Drzewo, wyrażenia regularne

Jonatan Kasperczak, Radosław Kasprzak

April 2025

1 Interpretacja tematu projektu

Drzewo wyrażenia regularnego jako narzędzie ekstrakcji cech tekstu w klasyfikacji ML

Temat zakłada wykorzystanie struktury drzewa składniowego wyrażenia regularnego jako źródła cech, które można wykorzystać w zadaniu klasyfikacji tekstu. Każde wyrażenie regularne zostanie rozłożone na drzewo reprezentujące jego składnię, z którego zostaną wyodrębnione cechy opisujące jego złożoność strukturalną.

Następnie, dla wybranego zbioru tekstów (wiadomości SMS), zostaną wykonane dopasowania tekstów do zbioru wyrażeń regularnych. Wyniki dopasowań zostaną potraktowane jako dodatkowe cechy klasyfikacyjne. Celem jest zbudowanie klasyfikatora uczącego się na podstawie tych danych oceniać potencjalną przydatność danego wyrażenia regularnego w kontekście rozróżniania klas (np. SPAM / nie-SPAM).

2 Wstęp

Celem projektu jest zbadanie możliwości wykorzystania wyrażeń regularnych jako źródła cech w klasyfikacji tekstów z użyciem algorytmów uczenia maszynowego. Każde wyrażenie regularne zostanie przekształcone do postaci drzewa składniowego, z którego zostaną wyodrębnione cechy opisujące jego strukturę (np. liczba poziomów, liczba operatorów alternatywy, liczba operatorów powtórzenia).

Następnie, dla każdego tekstu w zbiorze danych, zostanie wykonana próba dopasowania go do danego wyrażenia regularnego. Wynik dopasowania (tak/nie) będzie interpretowany jako dodatkowa cecha. W ten sposób powstanie zestaw cech opisujących zarówno właściwości strukturalne regexu, jak i jego zachowanie względem tekstów w zbiorze danych.

Jako zbiór danych wykorzystany zostanie zbiór SMS Spam Collection dostępny w repozytorium *UCI Machine Learning Repository*, zawierający ponad 5500 wiadomości SMS oznaczonych jako SPAM lub HAM.

Drugim datesetem będzie Phishing Email Detection dostępny na Kaggle, zawierający około 18500 wiadomosći oznaczonych jako SPAM lub HAM.

Trzeci dataset to Enron Spam Dataset to zbiór rzeczywistych e-maili z firmy Enron, który zawiera zarówno wiadomości spamowe, jak i niespamowe (ham), oznaczone ręcznie i przypisane do konkretnych użytkowników. Jest powszechnie wykorzystywany w badaniach nad filtrowaniem spamu oraz w uczeniu maszynowym, ponieważ odzwierciedla autentyczną strukturę i treść firmowej korespondencji.

Zestaw wyrażeń regularnych zostanie przygotowany ręcznie na podstawie analizy charakterystycznych wzorców występujących w wiadomościach typu SPAM. Będzie obejmował od 20 do 50 wyrażeń, które odpowiadają typowym schematom tego typu wiadomości, m.in

- Słowa kluczowe, takie jak `free`, `win`, `urgent`, `click`.
- Wyrażenia zawierające linki, np. `http[s]?://\S+`.
- Wzorce liczbowe, np. `\d{5,9}`.
- Kombinacje strukturalne, np. `win.*now`, `\bcall\b.*\bnow\b`.

Dodatkowo planowane jest rozszerzenie zbioru wyrażeń poprzez automatyczne generowanie kolejnych wariantów wzorców lub w oparciu o analizę statystyczną tokenów pojawiających się najczęściej w wiadomościach SPAM.

Na podstawie uzyskanych cech zostanie wytrenowany klasyfikator uczący się oceniać, czy dane wyrażenie regularne jest użyteczne w kontekście klasyfikacji wiadomości tekstowych. Porównane zostaną różne modele klasyfikacji, takie jak drzewa decyzyjne i lasy losowe, a uzyskane wyniki zostaną ocenione pod kątem dokładności i stabilności predykcji.

