Documentație ML-Email-Filter pe baza Algoritmul Naive Bayes

1 Introducere

Acest document oferă o documentație pentru proiectul meu realizat în limbajul de programare Python care implementează algoritmul Naive Bayes pentru a sorta emailurile în spam și non-spam, utilizând setul de date Ling-Spam.

2 Setul de Date Ling-Spam

Setul de date Ling-Spam conține mesaje email împărțite în 9 foldere pentru antrenare și una pentru testare (part10). Fiecare email are eticheta spam sau non-spam, iar aceasta poate fi dedusă din titlul fișierului, unde prefixul "spm" indică spam-ul.

Date statistice inițiale
Numarul emailurilor din setul de antrenare: 2602
Dintre care: non-spam 2170 și spam 432

Numarul de emailuri din setul de testare: 290
DIntre care: non-spam 242 și spam 48

Reieșind din aceste date putem simplu calcula ca probabilitatea ca un email sa fie spam este de 0.1660261337 respectiv probabilitatea ca un email sa fie non-spam este de 0.8339738663

3 Procesare Date

3.1 Scriptul run.py

Pentru a asigura corectitudinea și funcționalitatea scripturilor, am utilizat un script principal care rulează fiecare script în parte. Mai jos este atașat conținutul scriptului principal realizat în Python:

```
# Script 1: separate_subject_content
subprocess.run(["python", "separator.py"])
```

```
# Script 2: sort_emails
subprocess.run(["python", "sortareEmail.py"])

# Script 3: extract_subject_and_content
subprocess.run(["python", "SeparareSubiectDeContinut.py"])

# Script 4: calculate_probabilities
subprocess.run(["python", "CalculProbabilitati.py"])

# Script 5: classify_emails
subprocess.run(["python", "ClasificareaBayesiana.py"])
```

Acest script rulează fiecare script individual, asigurându-se că toate modulele funcționează corespunzător și că datele sunt corect procesate.

Scriptul `separator.py`

Utilizat asupra emailurilor din setul de testare

- Acest script are rolul de a separa subiectul și conținutul din textul unui email, utilizând expresii regulate.
- Se procesează recursiv folderele și fișierele text dintr-un director dat, apoi se salvează subiectul și conținutul în fișiere separate.

Scriptul `sortareEmail.py`

Utilizat asupra emailurilor din setul de antrenare

- Scopul acestui script este să sorteze emailurile în foldere distincte pentru spam și non-spam.
- Se definește o funcție care parcurge recursiv fișierele dintr-un director sursă și copiază fișierele în foldere destinație separate pentru spam și non-spam, în funcție de prezența cuvântului "spm" în numele fișierului.

Scriptul `SeparareSubiectDeContinut.py`

Utilizat asupra emailurilor din setul de antrenare

- Acest script extrage subiectul și conținutul emailurilor.
- Se definește o funcție care caută subiectul și conținutul într-un text de email și le salvează separat în foldere corespunzătoare.

Scriptul `CalculProbabilitati.py`

Utilizat asupra emailurilor din setul de antrenare

- Scopul acestui script este să calculeze probabilitățile de apariție ale cuvintelor în emailuri, pentru diferite categorii (spam, non-spam) și pentru diferite părți ale emailului (subiect, conținut).
- Sunt definite patru funcții, fiecare pentru calcularea probabilităților într-o categorie specifică și pentru o anumită parte a emailului.

Scriptul `ClasificareaBayesiana.py`

Utilizat asupra emailurilor din setul de testare

- Acest script efectuează clasificarea bayesiană a emailurilor.
- Se citesc informațiile precalculate despre probabilitățile cuvintelor în funcție de categoriile spam și non-spam pentru subiect, conținut și întregul email.
- Pentru fiecare email din setul de testare, scriptul determină probabilitatea de a fi spam sau non-spam, utilizând informațiile precalculate și regula lui Bayes.
- Rezultatele sunt salvate în fișiere separate pentru spam și non-spam în funcție de conținut, subiect și întregul email.

4 Algoritm Naive Bayes

Am ales să implementăm algoritmul Naive Bayes pentru a rezolva problema de clasificare a email-urilor spam. Alegerea se bazează pe teoria și experiența anterioară, având în vedere natura sa eficientă în gestionarea datelor textuale.

Pentru a atinge un rezultat mai bun în procesul de sortare a emailurilor din setul de date de testare, am ales să realizez aplicarea algoritmului Bayes separat pentru subiectul emailurilor, conținutul emailurilor, și întregul email (adică atât subiectul cât și conținutul).

La fel ca factor important în creșterea preciziei sortării am folosit niște variabile care îndeplineau funția regulii lui LaPlace cu valori deduse experimental prin rularea repetată a scriptului de sortare și compararea rezultatelor, astfel am ajuns la urmatoarele valori pentru fiecare dintre sortari:

După întregul email: 0.0001
După conținut: 0.0001
După subiect: 0.0004

5 Performanța pe Setul de Date de Testare

În urma rulării algoritmului Bayes Naiv pentru sortarile după conținut, subiect și în întregime am obținut următoarele rezultate:

Clasificare dupa Subiect:
dintre 242 de emailuri non-spam, 234 au fost clasificate corect, 8 greșit
dintre 48 de emailuri spam, 39 au fost clasificate corect, 9 greșit

Clasificare după Conținut:
dintre 242 de emailuri non-spam, 242 au fost clasificate corect, 0 greșit
dintre 48 de emailuri spam, 16 au fost clasificate corect, 32 greșit

Clasificare după întregul Email:
dintre 242 de emailuri non-spam, 242 au fost clasificate corect, 0 greșit
dintre 48 de emailuri spam, 15 au fost clasificate corect, 33 greșit

6 Concuzii

Putem observa că cea mai buna performanță la clasificarea emailurilor ca spam sau non-spam o obținem atunci când încercăm sa clasificam emailurile conform subiectului. În cazul meu **am obținut acuratețe de 81,25% în cazul emailurilor spam și acuratețe de 96,69% în cazul emailurilor non-spam.**