Raport Tehnic

Introducere în domeniu

În general, documentele scanate, inclusiv text, desene și regiuni grafice pot fi considerate documente de tip mixt. În multe aplicații practice, avem nevoie să recunoaștem sau să îmbunătățim în principal conținutul text al documentelor. În astfel de cazuri, este de preferat ca documentele să fie convertite într-o formă binară. Astfel binarizarea documentelor este primul pas în majoritatea sistemelor de analiză a documentelor. Scopul binarizării documentului este de a converti inputul dat, fie el document în tonuri de gri sau color, într-o reprezentare pe două niveluri. Această reprezentare este convenabilă deoarece majoritatea documentelor care apar în practică au o singură culoare pentru text (de exemplu, negru) și o culoare diferită (de exemplu, alb) pentru fundal.

Thresholding-ul devine un instrument simplu, dar eficient de a separa obiectele de fundal. Rezultatul operației de thresholding este o imagine binară a cărei stare va indica obiectele din prim-plan, adică textul tipărit, o legendă etc., în timp ce starea complementară va corespunde fundalului. În funcție de aplicație, prim-planul poate fi reprezentat de nivelul de gri 0, adică negru pentru text, și 1 pentru fundalul hârtiei, adică alb.

Binarizarea imaginii este procesul de separare a valorilor pixelilor în două grupuri, alb ca fundal și negru ca prim plan. Thresholding-ul joacă un rol important în binarizarea imaginilor. Thresholding-ul poate fi clasificat în Thresholding global și Thresholding local.

În imagini cu distribuție uniformă a contrastului de fundal și prim-plan precum imaginile cu documente, Thresholding-ul global este mai potrivit.

Metode de binarizare globală, cum ar fi cel propus de Otsu încearcă să găsească o singură valoare de prag pentru întregul document. Apoi fiecare pixel este atribuit
fie în prim-plan-ul imaginii, fie în fundalul acesteia pe baza valorii sale de gri. Metodele de binarizare globală sunt foarte rapide și dau rezultate bune pentru documentele scanate tipice. Pentru mulți ani, binarizarea documentelor în tonuri de gri s-a bazat pe algoritmii de prag global.

În imaginile degradate, unde există zgomot de fundal considerabil sau există variație de contrast și iluminare, sunt mulți pixeli care nu pot fi ușor clasificați ca pixeli de prim plan sau pixeli de fundal. În astfel de cazuri, binarizarea cu prag local este mai potrivită. Aceste tehnici estimează un prag diferit pentru fiecare pixel conform informațiilor în tonuri de gri ale pixelilor vecini. Tehnicile lui Bernsen, Chow și Kaneko, Eikvil, Mardia și Hainsworth, Niblack, Taxt, Yanowitz și Bruckstein și Sauvola și Pietikainen aparțin acestei categorii. Tehnici hibride: O'Gorman și Liu și Li, care combină informații despre pragurile globale și locale aparțin altei categorii.

Arhitectură

Pentru pregătirea mediului de dezvoltare al proiectului software creat cu ajutorul limbajului de programare Python, ne-am folosit de IDE-ul Visual Studio Code. IDE-ul a fost integrat cu GitHub, cu ajutorul extensiei Codespaces, facilitând preluarea și updatarea ultimei versiuni a codului. De asemenea, am utilizat biblioteci ajutătoare precum json (folosim funcții

implementate pentru importul și exportul de date în și din fișiere json), time (măsurarea timpului de execuție), csv (parsarea fișierelor de input de tip csv), os (preluarea tuturor fișierelor de input dintr-un folder), dar și itertools (funcția combinations() folosită pentru generarea tuturor combinațiilor de operatori: “+”, ”-”, “*”, “/”).

Specificațiile sistemului

- hardware: CPU: Intel i5-8250U, 1.60GHz base, overclock 1.80GHz, RAM 8GB, 4 Cores.
- software: OS: Windows 10, x86-64

