

Analiza porównawcza skuteczności wybranych modeli językowych w postprocessingu raportów z narzędzi do statycznej analizy kodu

Bartłomiej Ściseł

Promotor: dr inż. Krystian Radlak

Wydział Elektroniki i Technik Informacyjnych
Instytut Automatyki i Informatyki Stosowanej
Warszawa, 13 października 2026



Raporty przerastają możliwości ręcznego przeglądu

19 212

ostrzeżeń zgłosiły w Jetty trzy narzędzia



SonarQube	16 366
SpotBugs	1823
Error Prone	1023



8–20 min

na przegląd jednego ostrzeżenia (Google, Tencent)



> 30%

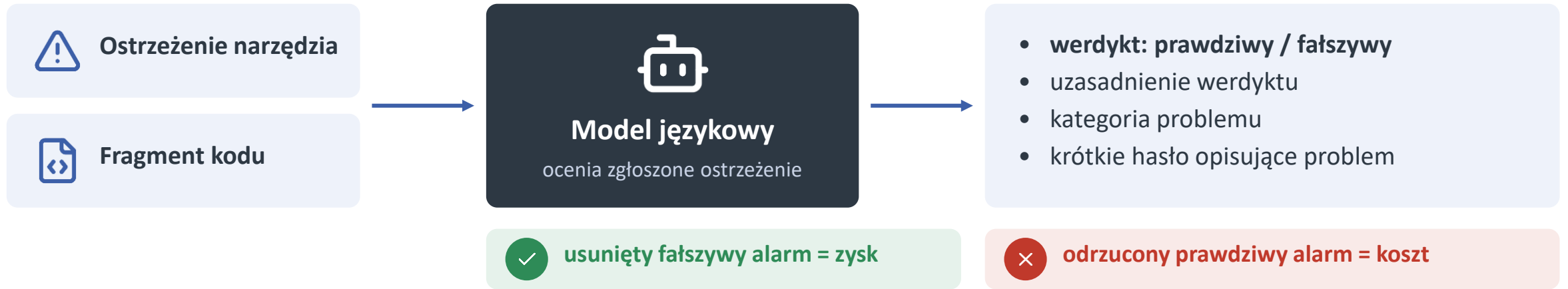
fałszywych alarmów i użytkownicy zaczynają ignorować narzędzie (twórcy analizatora Coverity)



19 z 20

badanych programistów: wiele narzędzi nie wyjaśnia, na czym polega problem

Model językowy jako filtr raportu



Dotychczas

zbiory syntetyczne; rzeczywistego kodu mniej

etykiety z historii zmian: trafne w 39 z 200 sprawdzonych

przeglądy: głównie FindBugs, Error Prone rzadziej

jedno wywołanie albo głosowanie; rozrzutu nie mierzono

W tej pracy

→ rzeczywisty projekt w Javie: Jetty

→ etykieta nadana ręcznie każdemu ostrzeżeniu

→ SonarQube, SpotBugs i Error Prone

→ trzy powtórzenia i pomiar stabilności

Cel i hipotezy

Hipoteza główna: postprocessing modelem językowym podnosi użyteczność raportów zawierających wyniki statycznej analizy kodu w języku Java, a wielkość tej poprawy zależy od zastosowanego modelu oraz od danych, które model otrzymuje

**H1**

filtr podnosi F1 względem surowego raportu

**H2**

różnice trafności; przewaga nad klasyfikatorem naiwnym

**H3**

różnice stabilności werdyktu w trzech powtórzeniach

**H4**

uzasadnienie odwołuje się do kodu częściej niż komunikat narzędzia

**H5**

lokalny 9B:
≥ 80% wartości F1 najdroższego modelu

**H6**

trafność zmienia się, gdy model nie widzi kodu

Pytania opisowe: trafność kategoryzacji · grupowanie ostrzeżeń · jakość uzasadnień

Dwa zbiory, trzy narzędzia

Baza przykładowych błędów

30

klas Javy, w każdej jeden ukryty błąd

43

ostrzeżenia zgłoszone przez narzędzia

43 prawdziwe

Mierzy: ile prawdziwych alarmów model wyrzuci?

