Appendix A

Extra Information

A.1 All models trained

A large number of models was trained as a result of this project. Most of the successful ones were uploaded to the `Huggingface Hub`. This allows for easy access, backup, and allows other people to use them for their projects and research.

A complete list of all the models created and uploaded is below:

1. MikolajDeja/gsarti-opus-mt-tc-en-pl-kde4-finetune
2. MikolajDeja/gsarti-opus-mt-tc-en-pl-opus100-accelerate-finetune
3. MikolajDeja/gsarti-opus-mt-tc-en-pl-opus100-finetune
4. MikolajDeja/gsarti-opus-mt-tc-en-pl-opus100-finetune-100
5. MikolajDeja/gsarti-opus-mt-tc-en-pl-opus100-finetune-50
6. MikolajDeja/gsarti-opus-mt-tc-en-pl-3-para_crawl-finetune
7. MikolajDeja/gsarti-opus-mt-tc-en-pl-para_crawl-finetune
8. MikolajDeja/gsarti-opus-mt-tc-en-pl-yhavinga-ccmatrix-finetune
9. MikolajDeja/Helsinki-NLP-opus-mt-pl-en-kde4-finetune
10. MikolajDeja/Helsinki-NLP-opus-mt-pl-en-opus100-accelerate
11. MikolajDeja/Helsinki-NLP-opus-mt-pl-en-opus100-finetune
12. MikolajDeja/Helsinki-NLP-opus-mt-pl-en-opus100-finetune-100

13. MikolajDeja/Helsinki-NLP-opus-mt-pl-en-opus100-finetune-50
14. MikolajDeja/Helsinki-NLP-opus-mt-pl-en-3-para_crawl-finetune
15. MikolajDeja/Helsinki-NLP-opus-mt-pl-en-para_crawl-finetune
16. MikolajDeja/Helsinki-NLP-opus-mt-pl-en-yhavinga-ccmatrix-finetune
17. MikolajDeja/Helsinki-NLP-opus-mt-en-mul-kde4-finetune
18. MikolajDeja/Helsinki-NLP-opus-mt-en-mul-opus100-accelerate
19. MikolajDeja/Helsinki-NLP-opus-mt-en-mul-opus100-finetune
20. MikolajDeja/Helsinki-NLP-opus-mt-en-mul-opus100-finetune-100
21. MikolajDeja/Helsinki-NLP-opus-mt-en-mul-opus100-finetune-50
22. MikolajDeja/Helsinki-NLP-opus-mt-en-mul-3-para_crawl-finetune
23. MikolajDeja/Helsinki-NLP-opus-mt-en-mul-para_crawl-finetune
24. MikolajDeja/Helsinki-NLP-opus-mt-en-mul-yhavinga-ccmatrix-finetune
25. MikolajDeja/Helsinki-NLP-opus-mt-mul-en-kde4-finetune
26. MikolajDeja/Helsinki-NLP-opus-mt-mul-en-opus100-accelerate
27. MikolajDeja/Helsinki-NLP-opus-mt-mul-en-opus100-finetune
28. MikolajDeja/Helsinki-NLP-opus-mt-mul-en-opus100-finetune-100
29. MikolajDeja/Helsinki-NLP-opus-mt-mul-en-opus100-finetune-50
30. MikolajDeja/Helsinki-NLP-opus-mt-mul-en-3-para_crawl-finetune
31. MikolajDeja/Helsinki-NLP-opus-mt-mul-en-para_crawl-finetune
32. MikolajDeja/Helsinki-NLP-opus-mt-mul-en-yhavinga-ccmatrix-finetune
33. MikolajDeja/alirezamsh-small100-en-pl-kde4-finetune
34. MikolajDeja/alirezamsh-small100-en-pl-opus100-accelerate-finetune
35. MikolajDeja/alirezamsh-small100-en-pl-opus100-finetune
36. MikolajDeja/alirezamsh-small100-en-pl-opus100-finetune-50

37. MikolajDeja/alirezamsh-small100-en-pl-3-para_crawl-finetune
38. MikolajDeja/alirezamsh-small100-en-pl-para_crawl-finetune
39. MikolajDeja/alirezamsh-small100-en-pl-yhavinga-ccmatrix-finetune
40. MikolajDeja/alirezamsh-small100-pl-en-kde4-finetune
41. MikolajDeja/alirezamsh-small100-pl-en-opus100-accelerate-finetune
42. MikolajDeja/alirezamsh-small100-pl-en-opus100-finetune
43. MikolajDeja/alirezamsh-small100-pl-en-opus100-finetune-50
44. MikolajDeja/alirezamsh-small100-pl-en-3-para_crawl-finetune
45. MikolajDeja/alirezamsh-small100-pl-en-para_crawl-finetune
46. MikolajDeja/alirezamsh-small100-pl-en-yhavinga-ccmatrix-finetune
47. MikolajDeja/facebook-nllb-200-distilled-600M-en-pl-opus100-finetune
48. MikolajDeja/facebook-nllb-200-distilled-600M-en-pl-3-para_crawl-finetune
49. MikolajDeja/facebook-nllb-200-distilled-600M-en-pl-para_crawl
50. MikolajDeja/facebook-nllb-200-distilled-600M-en-pl-yhavinga-ccmatrix-finetune
51. MikolajDeja/facebook-nllb-200-distilled-600M-pl-en-opus100-finetune
52. MikolajDeja/facebook-nllb-200-distilled-600M-pl-en-3-para_crawl-finetune
53. MikolajDeja/facebook-nllb-200-distilled-600M-pl-en-para_crawl-finetune-copy
54. MikolajDeja/facebook-nllb-200-distilled-600M-pl-en-yhavinga-ccmatrix-finetune

A.2 The design of the translator app

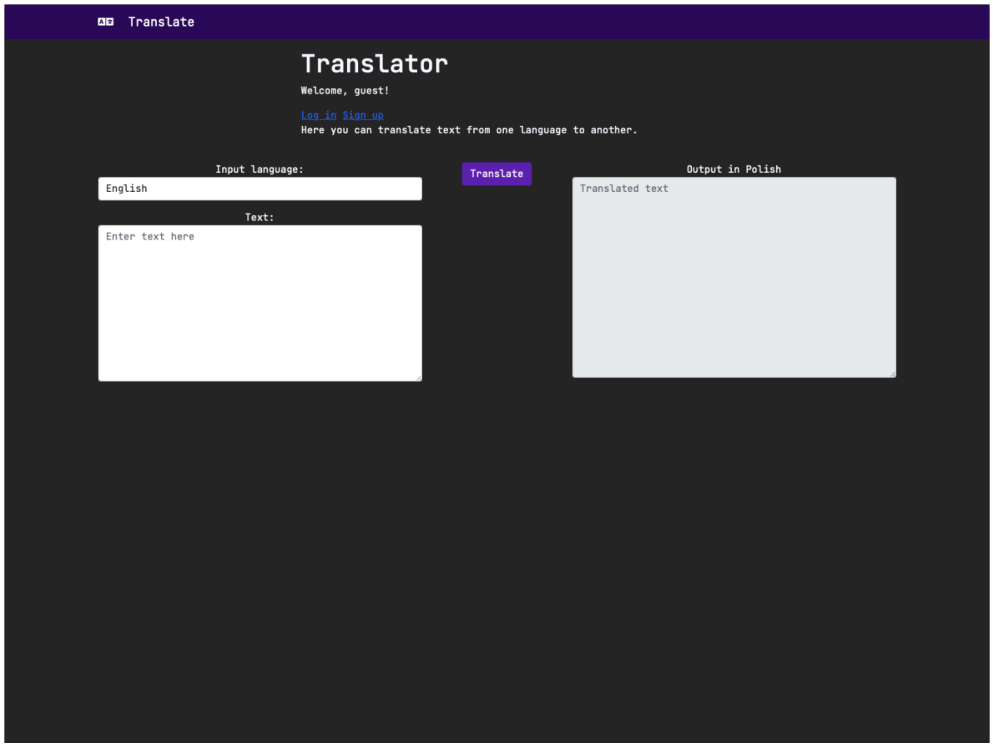

Figure A.1: The home page for a guest

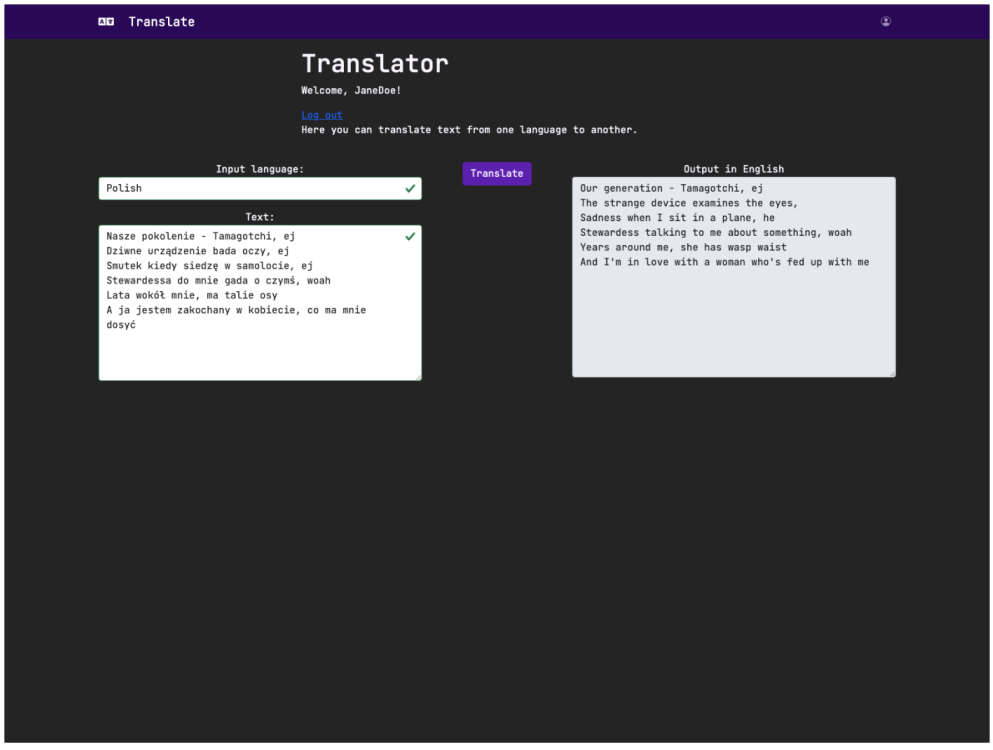

Figure A.2: The home page for a logged in user who translates from Polish to English

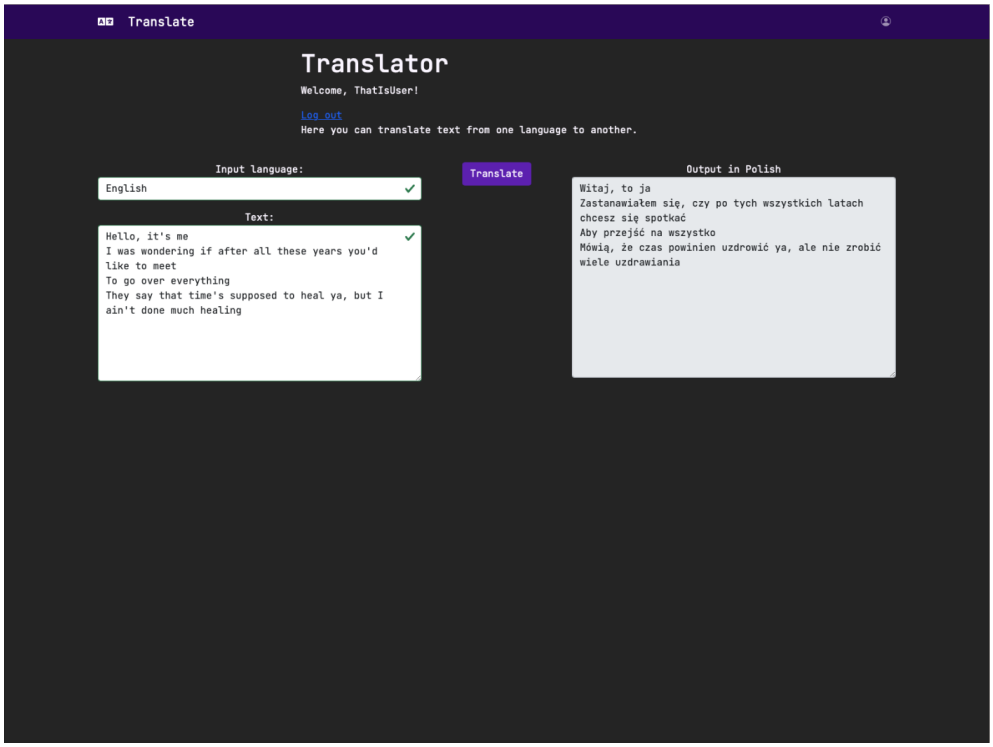

Figure A.3: The home page for a logged in user who translates from English to Polish

