POWERED BY Bitdefender

PCLP - BITDEFENDER CHALLENGE

PRIMUL TASK @BALIZA

Deadline hard: **30.01.2022 20:00**

CUPRINS

Changelog:

- **16.01.2022**: Precizare Task 1 - Cuvintele din lista **keywords** se consideră **case insensitive** și vor fi numărate doar aparițiile ca **substring** din **body**.

1 ENUNȚ

Date: Fri, 14 Jan 2022, 18:23:23 +0200
Subject: Primul task
From: baliza@pclp.ro

Body: Hello!

Aceasta este prima ta zi la internship-ul din cadrul startup-ului Baliza. Sper că totul să se desfășoare așa cum iți dorești și să te simți bine la noi!

A venit timpul și pentru primul tău task, pe care știu că ești nerăbdător să îl primești. Este o experiență nouă, dar sunt sigur că te vei descurca, pentru că ai învățat multe la materia PCLP în facultate și te-ai descurcat foarte bine la interviu.

Primul tău task este să implementezi un **spam detector**. Pentru acest lucru, ți-am atașat o arhivă cu email-uri, unele **spam**, altele nu. De asemenea, am atașat un fișier **keywords**, ce conține o listă de cuvinte frecvent întâlnite în emailurile spam. Proiectul nostru este în **C** sau **Python 3**, limbaje pe care le-ai folosit până acum și la facultate.

Fiecare email are o structură fixă:

```
Date: <data la care a fost trimis emailul>
Subject: <subiectul emailului>
From: <expeditorul emailului>

Body: <textul emailului>
```

Notă: La finalul fiecărui email se află o linie goală.

——— **Task 1** ———

Mereu când avem de-a face cu date, primul pas este să ne familiarizăm cu acele date. De aceea, pentru început va trebui să contorizezi pentru fiecare cuvânt din lista de **keywords** de câte ori apare ca **substring** în total în **body-urile** emailurilor. De exemplu, pentru keyword-ul **click** și cuvântul **click**ing va fi numărat, deoarece conține **click**. Vom considera cuvintele ca fiind **case insensitive** (de exemplu, și cuvintele **Free** sau **FREE** vor fi contorizate ca fiind **free**). De asemenea, trebuie să calculezi și deviația standard a numărului de apariții în fiecare email pentru fiecare keyword, deoarece poate oferi o bună înțelegere asupra setului de date.

Formula pentru deviația standard este:

$$stdev = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_i - m_{aritmetica})^2},$$

unde v este un vector ce conține numărul de apariții ale unui keyword în lista de mail-uri din folder (v[i] este numărul de apariții ale lui keyword ca subșir în body-ul emailului i).

Îți las și un exemplu de cum se calculează deviația standard:

Fie vectorul v = [4, 7, 2, 4, 1]. Prima dată calculăm media aritmetică: **m = 3.6**.

$$stdev = \sqrt{\frac{(4-3.6)^2 + (7-3.6)^2 + (2-3.6)^2 + (4-3.6)^2 + (1-3.6)^2}{5}} \approx 2.06$$

Deviația standard măsoară cât de dispersat este un set de valori. Astfel, o valoare mare reprezintă faptul că valorile sunt depărtate de medie, iar o valoare mică reprezintă faptul că valorile sunt apropiate de medie. Acest lucru poate fi util în taskul acesta deoarece, daca pentru un keyword obținem o deviație standard mare, ar putea însemna că emailurile spam conțin de multe ori acel cuvânt, iar celelalte nu.

De exemplu stdev([1, 2, 3]) = 0.81, iar stdev([1, 10, 40]) = 16.67

Date de intrare

Programul va fi rulat fără argumente. Toate datele de care ai nevoie se află în directorul **data/**. Email-urile o să le găsești în directorul **data/emails/** și sunt numerotate de la **0** la **n - 1**, unde **n** este numărul de emailuri (fiecare email este în câte un fișier). Spam detectorul va fi rulat pe mai multe seturi de date, deci **n** trebuie să fie variabil, egal cu numărul de fișiere din directorul **data/emails** (hint: **ls -al** && man readdir; exemplu: Stackoverflow: Counting the number of files in a directory using C).