Przykłady: T. Valeev · klasy: I. Okniński (2025) · podpowiedzi w nazwach i komentarzach usunięte

Jetty Core: serwer HTTP w Javie

jetty://**307 tys.**

wierszy kodu, 34 moduły

19 212

ostrzeżeń w raportach

360ostrzeżeń w próbie,
ze 123 reguł**144 prawdziwe****216 fałszywych**

Mierzy: ile fałszywych alarmów model usunie i ile prawdziwych straci?

Po 120 ostrzeżeń z każdego narzędzia · bez kodu testowego · każda etykieta nadana ręcznie

**SonarQube** – kod źródłowy**SpotBugs** – bajtkod**Error Prone** – wtyczka kompilatora

Fałszywy alarm = nieprawdziwe twierdzenie o kodzie

Error Prone · reguła OperatorPrecedence · Frame.java, wiersz 23 (Jetty)

```
22 return frame instanceof HeadersFrame && ((HeadersFrame)frame).isLast() ||  
23     frame instanceof DataFrame && ((DataFrame)frame).isLast();
```

„Użyj nawiasów grupujących, aby jawnie pokazać priorytet operatorów” komunikat narzędzia, tłum.



Czy twierdzenie narzędzia jest prawdziwe?

Tak: wzorzec występuje i szkodzi czytelności, bo kolejność działań wynika tylko z priorytetu operatorów.

Alarm prawdziwy. To oceniam.



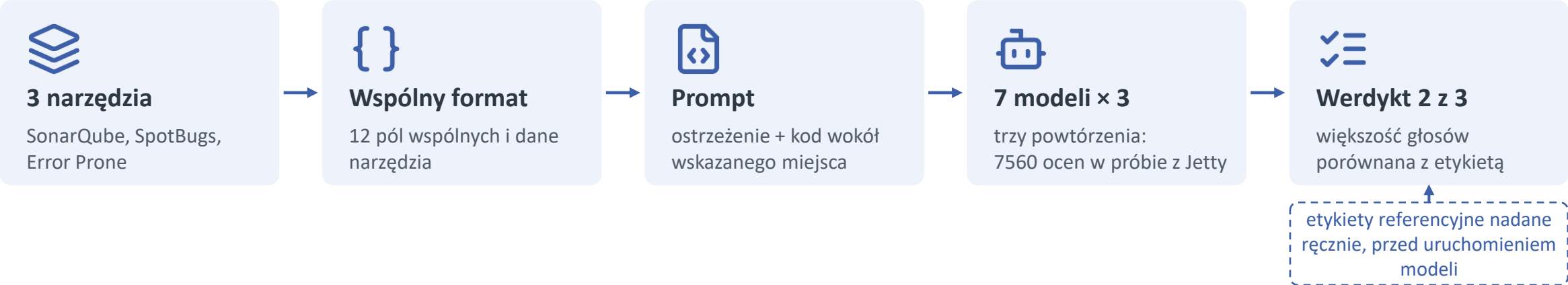
Czy warto to poprawić?

Zależy od zespołu i projektu. Program działa poprawnie.

Osobne pytanie, poza zakresem pracy.

Etykiety referencyjne nadane ręcznie, zanim uruchomiono modele. Niska waga poprawki nie czyni alarmu fałszywym.