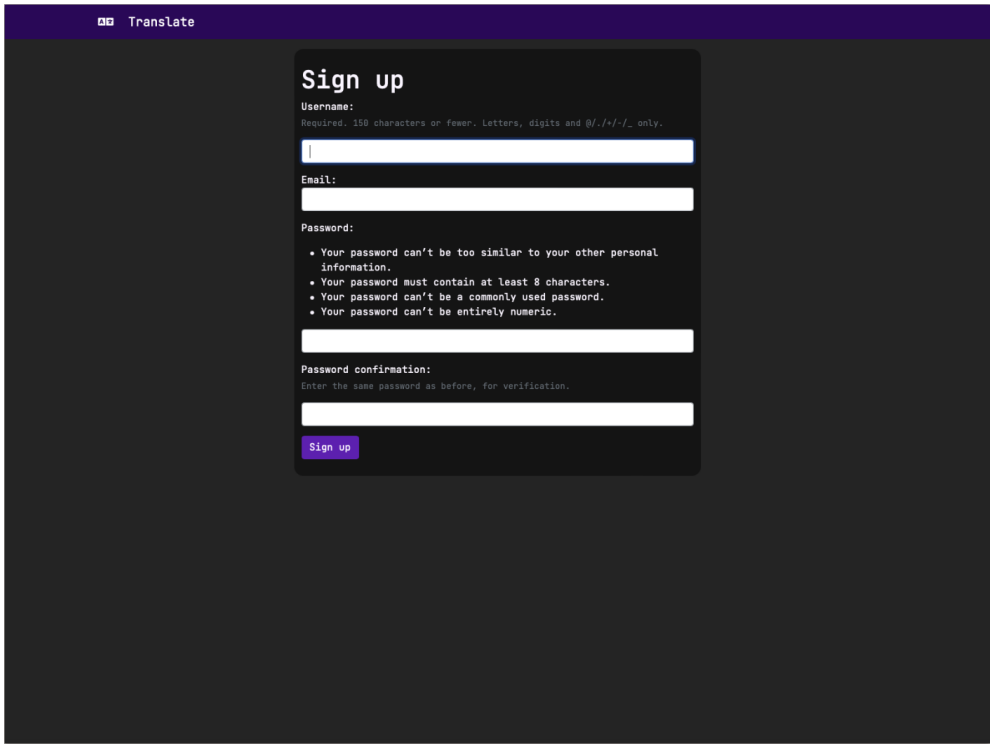

Figure A.4: The sign up page

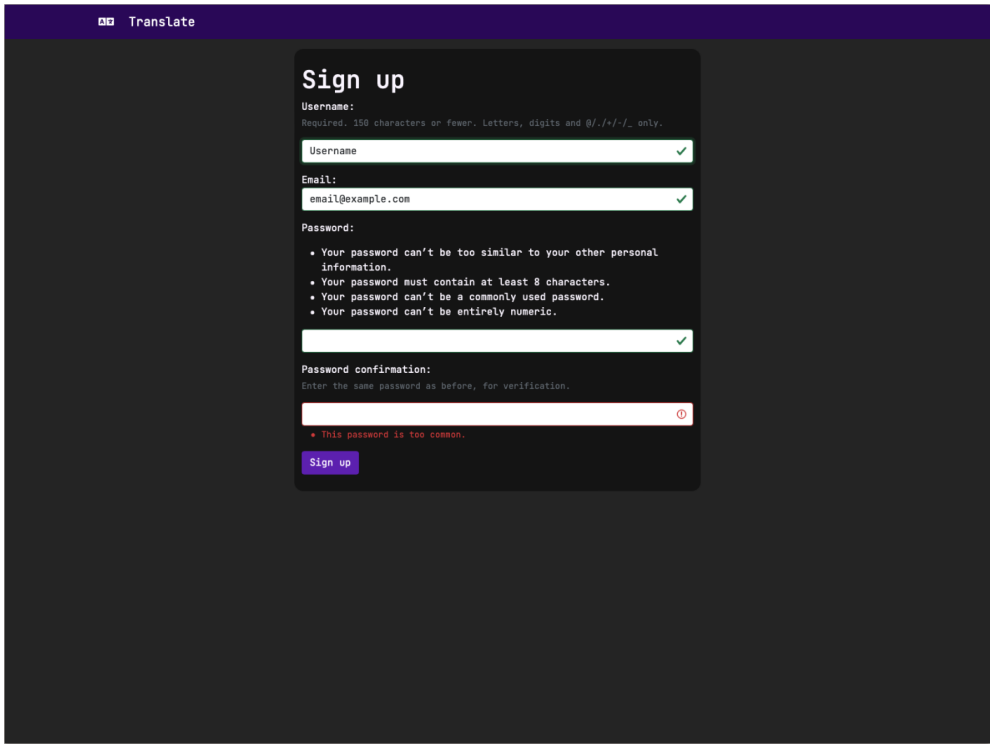

Figure A.5: The sign up page with a password that is too common password

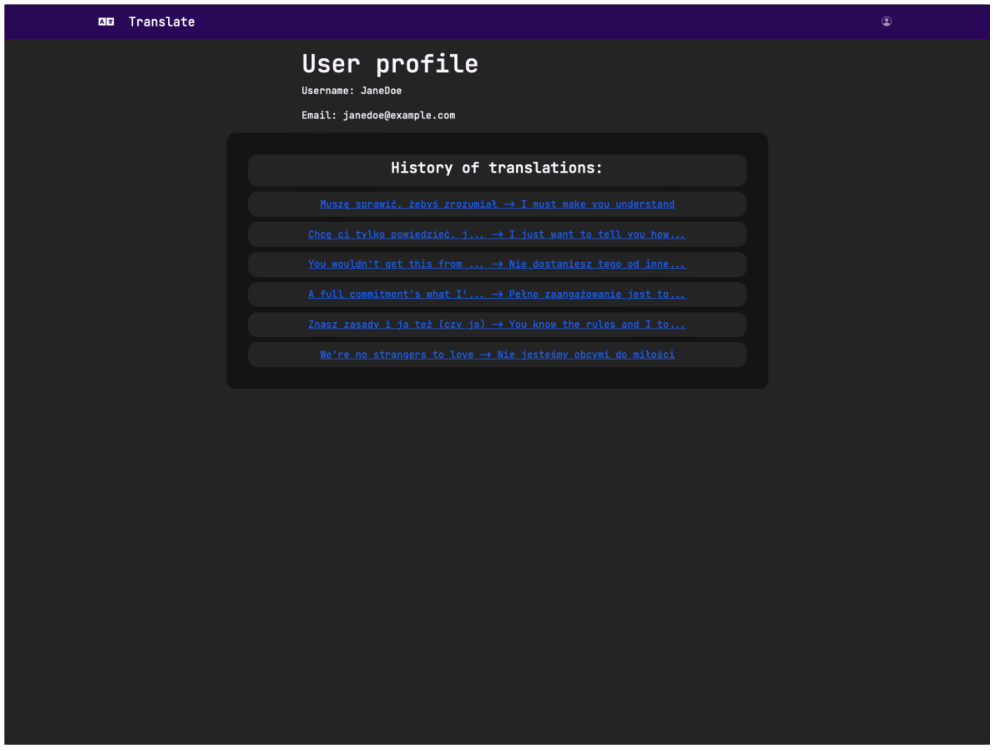

Figure A.6: The profile page with the history of translations

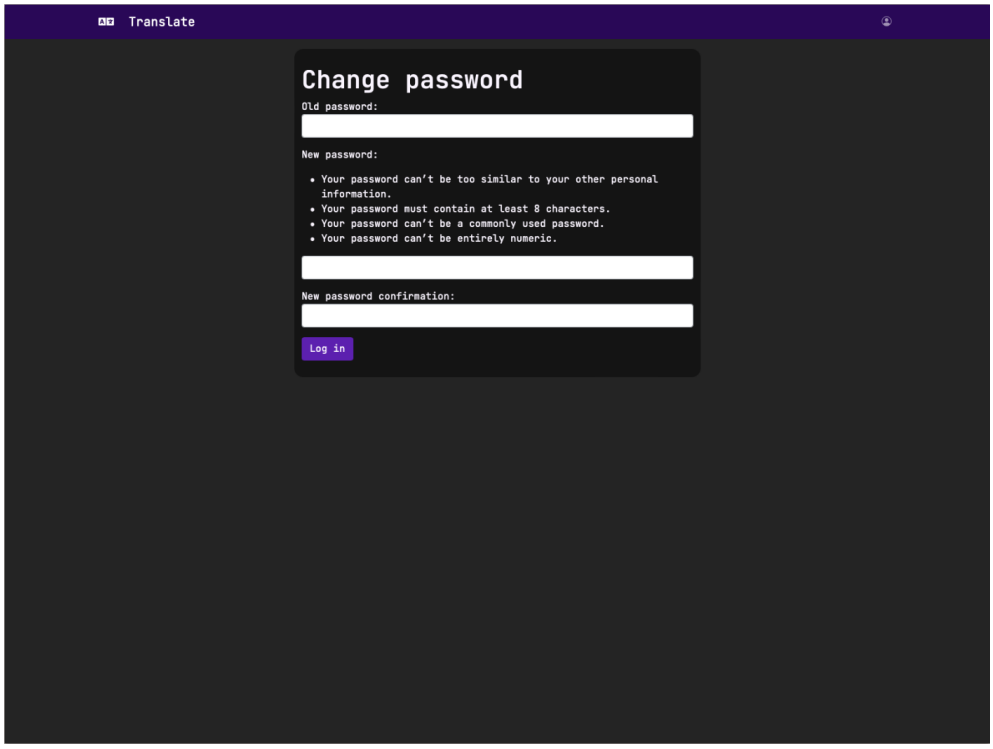

Figure A.7: The change password page

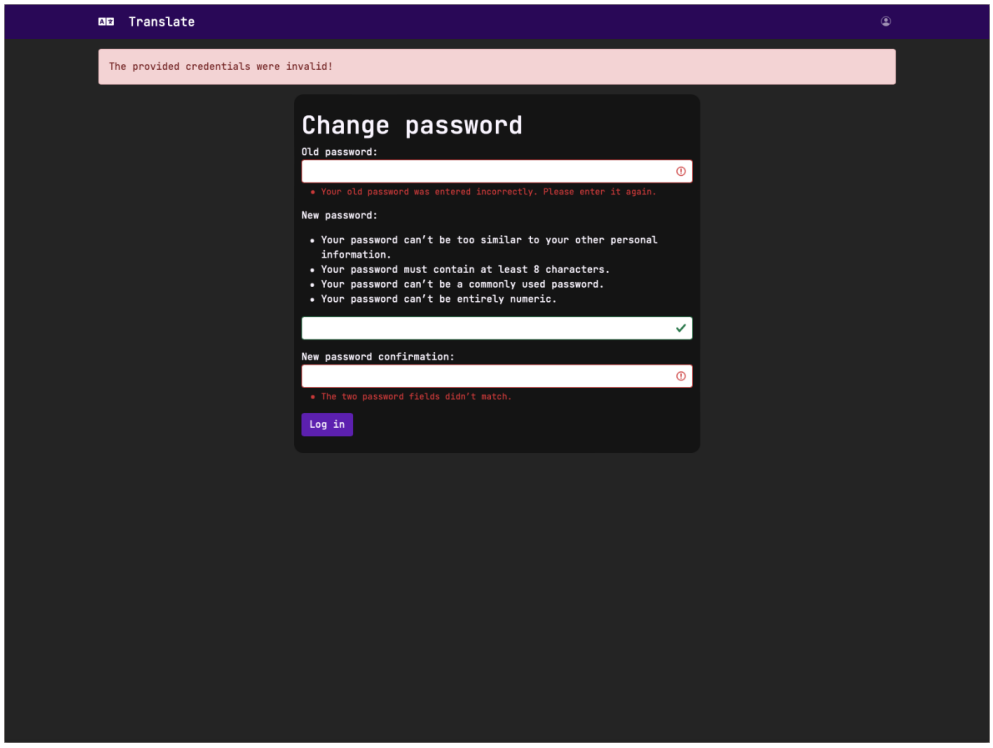

Figure A.8: The change password page with incorrect password confirmation

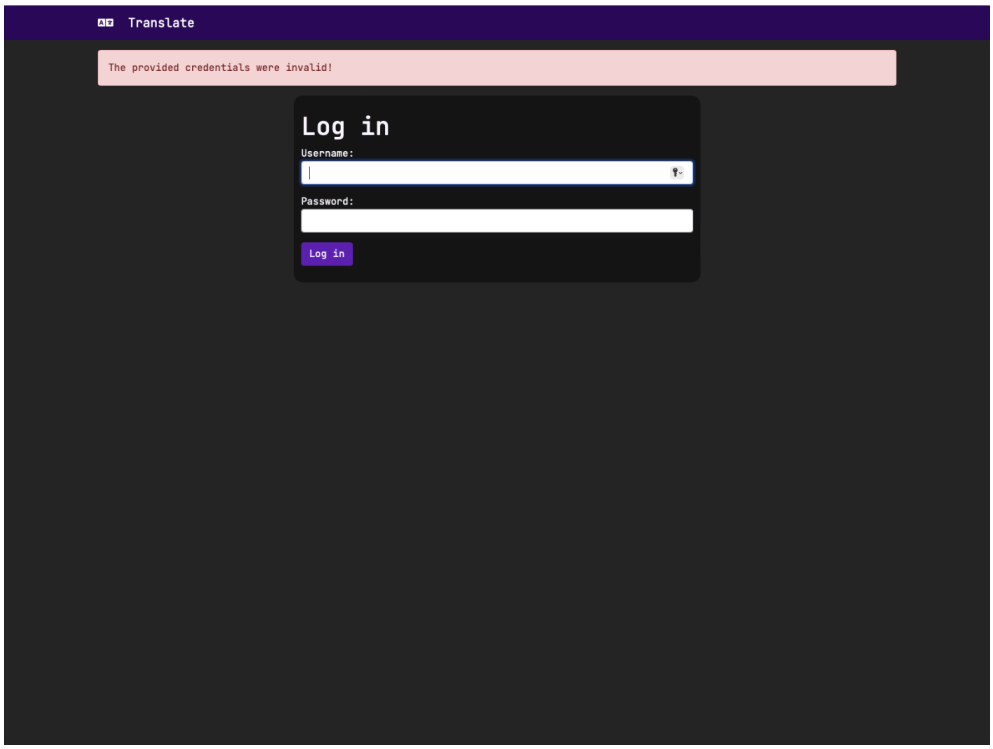

Figure A.9: Log in page with incorrect credentials

Appendix B

User Guide

B.1 Machine translation models

All of the models trained for this project (that were deemed valuable) were uploaded to the `Huggingface Hub`. A full list is available in Appendix A.1. All of them can easily be used with the `Inference API`.

```
pipeline_to_use = pipeline('translation', tokenizer='gsarti/opus-mt-tc-en-pl',
                                          model='MikolajDeja/gsarti-opus-mt-tc-en-
                                          pl-3-para_crawl-finetune', src_lang='en'
                                          )

pipeline_to_use = pipeline('translation', tokenizer='facebook/nllb-200-distilled
                                          -600M', model='MikolajDeja/facebook-nllb
                                          -200-distilled-600M-en-pl-yhavinga-
                                          ccmatrix-finetune', src_lang='en',
                                          tgt_lang='pl')
```

The pipelines for the `M2M-100` based models require an additional argument of `tgt_lang`.

B.1.1 Training

Google Colaboratory

Should someone like to train the models in the same way that it was done for this project, a suitable environment must be prepared. Below, necessary packages are listed.

- `datasets`

- `evaluate`
- `transformers[sentencepiece]`
- `sacrebleu`

For uploading files to `Huggingface Hub`, `git-lfs` is required.

The above allows training on Google Colaboratory. In the Jupyter notebook, one can first add a code cell to install all the libraries and packages with `!apt install git-lfs` and `!pip install datasets evaluate transformers[sentencepiece] sacrebleu`. Then, the code in `training.py` should be copied to a code cell. After that, the chosen entry script can be copied to a code cell. This allows for a training to run.

HPC Create

For training on HPC Create, one must be more precise with the packages and modules loaded.

After literal weeks of fighting with it, I do not dare touch it and uninstall something needed. The list of packages currently installed is:

```
_libgcc_mutex             0.1                  main
_openmp_mutex             5.1                  1_gnu
absl-py                   1.4.0                pypi_0    pypi
accelerate                0.15.0               pypi_0    pypi
aiohttp                   3.8.3                pypi_0    pypi
aiosignal                 1.3.1                pypi_0    pypi
alembic                   1.9.2                pypi_0    pypi
anyio                     3.5.0           py38h06a4308_0
argon2-cffi               21.3.0            pyhd3eb1b0_0
argon2-cffi-bindings      21.2.0          py38h7f8727e_0
asttokens                 2.0.5             pyhd3eb1b0_0
async-timeout             4.0.2           py38h06a4308_0
attrs                     22.1.0          py38h06a4308_0
babel                     2.11.0          py38h06a4308_0
backcall                  0.2.0             pyhd3eb1b0_0
beautifulsoup4            4.11.1          py38h06a4308_0
blas                      1.0                     mkl
bleach                    4.1.0             pyhd3eb1b0_0
bottleneck                1.3.5           py38h7deecbd_0
brotlipy                  0.7.0         py38h27cfd23_1003
bzip2                     1.0.8                h7b6447c_0
c-ares                    1.18.1               h7f8727e_0
ca-certificates           2023.01.10           h06a4308_0
cachetools                5.3.0                pypi_0    pypi
certifi                   2022.12.7       py38h06a4308_0
cffi                      1.15.1          py38h74dc2b5_0
charset-normalizer        2.0.4             pyhd3eb1b0_0
click                     8.1.3                pypi_0    pypi
cmaes                     0.9.1                pypi_0    pypi
colorama                  0.4.6           py38h06a4308_0
colorlog                  6.7.0                pypi_0    pypi
comm                      0.1.2           py38h06a4308_0
cryptography              38.0.4          py38h9ce1e76_0
cuda                      11.6.2                    0    nvidia
cuda-cccl                 11.6.55              hf6102b2_0    nvidia/label/cuda-11.6.2
cuda-command-line-tools   11.6.2                    0    nvidia/label/cuda-11.6.2
```

```
cuda-compiler                 11.6.2                        0    nvidia/label/cuda-11.6.2
cuda-cudart                   11.6.55                he381448_0    nvidia/label/cuda-11.6.2
cuda-cudart-dev               11.6.55                h42ad0f4_0    nvidia/label/cuda-11.6.2
cuda-cuobjdump                11.6.124               h2eeebcb_0    nvidia/label/cuda-11.6.2
cuda-cupti                    11.6.124               h86345e5_0    nvidia/label/cuda-11.6.2
cuda-cuxxfilt                 11.6.124               hecbf4f6_0    nvidia/label/cuda-11.6.2
cuda-driver-dev               11.6.55                       0    nvidia/label/cuda-11.6.2
cuda-gdb                      11.6.124               hcdc6958_0    nvidia/label/cuda-11.6.2
cuda-libraries                11.6.2                        0    nvidia/label/cuda-11.6.2
cuda-libraries-dev            11.6.2                        0    nvidia/label/cuda-11.6.2
cuda-memcheck                 11.6.124               ha4ac6c0_0    nvidia/label/cuda-11.6.2
cuda-nsight                   11.6.124                      0    nvidia/label/cuda-11.6.2
cuda-nsight-compute           11.6.2                        0    nvidia/label/cuda-11.6.2
cuda-nvcc                     11.6.124               hbba6d2d_0    nvidia/label/cuda-11.6.2
cuda-nvdisasm                 11.6.124               h75ac146_0    nvidia/label/cuda-11.6.2
cuda-nvml-dev                 11.6.55                haa9ef22_0    nvidia/label/cuda-11.6.2
cuda-nvprof                   11.6.124               h7c7a4e2_0    nvidia/label/cuda-11.6.2
cuda-nvprune                  11.6.124               he22ec0a_0    nvidia/label/cuda-11.6.2
cuda-nvrtc                    11.6.124               h020bade_0    nvidia/label/cuda-11.6.2
cuda-nvrtc-dev                11.6.124               h249d397_0    nvidia/label/cuda-11.6.2
cuda-nvtx                     11.6.124               h0630a44_0    nvidia/label/cuda-11.6.2
cuda-nvvp                     11.6.124               h38ac01c_0    nvidia/label/cuda-11.6.2
cuda-runtime                  11.6.2                        0    nvidia
cuda-samples                  11.6.101               h8efea70_0    nvidia/label/cuda-11.6.2
cuda-sanitizer-api            11.6.124               h3d04c53_0    nvidia/label/cuda-11.6.2
cuda-toolkit                  11.6.2                        0    nvidia/label/cuda-11.6.2
cuda-tools                    11.6.2                        0    nvidia/label/cuda-11.6.2
cuda-visual-tools             11.6.2                        0    nvidia/label/cuda-11.6.2
datasets                      2.10.1                   pypi_0    pypi
debugpy                       1.5.1               py38h295c915_0
decorator                     5.1.1                 pyhd3eb1b0_0
defusedxml                    0.7.1                 pyhd3eb1b0_0
deprecated                    1.2.13              py38h06a4308_0
dill                          0.3.6                    pypi_0    pypi
distlib                       0.3.6                    pypi_0    pypi
entrypoints                   0.4                 py38h06a4308_0
evaluate                      0.4.0                    pypi_0    pypi
executing                     0.8.3                 pyhd3eb1b0_0
ffmpeg                        4.2.2                   h20bf706_0
filelock                      3.9.0               py38h06a4308_0
flit-core                     3.6.0                 pyhd3eb1b0_0
freetype                      2.12.1                  h4a9f257_0
frozenlist                    1.3.3                    pypi_0    pypi
fsspec                        2022.11.0                pypi_0    pypi
gds-tools                     1.2.1.4                       0    nvidia/label/cuda-11.6.2
giflib                        5.2.1                   h5eee18b_1
gmp                           6.2.1                   h295c915_3
gnutls                        3.6.15                  he1e5248_0
google-auth                   2.16.0                   pypi_0    pypi
google-auth-oauthlib          0.4.6                    pypi_0    pypi
greenlet                      2.0.2                    pypi_0    pypi
grpcio                        1.51.1                   pypi_0    pypi
huggingface-hub               0.11.1                   pypi_0    pypi
icu                           58.2                    he6710b0_3
idna                          3.4                 py38h06a4308_0
importlib-metadata            4.11.3              py38h06a4308_0
importlib_resources           5.2.0                 pyhd3eb1b0_1
intel-openmp                  2021.4.0             h06a4308_3561
ipykernel                     6.19.2              py38hb070fc8_0
ipython                       8.8.0               py38h06a4308_0
ipython_genutils              0.2.0                 pyhd3eb1b0_1
ipywidgets                    8.0.4                    pypi_0    pypi
jedi                          0.18.1              py38h06a4308_1
jinja2                        3.1.2               py38h06a4308_0
jpeg                          9e                      h7f8727e_0
json5                         0.9.6                 pyhd3eb1b0_0
```