Fișierul **keywords** conține lista de **keywords**. Pe prima linie a fișierului se află numărul **C** de cuvinte din fișier, iar pe următoarele **C** linii se află cuvintele, câte un cuvânt pe fiecare linie.

Date de ieșire

Fiecare linie a fișierului de output **statistics.out** va conține o linie de forma:

```
word count stdev
...
```

cu semnificația că *word* apare de *count* ori în lista de email-uri și are deviația standard *stdev*, care va fi afișată cu 6 zecimale.

——— Task 2 ———

Acum că ai reușit să obții primele informații relevante pentru a începe munca de detecție, ți-am lăsat câteva idei cu care să începi lucrul la detectorul de emailuri spam. Pe baza a mai multor **criterii**, pentru fiecare email vei acorda un **scor**. Daca scorul obținut de un anumit email este mai mare decat un anumit **prag**, înseamnă că acesta este un email **spam**.

Criterii scor email

Keywords score

Astfel, definim funcția:

$$keywordsScore(mail, keywords) = \sum count(mail, keyword) \cdot \frac{avgSize}{len(mail.body)}$$

- $count(mail, keyword)$ reprezintă numărul de apariții ale lui **keyword** ca substring al body-ul emailul **mail**. De exemplu, pentru keyword-ul **click** și cuvântul **click**ing va fi numărat, deoarece conține **click**.
- **avgSize** reprezintă lungimea medie a body-urilor tuturor mailurilor
- $len(mail.body)$ reprezintă lungimea body-ului emailului **mail**

Uppercase

În general, când cineva trimite un email, nu folosește prea multe cuvinte scrise cu litere mari, însă în emailurile spam, acestea sunt destul de frecvente. Această informație este utilă pentru detectorul de spam.

Definim următoarea funcție:

$$hasCaps(mail) = (capsCount(mail.body) > 1/2 * len(mail.body))\ ?\ 1 : 0$$

Această funcție întoarce 1 dacă mai mult de jumătare din caracterele din body-ul emailului sunt litere mari și 0 altfel.

Spammers

Am atașat și o listă de adrese de email deja cunoscute de către comunitate ca fiind potențiale spammere. Fișierul **spammers** conține pe prima linie numărul **e** de adrese de email din fișier, apoi pe următoarele **e** linii sunt de forma:

```
<adresa de email> <scorul asociat>
```

Definim funcția:

$spammers(mail)$ = scorul din fișierul **spammers** pentru adresa de email (**mail.from**). Funcția are valoarea 0 dacă adresa respectiva de email nu se află în fișier.

Detecție

Scorul final al unui email va fi calculat pe baza scorurilor de până acum. Cum funcțiile definite anterior au ordine de mărime diferite, trebuie să facem o sumă ponderată:

$$score(mail) = 10 * keywordsScore(mail, kwords) + 30 * hasCaps(mail) + spammers(mail)$$

Dacă $score(mail) > prag$, atunci considerăm mailul ca fiind spam.
Pentru început să considerăm **prag = 35**.

Evaluare detecție

Modalitatea de evaluare a performanței modelului pe care o vom folosi este scorul F1. Pentru a îți putea verifica usor detectorul, ți-am pus la dispoziție și un checker, care va calcula acest scor pentru tine. Chiar dacă nu este nevoie să implementezi tu acest scor, îți recomand să te documentezi puțin despre el. Pe scurt, cu cât detectorul prezice corect (prezice ca fiind spam emailurile spam și ca nefiind spam cele care nu sunt spam) mai multe emailuri, cu atât scorul o sa fie mai mare.

Date de intrare

Acest task e o continuare a task-ului precedent, prin urmare vei avea la dispoziție atât lista de mail-uri, cât și fișierul de **keywords**. Pe lângă acestea, mai există și un fișier **data/spammers**, care conține o listă de adrese de email ce sunt deja considerate ca fiind potențiale spammere.

Date de ieșire

Fiecare linie a fișerului de output **prediction.out** va conține pe fiecare linie o singură valoare binară (1 sau 0). Răpunsul de pe linia **i** reprezintă clasificarea emailului și are valoarea 1 dacă acesta va fi considerat ca fiind spam și 0 în caz contrar.