Przebieg eksperymentu

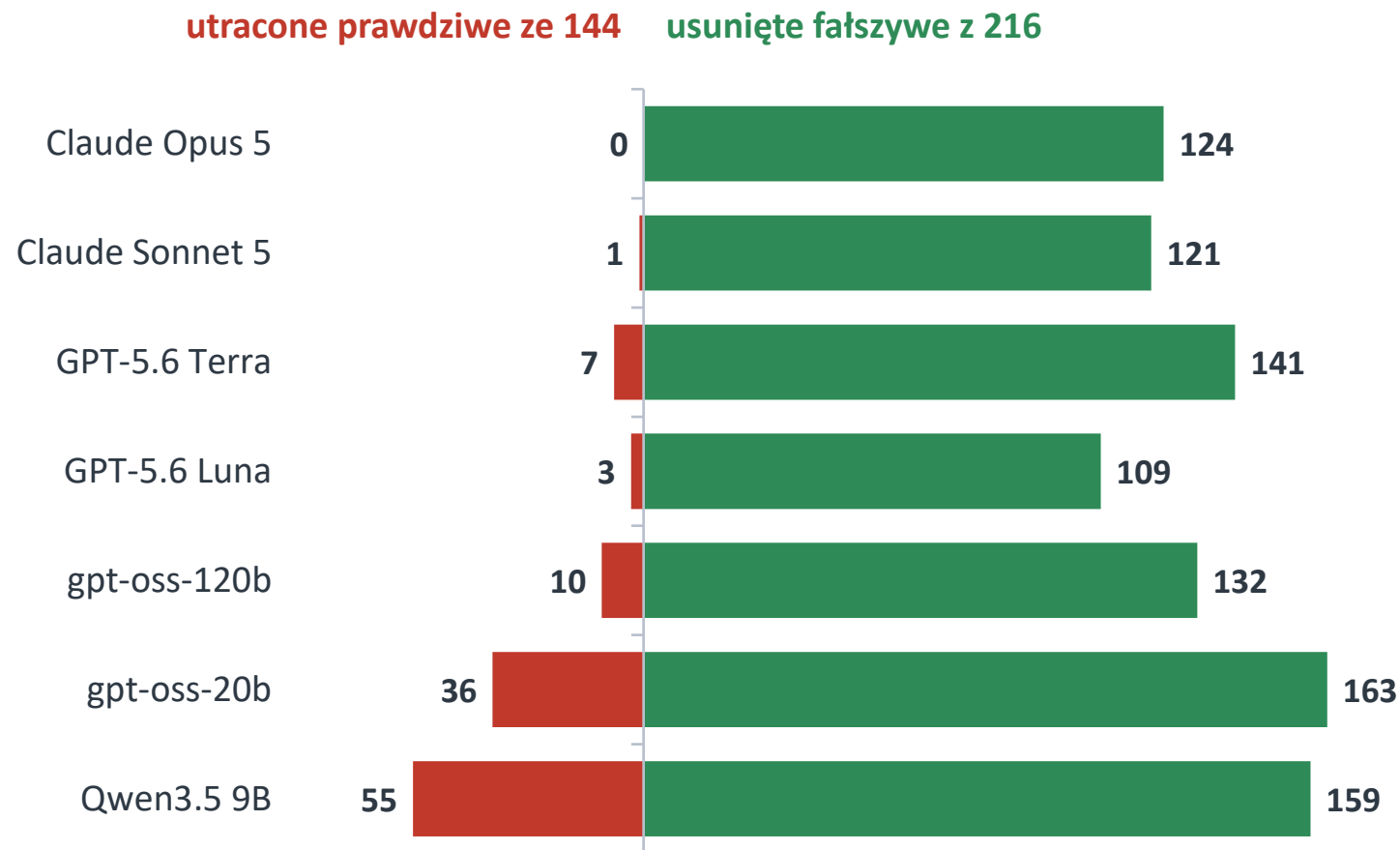


wagi zamknięte • API dostawcy				wagi otwarte		
Claude Opus 5 5 / 25 USD	Claude Sonnet 5 2 / 10 USD	GPT-5.6 Terra 2 / 12 USD	GPT-5.6 Luna 0,20 / 1,20 USD	gpt-oss 120b 0,037 / 0,17 USD	gpt-oss 20b 0,030 / 0,14 USD	Qwen3.5 9B lokalnie

Cena w USD za milion tokenów: wejście / wyjście. Qwen3.5 9B uruchamiany lokalnie (Ollama, RTX 5070 Ti, 16 GB).