jsonschema	4.16.0	py38h06a4308_0	
jupyter_client	7.4.8	py38h06a4308_0	
jupyter_core	5.1.1	py38h06a4308_0	
jupyter_server	1.23.4	py38h06a4308_0	
jupyterlab	3.5.2	pyhd8ed1ab_0	conda-forge
jupyterlab-widgets	3.0.5	pypi_0	pypi
jupyterlab_pygments	0.1.2	py_0	
jupyterlab_server	2.16.5	py38h06a4308_0	
jupyterlab_widgets	1.0.0	pyhd3eb1b0_1	
lame	3.100	h7b6447c_0	
lcms2	2.12	h3be6417_0	
ld_impl_linux-64	2.38	h1181459_1	
lerc	3.0	h295c915_0	
libcublas	11.9.2.110	h5e84587_0	nvidia/label/cuda-11.6.2
libcublas-dev	11.9.2.110	h5c901ab_0	nvidia/label/cuda-11.6.2
libcufft	10.7.2.124	h4fbf590_0	nvidia/label/cuda-11.6.2
libcufft-dev	10.7.2.124	h98a8f43_0	nvidia/label/cuda-11.6.2
libcufile	1.2.1.4	0	nvidia/label/cuda-11.6.2
libcufile-dev	1.2.1.4	0	nvidia/label/cuda-11.6.2
libcurand	10.2.9.124	h37c27f7_0	nvidia/label/cuda-11.6.2
libcurand-dev	10.2.9.124	h54cfa4b_0	nvidia/label/cuda-11.6.2
libcusolver	11.3.4.124	h33c3c4e_0	nvidia/label/cuda-11.6.2
libcusolver-dev	11.3.4.124	h203c794_0	nvidia/label/cuda-11.6.2
libcusparse	11.7.2.124	h7538f96_0	nvidia/label/cuda-11.6.2
libcusparse-dev	11.7.2.124	hbbe9722_0	nvidia/label/cuda-11.6.2
libdeflate	1.8	h7f8727e_5	
libffi	3.3	he6710b0_2	
libgcc-ng	11.2.0	h1234567_1	
libgomp	11.2.0	h1234567_1	
libidn2	2.3.2	h7f8727e_0	
libnpp	11.6.3.124	hd2722f0_0	nvidia/label/cuda-11.6.2
libnpp-dev	11.6.3.124	h3c42840_0	nvidia/label/cuda-11.6.2
libnvjpeg	11.6.2.124	hd473ad6_0	nvidia/label/cuda-11.6.2
libnvjpeg-dev	11.6.2.124	hb5906b9_0	nvidia/label/cuda-11.6.2
libopus	1.3.1	h7b6447c_0	
libpng	1.6.37	hbc83047_0	
libprotobuf	3.20.3	he621ea3_0	
libsodium	1.0.18	h7b6447c_0	
libstdcxx-ng	11.2.0	h1234567_1	
libtasn1	4.16.0	h27cfd23_0	
libtiff	4.5.0	h6a678d5_1	
libunistring	0.9.10	h27cfd23_0	
libvpx	1.7.0	h439df22_0	
libwebp	1.2.4	h11a3e52_0	
libwebp-base	1.2.4	h5eee18b_0	
libxml2	2.9.14	h74e7548_0	
libxslt	1.1.35	h4e12654_0	
lxml	4.9.2	pypi_0	pypi
lz4-c	1.9.4	h6a678d5_0	
mako	1.2.4	pypi_0	pypi
markdown	3.4.1	pypi_0	pypi
markupsafe	2.1.1	py38h7f8727e_0	
matplotlib-inline	0.1.6	py38h06a4308_0	
mistune	0.8.4	py38h7b6447c_1000	
mkl	2021.4.0	h06a4308_640	
mkl-service	2.4.0	py38h7f8727e_0	
mkl_fft	1.3.1	py38hd3c417c_0	
mkl_random	1.2.2	py38h51133e4_0	
msgpack	1.0.4	pypi_0	pypi
msgpack-python	1.0.3	py38hd09550d_0	
multidict	6.0.4	pypi_0	pypi
multiprocess	0.70.14	pypi_0	pypi
nbclassic	0.4.8	py38h06a4308_0	
nbclient	0.5.13	py38h06a4308_0	
nbconvert	6.5.4	py38h06a4308_0	
nbformat	5.7.0	py38h06a4308_0	

```
ncurses                   6.4                  h6a678d5_0
nest-asyncio              1.5.6             py38h06a4308_0
nettle                    3.7.3                hbbd107a_1
notebook                  6.5.2             py38h06a4308_0
notebook-shim             0.2.2             py38h06a4308_0
nsight-compute            2022.1.1.2                    0    nvidia/label/cuda-11.6.2
numexpr                   2.8.4             py38he184ba9_0
numpy                     1.23.5            py38h14f4228_0
numpy-base                1.23.5            py38h31eccc5_0
oauthlib                  3.2.2                   pypi_0    pypi
openh264                  2.1.1                h4ff587b_0
openssl                   1.1.1s               h7f8727e_0
optuna                    3.1.0                   pypi_0    pypi
packaging                 22.0              py38h06a4308_0
pandas                    1.5.2                   pypi_0    pypi
pandocfilters             1.5.0               pyhd3eb1b0_0
parso                     0.8.3               pyhd3eb1b0_0
pexpect                   4.8.0               pyhd3eb1b0_3
pickleshare               0.7.5            pyhd3eb1b0_1003
pillow                    9.3.0             py38h6a678d5_2
pip                       22.3.1            py38h06a4308_0
pkgutil-resolve-name      1.3.10            py38h06a4308_0
platformdirs              2.5.2             py38h06a4308_0
portalocker               2.6.0                   pypi_0    pypi
prometheus_client         0.14.1            py38h06a4308_0
prompt-toolkit            3.0.36            py38h06a4308_0
protobuf                  3.20.1                  pypi_0    pypi
psutil                    5.9.4                   pypi_0    pypi
ptyprocess                0.7.0               pyhd3eb1b0_2
pure_eval                 0.2.2               pyhd3eb1b0_0
pyarrow                   10.0.1                  pypi_0    pypi
pyasn1                    0.4.8                   pypi_0    pypi
pyasn1-modules            0.2.8                   pypi_0    pypi
pycparser                 2.21                pyhd3eb1b0_0
pygments                  2.11.2              pyhd3eb1b0_0
pyopenssl                 22.0.0              pyhd3eb1b0_0
pyrsistent                0.18.0            py38heee7806_0
pysocks                   1.7.1             py38h06a4308_0
python                    3.8.12               h12debd9_0
python-dateutil           2.8.2               pyhd3eb1b0_0
python-fastjsonschema     2.16.2            py38h06a4308_0
pytorch                   1.13.1         py3.8_cuda11.6_cudnn8.3.2_0    pytorch
pytorch-cuda              11.6                 h867d48c_0    pytorch
pytorch-mutex             1.0                        cuda    pytorch
pytz                      2022.7            py38h06a4308_0
pyyaml                    6.0               py38h5eee18b_1
pyzmq                     23.2.0            py38h6a678d5_0
ray                       2.2.0                   pypi_0    pypi
ray-core                  1.6.0             py38h295c915_0
ray-tune                  1.6.0             py38h06a4308_0
readline                  8.2                  h5eee18b_0
redis-py                  4.3.4             py38h06a4308_0
regex                     2022.10.31              pypi_0    pypi
requests                  2.28.1            py38h06a4308_0
requests-oauthlib         1.3.1                   pypi_0    pypi
responses                 0.18.0                  pypi_0    pypi
rsa                       4.9                     pypi_0    pypi
sacrebleu                 2.3.1                   pypi_0    pypi
scipy                     1.10.0                  pypi_0    pypi
send2trash                1.8.0               pyhd3eb1b0_1
sentencepiece             0.1.97                  pypi_0    pypi
setproctitle              1.1.10           py38h7b6447c_1001
setuptools                65.6.3            py38h06a4308_0
six                       1.16.0              pyhd3eb1b0_1
sniffio                   1.2.0             py38h06a4308_1
soupsieve                 2.3.2.post1       py38h06a4308_0
```

```
sqlalchemy                2.0.1                pypi_0    pypi
sqlite                    3.40.1               h5082296_0
stack_data                0.2.0              pyhd3eb1b0_0
tabulate                  0.9.0                pypi_0    pypi
tensorboard               2.11.2               pypi_0    pypi
tensorboard-data-server   0.6.1                pypi_0    pypi
tensorboard-plugin-wit    1.8.1                pypi_0    pypi
tensorboardx              2.5.1                pypi_0    pypi
terminado                 0.17.1           py38h06a4308_0
tinycss2                  1.2.1            py38h06a4308_0
tk                        8.6.12               h1ccaba5_0
tokenizers                0.13.2               pypi_0    pypi
tomli                     2.0.1            py38h06a4308_0
torchaudio                0.13.1             py38_cu116  pytorch
torchvision               0.14.1             py38_cu116  pytorch
tornado                   6.2              py38h5eee18b_0
tqdm                      4.64.1               pypi_0    pypi
traitlets                 5.7.1            py38h06a4308_0
transformers              4.25.1               pypi_0    pypi
typing-extensions         4.4.0            py38h06a4308_0
typing_extensions         4.4.0            py38h06a4308_0
urllib3                   1.26.14          py38h06a4308_0
virtualenv                20.17.1              pypi_0    pypi
wcwidth                   0.2.5              pyhd3eb1b0_0
webencodings              0.5.1                py38_1
websocket-client          0.58.0           py38h06a4308_4
werkzeug                  2.2.2                pypi_0    pypi
wheel                     0.37.1             pyhd3eb1b0_0
widgetsnbextension        4.0.5                pypi_0    pypi
wrapt                     1.14.1           py38h5eee18b_0
x264                      1!157.20191217       h7b6447c_0
xxhash                    3.2.0                pypi_0    pypi
xz                        5.2.10               h5eee18b_1
yaml                      0.2.5                h7b6447c_0
yarl                      1.8.2                pypi_0    pypi
zeromq                    4.3.4                h2531618_0
zipp                      3.11.0           py38h06a4308_0
zlib                      1.2.13               h5eee18b_0
zstd                      1.5.2                ha4553b6_0
```

Admittedly, `Jupyter` was also installed, since at the beginning it was used for training. Also, packages for hyperparameter search backend are installed and they are not necessary for training. The important pieces are the `Huggingface` packages, `torch`, and the `cuda` packages for running on the Nvidia GPUs and all of their dependencies.

To run the training scripts on HPC Create, the helper bash scripts have to be used to submit jobs with `sbatch`. To run one particular training, the `train.sh` script should be edited and all the other entry level scripts should be commented out. Only one line with `python scripts/file.py` should not be commented out.

B.1.2 Hyperparameter search

The setup for hyperparameter search is very similar to that of training, the only difference is that the hyperparameter search backend, `optuna` must be installed.

B.1.3 Testing

For testing, fewer packages are needed.

- `transformers`
- `datasets`
- `evaluate`

The scripts can be run in an analogous way to the training scripts. Install the packages, copy the contents of `testing.py` and the entry level script to Google Colaboratory to run it there. Use the `test.sh` to run on HPC Create.

B.2 Practical application

Running the practical application locally is the same as running any Django application.

The following script will install the necessary packages, create database migrations and run the server locally. For this project, Python 3 is required (3.9 recommended).

```
virtualenv venv --python=3.9
source venv/bin/activate
pip3 install -r requirements.txt
python3 manage.py migrate
python3 manage.py runserver
```

After having ran this set of commands, the user can navigate to *localhost:8000* to access the application.

To run the tests, use

```
python manage.py test
```

To then see the test coverage, run

```
coverage run manage.py test
coverage report
```

Appendix C

Source Code

1. Machine learning
   (a) Training - the training scripts
   (b) Hyper - the scripts used for hyperparameter search
   (c) Testing - the testing scripts
   (d) HelperScripts - the scripts made for running the other scripts on HPC Create
2. Translator - the code for the practical application

I verify that I am the sole author of the programs contained in this folder, except where explicitly stated to the contrary.

Mikolaj Deja,

April 6, 2023

43 files

```
Training/TrainAlirezaEnPlCC.py
Training/TrainAlirezaEnPlOpus.py
Training/TrainAlirezaEnPlOpusAcc.py
Training/TrainAlirezaEnPlPara.py
Training/TrainAlirezaEnPlPara3.py
Training/TrainAlirezaEnPlkde4.py
Training/TrainAlirezaPlEnCC.py
Training/TrainAlirezaPlEnOpus.py
Training/TrainAlirezaPlEnOpusAcc.py
Training/TrainAlirezaPlEnPara.py
Training/TrainAlirezaPlEnPara3.py
Training/TrainAlirezaPlEnkde4.py
Training/TrainEnMulCC.py
Training/TrainEnMulOpus.py
Training/TrainEnMulOpusAcc.py
Training/TrainEnMulPara.py
Training/TrainEnMulPara3.py
Training/TrainEnMulkde4.py
Training/TrainEnPlCC.py
Training/TrainEnPlOpus.py
Training/TrainEnPlOpusAcc.py
Training/TrainEnPlPara.py
Training/TrainEnPlPara3.py
Training/TrainEnPlkde4.py
Training/TrainMulEnCC.py
Training/TrainMulEnOpus.py
Training/TrainMulEnOpusAcc.py
Training/TrainMulEnPara.py
Training/TrainMulEnPara3.py
Training/TrainMulEnkde4.py
Training/TrainNLLBEnPlCC.py
Training/TrainNLLBEnPlOpus.py
Training/TrainNLLBEnPlPara3.py
Training/TrainNLLBPlEnCC.py
Training/TrainNLLBPlEnOpus.py
Training/TrainNLLBPlEnPara3.py
Training/TrainPlEnCC.py
Training/TrainPlEnOpus.py
Training/TrainPlEnOpusAcc.py
Training/TrainPlEnPara.py
Training/TrainPlEnPara3.py
Training/TrainPlEnkde4.py
Training/training.py
```

Training/TrainAlirezaEnPlCC.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'yhavinga/ccmatrix'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

```

```
model_name_cleaned = MODEL_NAME.replace('/', '-') + '-en-pl'

raw_dataset = load_dataset(DATASET, 'en-pl').filter(lambda example, idx: idx % 70 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='pl')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainAlirezaEnPlOpus.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'opus100'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-en-pl'

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='pl')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET,
```

```
EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainAlirezaEnPlOpusAcc.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_accelerate

DATASET = 'opus100'
EPOCH_NUM = 50
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-en-pl'

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='pl')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_accelerate(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Training/TrainAlirezaEnPlPara.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 50
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-en-pl'

raw_dataset = load_dataset(DATASET, 'enpl')
```

```

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='pl')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainAlirezaEnPlPara3.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-en-pl'

raw_dataset = load_dataset(DATASET, 'enpl').filter(lambda example, idx: idx % 3 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='pl')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET + '-3', EPOCH_NUM, tokenize_help,
            max_length)

```

Training/TrainAlirezaEnPlkde4.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'kde4'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-en-pl'

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='pl')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainAlirezaPlEnCC.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'yhavinga/ccmatrix'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-pl-en'

raw_dataset = load_dataset(DATASET, 'en-pl').filter(lambda example, idx: idx % 70 == 0, with_indices=True)

```

```
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainAlirezaPlEnOpus.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'opus100'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-pl-en'

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainAlirezaPlEnOpusAcc.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_accelerate

DATASET = 'opus100'
EPOCH_NUM = 50
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-pl-en'

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_accelerate(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Training/TrainAlirezaPlEnPara.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 50
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-pl-en'

raw_dataset = load_dataset(DATASET, 'enpl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')


# The preprocessing function was adapted from the huggingface example
```

```
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainAlirezaPlEnPara3.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-pl-en'

raw_dataset = load_dataset(DATASET, 'enpl').filter(lambda example, idx: idx % 3 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET + '-3', EPOCH_NUM, tokenize_help,
            max_length)

```

Training/TrainAlirezaPlEnkde4.py

```

```

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'kde4'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'alirezamsh/small100'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-pl-en'

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainEnMulCC.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'yhavinga/ccmatrix'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-en-mul'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'en-pl').filter(lambda example, idx: idx % 70 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
```

```
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainEnMulOpus.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'opus100'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-en-mul'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainEnMulOpusAcc.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_accelerate
```

```

DATASET = 'opus100'
EPOCH_NUM = 50
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-en-mul'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_accelerate(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Training/TrainEnMulPara.py

```

from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-en-mul'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'enpl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
```

```
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainEnMulPara3.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-en-mul'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'enpl').filter(lambda example, idx: idx % 3 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET + '-3', EPOCH_NUM, tokenize_help,
            max_length)

```

Training/TrainEnMulkde4.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'kde4'
EPOCH_NUM = 100
max_length = 128
```

```
MODEL_NAME = 'Helsinki-NLP/opus-mt-en-mul'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainEnPlCC.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'yhavinga/ccmatrix'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'gsarti/opus-mt-tc-en-pl'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'en-pl').filter(lambda example, idx: idx % 70 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


```

```
train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainEnPlOpus.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'opus100'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'gsarti/opus-mt-tc-en-pl'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainEnPlOpusAcc.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_accelerate

DATASET = 'opus100'
EPOCH_NUM = 50
max_length = 128
MODEL_NAME = 'gsarti/opus-mt-tc-en-pl'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'en-pl')
```

```
14
15 tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')
16
17
18 # The preprocessing function was adapted from the huggingface example
19 # https://huggingface.co/docs/transformers/tasks/translation
20 def tokenize_help(examples):
21     inputs = [ex['en'] for ex in examples['translation']]
22     targets = [ex['pl'] for ex in examples['translation']]
23     model_inputs = tokenizer(
24         inputs, text_target=targets, max_length=max_length, truncation=True
25     )
26     return model_inputs
27
28
29 train_accelerate(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET,
   EPOCH_NUM, tokenize_help)
30
```

Training/TrainEnPlPara.py

```
1  from datasets import load_dataset
2  from transformers import AutoTokenizer
3
4  from training import train_model
5
6  DATASET = 'para_crawl'
7  EPOCH_NUM = 100
8  max_length = 128
9  MODEL_NAME = 'gsarti/opus-mt-tc-en-pl'
10
11 model_name_cleaned = MODEL_NAME.replace('/', '-')
12
13 raw_dataset = load_dataset(DATASET, 'enpl')
14
15 tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')
16
17
18 # The preprocessing function was adapted from the huggingface example
19 # https://huggingface.co/docs/transformers/tasks/translation
20 def tokenize_help(examples):
21     inputs = [ex['en'] for ex in examples['translation']]
22     targets = [ex['pl'] for ex in examples['translation']]
23     model_inputs = tokenizer(
24         inputs, text_target=targets, max_length=max_length, truncation=True
25     )
26     return model_inputs
27
28
29 train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET,
   EPOCH_NUM, tokenize_help, max_length)
30
```

Training/TrainEnPlPara3.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'gsarti/opus-mt-tc-en-pl'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'enpl').filter(lambda example, idx: idx % 3 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET + '-3', EPOCH_NUM, tokenize_help,
            max_length)

```

Training/TrainEnPlkde4.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'kde4'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'gsarti/opus-mt-tc-en-pl'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
```

```
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainMulEnCC.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'yhavinga/ccmatrix'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-mul-en'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'en-pl').filter(lambda example, idx: idx % 70 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainMulEnOpus.py

```
from datasets import load_dataset
from transformers import AutoTokenizer
```

```

from training import train_model

DATASET = 'opus100'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-mul-en'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET,
EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainMulEnOpusAcc.py

```

from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_accelerate

DATASET = 'opus100'
EPOCH_NUM = 50
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-mul-en'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
```

```
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_accelerate(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Training/TrainMulEnPara.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-mul-en'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'enpl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainMulEnPara3.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 100
max_length = 128
```

```
MODEL_NAME = 'Helsinki-NLP/opus-mt-mul-en'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'enpl').filter(lambda example, idx: idx % 3 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET + '-3', EPOCH_NUM, tokenize_help,
            max_length)

```

Training/TrainMulEnkde4.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'kde4'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-mul-en'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs
```

```

train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainNLLBEnPlCC.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'yhavinga/ccmatrix'
EPOCH_NUM = 50
max_length = 128
MODEL_NAME = 'facebook/nllb-200-distilled-600M'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-en-pl'

raw_dataset = load_dataset(DATASET, 'en-pl').filter(lambda example, idx: idx % 70 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='pl')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainNLLBEnPlOpus.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'opus100'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'facebook/nllb-200-distilled-600M'
```

```
10
11 model_name_cleaned = MODEL_NAME.replace('/', '-') + '-en-pl'
12
13 raw_dataset = load_dataset(DATASET, 'en-pl')
14
15 tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt',
   tgt_lang='pl')
16
17
18 # The preprocessing function was adapted from the huggingface example
19 # https://huggingface.co/docs/transformers/tasks/translation
20 def tokenize_help(examples):
21     inputs = [ex['en'] for ex in examples['translation']]
22     targets = [ex['pl'] for ex in examples['translation']]
23     model_inputs = tokenizer(
24         inputs, text_target=targets, max_length=max_length, truncation=True
25     )
26     return model_inputs
27
28
29 train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET,
   EPOCH_NUM, tokenize_help, max_length)
30
```

Training/TrainNLLBEnPlPara3.py

```
1  from datasets import load_dataset
2  from transformers import AutoTokenizer
3
4  from training import train_model
5
6  DATASET = 'para_crawl'
7  EPOCH_NUM = 100
8  max_length = 128
9  MODEL_NAME = 'facebook/nllb-200-distilled-600M'
10
11 model_name_cleaned = MODEL_NAME.replace('/', '-') + '-en-pl'
12
13 raw_dataset = load_dataset(DATASET, 'enpl').filter(lambda example, idx: idx % 3 ==
   0, with_indices=True)
14
15 tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt',
   tgt_lang='pl')
16
17
18 # The preprocessing function was adapted from the huggingface example
19 # https://huggingface.co/docs/transformers/tasks/translation
20 def tokenize_help(examples):
21     inputs = [ex['en'] for ex in examples['translation']]
22     targets = [ex['pl'] for ex in examples['translation']]
23     model_inputs = tokenizer(
24         inputs, text_target=targets, max_length=max_length, truncation=True
25     )
26     return model_inputs
27
28
```

```
train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET + '-3', EPOCH_NUM, tokenize_help,
            max_length)

```

Training/TrainNLLBPlEnCC.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'yhavinga/ccmatrix'
EPOCH_NUM = 50
max_length = 128
MODEL_NAME = 'facebook/nllb-200-distilled-600M'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-pl-en'

raw_dataset = load_dataset(DATASET, 'en-pl').filter(lambda example, idx: idx % 70 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainNLLBPlEnOpus.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'opus100'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'facebook/nllb-200-distilled-600M'

```

```
model_name_cleaned = MODEL_NAME.replace('/', '-') + '-pl-en'

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainNLLBPlEnPara3.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'facebook/nllb-200-distilled-600M'

model_name_cleaned = MODEL_NAME.replace('/', '-') + '-pl-en'

raw_dataset = load_dataset(DATASET, 'enpl').filter(lambda example, idx: idx % 3 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET + '-3',
```

```
    EPOCH_NUM, tokenize_help,
30              max_length)
31
```

Training/TrainPlEnCC.py

```
 1  from datasets import load_dataset
 2  from transformers import AutoTokenizer
 3
 4  from training import train_model
 5
 6  DATASET = 'yhavinga/ccmatrix'
 7  EPOCH_NUM = 100
 8  max_length = 128
 9  MODEL_NAME = 'Helsinki-NLP/opus-mt-pl-en'
10
11  model_name_cleaned = MODEL_NAME.replace('/', '-')
12
13  raw_dataset = load_dataset(DATASET, 'en-pl').filter(lambda example, idx: idx % 70
    == 0, with_indices=True)
14
15  tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')
16
17
18  # The preprocessing function was adapted from the huggingface example
19  # https://huggingface.co/docs/transformers/tasks/translation
20  def tokenize_help(examples):
21      inputs = [ex['pl'] for ex in examples['translation']]
22      targets = [ex['en'] for ex in examples['translation']]
23      model_inputs = tokenizer(
24          inputs, text_target=targets, max_length=max_length, truncation=True
25      )
26      return model_inputs
27
28
29  train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET,
    EPOCH_NUM, tokenize_help, max_length)
30
```

Training/TrainPlEnOpus.py

```
 1  from datasets import load_dataset
 2  from transformers import AutoTokenizer
 3
 4  from training import train_model
 5
 6  DATASET = 'opus100'
 7  EPOCH_NUM = 100
 8  max_length = 128
 9  MODEL_NAME = 'Helsinki-NLP/opus-mt-pl-en'
10
11  model_name_cleaned = MODEL_NAME.replace('/', '-')
12
```

