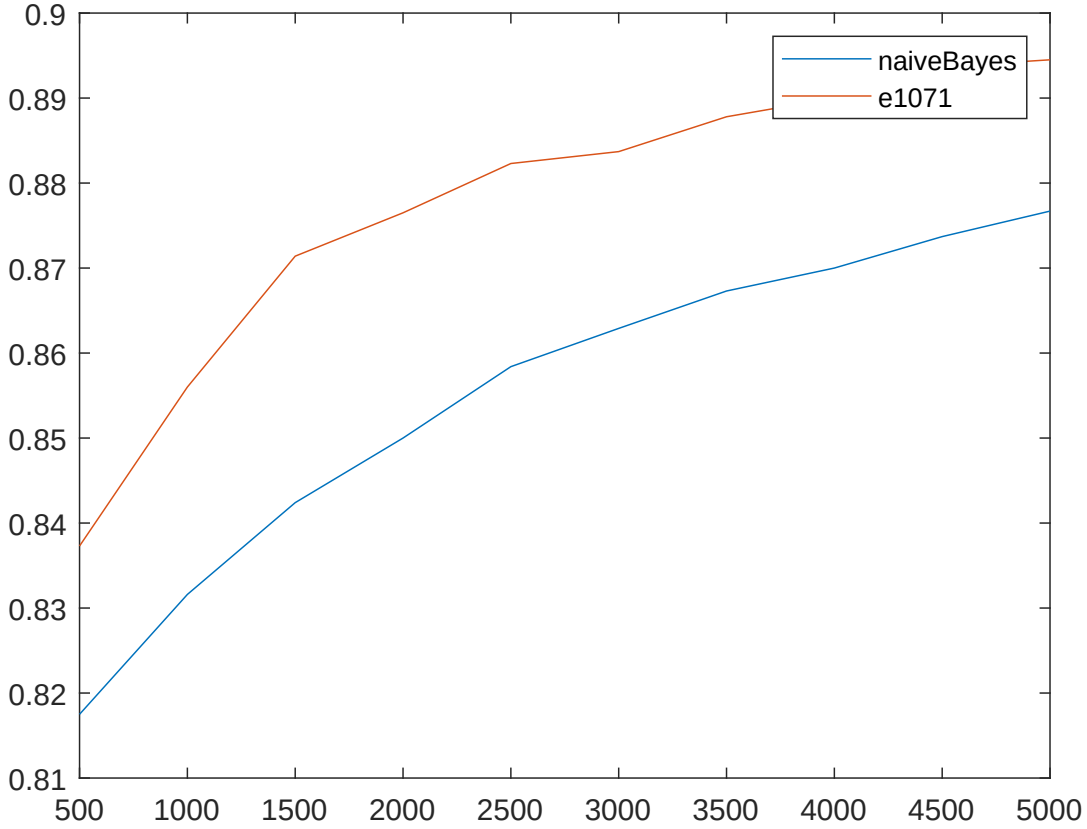


Rysunek 5: Wykres zależności dokładności klasyfikacji od rozmiaru wektora słów w wersji Bernoulli'ego

Poniższa tabela przedstawia porównanie dwóch klasyfikatorów w wersji Mulinomial'ego. Kolumna druga i trzecia opisują wyniki klasyfikatora przygotowanego przez nas, natomiast kolumna czwarta i piąta – klasyfikatora z pakietu języka R. Pierwsza kolumna opisuje rozmiar wektora słów.

#słów	dokładność (własny)	czas (własny)	dokładność (e1071)	czas (e1071)
500	0.8175	1.54	0.8373	1.21
1000	0.8316	2.65	0.8560	2.85
1500	0.8424	5.20	0.8714	5.07
2000	0.8500	6.47	0.8765	6.33
2500	0.8584	12.14	0.8823	14.47
3000	0.8629	8.90	0.8837	10.70
3500	0.8673	9.08	0.8878	11.60
4000	0.8704	11.90	0.8903	11.31
4500	0.8737	21.01	0.8937	19.16
5000	0.8767	15.63	0.8945	16.83

Poniższy wykres prezentuje zależność dokładności klasyfikacji od rozmiaru wektora słów w wersji Multinomial'ego.



Rysunek 6: Wykres zależności dokładności klasyfikacji od rozmiaru wektora słów w wersji Multinomial'ego

8 Wnioski

Po zaimplementowaniu klasyfikatora i wykonanych badaniach możemy stwierdzić, iż jest to bardzo prosty, a jednocześnie charakteryzujący się dużą skutecznością klasyfikator – wskazują na to małe błędy klasykacji. Jedną z wad tego klasyfikatora jest długi czas obliczeń. Obliczenia, szczególnie dla dużych zbiorów danych (np. *emotions*), są bardzo czasochłonne. Dzięki eksperymentom na różnych zbiorach danych mogliśmy zauważyć jak różne czynniki (takie jak: liczba klas, liczba plików tekstowych w danej klasie) wpływają na jakość klasyfikacji. Szczególnie istotnym elementem w analizowaniu treści tekstu jest wstępne przetwarzanie danych – mogliśmy to zrobić w prosty sposób dzięki paczkom języka R. W ten sposób klasyfikacja była dokładniejsza, gdyż np. słowa różniące się tylko końcówką fleksyjną były traktowane jako jedno słowo.

Dzięki przeprowadzonym eksperymentom możemy zauważyć, iż nasza implementacja klasyfikatora nie odbiega znacząco od klasyfikatora z pakietu języka R.

Warto zauważyć, że w przypadku klasyfikatora *Bernoulli NB* wyniki dla mniejszych rozmiarów słownika w obu implementacjach były praktycznie takie same. Wraz ze wzrostem wielkości słownika, nasza implementacja klasyfikatora działała coraz gorzej w porównaniu do klasyfikatora z paczki *e1071*. Najprawdopodobniej jest to spowodowane faktem, że klasyfikator z paczki *e1071* wykorzystuje logarytmiczną wersję klasyfikatora bayesowskiego, przez co jest bardziej odporny na błędy numeryczne. Wraz ze wzrostem wielkości słownika, wzrasta ilość obliczeń, co powoduje wzrost błędów związanych z faktem, że dokładność komputera jest ograniczona.

Jeśli chodzi o klasyfikator *Multinomial NB*, to ciężko jednoznacznie porównać go z wynikami klasyfikatora z paczki *e1071*. Jest to spowodowane faktem, że w paczce *e1071* nie jest zaimplementowany tego typu klasyfikator. Można jednak zauważyć, że czasem lepiej sprawiał się nasz klasyfikator *Multinomial NB*, a czasem klasyfikator z paczki *e1071*.