Wariant bez kodu (H6): te same 360 ostrzeżeń, jedna ocena na ostrzeżenie
Statystyka: ostrzeżenia jednej reguły mają podobne etykiety (ICC = 0,617), więc losuję całe reguły; korekta Holma dla 22 testów

Każdy model filtruje, nie każdy bezpiecznie

**H1** 5 z 7 modeli

wzrost F1 względem surowego raportu:
cały przedział ufności powyżej zera

gpt-oss-20b i Qwen3.5 9B

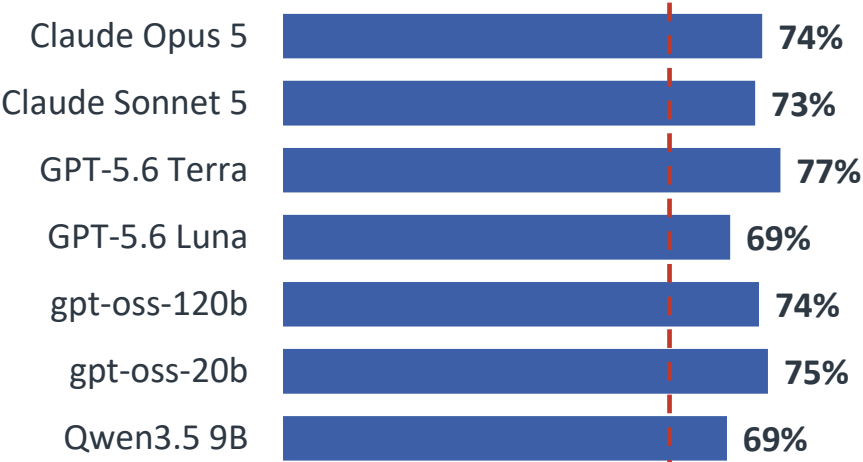
przedział przyrostu F1 obejmuje zero;
tracą 25% i 38% prawdziwych

Baza błędów: 43 prawdziwe

Opus, Sonnet i Luna tracą po 1, Qwen
aż 7: najwięcej w obu zbiorach

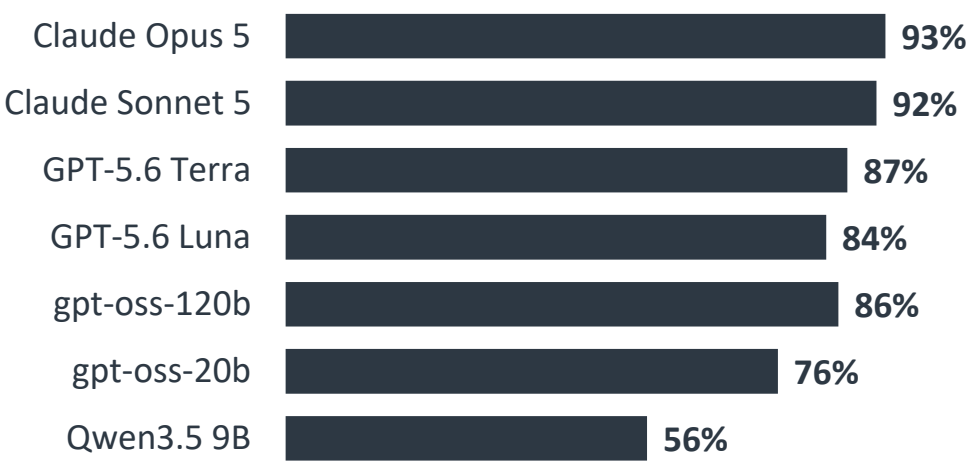
Modele różnią się stabilnością bardziej niż trafnością

Trafność · werdykt większościowy



klasyfikator naiwny 60%
(odrzuca wszystko)

