OCR-izare în Obsidian

Acest document exporează oportunitățile de OCR-izare pe care diferite pluginuri ale lui Obsidian le pun la îndemâna celor interesați.

1. Latex OCR

Acesta este un plugin care se folosește de infrastructura Python3 care trebuie construită la nivelul sistemului. Mai multe informații despre plugin aici: GitHub - lucasvanmol/latex-ocr-server. Tare bine prinde o placă video GForce pentru că vei avea suport de CUDA. Primul lucru care trebuie făcut este să instalezi software-ul necesar creării unui server local care rulează de pe mașina proprie. Pluginul mai permite apeluri API către un serviciu extern specializat, dar acest mod de operare este limitat la ceea ce îți oferă respectivul serviciu. Pentru scopul demonstrativ al acestui ghid de OCRizare cu Latex OCR în Obsidian, am ales rularea serverului local.
Vom porni prin instalarea unui mediu virtual dedicat rulării aplicațiilor Python care se numește miniconda.

2. Construcție infrastructură Python

Detalii de instalare a componentelor necesare în Linux/GNU distribuția Ubuntu 23.10. Primul lucru, dacă nu ai deja instalat miniconda (Miniconda — miniconda documentation), e timpul să o faci:

```
curl -O
https://repo.anaconda.com/miniconda/Miniconda3-
latest-Linux-x86_64.sh
sh Miniconda3-latest-Linux-x86_64.sh
```

Dacă nu ai instalat pachetul de sistem pentru pip, e timpul să-l instalezi:

```
sudo apt install python3-pip
```

Detaliile de instalare ale pachetelor pentru:

```
pip install https://github.com/lucasvanmol/latex-
ocr-
server/releases/download/0.1.0/latex_ocr_server-
0.1.0-py3-none-any.whl
```

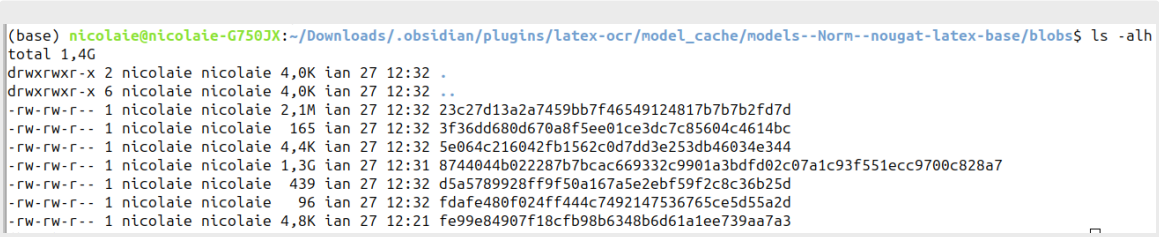
Se vor instala rând pe rând un set de pachete. Vezi Anexa 1. Pentru verificări se pot urma instrucțiunile de aici: Start Locally | PyTorch.

În acest moment, ai la dispoziție un server dedicat pentru a realiza OCR-izarea formulelor matematice cu scopul de a le transforma în fragmente de cod LaTeX. LaTeX-ul este formatul de

editare cel mai adesea folosit pentru elaborarea lucrărilor din domeniul științelor exacte. Există și alte tehnologii folosite pentru a redacta formule matematice, precum MathJAX, dar pentru acest plugin, transformarea este într-un fragment LaTeX.

3. Operarea OCR-ului

În momentul în care ai infrastructura Python deja disponibilă, dacă aplicația nu a fost închisă, este timpul să o repornești. În momentul în care repornește, se inițializează serverul care gestionează întreaga operațiune de OCR-izare.

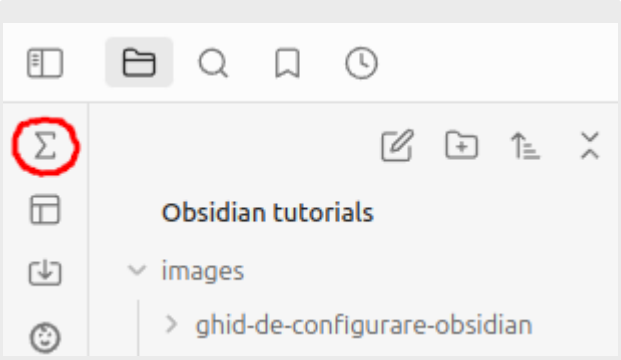


Screenshot from 2024-01-27 12-32-44

La prima pornire, se va descărca modelul necesar pentru a se face interpretarea imaginilor cu ajutorul algoritmilor de machine learning. Pe o mașină Linux/GNU, distribuția Ubuntu 23.10, acest model a fost descărcat în Downloads, unde a fost creat automat un subdirector `.obsidian`. În imaginea de mai sus sunt câteva detalii care vă permit să înțelegeți mai bine ceea ce s-a petrecut. Trebuie menționat că un computer cu resurse limitate de spațiu și de procesare va oferi o experiență de lucru vătămătoare nervilor operatorului. Cel puțin 8 nuclee de procesare, 16 Gb RAM și cel puțin 1T SSD-ul, ok?