Îmbunătățiri

De acum lucrurile devin și mai interesante. Noi aici la startup-ul Baliza ne dorim ca fiecare dintre noi să își folosească imaginația, nu doar să implementeze niște taskuri deja definite. De aceea, din acest punct, sarcina ta este să te gândești cum să faci detectorul cât mai bun.

Criteriile menționate la taskul 2 asigură un scor F1 = 0.75, însă sunt sigur că se poate obține mai mult de atât, deoarece criteriile mele nu folosesc toate informațiile oferite de emailuri. Te poți uita prin emailurile pe care ți le-am trimis. Poate observi detalii interesante ce te pot ajuta.

Poți încerca orice idee ai, cu scopul de a obține un scor F1 cât mai mare. Poți adăuga criterii noi și chiar să modifici criteriile sugerate de mine, inclusiv formula de detecție, dacă acest lucru duce la un scor mai mare. Poti de asemenea să adaugi si noi keywords, într-un fișier **additional_keywords** (pe care să îl trimiți împreună cu spam detectorul). Totuși, nu poți modifica fișierul **spammers**.

Deoarece ne dorim ca detectorul de spam să funcționeze bine pe orice email, nu doar pe cele pe care le-am atașat, după ce totul este gata, vom testa și pe un alt set de emailuri. Performanța pe acel set va reprezenta performanța reală a detectorului.
Împreună cu detectorul pe care o să îl implementezi, as vrea să îmi trimiți și un fișier în care descrii toate îmbunătățirile pe care le-ai adus față de cele pe care le-am sugerat eu. Sunt foarte curios să aflu ce idei o să ai.

Acesta este primul tău task. Mult succes!

2 PUNCTARE

• Punctajul temei pe checker este de 100 puncte indiferent de limbaj, distribuit astfel:

- Task1 1: 50p
  - Calculare număr de apariții keyword: 25p
  - Calculare deviație standard: 25p
- Task 2: 50p vor fi date proportional cu scorul F1 obținut după formula

$$min(max(2 * (F1 - 0.5), 0), 0.5) * 100$$

  astfel:
  - F1 $\leqslant$ 0.5 va fi punctat cu 0p
  - F1 $\geqslant$ 0.75 va fi punctat cu 50p
  - Un scor în (0.5, 0.75] obține un punctaj proporțional (de exemplu, pentru F1 = 0.6 vor fi acordate 20/50p).

Programul vostru va trebui să rezolve la o singură rulare atât task-ul 1, cât și task-ul 2. Cu alte cuvinte, executabilul generat va rezolva mai întâi răspunsul pentru task01 și apoi răspunsul pentru task02.
Implementarea corectă a criteriilor de la task02 va asigura un scor F1 $\geqslant$ 0.75 și implicit punctaj maxim pe acest task.

• Această temă este bonus, punctajul nefiind inclus în nota de pe parcurs. Cu toate acestea, tema valorează 0.5p bonus pentru implementarea în C **și** 0.2p bonus pentru implementarea în Python, care se adaugă la nota PCLP din regulamentele seriilor CA, respectiv CB+CD, după obținerea punctajului minim pe parcurs și promovarea examenului (acest punctaj nu poate fi folosit pentru a promova materia, însă poate ajuta la obținerea unei note finale mai mari).

3 COMPETIȚIE

Bitdefender va acorda **12 premii** pentru cele mai bune teme din punct de vedere al scorului F1 pe **testele private**, astfel:

- cele mai bune **3 teme în C** de la seria **CA**
- cele mai bune **3 teme în C** de la seria **CB**
- cele mai bune **3 teme în C** de la seria **CD**
- cele mai bune **3 teme în Python** la comun **CA + CB + CD**

Observație: un student poate primi un singur premiu.

Clasamentul se va face în funcție de scorul F1 obținut de către spam detector pe **testele private** de pe **vmchecker**. Link-ul către leaderboard va fi postat pe Teams. Vor fi eligibile pentru premii **DOAR** temele care au punctaj maxim pe vmchecker.