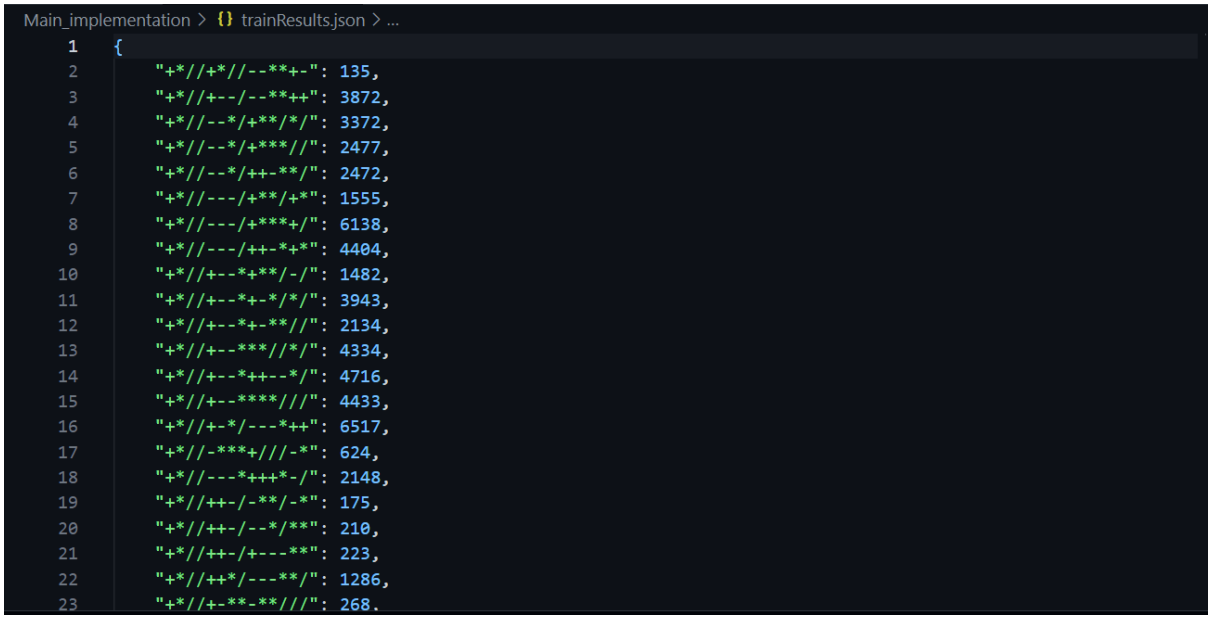
Descrierea soluției și rezultate intermediare

Etapa Globală

Pentru prima parte a proiectului, și anume binarizarea globală, am ales să luăm input-urile ca termeni ai unei funcții. Astfel, am ales o metodă de rezolvare de tip greedy: în primul rând am generat toate combinațiile posibile de 14 operatori, rezultând aproximativ 430.000 de combinații, dintre care și duplicate. După ce am eliminat tot ce era în plus și am pus condiția ca numărul de “+” să fie egal cu numărul de “-” și numărul de “*” să fie egal cu numărul de “\”, pentru a evita cat mai mult posibil apariția unor valori mai mari decat 1 și mai mici decat 0, valori ce nu ne-ar fi de folos. Combinațiile rămase le-am salvat apoi într-o listă pentru a le putea folosi mai ușor.

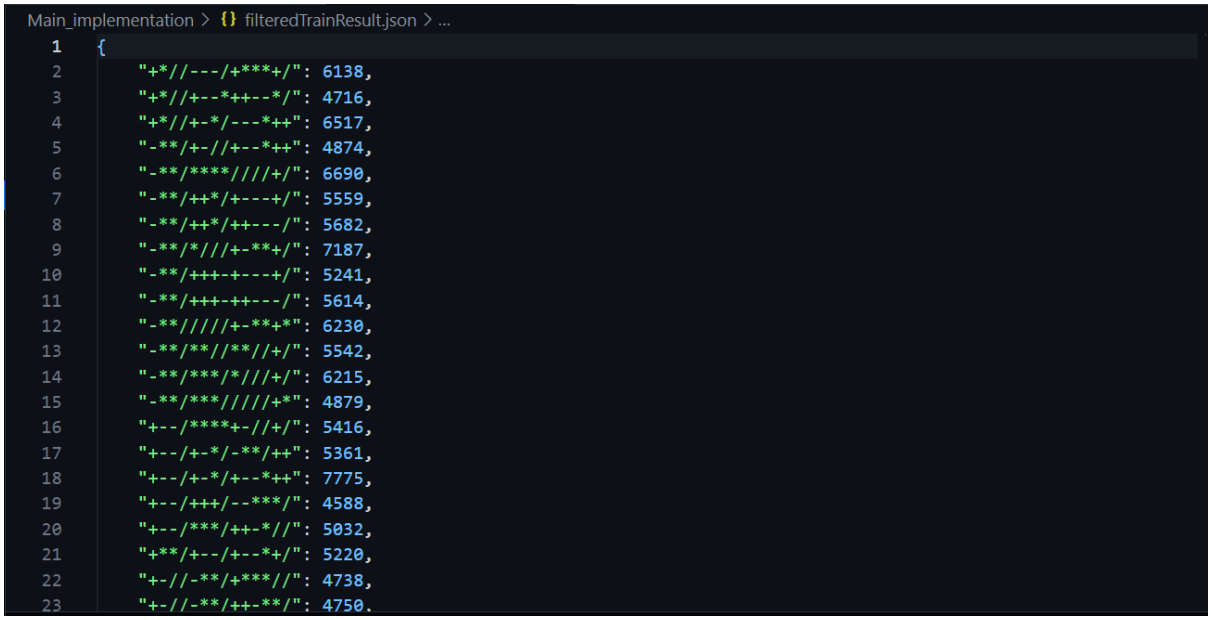
Totodată, am folosit un dicționar în care am ținut evidența numărului de fișiere pentru care fiecare combinație a avut o valoare F-measure de cel puțin 0.8 (80%). Astfel, am luat fiecare fișier de input și am calculat rezultatul pentru fiecare combinație de operatori și, dacă acesta se apropie de Tresholding-ul ideal, incrementăm frecvența corespunzătoare acesteia din dicționar.