Projekt zakłada pełną implementację parsera regexów do drzewa, ekstraktora cech oraz klasyfikatora, a także przeprowadzenie eksperymentów z udziałem rzeczywistego zbioru danych.

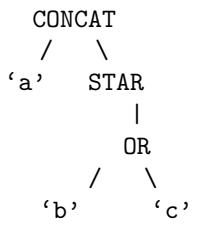
3 Opis algorytmów

1. **Parser wyrażenia regularnego do drzewa składniowego**

   Cel: przekształcenie napisu z wyrażeniem regularnym do drzewa reprezentującego jego strukturę.

   Opis krok po kroku:

   - Zainicjalizuj stos dla operatorów i dla operandów.
   - Iteruj po znakach w wyrażeniu regularnym:
     - Jeśli to znak literalny (np. litera, cyfra) → dodaj jako liść do stosu operandów.
     - Jeśli to nawias otwierający → dodaj do stosu operatorów.
     - Jeśli to nawias zamykający → rozwiń stos do nawiasu otwierającego.
     - Jeśli to operator (`|`, `*`, `+`, `.`) → zastosuj odpowiedni algorytm przetwarzania zgodnie z priorytetem.
   - Po przetworzeniu wszystkich znaków, rozwiń pozostałe operacje ze stosu operatorów.
   - Korzeniem drzewa będzie ostatni element stosu operandów.



2. **Ekstrakcja cech**

   Cel: uzyskanie wektora cech liczbowych opisujących strukturę drzewa

   - liczba poziomów drzewa,
   - liczba operatorów alternatywy (`|`),
   - liczba operatorów powtórzenia (`*`, `+`),
   - liczba znaków literalnych,
   - głębokość najdłuższej ścieżki,
   - liczba grupowania (nawiasy).

3. Klasyfikator drzewa decyzyjnego (Decision Tree Classifier)

   Drzewo decyzyjne jest klasyfikatorem opartym na strukturze hierarchicznej, w której każdy węzeł wewnętrzny reprezentuje test warunkowy na wartość cechy, a każdy liść — przypisaną klasę. W tym projekcie zastosowano uproszczony, binarny wariant drzewa, który dzieli dane względem jednej cechy i wartości progowej w każdym węźle.

   Kroki działania algorytmu:

   - **Warunki stopu:** Drzewo przestaje się rozwijać, gdy:
     - osiągnięto maksymalną głębokość (`max_depth`),
     - liczba przykładów w danym węźle jest mniejsza niż `min_samples_split`,
     - wszystkie etykiety w węźle są identyczne.
   - **Wybór najlepszego podziału:**
     - Dla każdej cechy iterowane są unikalne wartości (progi).
     - Obliczany jest zysk informacyjny dla każdego możliwego podziału.
     - Wybierany jest podział maksymalizujący zysk informacyjny.
   - **Zysk informacyjny (Information Gain):**

     $$IG = H(Y) - \left(\frac{n_L}{n} H(Y_L) + \frac{n_R}{n} H(Y_R)\right)$$

     gdzie $H(Y)$ to entropia oryginalnego zbioru, a $H(Y_L), H(Y_R)$ to entropie zbiorów po podziale.
   - **Budowa drzewa:** Dla najlepszego podziału tworzony jest węzeł z indeksem cechy i progiem. Rekurencyjnie budowane są lewe i prawe poddrzewa.
   - **Predykcja:** Każdy przykład testowy przechodzi kolejne warunki od korzenia do liścia, gdzie zwracana jest etykieta klasy.

Złożoność obliczeniowa

- Trenowanie: $\mathcal{O}(n \cdot m \cdot \log n)$,
- Predykcja: $\mathcal{O}(\log n)$.