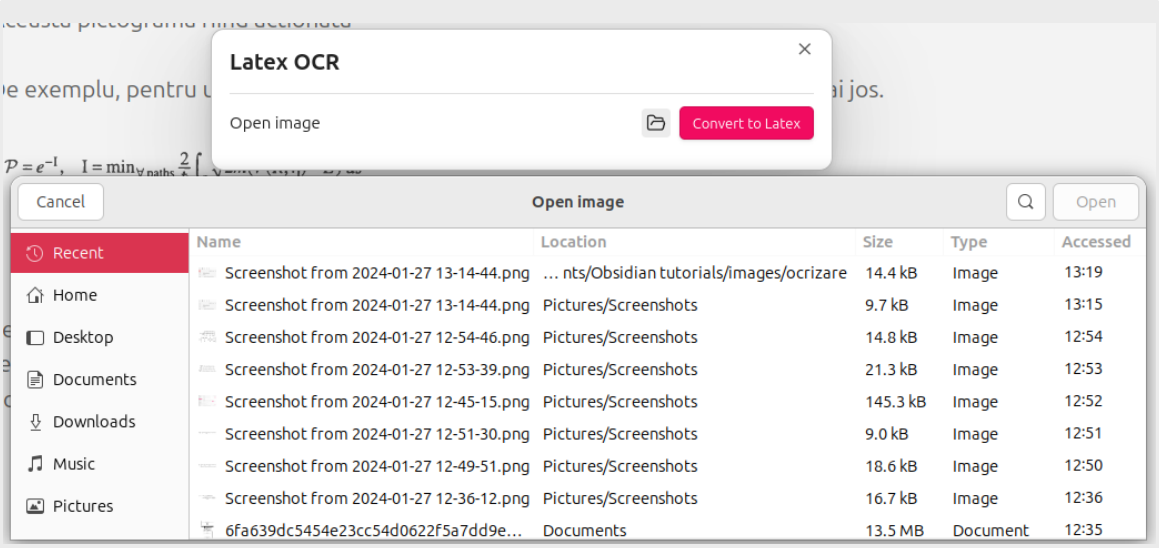
Pentru exemplificare, am făcut câteva capturi de ecran a unor formule matematice dintr-o revistă de cercetare științifică - Romanian Journal of Physics, care poate fi consultată de la Romanian Journal of Physics, vol. 47, Nos. 1-2 · Colecții BNF · Colectii BNF.

Pentru operare, va apărea o pictogramă în bara tip ribbon din stânga.



Screenshot from 2024-01-27 13-14-44

Această pictogramă fiind acționată va apărea un panou simplu cu două butoane.



Screenshot from 2024-01-27 13-21-01

Acționând pictograma folderului, vei naviga structura directoarelor și subdirectoarelor până unde ai făcut captura de imagine a formulei pentru care dorești să faci OCR-izarea. Acționarea butonului *Convert to Latex* va face această conversie care va pune rezultatul în memoria computerului. Un simplu *Paste* în editorul Obsidian sau dacă dorești în alt editor, va copia codul LaTeX și va avea drept rezultat instantaneu randarea (afișarea) rezultatului interpretării acelei secvențe de cod.

De exemplu, pentru următorul fragment de imagine capturat, avem rezultatul mai jos.

$$\mathcal{P} = e^{-\mathrm{I}}, \quad \mathrm{I} = \min_{\forall\,\text{paths}} \frac{2}{\hbar} \int_S \sqrt{2m(V(R,\eta) - E)}\, ds$$

$$= \min_{\forall\,\text{paths}} \frac{2}{\hbar} \int_0^1 \sqrt{\underbrace{2mg_{ij}}_{B_{ij}} (V(x_i(t) - E) \frac{dx_i}{dt} \frac{dx_j}{dt}}\, dt$$

Screenshot from 2024-01-27 12-36-12

Se observă faptul că în ciuda calității destul de reduse a imaginii, serverul a făcut o treabă excelentă în recunoașterea conținutului. Mai jos este fragmentul LaTeX interpretat și afișat corespunzător în prezenta notă de Obsidian.

$$\mathcal{P} = e^{-1}, \quad \mathrm{I} = min_{\forall\,\text{paths}} \frac{2}{\hbar} \int_S \sqrt{2m(V(R,\eta) - E)}\, ds$$

$$= min_{\forall\,\text{paths}} \frac{2}{\hbar} \int_0^1 \sqrt{\underbrace{2mg_{ij}}_{\widetilde{B}_{ij}} (V(x_i(t) - E) \frac{dx_i}{dt} \frac{dx_j}{dt}}\, dt$$

Ca în cazul utilizării oricărui software existent, lucrurile nu sunt perfecte. De exemplu, pentru fragmentul de cod de mai sus, a trebuit să-l *curăț* eliminând un adaos nedorit. Este vorba despre un fragment `operatorname*` care nu avea niciun rost. Totuși este un preț mic pe care îl plătești unui serviciu atât de util.