```
raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET,
EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainPlEnOpusAcc.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_accelerate

DATASET = 'opus100'
EPOCH_NUM = 50
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-pl-en'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_accelerate(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET,
EPOCH_NUM, tokenize_help)

```

Training/TrainPlEnPara.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-pl-en'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'enpl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/TrainPlEnPara3.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'para_crawl'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-pl-en'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, 'enpl').filter(lambda example, idx: idx % 3 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')

```

```

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET + '-3', EPOCH_NUM, tokenize_help,
            max_length)

```

Training/TrainPlEnkde4.py

```
from datasets import load_dataset
from transformers import AutoTokenizer

from training import train_model

DATASET = 'kde4'
EPOCH_NUM = 100
max_length = 128
MODEL_NAME = 'Helsinki-NLP/opus-mt-pl-en'

model_name_cleaned = MODEL_NAME.replace('/', '-')

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


train_model(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help, max_length)

```

Training/training.py

```python
import torch
from accelerate import Accelerator
from huggingface_hub import login, get_full_repo_name, Repository
from torch.utils.data.dataloader import DataLoader
from tqdm import tqdm
from transformers import AutoModelForSeq2SeqLM, DataCollatorForSeq2Seq, Seq2SeqTrainingArguments, Seq2SeqTrainer, AdamW, \
    get_scheduler
import evaluate
import numpy as np


# the train function is adapted from the huggingface example https://huggingface.co/docs/transformers/tasks/translation
def train_model(raw_dataset, tokenizer, model_name, model_name_cleaned, dataset_name, epoch_num,
                tokenize_help, max_length, resume_from_checkpoint=False):
    global metric
    login('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')
    split_datasets = raw_dataset['train'].train_test_split(train_size=0.9, seed=20)
    split_datasets['validation'] = split_datasets.pop('test')
    tokenized_datasets = split_datasets.map(
        tokenize_help,
        batched=True,
        remove_columns=split_datasets['train'].column_names,
    )
    model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
    data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)
    metric = evaluate.load('sacrebleu')

    dataset_name = dataset_name.replace('/', '-')

    def compute_metrics(eval_preds):
        preds, labels = eval_preds
        # In case the model returns more than the prediction logits
        if isinstance(preds, tuple):
            preds = preds[0]

        decoded_preds = tokenizer.batch_decode(preds, skip_special_tokens=True)

        # Replace -100s in the labels as we can't decode them
        labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
        decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

        # Some simple post-processing
        decoded_preds = [pred.strip() for pred in decoded_preds]
        decoded_labels = [[label.strip()] for label in decoded_labels]

        result = metric.compute(predictions=decoded_preds, references=decoded_labels)
        return {'bleu': result['score']}

    args = Seq2SeqTrainingArguments(
        f'{model_name_cleaned}-{dataset_name}-finetune',
        evaluation_strategy='no',
        save_strategy='epoch',
        learning_rate=2e-5,
```

```
        per_device_train_batch_size=32,
        per_device_eval_batch_size=64,
        weight_decay=0.01,
        save_total_limit=2,
        num_train_epochs=epoch_num,
        predict_with_generate=True,
        fp16=True,
        push_to_hub=True,
    )
    trainer = Seq2SeqTrainer(
        model,
        args,
        train_dataset=tokenized_datasets['train'],
        eval_dataset=tokenized_datasets['validation'],
        data_collator=data_collator,
        tokenizer=tokenizer,
        compute_metrics=compute_metrics,
    )
    before_training = trainer.evaluate(max_length=max_length)
    file = open(dataset_name + '-' + str(epoch_num) + '-' + model_name_cleaned + '.txt', 'w')
    file.write(str(before_training))
    file.write('\n')
    file.close()

    trainer.train(resume_from_checkpoint=resume_from_checkpoint)
    trainer.push_to_hub()

    after_training = trainer.evaluate(max_length=max_length)
    file = open(dataset_name + '-' + str(epoch_num) + '-' + model_name_cleaned + '.txt', 'a')
    file.write(str(after_training))
    file.close()


# the accelerate function is adapted from the huggingface example
# https://huggingface.co/docs/accelerate/index
def train_accelerate(raw_dataset, tokenizer, model_name, model_name_cleaned, dataset_name, epoch_num,
                     tokenize_help):
    login('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')

    split_datasets = raw_dataset['train'].train_test_split(train_size=0.9, seed=20)

    split_datasets['validation'] = split_datasets.pop('test')

    tokenized_datasets = split_datasets.map(
        tokenize_help,
        batched=True,
        remove_columns=split_datasets['train'].column_names,
    )

    model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

    data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)

    metric = evaluate.load('sacrebleu')
```

```
108
109     dataset_name = dataset_name.replace('/', '-')
110
111     def compute_metrics(eval_preds):
112         preds, labels = eval_preds
113         # In case the model returns more than the prediction logits
114         if isinstance(preds, tuple):
115             preds = preds[0]
116
117         decoded_preds = tokenizer.batch_decode(preds, skip_special_tokens=True)
118
119         # Replace -100s in the labels as we can't decode them
120         labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
121         decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)
122
123         # Some simple post-processing
124         decoded_preds = [pred.strip() for pred in decoded_preds]
125         decoded_labels = [[label.strip()] for label in decoded_labels]
126
127         result = metric.compute(predictions=decoded_preds,
    references=decoded_labels)
128         return {'bleu': result['score']}
129
130     tokenized_datasets.set_format('torch')
131     train_dataloader = DataLoader(
132         tokenized_datasets['train'],
133         shuffle=True,
134         collate_fn=data_collator,
135         batch_size=8,
136     )
137     eval_dataloader = DataLoader(
138         tokenized_datasets['validation'], collate_fn=data_collator, batch_size=8
139     )
140
141     model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
142
143     optimizer = AdamW(model.parameters(), lr=2e-5)
144
145     accelerator = Accelerator()
146     model, optimizer, train_dataloader, eval_dataloader = accelerator.prepare(
147         model, optimizer, train_dataloader, eval_dataloader
148     )
149
150     num_update_steps_per_epoch = len(train_dataloader)
151     num_training_steps = epoch_num * num_update_steps_per_epoch
152
153     lr_scheduler = get_scheduler(
154         'linear',
155         optimizer=optimizer,
156         num_warmup_steps=0,
157         num_training_steps=num_training_steps,
158     )
159
160     model_name = f'{model_name_cleaned}-{dataset_name}-accelerate'
161     repo_name = get_full_repo_name(model_name)
162
163     output_dir = f'{model_name_cleaned}-{dataset_name}-accelerate'
```

```
    repo = Repository(output_dir, clone_from=repo_name)

    def postprocess(predictions, labels):
        predictions = predictions.cpu().numpy()
        labels = labels.cpu().numpy()

        decoded_preds = tokenizer.batch_decode(predictions, skip_special_tokens=True)

        # Replace -100 in the labels as we can't decode them.
        labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
        decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

        # Some simple post-processing
        decoded_preds = [pred.strip() for pred in decoded_preds]
        decoded_labels = [[label.strip()] for label in decoded_labels]
        return decoded_preds, decoded_labels

    progress_bar = tqdm(range(num_training_steps))

    for epoch in range(epoch_num):
        # Training
        model.train()
        for batch in train_dataloader:
            outputs = model(**batch)
            loss = outputs.loss
            accelerator.backward(loss)

            optimizer.step()
            lr_scheduler.step()
            optimizer.zero_grad()
            progress_bar.update(1)

        # Evaluation
        model.eval()
        for batch in tqdm(eval_dataloader):
            with torch.no_grad():
                generated_tokens = accelerator.unwrap_model(model).generate(
                    batch['input_ids'],
                    attention_mask=batch['attention_mask'],
                    max_length=128,
                )
            labels = batch['labels']

            # Necessary to pad predictions and labels for being gathered
            generated_tokens = accelerator.pad_across_processes(
                generated_tokens, dim=1, pad_index=tokenizer.pad_token_id
            )
            labels = accelerator.pad_across_processes(labels, dim=1, pad_index=-100)

            predictions_gathered = accelerator.gather(generated_tokens)
            labels_gathered = accelerator.gather(labels)

            decoded_preds, decoded_labels = postprocess(predictions_gathered, labels_gathered)
            metric.add_batch(predictions=decoded_preds, references=decoded_labels)

```

```
        results = metric.compute()

        print(f"epoch {epoch}, BLEU score: {results['score']:.2f}")
        file = open(dataset_name + '-' + str(epoch_num) + '-' + model_name_cleaned + '.txt', 'a')
        file.write(f"epoch {epoch}, BLEU score: {results['score']:.2f}")
        file.write('\n')
        file.close()

        # Save and upload
        accelerator.wait_for_everyone()
        unwrapped_model = accelerator.unwrap_model(model)
        unwrapped_model.save_pretrained(output_dir, save_function=accelerator.save)
        if accelerator.is_main_process:
            tokenizer.save_pretrained(output_dir)
            repo.push_to_hub(
                commit_message=f'Training in progress epoch {epoch}', blocking=False
            )

```

19 files

```
Hyper/hyperAlirezaEnPlOpus.py
Hyper/hyperAlirezaEnPlPara.py
Hyper/hyperAlirezaEnPlkde4.py
Hyper/hyperAlirezaPlEnOpus.py
Hyper/hyperAlirezaPlEnPara.py
Hyper/hyperAlirezaPlEnkde4.py
Hyper/hyperEnMulOpus.py
Hyper/hyperEnMulPara.py
Hyper/hyperEnMulkde4.py
Hyper/hyperEnPlOpus.py
Hyper/hyperEnPlPara.py
Hyper/hyperEnPlkde4.py
Hyper/hyperMulEnOpus.py
Hyper/hyperMulEnPara.py
Hyper/hyperMulEnkde4.py
Hyper/hyperPlEnOpus.py
Hyper/hyperPlEnPara.py
Hyper/hyperPlEnkde4.py
Hyper/hyperparamSearch.py
```

Hyper/hyperAlirezaEnPlOpus.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'opus100'
MODEL_NAME = 'alirezamsh/small100'

raw_dataset = load_dataset(DATASET, 'en-pl').filter(lambda example, idx: idx % 20 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='pl')

model_name_cleaned = MODEL_NAME.replace('/', '-') + 'en-pl'

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperAlirezaEnPlPara.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'para_crawl'
MODEL_NAME = 'alirezamsh/small100'

raw_dataset = load_dataset(DATASET, 'enpl').filter(lambda example, idx: idx % 50 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='pl')

model_name_cleaned = MODEL_NAME.replace('/', '-') + 'en-pl'

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperAlirezaEnPlkde4.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'kde4'
MODEL_NAME = 'alirezamsh/small100'

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='pl')

```

```
model_name_cleaned = MODEL_NAME.replace('/', '-') + 'en-pl'

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperAlirezaPlEnOpus.py

```
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'opus100'
MODEL_NAME = 'alirezamsh/small100'

raw_dataset = load_dataset(DATASET, 'en-pl').filter(lambda example, idx: idx % 20 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')

model_name_cleaned = MODEL_NAME.replace('/', '-') + 'pl-en'

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperAlirezaPlEnPara.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'para_crawl'
MODEL_NAME = 'alirezamsh/small100'

raw_dataset = load_dataset(DATASET, 'enpl').filter(lambda example, idx: idx % 50 == 0, with_indices=True)

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')

model_name_cleaned = MODEL_NAME.replace('/', '-') + 'pl-en'

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperAlirezaPlEnkde4.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'kde4'
MODEL_NAME = 'alirezamsh/small100'

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')

model_name_cleaned = MODEL_NAME.replace('/', '-') + 'pl-en'

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):

```

```
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperEnMulOpus.py

```
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'para_crawl'
MODEL_NAME = 'Helsinki-NLP/opus-mt-en-mul'

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')

model_name_cleaned = MODEL_NAME.replace('/', '-')

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperEnMulPara.py

```
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128
```

```

DATASET = 'para_crawl'
MODEL_NAME = 'Helsinki-NLP/opus-mt-en-mul'

raw_dataset = load_dataset(DATASET, 'enpl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')

model_name_cleaned = MODEL_NAME.replace('/', '-')

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperEnMulkde4.py

```
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'kde4'
MODEL_NAME = 'Helsinki-NLP/opus-mt-en-mul'

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')

model_name_cleaned = MODEL_NAME.replace('/', '-')

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned,
```

```
DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperEnPlOpus.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'opus100'
MODEL_NAME = 'gsarti/opus-mt-tc-en-pl'

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')

model_name_cleaned = MODEL_NAME.replace('/', '-')

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperEnPlPara.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'para_crawl'
MODEL_NAME = 'gsarti/opus-mt-tc-en-pl'

raw_dataset = load_dataset(DATASET, 'enpl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')

model_name_cleaned = MODEL_NAME.replace('/', '-')
```

```

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperEnPlkde4.py

```
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'kde4'
MODEL_NAME = 'gsarti/opus-mt-tc-en-pl'

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')

model_name_cleaned = MODEL_NAME.replace('/', '-')

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['en'] for ex in examples['translation']]
    targets = [ex['pl'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperMulEnOpus.py

```
from datasets import load_dataset
from transformers import AutoTokenizer
```

```
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'opus100'
MODEL_NAME = 'Helsinki-NLP/opus-mt-mul-en'

raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')

model_name_cleaned = MODEL_NAME.replace('/', '-')


# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperMulEnPara.py

```
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'para_crawl'
MODEL_NAME = 'Helsinki-NLP/opus-mt-mul-en'

raw_dataset = load_dataset(DATASET, 'enpl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')

model_name_cleaned = MODEL_NAME.replace('/', '-')

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
```

```
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperMulEnkde4.py

```
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'kde4'
MODEL_NAME = 'Helsinki-NLP/opus-mt-mul-en'

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt')

model_name_cleaned = MODEL_NAME.replace('/', '-')

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperPlEnOpus.py

```
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'opus100'
MODEL_NAME = 'Helsinki-NLP/opus-mt-pl-en'

```

```
raw_dataset = load_dataset(DATASET, 'en-pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')

model_name_cleaned = MODEL_NAME.replace('/', '-')

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperPlEnPara.py

```
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'para_crawl'
MODEL_NAME = 'Helsinki-NLP/opus-mt-pl-en'

raw_dataset = load_dataset(DATASET, 'enpl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')

model_name_cleaned = MODEL_NAME.replace('/', '-')

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperPlEnkde4.py

```python
from datasets import load_dataset
from transformers import AutoTokenizer
from hyperparamSearch import hyperparameter_search

EPOCH_NUM = 50
max_length = 128

DATASET = 'kde4'
MODEL_NAME = 'Helsinki-NLP/opus-mt-pl-en'

raw_dataset = load_dataset(DATASET, lang1='en', lang2='pl')

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, return_tensors='pt', tgt_lang='en')

model_name_cleaned = MODEL_NAME.replace('/', '-')

# The preprocessing function was adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def tokenize_help(examples):
    inputs = [ex['pl'] for ex in examples['translation']]
    targets = [ex['en'] for ex in examples['translation']]
    model_inputs = tokenizer(
        inputs, text_target=targets, max_length=max_length, truncation=True
    )
    return model_inputs


hyperparameter_search(raw_dataset, tokenizer, MODEL_NAME, model_name_cleaned, DATASET, EPOCH_NUM, tokenize_help)

```

Hyper/hyperparamSearch.py

```python
import evaluate
import numpy as np
from transformers import AutoModelForSeq2SeqLM, DataCollatorForSeq2Seq, Seq2SeqTrainingArguments, Seq2SeqTrainer
from huggingface_hub import login


# the beginning of the search function is adapted from the huggingface example
# https://huggingface.co/docs/transformers/tasks/translation
def hyperparameter_search(raw_dataset, tokenizer, model_name, model_name_cleaned, dataset_name, epoch_num,
                          tokenize_help):
    split_dataset = raw_dataset['train'].train_test_split(train_size=0.9, seed=20)

    split_dataset['validation'] = split_dataset.pop('test')

    login('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')

```

```
    def init_model(trial):
        return AutoModelForSeq2SeqLM.from_pretrained(model_name)

    global metric
    tokenized_datasets = split_dataset.map(
        tokenize_help,
        batched=True,
        remove_columns=split_dataset['train'].column_names,
    )
    model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
    data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)
    metric = evaluate.load('sacrebleu')

    def compute_metrics(eval_preds):
        preds, labels = eval_preds
        # In case the model returns more than the prediction logits
        if isinstance(preds, tuple):
            preds = preds[0]

        decoded_preds = tokenizer.batch_decode(preds, skip_special_tokens=True)

        # Replace -100s in the labels as we can't decode them
        labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
        decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

        # Some simple post-processing
        decoded_preds = [pred.strip() for pred in decoded_preds]
        decoded_labels = [[label.strip()] for label in decoded_labels]

        result = metric.compute(predictions=decoded_preds,
references=decoded_labels)
        return {'bleu': result['score']}

    args = Seq2SeqTrainingArguments(
        f'{model_name_cleaned}-{dataset_name}-finetune',
        evaluation_strategy='no',
        save_strategy='epoch',
        learning_rate=2e-5,
        per_device_train_batch_size=32,
        per_device_eval_batch_size=64,
        weight_decay=0.01,
        save_total_limit=3,
        num_train_epochs=epoch_num,
        predict_with_generate=True,
        fp16=True,
        push_to_hub=True,
    )
    trainer = Seq2SeqTrainer(
        model=None,
        args=args,
        train_dataset=tokenized_datasets['train'],
        eval_dataset=tokenized_datasets['validation'],
        data_collator=data_collator,
        tokenizer=tokenizer,
        compute_metrics=compute_metrics,
        model_init=init_model,
    )
```

```
    best_trial = trainer.hyperparameter_search(
        direction='maximize',
        backend='optuna',
        n_trials=10,
    )
    print(best_trial)
    # save to file
    with open(f'{model_name_cleaned}-{dataset_name}-hyper.txt', 'w') as f:
        f.write(str(best_trial))

```

16 files

```
Testing/TestAllModelsFromHub.py
Testing/TestModelEfficiency.py
Testing/TestModelScoresFlores.py
Testing/TestModelScoresFlores101.py
Testing/TestModelScoresFloresDev.py
Testing/TestModelScoresFloresTest.py
Testing/TestModelScoresIVA.py
Testing/TestModelScoresMix.py
Testing/TestModelScoresOpus.py
Testing/TestModelScoresTED14.py
Testing/TestModelScoresTED15.py
Testing/TestModelScoresTED16.py
Testing/TestModelScoresTatoeba.py
Testing/TestModelScoresTatoebaDev.py
Testing/TestModelScoresWMT.py
Testing/testing.py
```

Testing/TestAllModelsFromHub.py

```python
from transformers import pipeline

from testing import get_model_params

sentence_en_to_pl = '''Hello, it's me
I was wondering if after all these years you'd like to meet
To go over everything
They say that time's supposed to heal ya, but I ain't done much healing
Hello, can you hear me?
I'm in California dreaming about who we used to be
When we were younger and free
I've forgotten how it felt before the world fell at our feet
There's such a difference between us
And a million miles
Hello from the other side
I must've called a thousand times
To tell you I'm sorry for everything that I've done
But when I call, you never seem to be home'''.split('\n')

sentence_pl_to_en = '''Ktoś pogasił wszystkie światła
Świat się ugiął od zamieci
I pomimo wielkich chęci
Nie mam dobrych wieści
Chciałoby się uciec
Nie przed wszystkim się da
Idzie zima
Wiem, że nigdy nie jest łatwa
Niesie ciemny dzień przy sobie
Nieruchome myśli w głowie
Żywe są o Tobie
Byle śpiewać było o czym
Serce było wciąż gorące
Znowu krótsze będą noce
Zobaczymy słońce
```

```
Chciałoby się uciec
Nie przed wszystkim się da
Chciałoby się uciec
Nie przed wszystkim się da
Idzie zima
Idzie zima'''.split('\n')


def translate(sentence, model_name, tokenizer_to_use, source_lang=None, target_lang=None):
    translator = pipeline('translation', model=model_name, tokenizer=tokenizer_to_use)
    print(model_name)
    if source_lang is not None:
        results = translator(sentence, src_lang=source_lang, tgt_lang=target_lang)
    else:
        results = translator(sentence)
    for index, result in enumerate(results):
        print(result['translation_text'], ' -> ', sentence[index])
        file.write(f"{result['translation_text']} -> {sentence[index]}")


params = get_model_params()

# open file
with open('results-translation.txt', 'w') as file:
    for param in params:
        translate(param['model_name'], param['tokenizer_to_use'], param['source_lang'],
                  param['additional_model_data'])