Jednomyślność trzech powtórzeń



H2 · różnice trafności; przewaga nad klasyfikatorem naiwnym

nie wykazano ani różnic ($p = 0,70$), ani przewagi nad naiwnym



H3 · różnice stabilności werdyktu w trzech powtórzeniach

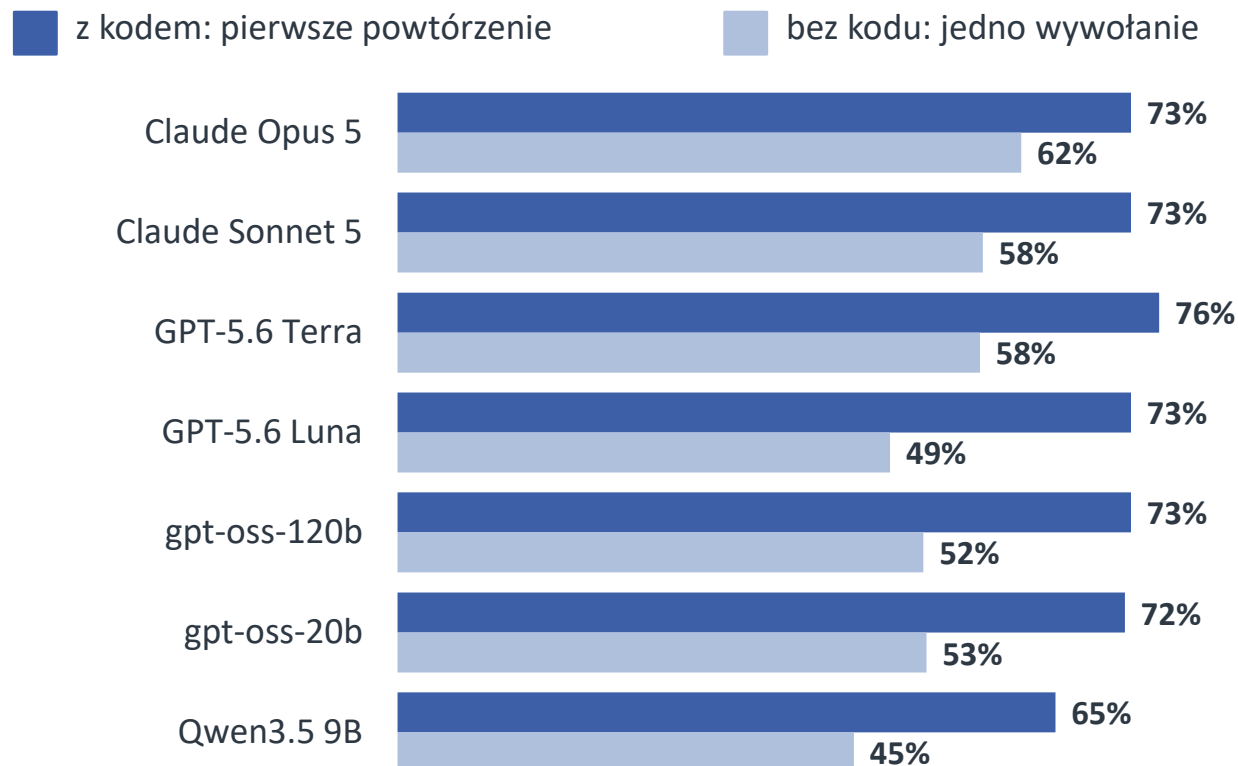
potwierdzona: różnica w 15 z 21 par modeli



H5 · lokalny 9B: $\geq 80\%$ F1 najdroższego modelu

nie wykazano: 81% F1 Opusa, dolna granica przedziału 71%

Kontekst jest ważniejszy niż wybór modelu



-11 do -24 p.p.

spadek trafności po usunięciu fragmentu kodu

Dla porównania: najlepszy i najgorszy model z kodem dzieli 10 p.p. (76% i 65%)



H6 · zmiana trafności bez kodu
istotny spadek u 6 z 7 modeli

Bez kodu modele częściej zostawiają ostrzeżenie: Luna zostawiła w raporcie 91% ostrzeżeń.