Un alt exemplu este imagine următoare

$$\frac{d\rho(t)}{dt} = -\frac{i}{\hbar}[H,\rho(t)] + \frac{1}{2\hbar}\sum_j ([V_j\rho(t),V_j^\dagger] + [V_j,\rho(t)V_j^\dagger]).$$

Screenshot from 2024-01-27 12-51-30

pentru care serverul a generat corect codul LaTeX.

$$\frac{d\rho(t)}{dt} = -\frac{i}{\hbar}[H,\rho(t)] + \frac{1}{2\hbar}\sum_j (\{V_j\rho(t),V_j^\dagger\} + [V_j,\rho(t)V_j^\dagger]).$$

Un alt exemplu ceva mai complex

$$\frac{d\sigma_{qq}(t)}{dt} = -2(\lambda-\mu)\sigma_{qq}(t) + \frac{2}{m}\sigma_{pq}(t) + 2D_{qq},$$
$$\frac{d\sigma_{pp}}{dt} = -2(\lambda+\mu)\sigma_{pp}(t) - 2m\omega^2\sigma_{pq}(t) + 2D_{pp},$$
$$\frac{d\sigma_{pq}(t)}{dt} = -m\omega^2\sigma_{qq}(t) + \frac{1}{m}\sigma_{pp}(t) - 2\lambda\sigma_{pq}(t) + 2D_{pq}.$$

Screenshot from 2024-01-27 12-53-39

pentru care a fost creat corect codul necesar

$$\frac{d\sigma_{qq}(t)}{dt} = -2(\lambda-\mu)\sigma_{qq}(t) + \frac{2}{m}\sigma_{pq}(t) + 2D_{qq},$$
$$\frac{d\sigma_{pp}}{dt} = -2(\lambda+\mu)\sigma_{pp}(t) - 2m\omega^2\sigma_{pq}(t) + 2D_{pp},$$
$$\frac{d\sigma_{pq}(t)}{dt} = -m\omega^2\sigma_{qq}(t) + \frac{1}{m}\sigma_{pp}(t) - 2\lambda\sigma_{pq}(t) + 2D_{pq}.$$

Și un ultim exemplu

$$T = \frac{1}{2i\Omega}\begin{pmatrix} \mu+i\Omega & \mu-i\Omega & 2\omega \\ \mu-i\Omega & \mu+i\Omega & 2\omega \\ -\omega & -\omega & -2\mu \end{pmatrix},$$
$$K = \begin{pmatrix} -2(\lambda-i\Omega) & 0 & 0 \\ 0 & -2(\lambda+i\Omega) & 0 \\ 0 & 0 & -2\lambda \end{pmatrix}$$

Screenshot from 2024-01-27 12-54-46

Pentru care a fost generat:

$$T = \frac{1}{2i\Omega}\begin{pmatrix} \mu + i\Omega & \mu - i\Omega & 2\omega \\ \mu - i\Omega & \mu + i\Omega & 2\omega \\ -\omega & -\omega & -2\mu \end{pmatrix},$$
$$K = \begin{pmatrix} -2(\lambda - i\Omega) & 0 & 0 \\ 0 & -2(\lambda + i\Omega) & 0 \\ 0 & 0 & -2\lambda \end{pmatrix}$$

Anexa 1

Installing collected packages: mpmath, uritemplate, typing-extensions, sympy, safetensors, regex, pyyaml, pyparsing, pyasn1, protobuf, pillow, nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-cu12, numpy, networkx, MarkupSafe, grpcio, fsspec, filelock, cachetools, triton, rsa, pyasn1-modules, nvidia-cusparse-cu12, nvidia-cudnn-cu12, jinja2, huggingface-hub, httplib2, googleapis-common-protos, tokenizers, nvidia-cusolver-cu12, google-auth, transformers, torch, google-auth-httplib2, google-api-core, google-api-python-client, latex-ocr-server

Successfully installed MarkupSafe-2.1.4 cachetools-5.3.2 filelock-3.13.1 fsspec-2023.12.2 google-api-core-2.15.0 google-api-python-client-2.107.0 google-auth-2.27.0 google-auth-httplib2-0.2.0 googleapis-common-protos-1.62.0 grpcio-1.59.3 httplib2-0.22.0 huggingface-hub-0.20.3 jinja2-3.1.3 latex-ocr-server-0.1.0 mpmath-1.3.0 networkx-3.2.1 numpy-1.26.3 nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-12.1.105 nvidia-cuda-runtime-cu12-12.1.105 nvidia-cudnn-cu12-8.9.2.26 nvidia-cufft-cu12-11.0.2.54 nvidia-curand-cu12-10.3.2.106 nvidia-cusolver-cu12-11.4.5.107 nvidia-cusparse-cu12-12.1.0.106 nvidia-nccl-cu12-2.18.1 nvidia-nvjitlink-cu12-12.3.101 nvidia-nvtx-cu12-12.1.105 pillow-10.1.0 protobuf-4.25.2 pyasn1-0.5.1 pyasn1-modules-0.3.0 pyparsing-3.1.1 pyyaml-6.0.1 regex-2023.12.25 rsa-4.9 safetensors-0.4.2 sympy-1.12 tokenizers-0.15.1 torch-2.1.2 transformers-4.35.2 triton-2.1.0 typing-extensions-4.9.0 uritemplate-4.1.1