```

Testing/TestModelEfficiency.py

```
import time
from transformers import pipeline
from testing import get_model_params


def time_model(model_name, tokenizer_to_use, source_lang=None, additional_model_data=None):
    second_lang = 'pl' if source_lang == 'en' else 'en'
    if additional_model_data is not None:
        pipeline_to_use = pipeline('translation', tokenizer=tokenizer_to_use, model=model_name, src_lang=source_lang,
                                   tgt_lang=second_lang)
    else:
        pipeline_to_use = pipeline('translation', tokenizer=tokenizer_to_use, model=model_name)

    # time the model for 10 characters, 100 characters, and 300 characters
    times = ''
    for c in [10, 100, 300]:
        # take an average of 10 runs
        runs = []
        for j in range(10):
```

```
            text = 'a' * c
            start = time.time()
            pipeline_to_use(text)
            end = time.time()
            runs.append(end - start)
        times += f'{sum(runs) / len(runs):.2f}, '

    print(f'{model_name}: {times}')
    with open('results-efficiency.txt', 'a') as file:
        file.write(f'{model_name}: {times}\n')


params = get_model_params()

for param in params:
    time_model(param['model_name'], param['tokenizer_to_use'], param['source_lang'], param['additional_model_data'])

```

Testing/TestModelScoresFlores.py

```
from datasets import load_dataset, concatenate_datasets
from testing import evaluate_models

polish = load_dataset('facebook/flores', 'pol_Latn')
polish_dev = polish['dev'].select_columns(['sentence'])
polish_test = polish['devtest'].select_columns(['sentence'])
english = load_dataset('facebook/flores', 'eng_Latn')
english_dev = english['dev'].select_columns(['sentence']).rename_column('sentence', 'reference')
english_test = english['devtest'].select_columns(['sentence']).rename_column('sentence', 'reference')
combined_dataset_dev = concatenate_datasets([english_dev, polish_dev], axis=1)
combined_dataset_test = concatenate_datasets([english_test, polish_test], axis=1)
combined_dataset = concatenate_datasets([combined_dataset_dev, combined_dataset_test], axis=0)

pl_en_dataset = combined_dataset.rename_column('sentence', 'source')

en_pl_dataset = combined_dataset.rename_column('reference', 'source')
en_pl_dataset = en_pl_dataset.rename_column('sentence', 'reference')

evaluate_models(en_pl_dataset, pl_en_dataset, 'results-flores.txt')

```

Testing/TestModelScoresFlores101.py

```
from datasets import load_dataset, concatenate_datasets
from testing import evaluate_models

polish = load_dataset('gsarti/flores_101', 'pol')
polish_dev = polish['dev'].select_columns(['sentence'])
polish_test = polish['devtest'].select_columns(['sentence'])
```

```
english = load_dataset('gsarti/flores_101', 'eng')
english_dev = english['dev'].select_columns(['sentence']).rename_column('sentence', 'reference')
english_test = english['devtest'].select_columns(['sentence']).rename_column('sentence', 'reference')
combined_dataset_dev = concatenate_datasets([english_dev, polish_dev], axis=1)
combined_dataset_test = concatenate_datasets([english_test, polish_test], axis=1)
combined_dataset = concatenate_datasets([combined_dataset_dev, combined_dataset_test], axis=0)

pl_en_dataset = combined_dataset.rename_column('sentence', 'source')

en_pl_dataset = combined_dataset.rename_column('reference', 'source')
en_pl_dataset = en_pl_dataset.rename_column('sentence', 'reference')

evaluate_models(en_pl_dataset, pl_en_dataset, 'results-flores101.txt')

```

Testing/TestModelScoresFloresDev.py

```
from datasets import load_dataset, concatenate_datasets
from testing import evaluate_models

polish = load_dataset('facebook/flores', 'pol_Latn')
polish_dev = polish['dev'].select_columns(['sentence'])
english = load_dataset('facebook/flores', 'eng_Latn')
english_dev = english['dev'].select_columns(['sentence']).rename_column('sentence', 'reference')
combined_dataset_dev = concatenate_datasets([english_dev, polish_dev], axis=1)

pl_en_dataset_dev = combined_dataset_dev.rename_column('sentence', 'source')

en_pl_dataset_dev = combined_dataset_dev.rename_column('reference', 'source')
en_pl_dataset_dev = en_pl_dataset_dev.rename_column('sentence', 'reference')

evaluate_models(en_pl_dataset_dev, pl_en_dataset_dev, 'results-flores-dev.txt')

```

Testing/TestModelScoresFloresTest.py

```
from datasets import load_dataset, concatenate_datasets
from testing import evaluate_models

polish = load_dataset('facebook/flores', 'pol_Latn')
polish_test = polish['devtest'].select_columns(['sentence'])
english = load_dataset('facebook/flores', 'eng_Latn')
english_test = english['devtest'].select_columns(['sentence']).rename_column('sentence', 'reference')
combined_dataset_test = concatenate_datasets([english_test, polish_test], axis=1)

pl_en_dataset_test = combined_dataset_test.rename_column('sentence', 'source')

```

```
en_pl_dataset_test = combined_dataset_test.rename_column('reference', 'source')
en_pl_dataset_test = en_pl_dataset_test.rename_column('sentence', 'reference')

evaluate_models(en_pl_dataset_test, pl_en_dataset_test, 'results-flores-test.txt')

```

Testing/TestModelScoresIVA.py

```
from datasets import load_dataset
from testing import evaluate_models

raw_dataset = load_dataset('cartesinus/iva_mt_wslot', split='test').flatten()
raw_dataset = raw_dataset.select_columns(['translation_utt.en',
'translation_utt.pl'])

en_pl_dataset = raw_dataset.rename_column('translation_utt.en', 'source')
en_pl_dataset = en_pl_dataset.rename_column('translation_utt.pl', 'reference')

pl_en_dataset = raw_dataset.rename_column('translation_utt.pl', 'source')
pl_en_dataset = pl_en_dataset.rename_column('translation_utt.en', 'reference')

evaluate_models(en_pl_dataset, pl_en_dataset, 'results-iva.txt')

```

Testing/TestModelScoresMix.py

```
from datasets import load_dataset, concatenate_datasets, Dataset
from testing import evaluate_models

kde4 = load_dataset('kde4', lang1='en', lang2='pl',
split='train').select(range(100))
kde4 = kde4.flatten().select_columns(['translation.en', 'translation.pl'])
para_crawl = load_dataset('para_crawl', 'enpl', split='train').select(range(1300))
para_crawl = para_crawl.flatten().select_columns(['translation.en',
'translation.pl'])
opus100 = load_dataset('opus100', 'en-pl', split='test').select(range(1300))
opus100 = opus100.flatten().select_columns(['translation.en', 'translation.pl'])
ccmatrix = Dataset.from_list(list(load_dataset('yhavinga/ccmatrix', 'en-pl',
split='train', streaming=True).take(1300)))
ccmatrix = ccmatrix.flatten().select_columns(['translation.en', 'translation.pl'])

raw_dataset = concatenate_datasets([kde4, para_crawl, opus100, ccmatrix])

en_pl_dataset = raw_dataset.rename_column('translation.en', 'source')
en_pl_dataset = en_pl_dataset.rename_column('translation.pl', 'reference')

pl_en_dataset = raw_dataset.rename_column('translation.pl', 'source')
pl_en_dataset = pl_en_dataset.rename_column('translation.en', 'reference')

evaluate_models(en_pl_dataset, pl_en_dataset, 'results-mix.txt')

```

Testing/TestModelScoresOpus.py

```
from datasets import load_dataset
from testing import evaluate_models

raw_dataset = load_dataset('opus_euconst', 'en-pl', split='train').flatten()

en_pl_dataset = raw_dataset.rename_column('translation.en', 'source')
en_pl_dataset = en_pl_dataset.rename_column('translation.pl', 'reference')

pl_en_dataset = raw_dataset.rename_column('translation.pl', 'source')
pl_en_dataset = pl_en_dataset.rename_column('translation.en', 'reference')

evaluate_models(en_pl_dataset, pl_en_dataset, 'results-euconst.txt')

```

Testing/TestModelScoresTED14.py

```
from datasets import load_dataset
from testing import evaluate_models

raw_dataset = load_dataset('ted_talks_iwslt', language_pair=('en', 'pl'), year='2014', split='test').flatten()

en_pl_dataset = raw_dataset.rename_column('translation.en', 'source')
en_pl_dataset = en_pl_dataset.rename_column('translation.pl', 'reference')

pl_en_dataset = raw_dataset.rename_column('translation.pl', 'source')
pl_en_dataset = pl_en_dataset.rename_column('translation.en', 'reference')

evaluate_models(en_pl_dataset, pl_en_dataset, 'results-ted-14.txt')

```

Testing/TestModelScoresTED15.py

```
from datasets import load_dataset
from testing import evaluate_models

raw_dataset = load_dataset('ted_talks_iwslt', language_pair=('en', 'pl'), year='2015', split='test').flatten()

en_pl_dataset = raw_dataset.rename_column('translation.en', 'source')
en_pl_dataset = en_pl_dataset.rename_column('translation.pl', 'reference')

pl_en_dataset = raw_dataset.rename_column('translation.pl', 'source')
pl_en_dataset = pl_en_dataset.rename_column('translation.en', 'reference')

evaluate_models(en_pl_dataset, pl_en_dataset, 'results-ted-15.txt')

```

Testing/TestModelScoresTED16.py

```
from datasets import load_dataset
from testing import evaluate_models

raw_dataset = load_dataset('ted_talks_iwslt', language_pair=('en', 'pl'), year='2016', split='test').flatten()

en_pl_dataset = raw_dataset.rename_column('translation.en', 'source')
en_pl_dataset = en_pl_dataset.rename_column('translation.pl', 'reference')

pl_en_dataset = raw_dataset.rename_column('translation.pl', 'source')
pl_en_dataset = pl_en_dataset.rename_column('translation.en', 'reference')

evaluate_models(en_pl_dataset, pl_en_dataset, 'results-ted-16.txt')

```

Testing/TestModelScoresTatoeba.py

```
from datasets import load_dataset
from testing import evaluate_models

raw_dataset = load_dataset('Helsinki-NLP/tatoeba_mt', 'eng-pol', split='test').select(range(4000))

pl_en_dataset = raw_dataset.rename_column('sourceString', 'reference')
pl_en_dataset = pl_en_dataset.rename_column('targetString', 'source')

en_pl_dataset = raw_dataset.rename_column('targetString', 'reference')
en_pl_dataset = en_pl_dataset.rename_column('sourceString', 'source')

evaluate_models(en_pl_dataset, pl_en_dataset, 'results-tatoeba.txt')

```

Testing/TestModelScoresTatoebaDev.py

```
from datasets import load_dataset
from testing import evaluate_models

raw_dataset = load_dataset('Helsinki-NLP/tatoeba_mt', 'eng-pol', split='dev').select(range(4000))

pl_en_dataset = raw_dataset.rename_column('sourceString', 'reference')
pl_en_dataset = pl_en_dataset.rename_column('targetString', 'source')

en_pl_dataset = raw_dataset.rename_column('targetString', 'reference')
en_pl_dataset = en_pl_dataset.rename_column('sourceString', 'source')

evaluate_models(en_pl_dataset, pl_en_dataset, 'results-tatoeba-dev.txt')

```

Testing/TestModelScoresWMT.py

```python
from datasets import load_dataset
from testing import evaluate_models

en_pl_dataset = load_dataset('gsarti/wmt_vat', 'wmt20_en_pl', split='test')
pl_en_dataset = load_dataset('gsarti/wmt_vat', 'wmt20_pl_en', split='test')

evaluate_models(en_pl_dataset, pl_en_dataset, 'results-wmt.txt')

```

Testing/testing.py

```python
import evaluate
from transformers import pipeline


def evaluate_models(dataset_en_pl, dataset_pl_en, filename):
    model_params = get_model_params()
    for param in model_params:
        evaluate_model(param['model_name'], param['tokenizer_to_use'], filename, dataset_pl_en, dataset_en_pl,
                       param['source_lang'], param['additional_model_data'])


def evaluate_model(model_name, tokenizer_to_use, filename, pl_en_dataset, en_pl_dataset, source_lang=None,
                   additional_model_data=None):
    second_lang = 'pl' if source_lang == 'en' else 'en'
    if source_lang == 'pl':
        dataset = pl_en_dataset
    else:
        dataset = en_pl_dataset
    if additional_model_data is not None:
        pipeline_to_use = pipeline('translation', tokenizer=tokenizer_to_use, model=model_name, src_lang=source_lang,
                                   tgt_lang=second_lang)
    else:
        pipeline_to_use = pipeline('translation', tokenizer=tokenizer_to_use, model=model_name)

    task_evaluator = evaluate.evaluator('translation')

    with open(filename, 'a') as file:
        file.write('Evaluating model: ' + model_name + '\n')
    print('Evaluating model: ' + model_name)

    results = get_results(task_evaluator, pipeline_to_use, dataset)

    with open(filename, 'a') as file:
        file.write('Results ' + str(results['score']) + '\n')
    print('Results ' + str(results['score']))


```

```
def get_results(task_evaluator_to_use, pipeline_to_use, dataset_to_use):
    results = task_evaluator_to_use.compute(
        model_or_pipeline=pipeline_to_use,
        data=dataset_to_use,
        input_column='source',
        label_column='reference',
        metric='sacrebleu'
    )

    return results


def get_model_params():
    return [
        {'model_name': 'gsarti/opus-mt-tc-en-pl', 'tokenizer_to_use': 'gsarti/opus-mt-tc-en-pl', 'source_lang': 'en',
         'additional_model_data': None},
        {'model_name': 'MikolajDeja/gsarti-opus-mt-tc-en-pl-kde4-finetune',
         'tokenizer_to_use': 'gsarti/opus-mt-tc-en-pl',
         'source_lang': 'en', 'additional_model_data': None},
        {'model_name': 'MikolajDeja/gsarti-opus-mt-tc-en-pl-opus100-finetune',
         'tokenizer_to_use': 'gsarti/opus-mt-tc-en-pl',
         'source_lang': 'en', 'additional_model_data': None},
        {'model_name': 'MikolajDeja/gsarti-opus-mt-tc-en-pl-3-para_crawl-finetune',
         'tokenizer_to_use': 'gsarti/opus-mt-tc-en-pl', 'source_lang': 'en', 'additional_model_data': None},
        {'model_name': 'MikolajDeja/gsarti-opus-mt-tc-en-pl-para_crawl-finetune',
         'tokenizer_to_use': 'gsarti/opus-mt-tc-en-pl', 'source_lang': 'en', 'additional_model_data': None},
        {'model_name': 'MikolajDeja/gsarti-opus-mt-tc-en-pl-yhavinga-ccmatrix-finetune',
         'tokenizer_to_use': 'gsarti/opus-mt-tc-en-pl', 'source_lang': 'en', 'additional_model_data': None},

        {'model_name': 'Helsinki-NLP/opus-mt-pl-en', 'tokenizer_to_use': 'Helsinki-NLP/opus-mt-pl-en',
         'source_lang': 'pl',
         'additional_model_data': None},
        {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-pl-en-kde4-finetune',
         'tokenizer_to_use': 'Helsinki-NLP/opus-mt-pl-en', 'source_lang': 'pl',
         'additional_model_data': None},
        {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-pl-en-opus100-finetune',
         'tokenizer_to_use': 'Helsinki-NLP/opus-mt-pl-en', 'source_lang': 'pl',
         'additional_model_data': None},
        {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-pl-en-3-para_crawl-finetune',
         'tokenizer_to_use': 'Helsinki-NLP/opus-mt-pl-en', 'source_lang': 'pl',
         'additional_model_data': None},
        {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-pl-en-para_crawl-finetune',
         'tokenizer_to_use': 'Helsinki-NLP/opus-mt-pl-en', 'source_lang': 'pl',
         'additional_model_data': None},
        {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-pl-en-yhavinga-ccmatrix-finetune',
         'tokenizer_to_use': 'Helsinki-NLP/opus-mt-pl-en', 'source_lang': 'pl',
         'additional_model_data': None},

        {'model_name': 'Helsinki-NLP/opus-mt-en-mul',
         'tokenizer_to_use': 'Helsinki-NLP/opus-mt-en-mul', 'source_lang': 'en',
```

```
 88          'additional_model_data': None},
 89         {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-en-mul-kde4-finetune',
 90          'tokenizer_to_use': 'Helsinki-NLP/opus-mt-en-mul', 'source_lang': 'en',
 91          'additional_model_data': None},
 92         {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-en-mul-opus100-finetune',
 93          'tokenizer_to_use': 'Helsinki-NLP/opus-mt-en-mul', 'source_lang': 'en',
 94          'additional_model_data': None},
 95         {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-en-mul-3-para_crawl-
finetune',
 96          'tokenizer_to_use': 'Helsinki-NLP/opus-mt-en-mul', 'source_lang': 'en',
 97          'additional_model_data': None},
 98         {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-en-mul-para_crawl-
finetune',
 99          'tokenizer_to_use': 'Helsinki-NLP/opus-mt-en-mul', 'source_lang': 'en',
100          'additional_model_data': None},
101         {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-en-mul-yhavinga-ccmatrix-
finetune',
102          'tokenizer_to_use': 'Helsinki-NLP/opus-mt-en-mul', 'source_lang': 'en',
103          'additional_model_data': None},
104
105         {'model_name': 'Helsinki-NLP/opus-mt-mul-en',
106          'tokenizer_to_use': 'Helsinki-NLP/opus-mt-mul-en', 'source_lang': 'pl',
107          'additional_model_data': None},
108         {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-mul-en-kde4-finetune',
109          'tokenizer_to_use': 'Helsinki-NLP/opus-mt-mul-en', 'source_lang': 'pl',
110          'additional_model_data': None},
111         {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-mul-en-opus100-finetune',
112          'tokenizer_to_use': 'Helsinki-NLP/opus-mt-mul-en', 'source_lang': 'pl',
113          'additional_model_data': None},
114         {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-mul-en-3-para_crawl-
finetune',
115          'tokenizer_to_use': 'Helsinki-NLP/opus-mt-mul-en', 'source_lang': 'pl',
116          'additional_model_data': None},
117         {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-mul-en-para_crawl-
finetune',
118          'tokenizer_to_use': 'Helsinki-NLP/opus-mt-mul-en', 'source_lang': 'pl',
119          'additional_model_data': None},
120         {'model_name': 'MikolajDeja/Helsinki-NLP-opus-mt-mul-en-yhavinga-ccmatrix-
finetune',
121          'tokenizer_to_use': 'Helsinki-NLP/opus-mt-mul-en', 'source_lang': 'pl',
122          'additional_model_data': None},
123
124         {'model_name': 'alirezamsh/small100', 'tokenizer_to_use':
'alirezamsh/small100', 'source_lang': 'pl',
125          'additional_model_data': 'pl'},
126         {'model_name': 'MikolajDeja/alirezamsh-small100-pl-en-kde4-finetune',
'tokenizer_to_use': 'alirezamsh/small100',
127          'source_lang': 'pl', 'additional_model_data': 'pl'},
128         {'model_name': 'MikolajDeja/alirezamsh-small100-pl-en-opus100-finetune',
129          'tokenizer_to_use': 'alirezamsh/small100', 'source_lang': 'pl',
'additional_model_data': 'pl'},
130         {'model_name': 'MikolajDeja/alirezamsh-small100-pl-en-3-para_crawl-
finetune',
131          'tokenizer_to_use': 'alirezamsh/small100', 'source_lang': 'pl',
'additional_model_data': 'pl'},
132         {'model_name': 'MikolajDeja/alirezamsh-small100-pl-en-para_crawl-
finetune',
133          'tokenizer_to_use': 'alirezamsh/small100', 'source_lang': 'pl',
'additional_model_data': 'pl'},
```

```
        {'model_name': 'MikolajDeja/alirezamsh-small100-pl-en-yhavinga-ccmatrix-finetune',
         'tokenizer_to_use': 'alirezamsh/small100', 'source_lang': 'pl', 'additional_model_data': 'pl'},

        {'model_name': 'alirezamsh/small100', 'tokenizer_to_use': 'alirezamsh/small100', 'source_lang': 'en',
         'additional_model_data': 'en'},
        {'model_name': 'MikolajDeja/alirezamsh-small100-en-pl-kde4-finetune', 'tokenizer_to_use': 'alirezamsh/small100',
         'source_lang': 'en', 'additional_model_data': 'en'},
        {'model_name': 'MikolajDeja/alirezamsh-small100-en-pl-opus100-finetune',
         'tokenizer_to_use': 'alirezamsh/small100', 'source_lang': 'en', 'additional_model_data': 'en'},
        {'model_name': 'MikolajDeja/alirezamsh-small100-en-pl-3-para_crawl-finetune',
         'tokenizer_to_use': 'alirezamsh/small100', 'source_lang': 'en', 'additional_model_data': 'en'},
        {'model_name': 'MikolajDeja/alirezamsh-small100-en-pl-para_crawl-finetune',
         'tokenizer_to_use': 'alirezamsh/small100', 'source_lang': 'en', 'additional_model_data': 'en'},
        {'model_name': 'MikolajDeja/alirezamsh-small100-en-pl-yhavinga-ccmatrix-finetune',
         'tokenizer_to_use': 'alirezamsh/small100', 'source_lang': 'en', 'additional_model_data': 'en'},

        {'model_name': 'facebook/nllb-200-distilled-600M', 'tokenizer_to_use': 'facebook/nllb-200-distilled-600M',
         'source_lang': 'pl', 'additional_model_data': 'pl'},
        {'model_name': 'MikolajDeja/facebook-nllb-200-distilled-600M-pl-en-opus-finetune',
         'tokenizer_to_use': 'facebook/nllb-200-distilled-600M', 'source_lang': 'pl', 'additional_model_data': 'pl'},
        {'model_name': 'MikolajDeja/facebook-nllb-200-distilled-600M-pl-en-3-para_crawl-finetune',
         'tokenizer_to_use': 'facebook/nllb-200-distilled-600M', 'source_lang': 'pl', 'additional_model_data': 'pl'},
        {'model_name': 'MikolajDeja/facebook-nllb-200-distilled-600M-pl-en-yhavinga-ccmatrix-finetune',
         'tokenizer_to_use': 'facebook/nllb-200-distilled-600M', 'source_lang': 'pl', 'additional_model_data': 'pl'},

        {'model_name': 'facebook/nllb-200-distilled-600M', 'tokenizer_to_use': 'facebook/nllb-200-distilled-600M',
         'source_lang': 'en', 'additional_model_data': 'en'},
        {'model_name': 'MikolajDeja/facebook-nllb-200-distilled-600M-en-pl-opus-finetune',
         'tokenizer_to_use': 'facebook/nllb-200-distilled-600M', 'source_lang': 'en', 'additional_model_data': 'en'},
        {'model_name': 'MikolajDeja/facebook-nllb-200-distilled-600M-en-pl-3-para_crawl-finetune',
         'tokenizer_to_use': 'facebook/nllb-200-distilled-600M', 'source_lang': 'en', 'additional_model_data': 'en'},
        {'model_name': 'MikolajDeja/facebook-nllb-200-distilled-600M-en-pl-yhavinga-ccmatrix-finetune',
         'tokenizer_to_use': 'facebook/nllb-200-distilled-600M', 'source_lang': 'en', 'additional_model_data': 'en'},
    ]