✓ **H4 · odwołania do kodu (5 z 7):** identyfikator z pokazanego kodu zawiera 66–86% uzasadnień i 52% komunikatów narzędzi

Modele częściej mylą się we wniosku niż w odczycie kodu

Mechanizmy pomyłek wyróżnione w lekturze



Brak kontekstu
fakt rozstrzygający leży poza wycinkiem kodu



Złe kryterium
stosuje inne kryterium niż prompt, np. „czy program działa”



Błędna interpretacja reguły
prawdziwe fakty źle odniesione do reguły



Błędne fakty
nieprawda o kodzie, Javie lub bibliotece

Przykład złego kryterium (slajd 6) · etykieta: alarm prawdziwy

Claude Opus 5	● ● ●	zachował 3:0
Claude Sonnet 5	● ● ●	zachował 3:0
GPT-5.6 Terra	● ● ●	zachował 3:0
GPT-5.6 Luna	● ● ●	zachował 3:0
gpt-oss-120b	● ● ●	odrzucił 2:1
gpt-oss-20b	● ● ●	odrzucił 3:0
Qwen3.5 9B	● ● ●	odrzucił 2:1
● powtórzenie zachowuje ostrzeżenie ● odrzuca		

„Dodanie nawiasów nie zmienia semantyki warunku”
gpt-oss-20b (tłum.)

Ręczna ocena 42 uzasadnień, po 6 z każdego modelu

konkretny fakt z kodu		39/42
przywołane fakty prawdziwe		35/42
wniosek wynika z faktów		21/42

Kategoryzacja i grupowanie ostrzeżeń



Kategoria problemu (6 klas)

74–90%

trafności kategorii u siedmiu modeli

poprawność

współbieżność

bezpieczeństwo

wydajność

utrzymywalność

pozostałe

Pomyłki są asymetryczne:

inna → utrzymywalność



324 oceny

utrzymywalność → inna



9 ocen

kategoria referencyjna → wybór modelu; suma ocen siedmiu modeli



Krótkie hasło (grupowanie)

27

grup łączy ostrzeżenia różnych narzędzi (łącznie dla siedmiu modeli, próg podobieństwa haseł 0,80)



25 z 27 dotyczy tego samego problemu



Ale takie grupy są rzadkie, a ostrzeżenia tej samej reguły często dostają różne hasła. Dosłowne brzmienie hasła nie nadaje się na stabilny identyfikator grupy.

Wnioski

H1  5 z 7 modeli wzrost F1	H2  nie wykazano różnice trafności	H3  potwierdzona różnice stabilności	H4  5 z 7 modeli odwołania do kodu	H5  nie wykazano $9B \geq 80\%$ F1 Opusa	H6  6 z 7 modeli spadek bez kodu
 Zachowawcze Claude Opus 5 · Claude Sonnet 5 0–1 utraconych prawdziwych - najwyższa stabilność 0,07–0,16 USD za ostrzeżenie			 Pośrednie GPT-5.6 Terra · GPT-5.6 Luna · gpt-oss-120b 3–10 utraconych prawdziwych - najwyższe F1 w próbie: Terra 0,001–0,03 USD za ostrzeżenie		 Agresywne gpt-oss-20b · Qwen3.5 9B 36–55 utraconych prawdziwych - najwięcej usuniętych fałszywych, najniższa stabilność $\leq 0,001$ USD za ostrzeżenie albo lokalnie

Model wspiera przegląd raportu; automatyczne odrzucanie ostrzeżeń wymaga dalszej weryfikacji

Ograniczenia i kierunki dalszych badań

Ograniczenia



Jeden projekt: liczb nie da się wprost uogólnić na inne



Etykiety i ocena uzasadnień: jedna osoba



Jeden prompt i konkretne wersje modeli



Brak badania z udziałem programistów



Kierunki dalszych badań



Agent, który sam szuka kontekstu w repozytorium



Ocena krok po kroku: sens reguły, fakty z kodu, dopiero potem werdykt



Trzeci werdykt: „brak danych do rozstrzygnięcia”



Kaskada: tani model ocenia wszystko, drogi tylko przypadki niepewne

Politechnika Warszawska

Dziękuję za uwagę

Bartłomiej Ściseł

Analiza porównawcza skuteczności wybranych modeli językowych w postprocessingu raportów z narzędzi do statycznej analizy kodu