În caz de egalitate, evaluarea se va face manual/subiectiv pe baza mai mulor criterii, printre care:

- explicații README
- calitatea codului
- prezența / calitatea log-urilor
- eficiență
- inventivitate
- implementarea temei în ambele limbaje de programare
- etc

4 PRECIZĂRI

Orice încercare de rezolvare frauduloasă a temei va fi notată cu 0p și nu va putea fi considerată pentru stabilirea câștigătorilor. Metode frauduloase presupun (dar nu sunt limitate la): hardcodare (stabilirea manuală pentru fiecare mail dacă este spam sau nu), modificarea fișierelor de referință, rularea de alte procese care au scop rezolvarea temei, modificarea checker-ului, atacul asupra infrastructurii pentru a prelua codul altor colegi sau testele private, copierea din orice sursă.

Pentru detalii puteți să vă uitați și peste regulile generale de trimitere a temelor (regulile generale CA și reguli generale CB-CD).

- O temă care **NU** compilează va fi punctată cu 0. O temă care **NU** trece niciun test pe vmchecker va fi punctată cu 0.
- Vor exista atât teste publice, cât și teste private pentru fiecare problemă în parte. Punctajul pe temă și scorul final **se calculează exclusiv pe baza rulării programului pe testele private**.
- Pe lângă codul sursă, este **obligatorie** trimiterea unui fișier README, în care să descrieți soluția voastră și să descrieți eventualele euristici implementate, diferite de cele menționate în enunț.

Checker

- Pentru testarea locala, aveți disponibil un set de teste publice (de o dificultate similară) în arhiva temei.

```
.
├── check                       # [IMPORTANT] ./check - ruleaza checkerul
├── check_utils                 # [ignore me] director cu scripturi interne ale checkerului
├── cs                          # [ignore me] director cu scripturi interne ale checkerului
├── data                        # setul de date
│   ├── emails                  # * director cu emailuri
│   ├── keywords                # * fisierul cu cuvintele cheie
│   └── spammers                # * fisierul cu adrese potentiale spam
├── install.sh                  # [IMPORTANT] ./install.sh - instaleaza checkerul
├── README.md                   # [IMPORTANT] instructiuni utilizare checker
└── tasks
    └── spam_detector
        └── tests/
            └── 00-spam_detector/   # [IMPORTANT] testele pentru task01
            └── 01-spam_detector/   # [IMPORTANT] testele pentru task02
```

* Rulați comenzile **./install.sh** și **./check** pentru a instala / rula checkerul. Mai multe detalii în README.md.

5 FORMAT ARHIVĂ SUBMISĂ

- Arhiva cu rezolvarea temei trebuie să fie **.zip**, având un nume de forma **Grupa_NumePrenume_SpamDetector.zip** și va conține:
  - Fișierul/fișierele sursă. Sursa pentru C în care se află funcția main() trebuie denumită obligatoriu **spam_detector.c**, iar pentru Python 3 trebuie denumită obligatoriu **spam_detector.py**.
  - Fișierul **Makefile**
  - Fișierul **README** (fără extensie)
  - Fișierul **additional_keywords**, dacă este cazul

- Fișierul pentru make trebuie denumit obligatoriu **Makefile** și trebuie să conțină următoarele reguli:
  - **build**, care va compila sursele C și va obține executabilul numit **spam_detector** pentru C. Dacă ați implementat doar în Python 3 această regulă nu face ceva, dar prezența sa este obligatorie!
  - **clean**, care șterge executabilele și fișierele obiect generate.

- Toate fișierele menționate trebuie să fie în **rădăcina** arhivei.
- **ATENȚIE!** Tema va fi compilată și testată **DOAR într-un mediu Linux**.
- **ATENȚIE!** Numele regulilor și a surselor trebuie să fie exact cele de mai sus. Absența sau denumirea diferită a acestora va avea drept consecință obținerea a 0 puncte pe temă.
- **ATENȚIE! NU** este permisă compilarea cu opțiuni de optimizare a codului (O1, O2, etc.).
- **ATENȚIE!** Orice nerespectare a restricțiilor duce la pierderea punctajului (după regulile de mai sus).