```

4 files

```
HelperScripts/acc.sh
HelperScripts/hyper.sh
HelperScripts/test.sh
HelperScripts/train.sh
```

HelperScripts/acc.sh

```
#!/bin/bash -l

#SBATCH --job-name=translation-training
#SBATCH --partition=nmes_gpu
#SBATCH --gres=gpu
#SBATCH --signal=USR2
#SBATCH --time=48:00:00
#SBATCH --mem=10240

module load anaconda3/2021.05-gcc-9.4.0
ml test_switch_kcl
source test_switch
ml git-lfs


source /users/${USER}/.bashrc
source activate /scratch/users/${USER}/conda/torch-env

pip install accelerate
pip install tqdm

python scripts/TrainAlirezaEnPlkde4.py
python scripts/TrainAlirezaPlEnkde4.py
python scripts/TrainEnMulkde4.py
python scripts/TrainEnPlkde4.py
python scripts/TrainMulEnkde4.py
python scripts/TrainPlEnkde4.py

python scripts/TrainAlirezaEnPlOpusAcc.py
python scripts/TrainAlirezaPlEnOpusAcc.py
python scripts/TrainPlEnOpusAcc.py
python scripts/TrainEnPlOpusAcc.py
python scripts/TrainEnMulOpusAcc.py
python scripts/TrainMulEnOpusAcc.py

python scripts/TrainAlirezaEnPlParaCrawl.py
python scripts/TrainAlirezaPlEnParaCrawl.py
python scripts/TrainPlEnPara.py
python scripts/TrainEnPlPara.py
python scripts/TrainEnMulPara.py
python scripts/TrainMulEnPara.py

python scripts/TrainAlirezaEnPlParaCrawl3.py
python scripts/TrainAlirezaPlEnParaCrawl3.py
```

```
python scripts/TrainPlEnPara3.py
python scripts/TrainEnPlPara3.py
python scripts/TrainEnMulPara3.py
python scripts/TrainMulEnPara3.py
```

HelperScripts/hyper.sh

```
#!/bin/bash -l

#SBATCH --job-name=hyperparam
#SBATCH --partition=nmes_gpu
#SBATCH --gres=gpu
#SBATCH --signal=USR2
#SBATCH --time=48:00:00
#SBATCH --mem=40960

module load anaconda3/2021.05-gcc-9.4.0

source /users/${USER}/.bashrc
source activate /scratch/users/${USER}/conda/torch-env


python scripts/hyperEnPlkde4.py
python scripts/hyperEnPlopus.py
python scripts/hyperEnPlpara.py
python scripts/hyperPlEnkde4.py
python scripts/hyperPlEnOpus.py
python scripts/hyperPlEnPara.py

python scripts/hyperEnMulkde4.py
python scripts/hyperEnMulOpus.py
python scripts/hyperEnMulPara.py

python scripts/hyperMulEnkde4.py
python scripts/hyperMulEnOpus.py
python scripts/hyperMulEnPara.py

```

HelperScripts/test.sh

```
#!/bin/bash -l

#SBATCH --job-name=testing
#SBATCH --partition=nmes_gpu
#SBATCH --gres=gpu
#SBATCH --signal=USR2
#SBATCH --time=48:00:00
#SBATCH --mem=10240

module load anaconda3/2021.05-gcc-9.4.0

source /users/${USER}/.bashrc
```

```
source activate /scratch/users/${USER}/conda/torch-env

pip install scipy>1.17
pip install --upgrade datasets

export TRANSFORMERS_CACHE=/scratch/users/k20010020/cache/

python scripts/TestModelScoresWMT.py
python scripts/TestModelScoresOpus.py
python scripts/TestModelScoresTatoeba.py
python scripts/TestModelScoresMix.py
python scripts/TestModelEfficiency.py
python scripts/TestModelScoresFlores.py
python scripts/TestModelScoresFloresTwo.py
python scripts/TestModelScoresFlores101.py
python scripts/TestModelScoresFlores101Two.py
python scripts/TestModelScoresOpus100.py
python scripts/TestModelScoresIVA.py
python scripts/TestModelScoresTED14.py
python scripts/TestModelScoresTED15.py
python scripts/TestModelScoresTED16.py
```

HelperScripts/train.sh

```
#!/bin/bash -l

#SBATCH --job-name=training
#SBATCH --partition=nmes_gpu
#SBATCH --gres=gpu
#SBATCH --signal=USR2
#SBATCH --time=48:00:00
#SBATCH --mem=40960

module load anaconda3/2021.05-gcc-9.4.0
ml test_switch_kcl
source test_switch
ml git-lfs


source /users/${USER}/.bashrc
source activate /scratch/users/${USER}/conda/torch-env

python scripts/TrainNLLBEnPlPara.py
python scripts/TrainNLLBEnPlOpus.py
python scripts/TrainNLLBEnPlPara3.py
python scripts/TrainNLLBPlEnPara3.py
python scripts/TrainNLLBPlEnOpus.py
python scripts/TrainNLLBEnPlCC.py
python scripts/TrainNLLBPlEnCC.py


python scripts/TrainAlirezaEnPlCC.py
python scripts/TrainAlirezaPlEnCC.py
python scripts/TrainEnPlCC.py
python scripts/TrainPlEnCC.py
```

```
python scripts/TrainEnMulCC.py
python scripts/TrainMulEnCC.py

python scripts/TrainAlirezaEnPlkde4.py
python scripts/TrainAlirezaPlEnkde4.py
python scripts/TrainEnMulkde4.py
python scripts/TrainEnPlkde4.py
python scripts/TrainMulEnkde4.py
python scripts/TrainPlEnkde4.py

python scripts/TrainAlirezaEnPlOpus100.py
python scripts/TrainAlirezaPlEnOpus100.py
python scripts/TrainPlEnOpus100.py
python scripts/TrainEnPlOpus100.py
python scripts/TrainEnMulOpus100.py
python scripts/TrainMulEnOpus100.py

python scripts/TrainAlirezaEnPlParaCrawl.py
python scripts/TrainAlirezaPlEnParaCrawl.py
python scripts/TrainPlEnPara.py
python scripts/TrainEnPlPara.py
python scripts/TrainEnMulPara.py
python scripts/TrainMulEnPara.py

python scripts/TrainAlirezaEnPlParaCrawl3.py
python scripts/TrainAlirezaPlEnParaCrawl3.py
python scripts/TrainPlEnPara3.py
python scripts/TrainEnPlPara3.py
python scripts/TrainEnMulPara3.py
python scripts/TrainMulEnPara3.py
```

60 files

```
Translator/.github/workflows/django.yml
Translator/.gitignore
Translator/README.md
Translator/manage.py
Translator/requirements.txt
Translator/static/custom.css
Translator/static/favicon/browserconfig.xml
Translator/static/favicon/safari-pinned-tab.svg
Translator/static/favicon/site.webmanifest
Translator/translate/__init__.py
Translator/translate/__pycache__/views.cpython-39.pyc.140357439501584
Translator/translate/admin.py
Translator/translate/apps.py
Translator/translate/forms.py
Translator/translate/migrations/0001_initial.py
Translator/translate/migrations/0002_translation.py
Translator/translate/migrations/0003_alter_translation_user.py
Translator/translate/migrations/0004_alter_translation_user.py
Translator/translate/migrations/0005_translation_input_language.py
Translator/translate/migrations/0006_remove_translation_created_at_and_more.py
Translator/translate/migrations/0007_translation_created_at_translation_updated_at.py
Translator/translate/migrations/__init__.py
Translator/translate/models.py
Translator/translate/templates/base.html
Translator/translate/templates/base_content.html
Translator/translate/templates/home.html
Translator/translate/templates/login.html
Translator/translate/templates/partials/bootstrap_form.html
Translator/translate/templates/partials/messages.html
Translator/translate/templates/partials/navbar.html
Translator/translate/templates/password_change.html
Translator/translate/templates/profile.html
Translator/translate/templates/signup.html
Translator/translate/tests/__init__.py
Translator/translate/tests/fixtures/default_translation.json
Translator/translate/tests/fixtures/default_user.json
Translator/translate/tests/fixtures/other_translations.json
Translator/translate/tests/fixtures/other_users.json
Translator/translate/tests/forms/__init__.py
Translator/translate/tests/forms/test_log_in_form.py
Translator/translate/tests/forms/test_sign_up_form.py
Translator/translate/tests/forms/test_translator_form.py
Translator/translate/tests/helpers.py
Translator/translate/tests/models/__init__.py
Translator/translate/tests/models/test_translation_model.py
Translator/translate/tests/models/test_user_model.py
Translator/translate/tests/views/__init__.py
Translator/translate/tests/views/test_change_password_view.py
Translator/translate/tests/views/test_home_view.py
Translator/translate/tests/views/test_log_in_view.py
Translator/translate/tests/views/test_logout_view.py
Translator/translate/tests/views/test_profile_view.py
Translator/translate/tests/views/test_sign_up_view.py
Translator/translate/views.py
Translator/translator/__init__.py
Translator/translator/asgi.py
Translator/translator/settings.py
Translator/translator/urls.py
Translator/translator/wsgi.py
Translator/translator_backend.py
```

Translator/.github/workflows/django.yml

```yaml
name: Django CI

on:
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]

jobs:
  build:

    runs-on: ubuntu-latest
    strategy:
      max-parallel: 4
      matrix:
        python-version: [3.8, 3.9]

```

```
    steps:
    - uses: actions/checkout@v3
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v3
      with:
        python-version: ${{ matrix.python-version }}
    - name: Install Dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt
    - name: Run Tests
      run: |
        python manage.py test

```

Translator/.gitignore

```
# Created by https://www.toptal.com/developers/gitignore/api/macos,pycharm,jetbrains,django,python
# Edit at https://www.toptal.com/developers/gitignore?templates=macos,pycharm,jetbrains,django,python

### Django ###
*.log
*.pot
*.pyc
__pycache__/
local_settings.py
db.sqlite3
db.sqlite3-journal
media

# If your build process includes running collectstatic, then you probably don't need or want to include staticfiles/
# in your Git repository. Update and uncomment the following line accordingly.
# <django-project-name>/staticfiles/

### Django.Python Stack ###
# Byte-compiled / optimized / DLL files
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST

# PyInstaller
#  Usually these files are written by a python script from a template
#  before PyInstaller builds the exe, so as to inject date/other infos into it.
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.nox/
.coverage
```

```
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
*.py,cover
.hypothesis/
.pytest_cache/
cover/

# Translations
*.mo

# Django stuff:

# Flask stuff:
instance/
.webassets-cache

# Scrapy stuff:
.scrapy

# Sphinx documentation
docs/_build/

# PyBuilder
.pybuilder/
target/

# Jupyter Notebook
.ipynb_checkpoints

# IPython
profile_default/
ipython_config.py

# pyenv
#   For a library or package, you might want to ignore these files since the code is
#   intended to run in multiple environments; otherwise, check them in:
# .python-version

# pipenv
#   According to pypa/pipenv#598, it is recommended to include Pipfile.lock in version control.
#   However, in case of collaboration, if having platform-specific dependencies or dependencies
#   having no cross-platform support, pipenv may install dependencies that don't work, or not
#   install all needed dependencies.
#Pipfile.lock

# poetry
#   Similar to Pipfile.lock, it is generally recommended to include poetry.lock in version control.
#   This is especially recommended for binary packages to ensure reproducibility, and is more
#   commonly ignored for libraries.
#   https://python-poetry.org/docs/basic-usage/#commit-your-poetrylock-file-to-version-control
#poetry.lock

# pdm
#   Similar to Pipfile.lock, it is generally recommended to include pdm.lock in version control.
#pdm.lock
#   pdm stores project-wide configurations in .pdm.toml, but it is recommended to not include it
#   in version control.
#   https://pdm.fming.dev/#use-with-ide
.pdm.toml

# PEP 582; used by e.g. github.com/David-OConnor/pyflow and github.com/pdm-project/pdm
__pypackages__/

# Celery stuff
celerybeat-schedule
celerybeat.pid

# SageMath parsed files
*.sage.py

# Environments
.env
.venv
env/
venv/
ENV/
```

```
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
#  JetBrains specific template is maintained in a separate JetBrains.gitignore that can
#  be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
#  and can be added to the global gitignore or merged into this file.  For a more nuclear
#  option (not recommended) you can uncomment the following to ignore the entire idea folder.
#.idea/

### JetBrains ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml

# Generated files
.idea/**/contentModel.xml

# Sensitive or high-churn files
.idea/**/dataSources/
.idea/**/dataSources.ids
.idea/**/dataSources.local.xml
.idea/**/sqlDataSources.xml
.idea/**/dynamic.xml
.idea/**/uiDesigner.xml
.idea/**/dbnavigator.xml

# Gradle
.idea/**/gradle.xml
.idea/**/libraries

# Gradle and Maven with auto-import
# When using Gradle or Maven with auto-import, you should exclude module files,
# since they will be recreated, and may cause churn.  Uncomment if using
# auto-import.
# .idea/artifacts
# .idea/compiler.xml
# .idea/jarRepositories.xml
# .idea/modules.xml
# .idea/*.iml
# .idea/modules
# *.iml
# *.ipr

# CMake
cmake-build-*/
```

```
# Mongo Explorer plugin
.idea/**/mongoSettings.xml

# File-based project format
*.iws

# IntelliJ
out/

# mpeltonen/sbt-idea plugin
.idea_modules/

# JIRA plugin
atlassian-ide-plugin.xml

# Cursive Clojure plugin
.idea/replstate.xml

# SonarLint plugin
.idea/sonarlint/

# Crashlytics plugin (for Android Studio and IntelliJ)
com_crashlytics_export_strings.xml
crashlytics.properties
crashlytics-build.properties
fabric.properties

# Editor-based Rest Client
.idea/httpRequests

# Android studio 3.1+ serialized cache file
.idea/caches/build_file_checksums.ser

### JetBrains Patch ###
# Comment Reason: https://github.com/joeblau/gitignore.io/issues/186#issuecomment-215987721

# *.iml
# modules.xml
# .idea/misc.xml
# *.ipr

# Sonarlint plugin
# https://plugins.jetbrains.com/plugin/7973-sonarlint
.idea/**/sonarlint/

# SonarQube Plugin
# https://plugins.jetbrains.com/plugin/7238-sonarqube-community-plugin
.idea/**/sonarIssues.xml

# Markdown Navigator plugin
# https://plugins.jetbrains.com/plugin/7896-markdown-navigator-enhanced
.idea/**/markdown-navigator.xml
.idea/**/markdown-navigator-enh.xml
.idea/**/markdown-navigator/

# Cache file creation bug
# See https://youtrack.jetbrains.com/issue/JBR-2257
.idea/$CACHE_FILE$

# CodeStream plugin
# https://plugins.jetbrains.com/plugin/12206-codestream
.idea/codestream.xml

# Azure Toolkit for IntelliJ plugin
# https://plugins.jetbrains.com/plugin/8053-azure-toolkit-for-intellij
.idea/**/azureSettings.xml

### macOS ###
# General
.DS_Store
.AppleDouble
.LSOverride

# Icon must end with two \r
Icon


# Thumbnails
```

```
._*

# Files that might appear in the root of a volume
.DocumentRevisions-V100
.fseventsd
.Spotlight-V100
.TemporaryItems
.Trashes
.VolumeIcon.icns
.com.apple.timemachine.donotpresent

# Directories potentially created on remote AFP share
.AppleDB
.AppleDesktop
Network Trash Folder
Temporary Items
.apdisk

### macOS Patch ###
# iCloud generated files
*.icloud

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff

# AWS User-specific

# Generated files

# Sensitive or high-churn files

# Gradle

# Gradle and Maven with auto-import
# When using Gradle or Maven with auto-import, you should exclude module files,
# since they will be recreated, and may cause churn.  Uncomment if using
# auto-import.
# .idea/artifacts
# .idea/compiler.xml
# .idea/jarRepositories.xml
# .idea/modules.xml
# .idea/*.iml
# .idea/modules
# *.iml
# *.ipr

# CMake

# Mongo Explorer plugin

# File-based project format

# IntelliJ

# mpeltonen/sbt-idea plugin

# JIRA plugin

# Cursive Clojure plugin

# SonarLint plugin

# Crashlytics plugin (for Android Studio and IntelliJ)

# Editor-based Rest Client

# Android studio 3.1+ serialized cache file

### PyCharm Patch ###
# Comment Reason: https://github.com/joeblau/gitignore.io/issues/186#issuecomment-215987721

# *.iml
# modules.xml
# .idea/misc.xml
# *.ipr

```

```
# Sonarlint plugin
# https://plugins.jetbrains.com/plugin/7973-sonarlint

# SonarQube Plugin
# https://plugins.jetbrains.com/plugin/7238-sonarqube-community-plugin

# Markdown Navigator plugin
# https://plugins.jetbrains.com/plugin/7896-markdown-navigator-enhanced

# Cache file creation bug
# See https://youtrack.jetbrains.com/issue/JBR-2257

# CodeStream plugin
# https://plugins.jetbrains.com/plugin/12206-codestream

# Azure Toolkit for IntelliJ plugin
# https://plugins.jetbrains.com/plugin/8053-azure-toolkit-for-intellij

### Python ###
# Byte-compiled / optimized / DLL files

# C extensions

# Distribution / packaging

# PyInstaller
#  Usually these files are written by a python script from a template
#  before PyInstaller builds the exe, so as to inject date/other infos into it.

# Installer logs

# Unit test / coverage reports

# Translations

# Django stuff:

# Flask stuff:

# Scrapy stuff:

# Sphinx documentation

# PyBuilder

# Jupyter Notebook

# IPython

# pyenv
#   For a library or package, you might want to ignore these files since the code is
#   intended to run in multiple environments; otherwise, check them in:
# .python-version

# pipenv
#   According to pypa/pipenv#598, it is recommended to include Pipfile.lock in version control.
#   However, in case of collaboration, if having platform-specific dependencies or dependencies
#   having no cross-platform support, pipenv may install dependencies that don't work, or not
#   install all needed dependencies.

# poetry
#   Similar to Pipfile.lock, it is generally recommended to include poetry.lock in version control.
#   This is especially recommended for binary packages to ensure reproducibility, and is more
#   commonly ignored for libraries.
#   https://python-poetry.org/docs/basic-usage/#commit-your-poetrylock-file-to-version-control

# pdm
#   Similar to Pipfile.lock, it is generally recommended to include pdm.lock in version control.
#   pdm stores project-wide configurations in .pdm.toml, but it is recommended to not include it
#   in version control.
#   https://pdm.fming.dev/#use-with-ide

# PEP 582; used by e.g. github.com/David-OConnor/pyflow and github.com/pdm-project/pdm

# Celery stuff

# SageMath parsed files

# Environments
```

```
# Spyder project settings

# Rope project settings

# mkdocs documentation

# mypy

# Pyre type checker

# pytype static type analyzer

# Cython debug symbols

# PyCharm
#  JetBrains specific template is maintained in a separate JetBrains.gitignore that can
#  be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
#  and can be added to the global gitignore or merged into this file.  For a more nuclear
#  option (not recommended) you can uncomment the following to ignore the entire idea folder.

### Python Patch ###
# Poetry local configuration file - https://python-poetry.org/docs/configuration/#local-configuration
poetry.toml

# ruff
.ruff_cache/

# End of https://www.toptal.com/developers/gitignore/api/macos,pycharm,jetbrains,django,python
```

Translator/README.md

Translator

This is an app utilising the models I trained for my final year project.