Wyniki szczegółowe: 360 ostrzeżeń z Jetty

Model	Precyzja	Czułość	F1	Trafność	Redukcja fałszywych	Utracone prawdziwe	Jednomysłność	Koszt [USD]
Claude Opus 5	0,610	1,000	0,758	0,744	0,574	0	0,931	0,1630
Claude Sonnet 5	0,601	0,993	0,749	0,733	0,560	1	0,917	0,0664
GPT-5.6 Terra	0,646	0,951	0,770	0,772	0,653	7	0,872	0,0331
GPT-5.6 Luna	0,569	0,979	0,719	0,694	0,505	3	0,839	0,0030
gpt-oss-120b	0,615	0,931	0,740	0,739	0,611	10	0,858	0,0009
gpt-oss-20b	0,671	0,750	0,708	0,753	0,755	36	0,764	0,0009
Qwen3.5 9B	0,610	0,618	0,614	0,689	0,736	55	0,561	lokalnie
Surowy raport	0,400	1,000	0,571	0,400	0,000	0	–	–
Klasyfikator naiwny	–	0,000	0,000	0,600	1,000	144	–	–

Werdykt większościowy z trzech powtórzeń. Koszt: trzy wywołania na jedno ostrzeżenie, według cennika z dnia uruchomienia; kosztu sprzętu i energii dla Qwena nie mierzono.
Tabele 6.1–6.6 w pracy.

Dlaczego losuję całe reguły



360 ostrzeżeń, ale tylko 123 reguły; ICC etykiety w obrębie reguły: 0,617



połowę ostrzeżeń SpotBugs dają 4 z 42 reguł; Error Prone 5 z 32; SonarQube 6 z 49



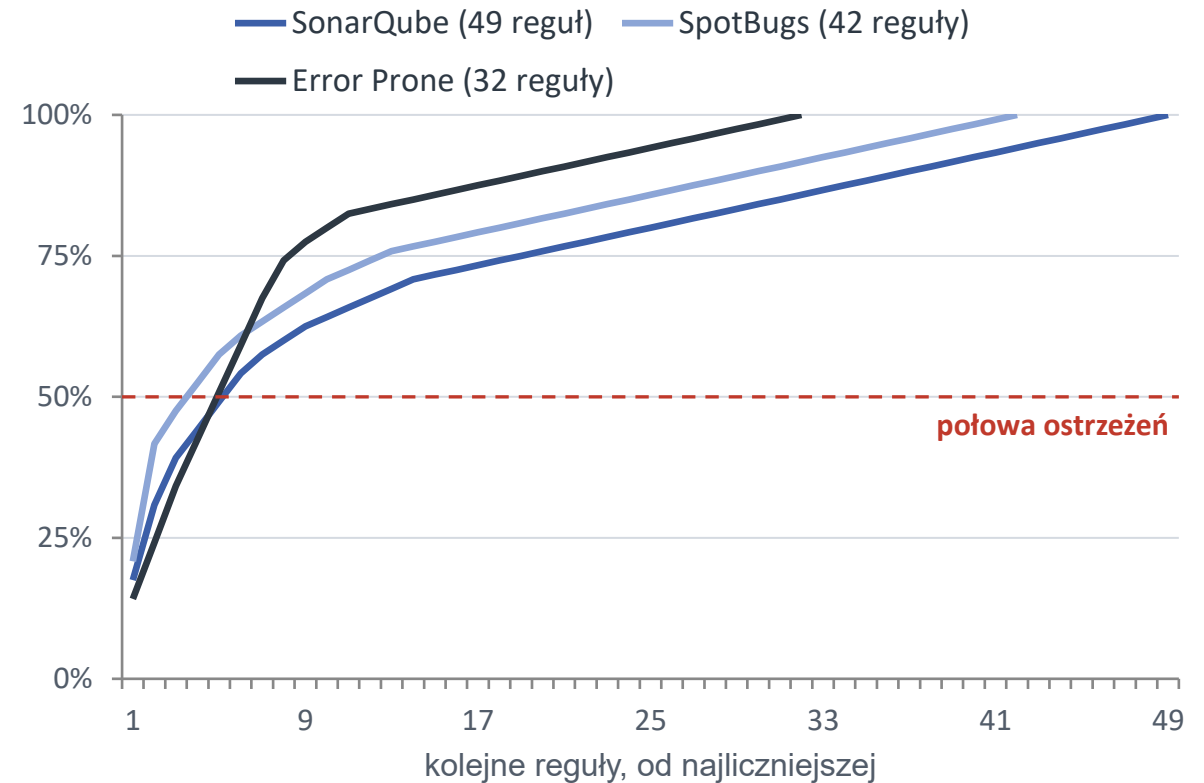
przedziały 95%: bootstrap, 10 000 losowań całych reguł osobno w każdym narzędziu, bez korekty na wielokrotność