Zalety i ograniczenia

Cecha	Decision Tree
Interpretowalność	Wysoka
Szybkość predykcji	Bardzo szybka
Odporność na szum	Niska
Ryzyko przeuczenia	Wysokie

Algorithm 1 Trenowanie uproszczonego drzewa decyzyjnego

Require: Macierz cech X, etykiety y, głębokość d
if d osiągnęła maksymalną głębokość lub wszystkie etykiety są identyczne **then**
  Zwróć liść z najczęstszą etykietą
end if
for każda cecha i **do**
  for każda unikalna wartość progu t w kolumnie i **do**
    Podziel dane względem t: lewa i prawa część
    Oblicz zysk informacyjny dla tego podziału
  end for
end for
Wybierz najlepszy podział i, t z największym zyskiem
Zbuduj lewe i prawe poddrzewa rekurencyjnie
Zwróć węzeł z cechą i, progiem t i dwoma poddrzewami

3.1 Szczegóły etykietowania skuteczności regexów

Definicja skuteczności regexu (dla SPAM):

$$\text{skuteczność} = \frac{TP}{TP + FP + FN}$$

Gdzie:

- **TP** – regex dopasował wiadomość SPAM,
- **FP** – regex dopasował wiadomość HAM,
- **FN** – regex nie dopasował wiadomości SPAM.

Przykład (regex `win.*now`):

- SPAM dopasowane: 30
- HAM dopasowane: 10
- SPAM łącznie: 50

- Skuteczność: $\frac{30}{30+10+20} = 0{,}5$

Etykiety klas:

- skuteczność $< 0.3 \rightarrow$ **słaby**,
- $0.3 \leq$ skuteczność $< 0.6 \rightarrow$ **przeciętny**,
- skuteczność $\geq 0.6 \rightarrow$ **dobry**.

Alternatywnie możliwe jest podejście regresyjne: przewidywanie skuteczności jako wartość ciągła z przedziału [0, 1].

3.2 Podział zbioru treningowego i testowego

Podejście:

- 80% regexów do treningu, 20% do testu,
- alternatywnie: k-krotna walidacja krzyżowa (np. k = 5).

Możliwe strategie podziału:

- podział na poziomie regexów (np. 40 treningowych, 10 testowych),
- podział na poziomie wiadomości (cross-data).

3.3 Miary jakości klasyfikatora regexów

Dla modelu klasyfikującego regexy:

- **Accuracy** – ogólna trafność etykiet (dobry / przeciętny / słaby),
- **Precision** – trafność przewidywanych regexów jako „dobrych”,
- **Recall** – wykrywalność faktycznie dobrych regexów,
- **F1-score** – średnia harmoniczna precision i recall,
- **Confusion matrix** – macierz pomyłek 3×3 dla klas.

Dla klasyfikacji SPAM/HAM na poziomie pojedynczego regexu (opcjonalnie):

- Precision, recall, accuracy dla dopasowań regexu.

Wyniki będą prezentowane jako:

- tabele,
- macierze pomyłek (confiusion matrix)
- wykresy (biblioteki `matplotlib`, `seaborn`).

4 Środowisko implementacji

Implementacja zostanie wykonana w języku **Python 3.12**, z wykorzystaniem bibliotek:

- `re` – dopasowywanie wyrażeń regularnych,
- `scikit-learn` – algorytmy ML i metryki,
- `pandas`, `numpy` – przetwarzanie danych,
- `matplotlib`, `seaborn` – wizualizacja wyników.

Kod będzie działał w systemie Linux (Ubuntu 24.04), zgodnie z wymaganiami przedmiotu.

5 Otwarte kwestie

- Ostateczny dobór zbioru regexów – wstępnie zostanie przygotowany zestaw 20–50 ręcznie wybranych regexów, ale możliwe jest automatyczne ich rozszerzenie.
- Sposób etykietowania skuteczności regexu – możliwe są różne podejścia (próg trafień, grupowanie jakościowe, klasyfikacja ciągła).
- Rozszerzenie projektu o automatyczne generowanie wyrażeń regularnych w drugiej fazie – zależne od dostępnego czasu.
- Weryfikacja jakość zbioru danych