It is a simple Django app, with simple user management. It allows translating between English and Polish, both ways. It also stores past translations for logged-in users.

Installation

To install the app, you need to have Python 3 installed (3.9 recommended). Then, you need to install the requirements:

```
virtualenv venv --python=3.9
source venv/bin/activate
pip3 install -r requirements.txt
```

Then, you need to create a database:

```
python3 manage.py migrate
```

Then, you need to run the server:

```
python3 manage.py runserver
```

To run tests, use the following command:

```
python3 manage.py test
```

To run coverage, use the following command:

```
coverage run manage.py test
coverage report
```

Usage

To use the app, navigate to `localhost:8000` in your browser. You can then register an account, and log in if you choose to do so. You can translate between English and Polish, both ways.

Translator/manage.py

```python
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'translator.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc: # pragma: no cover
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()

```

Translator/requirements.txt

```
Django==4.1.5
django-widget-tweaks==1.4.12
django-extensions==3.2.1
transformers==4.26.0
torch==1.13.1
sentencepiece==0.1.97
coverage==7.1.0
django-extensions==3.2.1

```

Translator/static/custom.css

```css
.form-box {
    background-color: #141414;
    padding-bottom: 2%;
    padding-top: 2%;
    border-radius: 15px;
}

.form-button {
    background-color: #6D1DBC;
    border-color: #6D1DBC;
    color: #FBF6FF;
}

.form-button:hover {
    background-color: #5f1aa3;
    border-color: #5f1aa3;
    color: #FBF6FF;
}

.form-button:active {
    background-color: #551791;
    border-color: #551791;
    color: #FBF6FF;
}

.translation-box {
    background-color: #141414;
    padding: 2%;
    border-radius: 15px;
}

.translation-item {
    background-color: #282828;
    padding: 1%;
    margin: 1%;
```

```
    border-radius: 15px;
    text-align: center;
}

.translation-item:hover {
    background-color: #3a3a3a;
}
```

Translator/static/favicon/browserconfig.xml

```
<?xml version="1.0" encoding="utf-8"?>
<browserconfig>
    <msapplication>
        <tile>
            <square150x150logo src="/mstile-150x150.png"/>
            <TileColor>#fbf6ff</TileColor>
        </tile>
    </msapplication>
</browserconfig>

```

Translator/static/favicon/safari-pinned-tab.svg

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg version="1.0" xmlns="http://www.w3.org/2000/svg"
 width="640.000000pt" height="640.000000pt" viewBox="0 0 640.000000 640.000000"
 preserveAspectRatio="xMidYMid meet">
<metadata>
Created by potrace 1.14, written by Peter Selinger 2001-2017
</metadata>
<g transform="translate(0.000000,640.000000) scale(0.100000,-0.100000)"
fill="#000000" stroke="none">
<path d="M604 5121 c-2 -2 -23 -5 -46 -7 -52 -5 -208 -54 -208 -65 0 -5 -6 -9
-12 -9 -24 -1 -103 -61 -166 -127 -64 -68 -136 -196 -147 -260 -3 -18 -9 -44
-15 -58 -6 -16 -10 -525 -9 -1383 0 -1496 -3 -1425 63 -1565 41 -87 52 -103
138 -188 44 -43 93 -79 144 -105 42 -21 81 -40 86 -40 4 -1 9 -2 11 -4 1 -1 9
-3 17 -5 8 -2 29 -6 45 -10 17 -4 72 -8 122 -9 51 0 93 -3 93 -6 0 -3 26 -4
57 -3 32 1 58 2 58 3 0 1 28 2 63 3 34 1 62 0 62 -3 0 -3 26 -4 57 -3 32 1 58
2 58 3 0 1 28 2 63 3 34 1 62 0 62 -3 0 -3 26 -4 57 -3 32 1 58 2 58 3 0 1 28
2 63 3 34 1 62 0 62 -3 0 -3 26 -4 57 -3 32 1 58 2 58 3 0 1 28 2 63 3 34 1
62 0 62 -3 0 -3 26 -4 57 -3 32 1 58 2 58 3 0 1 28 2 63 3 34 1 62 0 62 -3 0
-3 26 -4 57 -3 32 1 58 2 58 3 0 1 28 2 63 3 34 1 62 0 62 -3 0 -3 26 -4 57
-3 32 1 58 2 58 3 0 1 28 2 63 3 34 1 62 0 62 -3 0 -3 26 -4 57 -3 32 1 58 2
58 3 0 1 28 2 63 3 34 1 62 0 62 -3 0 -3 26 -4 57 -3 32 1 58 2 58 3 0 1 28 2
63 3 34 1 62 0 62 -3 0 -3 26 -4 57 -3 32 1 58 2 58 3 0 1 28 2 63 3 34 1 62
0 62 -3 0 -3 26 -4 57 -3 32 1 58 2 58 3 0 1 28 2 63 3 34 1 62 0 62 -3 0 -3
26 -4 57 -3 32 1 58 2 58 3 0 1 28 2 63 3 34 1 62 0 62 -3 0 -3 26 -4 57 -3
32 1 58 2 58 3 0 1 28 2 63 3 34 1 62 0 62 -3 0 -3 26 -4 57 -3 32 1 58 2 58
3 0 1 28 2 63 3 34 1 62 0 62 -3 0 -3 26 -4 57 -3 32 1 58 2 58 3 0 1 28 2 63
3 34 1 62 0 62 -3 0 -3 26 -4 57 -3 32 1 58 2 58 3 0 1 28 2 63 3 34 1 62 0
62 -3 0 -3 26 -4 57 -3 32 1 58 2 58 3 0 1 28 2 63 3 34 1 62 0 62 -3 0 -3 26
-4 57 -3 32 1 58 2 58 3 0 1 28 2 63 3 34 1 62 0 62 -3 0 -6 109 -4 116 2 2 3
24 2 49 -2 24 -4 50 -3 57 1 6 4 15 6 19 4 5 -4 67 -6 119 -4 8 0 33 0 55 0
22 0 47 0 55 0 8 0 40 0 70 0 30 0 64 -1 75 -1 11 0 36 0 55 0 63 0 167 7 177
13 5 3 25 9 44 12 66 11 193 82 262 147 66 63 126 142 127 166 0 6 4 12 8 12
5 0 22 37 38 82 l29 83 2 1375 c3 1518 6 1442 -61 1583 -39 84 -52 101 -127
178 -46 46 -157 129 -174 129 -2 0 -16 5 -32 12 -15 6 -31 12 -35 13 -5 2 -9
3 -10 5 -2 1 -10 3 -18 5 -8 2 -28 6 -45 10 -32 7 -109 14 -160 15 -34 0 -49
0 -120 -1 -27 0 -57 0 -65 0 -8 1 -33 1 -55 0 -38 -1 -47 0 -77 2 -14 1 -112
0 -168 -2 -19 -1 -41 -1 -47 0 -7 1 -19 1 -25 2 -14 1 -112 0 -168 -2 -19 -1
-41 -1 -47 0 -7 1 -19 1 -25 2 -14 1 -112 0 -168 -2 -19 -1 -41 -1 -47 0 -7 1
-19 1 -25 2 -14 1 -112 0 -168 -2 -19 -1 -41 -1 -47 0 -7 1 -19 1 -25 2 -14 1
-112 0 -168 -2 -19 -1 -41 -1 -47 0 -7 1 -19 1 -25 2 -14 1 -112 0 -168 -2
-19 -1 -39 -1 -45 -1 -13 1 -395 1 -415 1 -8 -1 -80 0 -160 0 -80 0 -152 0
-160 0 -8 0 -80 0 -160 0 -80 0 -152 0 -160 0 -8 0 -80 0 -160 0 -178 1 -362
2 -375 2 -82 -1 -103 -1 -190 0 -33 0 -67 0 -75 0 -8 -1 -31 -1 -50 0 -19 0
-42 0 -50 0 -8 0 -40 0 -70 0 -30 0 -62 0 -70 0 -8 -1 -31 -1 -50 0 -19 0 -42
0 -50 0 -8 0 -40 0 -70 0 -30 0 -62 0 -70 0 -8 -1 -31 -1 -50 0 -19 0 -42 0
-50 0 -8 0 -40 0 -70 0 -30 0 -62 0 -70 0 -8 -1 -31 -1 -50 0 -19 1 -39 1 -45
0 -70 -4 -135 -5 -135 -1 0 3 -26 4 -57 3 -32 -1 -58 -2 -58 -3 0 -1 -28 -2
-63 -3 -34 -1 -62 0 -62 3 0 3 -26 4 -57 3 -32 -1 -58 -2 -58 -3 0 -1 -28 -2
-63 -3 -34 -1 -62 0 -62 3 0 6 -110 7 -116 1z m2818 -640 c6 4 15 6 19 4 6 -4
80 -6 119 -4 8 1 29 0 45 0 26 -1 130 -3 160 -3 6 0 71 1 145 1 74 0 144 0
```

```
155 0 11 0 85 0 165 0 80 0 151 0 158 0 19 0 181 0 195 0 6 -1 37 -1 67 0 30
0 62 0 70 0 8 -1 31 -1 50 0 19 1 42 1 50 0 8 -1 40 -1 70 0 30 0 62 0 70 0 8
-1 31 -1 50 0 19 1 42 1 50 0 8 -1 40 -1 70 0 30 0 62 0 70 0 8 -1 31 -1 50 0
19 1 42 1 50 0 8 -1 40 -1 70 0 30 0 62 0 70 0 8 -1 31 -1 50 0 19 1 42 1 50
0 8 -1 38 -1 65 0 153 3 154 3 154 -16 0 -10 0 -36 0 -58 0 -22 0 -47 0 -55 0
-8 0 -31 0 -50 1 -19 1 -42 0 -50 0 -8 0 -42 0 -75 1 -85 1 -131 0 -190 0 -5
0 -66 1 -135 0 -69 0 -127 0 -130 -4 -19 -4 -120 0 -120 3 0 4 -30 2 -67 -4
-90 -2 -165 4 -175 3 -4 0 -8 -6 -8 -6 0 -9 -4 -6 -9 5 -7 7 -82 5 -131 0 -8
0 -40 0 -70 0 -30 0 -62 0 -70 0 -8 0 -31 0 -50 1 -19 1 -42 0 -50 0 -8 0 -42
0 -75 1 -85 1 -131 0 -190 0 -5 0 -66 1 -135 0 -69 0 -127 0 -130 -4 -19 -4
-120 0 -120 3 0 4 -28 3 -62 -1 -35 -2 -63 -3 -63 -1 0 -2 -26 -3 -58 -1 -31
1 -57 4 -57 3 0 3 -20 -1 -45 -4 -24 -4 -49 0 -55 4 -6 5 -22 2 -36 -4 -20
-11 -24 -41 -25 -20 -1 -40 0 -46 1 -5 1 -32 1 -60 0 -27 -1 -57 -1 -65 -1 -8
1 -33 1 -55 0 -38 -1 -47 0 -77 2 -7 1 -38 1 -68 0 -30 0 -62 0 -70 0 -8 1
-31 1 -50 0 -19 -1 -42 -1 -50 0 -8 1 -40 1 -70 0 -30 0 -62 0 -70 0 -8 1 -31
1 -50 0 -19 -1 -42 -1 -50 0 -8 1 -40 1 -70 0 -30 0 -62 0 -70 0 -8 1 -31 1
-50 0 -19 -1 -42 -1 -50 0 -8 1 -40 1 -70 0 -30 0 -62 -1 -70 0 -14 0 -264 1
-305 0 -11 0 -85 0 -165 0 -80 0 -152 0 -160 0 -8 0 -62 0 -120 0 -58 0 -109
0 -115 0 -82 -1 -103 -1 -190 0 -73 1 -75 1 -100 -1 -11 -1 -38 -1 -60 -1 -22
0 -47 0 -55 0 -8 0 -32 1 -53 2 l-38 2 1 46 c0 25 -1 53 -1 61 -1 8 -1 38 0
65 0 28 1 57 0 65 -1 8 -1 31 0 50 2 56 3 154 2 168 -1 6 -1 18 -2 25 -1 6 -1
28 0 47 2 56 3 154 2 168 -1 6 -1 18 -2 25 -1 6 -1 28 0 47 2 56 3 154 2 168
-1 6 -1 18 -2 25 -1 6 -1 28 0 47 2 48 3 152 2 168 0 13 0 175 0 195 0 6 0 77
0 157 0 80 0 152 0 160 0 8 0 80 0 160 0 80 0 152 0 160 0 8 0 80 0 160 0 80
0 152 0 160 0 8 0 59 -1 112 -1 54 2 103 5 109 4 6 29 9 56 8 27 -2 51 0 54 3
3 2 25 2 50 -2 24 -4 50 -3 57 1z m-1711 -396 c21 -15 44 -37 52 -49 8 -11 58
-120 112 -241 53 -121 107 -240 118 -265 11 -25 21 -47 22 -50 1 -3 8 -21 17
-40 8 -19 43 -96 76 -170 33 -74 71 -160 85 -190 13 -30 50 -113 81 -185 58
-131 67 -152 126 -280 51 -111 49 -185 -8 -262 -67 -91 -233 -92 -303 -1 -13
16 -45 78 -71 136 l-47 107 -138 3 c-169 3 -135 3 -391 2 l-213 -1 -9 -25 c-9
-23 -26 -61 -78 -174 -34 -76 -100 -117 -182 -115 -117 3 -188 70 -196 182 -3
57 0 68 95 276 17 37 31 69 31 72 0 2 15 36 34 77 18 40 56 125 84 188 27 63
64 147 82 185 17 39 38 84 45 100 7 17 26 59 42 95 16 36 79 177 139 314 139
318 169 351 303 341 37 -3 65 -12 92 -30z"/>
<path d="M4395 4100 c-70 -36 -104 -86 -112 -165 l-6 -53 -261 0 c-143 0 -277
-3 -296 -7 -88 -15 -158 -102 -155 -195 2 -85 33 -137 106 -177 32 -17 70 -18
592 -20 307 -1 561 -3 563 -6 14 -13 -127 -249 -211 -352 -84 -105 -67 -105
-176 1 -103 99 -142 119 -217 111 -68 -8 -110 -34 -152 -96 -19 -27 -24 -48
-24 -95 0 -78 16 -108 106 -198 53 -53 67 -73 57 -79 -8 -5 -71 -38 -140 -74
-74 -38 -134 -76 -145 -93 -62 -84 -51 -202 24 -267 84 -73 153 -71 313 8 63
31 148 79 191 107 79 52 96 58 115 39 6 -6 20 -16 31 -23 11 -6 60 -35 108
-65 49 -30 105 -63 124 -74 19 -10 40 -23 45 -27 19 -15 109 -20 145 -9 91 30
141 104 136 201 -4 87 -36 126 -166 204 -58 35 -113 68 -123 74 -17 10 -14 16
30 68 73 87 113 140 113 151 0 5 3 11 8 13 10 4 62 85 62 97 0 5 4 11 8 13 17
7 134 254 166 351 3 9 16 17 28 18 108 3 187 79 193 184 4 67 -6 99 -43 141
-63 73 -78 75 -387 75 -82 0 -157 0 -165 0 -22 0 -151 0 -177 0 -13 -1 -23 5
-24 12 -3 86 -15 123 -56 166 -53 57 -160 76 -228 41z"/>
<path d="M1585 3398 c-4 -13 -22 -54 -39 -93 -18 -38 -48 -106 -67 -150 -19
-44 -41 -96 -51 -115 -9 -19 -15 -36 -13 -38 3 -4 367 -3 371 1 1 1 -22 56
-52 122 -76 170 -88 199 -86 208 1 4 -2 7 -6 7 -5 0 -15 18 -22 40 -15 45 -26
50 -35 18z"/>
</g>
</svg>
```

Translator/static/favicon/site.webmanifest

```
{
    "name": "",
    "short_name": "",
    "icons": [
        {
            "src": "/android-chrome-192x192.png",
            "sizes": "192x192",
            "type": "image/png"
        },
        {
            "src": "/android-chrome-512x512.png",
            "sizes": "512x512",
            "type": "image/png"
        }
    ],
    "theme_color": "#ffffff",
    "background_color": "#ffffff",
    "display": "standalone"
}
```

```
20
```

Translator/translate/__init__.py

```
1
```

Translator/translate/__pycache__/views.cpython-39.pyc.140357439501584

Translator/translate/admin.py

```python
from django.contrib import admin

# Register your models here.

```

Translator/translate/apps.py

```python
from django.apps import AppConfig


class TranslateConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'translate'

```

Translator/translate/forms.py

```python
from django import forms
from django.contrib.auth import authenticate
from django.contrib.auth.forms import UserCreationForm

from translate.models import User

import translator_backend


class LogInForm(forms.Form):
    """Form to log in users. Adapted from the Clucker project from 5CCS2SEG."""
    username = forms.CharField(label='Username', widget=forms.TextInput())
    password = forms.CharField(label='Password', widget=forms.PasswordInput())

    def get_user(self):
        user = None
        if self.is_valid():
            username = self.cleaned_data.get('username')
            password = self.cleaned_data.get('password')
            user = authenticate(username=username, password=password)
        return user


class SignUpForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['username', 'email']

    def clean(self):
        super().clean()
        password = self.cleaned_data.get('password1')
        password_confirmation = self.cleaned_data.get('password2')
        if password != password_confirmation:
            self.add_error('password1', 'Confirmation does not match password.')

    def save(self, commit=True):
        user = super().save(commit=False)
        user.set_password(self.cleaned_data['password1'])
        if commit:
            user.save()
```