korekta Holma, 22 testy: Q Cochрана i 7 porównań z naiwnym (H2), po 7 w H4 i H6; permutacje po regułach (20 000)



H1 i H3 rozstrzygają przedziały ufności, H2, H4 i H6 wartości p po korekcie Holma



Skumulowany udział ostrzeżeń w kolejnych regułach, jak na rysunku 5.2 w pracy

Prompt i odpowiedź modelu

Prompt systemowy

- zadanie i rola modelu
- definicja alarmu prawdziwego i fałszywego
- czego nie traktować jako dowodu fałszywości: niska waga, mała poprawka, inny błąd obok
- niepełny kontekst: zaznacz niepewność, ale wydaj werdykt
- sześć kategorii i zasady ich wyboru

Wiadomość użytkownika

- narzędzie, reguła, komunikat, lokalizacja
- pola właściwe dla narzędzia: CWE, waga, sugerowana poprawka...
- kod: plik ≤ 200 wierszy w całości, dłuższy ± 150 wierszy
- numery wierszy, zakres ostrzeżenia oznaczony

Odpowiedź: JSON, sześć pól

```
{
  "reasoning": "The return expression mi...",
  "is_true_positive": true,
  "confidence": 0.93,
  "category": "MAINTAINABILITY",
  "issue_signature": "mixed logical and/or...",
  "developer_explanation": "The boolean..."
}
```

Claude Opus 5, Frame.java:23, pierwsze powtórzenie.
Analizowane pola: werdykt, uzasadnienie, kategoria i hasło.

Ustawienia: domyślne parametry dostawców, limit odpowiedzi 8192 tokenów, nieudane wywołania ponawiane bez zmiany promptu.

Dobór próby z Jetty

Populacja

19 212 ostrzeżeń w raportach
bez kodu testowego
zwykłe losowanie: kilka licznych reguł
zdominowałoby próbę

Etap 1: losowanie w grupach

po 100 ostrzeżeń z narzędzia
SpotBugs: kategoria × priorytet
SonarQube: typ × waga
Error Prone: reguła
→ 300 ostrzeżeń, 63 reguły

Etap 2: nowe reguły

po 20 reguł nieobecnych w etapie 1
po jednym ostrzeżeniu z reguły
→ 60 ostrzeżeń

Razem: 360 ostrzeżeń, 123 reguły, po 120 z narzędzia

Udział alarmów prawdziwych: Error Prone 0,567 · SonarQube 0,325 · SpotBugs 0,308

Etap 2 dokłada reguły, bo w statystyce to reguła jest jednostką losowania.

Jednomyślność błędnych werdyktów

Model	Błędne werdykty	Jednomyślne 3:0	Rozbieżne 2:1	Udział jednomyślnych
Claude Opus 5	92	80	12	0,870
Claude Sonnet 5	96	83	13	0,865
GPT-5.6 Terra	82	68	14	0,829
GPT-5.6 Luna	110	79	31	0,718
gpt-oss-120b	94	72	22	0,766
gpt-oss-20b	89	55	34	0,618
Qwen3.5 9B	112	46	66	0,411

Rozbieżność powtórzeń wskazuje część niepewnych przypadków, ale większość pomyłek u sześciu modeli jest jednomyślna: kolejne powtórzenia ich nie wychwycą.

Błędne werdykty na 360 ostrzeżeniach z Jetty, werdykt większościowy. Tabela 7.1 w pracy.