Am împărțit fișierele de input în 3 categorii: 70% din ele pentru partea de train, 25% pentru validation și ultimele 5% pentru partea de testing. Astfel, am rulat procesul descris mai sus pentru partea de training, iar dicționarul rezultat l-am exportat într-un fișier .json.

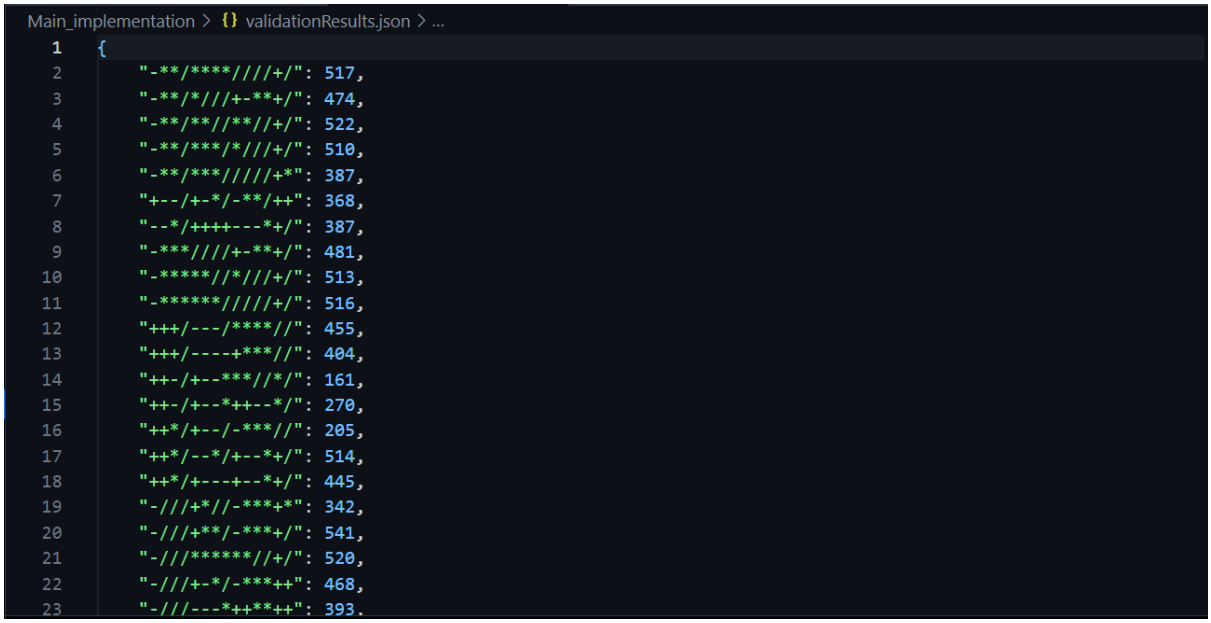


Pentru a trece la următoarea etapă, am ales doar combinațiile care erau cel puțin egale cu 75% din maximul obținut. Același procedeu l-am aplicat și pentru următoarele două etape, însă scopul a fost ca după fiecare etapă să eliminăm cât mai multe valori și să rămânem doar cu cele mai relevante. Astfel, am obtinut următoarele rezultate parțiale:

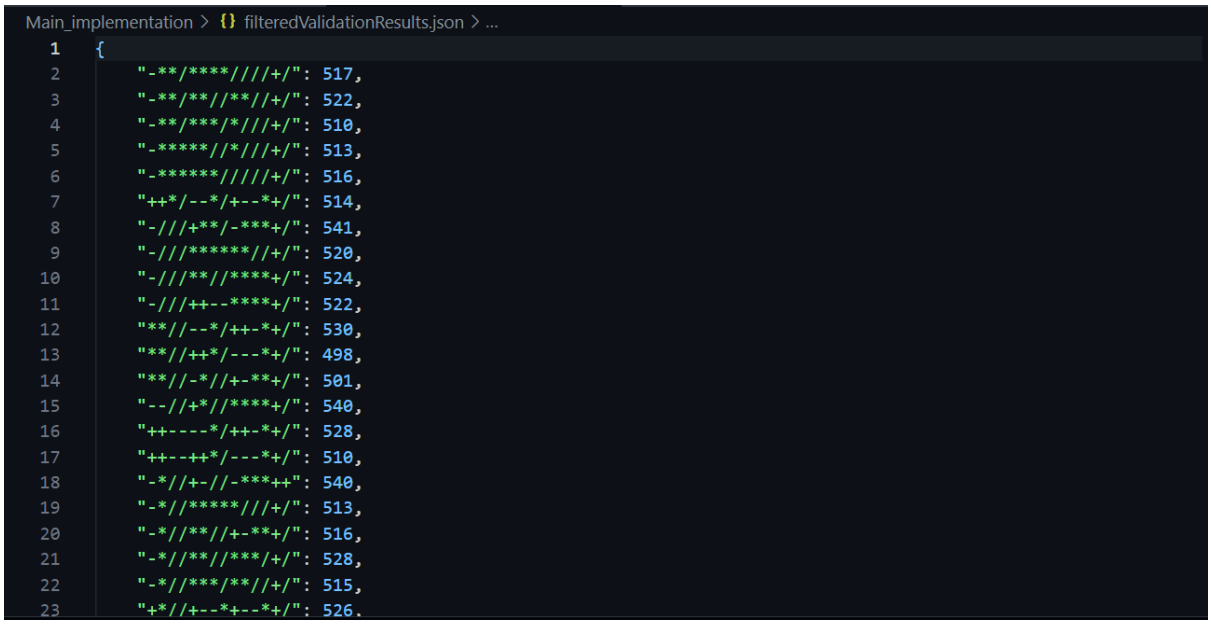
Rezultatele înainte de etapa validation:



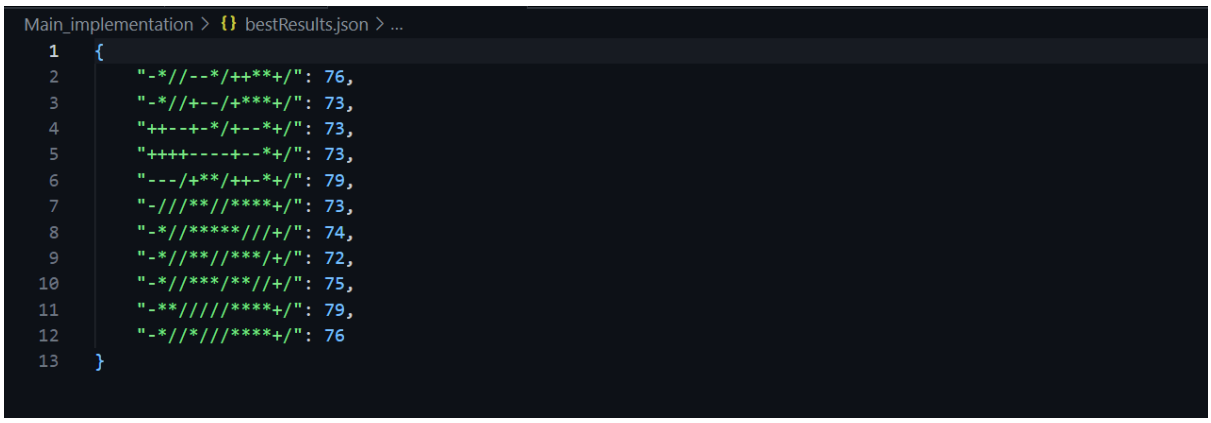
Rezultatele după etapa validation:



Rezultatele înainte de etapa test:



Rezultatele finale ale etapei globale



Astfel, am reușit să ajungem la 11 posibile rezultate finale.

Etapa Locală

Pentru partea de binarizare locală am reușit să punem baza implementării, ideea fiind bazată tot pe o strategie de tip greedy, similară cu cea de la etapa precedentă.Deși au fost mai puțini operatori, principala problemă pe care am întâmpinat-o a fost timpul de execuție al programului, îngreunat de numărul imens de linii din fișierele de input. Modul de abordare este unul similar, diferența fiind faptul că în dicționar nu mai folosim frecvența unei combinații într-un set de teste, ci procentajul de pixeli pentru care rezultatul fiecărei combinații corespunde celui din input. Am început implementarea pentru partea de train, însă abordarea noastră lasă loc de îmbunătățiri pentru următorul milestone.