```
        return user


class TranslatorForm(forms.Form):
    language = forms.ChoiceField(label='Input language', choices=[('English', 'English'), ('Polish', 'Polish')])
    text = forms.CharField(label='Text',
                           widget=forms.Textarea(attrs={'rows': 10, 'cols': 50, 'placeholder': 'Enter text here'}))

    def clean(self):
        super().clean()
        text = self.cleaned_data.get('text')
        if not text:
            self.add_error('text', 'Please enter some text.')

    def translate(self):
        text = self.cleaned_data.get('text')
        if self.cleaned_data['language'] == 'English':
            translated_texts = translator_backend.translateEnglishToPolish(text)
        else:
            translated_texts = translator_backend.translatePolishToEnglish(text)
        whole_text = []
        for translated_text in translated_texts:
            whole_text.append(translated_text['translation_text'])
        return '\n'.join(whole_text)

```

Translator/translate/migrations/0001_initial.py

```
# Generated by Django 4.1.5 on 2023-01-28 19:22

import django.contrib.auth.models
import django.contrib.auth.validators
from django.db import migrations, models
import django.utils.timezone


class Migration(migrations.Migration):
    initial = True

    dependencies = [
        ('auth', '0012_alter_user_first_name_max_length'),
    ]

    operations = [
        migrations.CreateModel(
            name='User',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('password', models.CharField(max_length=128, verbose_name='password')),
                ('last_login', models.DateTimeField(blank=True, null=True, verbose_name='last login')),
                ('is_superuser', models.BooleanField(default=False,
                                                     help_text='Designates that this user has all permissions without
explicitly assigning them.',
                                                     verbose_name='superuser status')),
                ('username', models.CharField(error_messages={'unique': 'A user with that username already exists.'},
                                              help_text='Required. 150 characters or fewer. Letters, digits and
@/./+/-/_ only.',
                                              max_length=150, unique=True,
                                              validators=[django.contrib.auth.validators.UnicodeUsernameValidator()],
                                              verbose_name='username')),
                ('first_name', models.CharField(blank=True, max_length=150, verbose_name='first name')),
                ('last_name', models.CharField(blank=True, max_length=150, verbose_name='last name')),
                ('is_staff', models.BooleanField(default=False,
                                                 help_text='Designates whether the user can log into this admin
site.',
                                                 verbose_name='staff status')),
                ('is_active', models.BooleanField(default=True,
                                                  help_text='Designates whether this user should be treated as
active. Unselect this instead of deleting accounts.',
                                                  verbose_name='active')),
                ('date_joined', models.DateTimeField(default=django.utils.timezone.now, verbose_name='date joined')),

                ('email', models.EmailField(max_length=254, unique=True)),
                ('groups', models.ManyToManyField(blank=True,
                                                  help_text='The groups this user belongs to. A user will get all
permissions granted to each of their groups.',
                                                  related_name='user_set', related_query_name='user',
to='auth.group',
                                                  verbose_name='groups')),
```

```
                ('user_permissions', models.ManyToManyField(blank=True, help_text='Specific permissions for this user.',
                                                            related_name='user_set', related_query_name='user',
                                                            to='auth.permission', verbose_name='user permissions')),
            ],
            options={
                'verbose_name': 'user',
                'verbose_name_plural': 'users',
                'abstract': False,
            },
            managers=[
                ('objects', django.contrib.auth.models.UserManager()),
            ],
        ),
    ]

```

Translator/translate/migrations/0002_translation.py

```
# Generated by Django 4.1.5 on 2023-02-02 10:24

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):
    dependencies = [
        ('translate', '0001_initial'),
    ]

    operations = [
        migrations.CreateModel(
            name='Translation',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('text', models.CharField(max_length=255)),
                ('translated_text', models.CharField(max_length=255)),
                ('created_at', models.DateTimeField(auto_now_add=True)),
                ('updated_at', models.DateTimeField(auto_now=True)),
                ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),
            ],
        ),
    ]

```

Translator/translate/migrations/0003_alter_translation_user.py

```
# Generated by Django 4.1.5 on 2023-02-02 11:38

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):
    dependencies = [
        ('translate', '0002_translation'),
    ]

    operations = [
        migrations.AlterField(
            model_name='translation',
            name='user',
            field=models.ForeignKey(null=True, on_delete=django.db.models.deletion.CASCADE,
                                    to=settings.AUTH_USER_MODEL),
        ),
    ]

```

Translator/translate/migrations/0004_alter_translation_user.py

```python
# Generated by Django 4.1.5 on 2023-02-02 11:45

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):
    dependencies = [
        ('translate', '0003_alter_translation_user'),
    ]

    operations = [
        migrations.AlterField(
            model_name='translation',
            name='user',
            field=models.ForeignKey(blank=True, null=True, on_delete=django.db.models.deletion.CASCADE,
                                    to=settings.AUTH_USER_MODEL),
        ),
    ]

```

Translator/translate/migrations/0005_translation_input_language.py

```python
# Generated by Django 4.1.5 on 2023-02-09 12:32

from django.db import migrations, models


class Migration(migrations.Migration):
    dependencies = [
        ('translate', '0004_alter_translation_user'),
    ]

    operations = [
        migrations.AddField(
            model_name='translation',
            name='input_language',
            field=models.CharField(choices=[('English', 'English'), ('Polish', 'Polish')], default='English',
                                   max_length=7),
            preserve_default=False,
        ),
    ]

```

Translator/translate/migrations/0006_remove_translation_created_at_and_more.py

```python
# Generated by Django 4.1.5 on 2023-02-09 15:33

from django.db import migrations


class Migration(migrations.Migration):
    dependencies = [
        ('translate', '0005_translation_input_language'),
    ]

    operations = [
        migrations.RemoveField(
            model_name='translation',
            name='created_at',
        ),
        migrations.RemoveField(
            model_name='translation',
            name='updated_at',
        ),
    ]

```

Translator/translate/migrations/0007_translation_created_at_translation_updated_at.py

```python
# Generated by Django 4.1.5 on 2023-02-14 12:32
```

```

from django.db import migrations, models
import django.utils.timezone


class Migration(migrations.Migration):
    dependencies = [
        ('translate', '0006_remove_translation_created_at_and_more'),
    ]

    operations = [
        migrations.AddField(
            model_name='translation',
            name='created_at',
            field=models.DateTimeField(auto_now_add=True, default=django.utils.timezone.now),
            preserve_default=False,
        ),
        migrations.AddField(
            model_name='translation',
            name='updated_at',
            field=models.DateTimeField(auto_now=True),
        ),
    ]

```

Translator/translate/migrations/__init__.py

```

```

Translator/translate/models.py

```
from django.contrib.auth.models import AbstractUser
from django.db import models


class User(AbstractUser):
    """A simple user model that extends the default Django user model."""
    email = models.EmailField(max_length=254, unique=True)


class Translation(models.Model):
    """A translation of a word or phrase."""

    class LanguageChoices(models.TextChoices):
        ENGLISH = 'English'
        POLISH = 'Polish'

    text = models.CharField(max_length=255)
    translated_text = models.CharField(max_length=255)
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True, blank=True)
    input_language = models.CharField(max_length=7, choices=LanguageChoices.choices)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def text_display(self):
        return self.text[:27] + '...' if len(self.text) > 30 else self.text

    def translated_text_display(self):
        return self.translated_text[:27] + '...' if len(self.translated_text) > 30 else self.translated_text

```

Translator/translate/templates/base.html

```
{% load static %}
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
          integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">

```

```
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.6.1/font/bootstrap-icons.css">

    <link href="{% static 'custom.css' %}" rel="stylesheet">
    <link rel="apple-touch-icon" sizes="180x180" href="{% static 'favicon/apple-touch-icon.png' %}">
    <link rel="icon" type="image/png" sizes="32x32" href="{% static 'favicon/favicon-32x32.png' %}">
    <link rel="icon" type="image/png" sizes="16x16" href="{% static 'favicon/favicon-16x16.png' %}">
    <link rel="manifest" href="{% static 'favicon/site.webmanifest' %}">
    <link rel="mask-icon" href="{% static 'favicon/safari-pinned-tab.svg' %}" color="#320064">
    <meta name="msapplication-TileColor" content="#fbf6ff">
    <meta name="theme-color" content="#ffffff">
    <meta name="apple-mobile-web-app-title" content="Translator">
    <meta name="application-name" content="Translator">

    <script src="https://kit.fontawesome.com/29f2d97b0d.js" crossorigin="anonymous"></script>

    <title>
        Translator
        {% block title %}
        {% endblock %}
    </title>

    <meta property="og:url" content="{{ request.build_absolute_uri }}"/>
    <meta property="og:type" content="website"/>
    <meta property="og:title" content="Translate"/>
    <meta property="og:description"
          content="The implementation part of my final year project."/>

    <style>
        @import url('https://fonts.googleapis.com/css2?family=JetBrains+Mono:wght@300;400;500;700&display=swap');

        * {
            font-family: 'JetBrains Mono', sans-serif;
        }
    </style>
</head>
<body style="background-color: #282828; color: #FBF6FF">
<div id="background">
    {% block body %}
    {% endblock %}
</div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js"
        integrity="sha384-QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13"
        crossorigin="anonymous"></script>
</body>
```

Translator/translate/templates/base_content.html

```
{% extends 'base.html' %}
{% block title %}
{% endblock %}
{% block body %}
{% include 'partials/navbar.html' %}
{% include 'partials/messages.html' %}
{% block content %}
{% endblock %}
{% endblock %}
```

Translator/translate/templates/home.html

```
{% extends 'base_content.html' %}
{% block title %}{% endblock %}
{% block content %}
{% load widget_tweaks %}
<div class="container" style="height: 100%">
    <div class="row">
        <div class="col-sm-12 col-md-6 offset-md-3">
            <h1>Translator</h1>
            {% if user.is_authenticated %}
            <p>Welcome, {{ user.username }}!</p>
            <a href="{% url 'logout' %}">Log out</a>
            {% else %}
            <p>Welcome, guest!</p>
            <a href="{% url 'login' %}">Log in</a>
```

```
            <a href="{% url 'signup' %}">Sign up</a>
            {% endif %}
            <p>Here you can translate text from one language to another.</p>
        </div>
    </div>
    <br>
    <!-- This is the translation form -->
    <form action="{% url 'home' %}" method="post">
        {% csrf_token %}
        <div class="row" style="text-align: center">
            <div class="col-sm-12 col-md-5">
                {% include 'partials/bootstrap_form.html' with form=form %}
            </div>

            <div class="col-sm-12 col-md-2">
                <input class="btn btn-secondary form-button" type="submit" value="Translate">
            </div>

            <div class="col-sm-12 col-md-5">
                <label>Output in {{second_language}}
                    <textarea class="form-control" readonly="readonly" placeholder="Translated text"
                              style="height: auto" rows="13" cols="50">{{ translations }}</textarea>
                </label>
            </div>
        </div>
    </form>


</div>
{% endblock %}
```

Translator/translate/templates/login.html

```
{% extends 'base_content.html' %}
{% block title %} | Log in{% endblock %}
{% block content %}
<div class="container">
    <div class="row">
        <div class="col-sm-12 col-md-6 offset-md-3 form-box">
            <h1>Log in</h1>
            <form action="{% url 'login' %}" method="post">
                {% csrf_token %}
                <input type="hidden" name="next" value="{{ next }}">
                {% include 'partials/bootstrap_form.html' with form=form %}
                <input class="btn btn-secondary form-button" type="submit" value="Log in">
            </form>
        </div>
    </div>
</div>
{% endblock %}
```

Translator/translate/templates/partials/bootstrap_form.html

```
<!--This code was adapted from the 5CCS2SEG project Chess Club-->

{% load widget_tweaks %}
{% for field in form %}
<div class="mb-3">
    {{ field.label_tag }}

    {% if field.help_text %}
    <p class="form-text text-muted">{{ field.help_text }}</p>
    {% endif %}

    {% if form.is_bound %}
    {% if field.errors %}
    {% render_field field class="form-control is-invalid" %}
    {% else %}
    {% render_field field class="form-control is-valid" %}
    {% endif %}
    {% else %}
    {% render_field field class="form-control" %}
    {% endif %}
    <div class="valid-feedback"></div>
    <div class="invalid-feedback">
```

```
        {{ field.errors }}
    </div>
</div>
{% endfor %}

```

Translator/translate/templates/partials/messages.html

```
<!--This code was adapted from the 5CCS2SEG project Chess Club-->

{% if messages %}
<div class="container">
    <div class="row">
        <div class="col-12">
            {% for message in messages %}
            <div class="alert alert-{{ message.level_tag }}" role="alert">
                {{ message }}
            </div>
            {% endfor %}
        </div>
    </div>
</div>
{% endif %}
```

Translator/translate/templates/partials/navbar.html

```
<!--This code was adapted from the 5CCS2SEG project Chess Club-->

<nav class="navbar navbar-expand-lg navbar-dark mb-3" style="background-color: #320064;">
    <div class="container">

        <a class="navbar-brand" href="{% url 'home' %}">
            <span class="fas fa-solid fa-language"></span>  Translate
        </a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent"
                aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
                {% if user.is_authenticated %}
                <li class="nav-item dropdown">
                    <a class="nav-link" href="#" id="user_account_dropdown" role="button" data-bs-toggle="dropdown"
                       aria-expanded="false">
                        <span class="bi-person-circle"></span>
                    </a>
                    <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="user_account_dropdown">
                        <li><a class="dropdown-item" href="{% url 'profile' user.id %}">View profile</a></li>
                        <li><a class="dropdown-item" href="{% url 'password_change' %}">Change password</a></li>
                        <li>
                            <hr class="dropdown-divider">
                        </li>
                        <li><a class="dropdown-item" href="{% url 'logout' %}">Log out</a></li>
                    </ul>
                </li>
                {% endif %}
            </ul>
        </div>
    </div>
</nav>

```

Translator/translate/templates/password_change.html

```
{% extends 'base_content.html' %}
{% block title %} | Password change{% endblock %}
{% block content %}
<div class="container">
    <div class="row">
        <div class="col-sm-12 col-md-6 offset-md-3 form-box">
            <h1>Change password</h1>

```

```
            <form action="{% url 'password_change' %}" method="post">
                {% csrf_token %}
                <input type="hidden" name="next" value="{{ next }}">
                {% include 'partials/bootstrap_form.html' with form=form %}
                <input type="submit" value="Password change" class="btn btn-secondary form-button">
            </form>
        </div>
    </div>
</div>
{% endblock %}
```

Translator/translate/templates/profile.html

```
{% extends 'base_content.html' %}
{% block title %} | Profile{% endblock %}
{% block content %}
<div class="container">
    <div class="row">
        <div class="col-sm-12 col-md-6 offset-md-3">
            <h1>User profile</h1>
            <p>Username: {{ user.username }}</p>
            <p>Email: {{ user.email }}</p>
        </div>
    </div>
    <div class="row">
        <div class="col-sm-12 col-md-8 offset-md-2 translation-box">
            <!--Print all related translations-->
            <div class="translation-item">
                <h4>History of translations:</h4>
            </div>

            {% for translation in translations %}
            <div class="translation-item">
                <a class="text" href="{% url 'home' %}?translation={{translation.pk}}">{{ translation.text_display }}
-> {{ translation.translated_text_display }}</a>
            </div>
            {% endfor %}
        </div>
    </div>
</div>
{% endblock %}
```

Translator/translate/templates/signup.html

```
{% extends 'base_content.html' %}
{% block title %} | Sign up{% endblock %}
{% block content %}
<div class="container">
    <div class="row">
        <div class="col-sm-12 col-md-6 offset-md-3 form-box">
            <h1>Sign up</h1>
            <form action="{% url 'signup' %}" method="post">
                {% csrf_token %}
                <input type="hidden" name="next" value="{{ next }}">
                {% include 'partials/bootstrap_form.html' with form=form %}
                <input type="submit" value="Sign up" class="btn btn-secondary form-button">
            </form>
        </div>
    </div>
</div>
{% endblock %}
```

Translator/translate/tests/__init__.py

```

```

Translator/translate/tests/fixtures/default_translation.json

```
[
  {
```

```
    "model": "translate.translation",
    "pk": 1,
    "fields": {
      "text": "Hello world!",
      "input_language": "English",
      "translated_text": "Witaj świat!",
      "user": 1,
      "created_at": "2023-02-14T12:00:00Z",
      "updated_at": "2023-02-14T12:00:00Z"
    }
  }
]
```

Translator/translate/tests/fixtures/default_user.json

```
[
  {
    "model": "translate.user",
    "pk": 1,
    "fields": {
      "username": "MichaelScott",
      "email": "michaelscott@dundermifflin.com",
      "password": "pbkdf2_sha256$260000$VEDi9wsMYG6eNVeL8WSPqj$LHEiR2iUkusHCIeiQdWS+xQGC9/CjhhrjEOESMMp+c0=",
      "is_active": true
    }
  }
]
```

Translator/translate/tests/fixtures/other_translations.json

```
[
  {
    "model": "translate.translation",
    "pk": 2,
    "fields": {
      "text": "Witaj świecie!",
      "input_language": "Polish",
      "translated_text": "Hello world!",
      "user": 1,
      "created_at": "2023-02-14T12:00:00Z",
      "updated_at": "2023-02-14T13:00:00Z"
    }
  },
  {
    "model": "translate.translation",
    "pk": 3,
    "fields": {
      "text": "My name is Mikolaj and I like flowers.",
      "input_language": "English",
      "translated_text": "Nazywam się Mikolaj i lubię kwiaty.",
      "user": 2,
      "created_at": "2023-02-14T12:00:00Z",
      "updated_at": "2023-02-14T15:00:00Z"
    }
  },
  {
    "model": "translate.translation",
    "pk": 4,
    "fields": {
      "text": "Mój ulubiony kolor to czerwony. Jest to kolor miłości i zazdrości.",
      "input_language": "Polish",
      "translated_text": "My favorite color is red. It is the color of love and jealousy.",
      "user": 2,
      "created_at": "2023-02-14T12:00:00Z",
      "updated_at": "2023-02-14T16:00:00Z"
    }
  },
  {
    "model": "translate.translation",
    "pk": 5,
    "fields": {
      "text": "This is a very long text. It is long enough that it needs to be shortened to be displayed.",
      "input_language": "English",
      "translated_text": "To jest bardzo długi tekst. Jest wystarczająco długi, aby musiał zostać skrócony, aby
został wyświetlony.",
```

```
      "user": 1,
      "created_at": "2023-02-14T12:00:00Z",
      "updated_at": "2023-02-14T18:00:00Z"
    }
  }
]
```

Translator/translate/tests/fixtures/other_users.json

```
[
  {
    "model": "translate.user",
    "pk": 2,
    "fields": {
      "username": "DwightSchrute",
      "email": "dwightschrute@dundermifflin.com",
      "password": "pbkdf2_sha256$260000$VEDi9wsMYG6eNVeL8WSPqj$LHEiR2iUkusHCIeiQdWS+xQGC9/CjhhrjEOESMMp+c0=",
      "is_active": true
    }
  },
  {
    "model": "translate.user",
    "pk": 3,
    "fields": {
      "username": "JimHalpert",
      "email": "jimhalpert@dundermifflin.com",
      "password": "pbkdf2_sha256$260000$VEDi9wsMYG6eNVeL8WSPqj$LHEiR2iUkusHCIeiQdWS+xQGC9/CjhhrjEOESMMp+c0=",
      "is_active": true
    }
  },
  {
    "model": "translate.user",
    "pk": 4,
    "fields": {
      "username": "PamBeesly",
      "email": "pambeesly@dundermifflin.com",
      "password": "pbkdf2_sha256$260000$VEDi9wsMYG6eNVeL8WSPqj$LHEiR2iUkusHCIeiQdWS+xQGC9/CjhhrjEOESMMp+c0=",
      "is_active": true
    }
  }
]
```

Translator/translate/tests/forms/__init__.py

```

```

Translator/translate/tests/forms/test_log_in_form.py

```python
from django.test import TestCase

from django import forms
from translate.forms import LogInForm

"""Tests of the log in form. The structure of the tests was inspired by the Clucker project from 5CCS2SEG."""


class TestLogInForm(TestCase):
    fixtures = ['translate/tests/fixtures/default_user.json']

    def setUp(self) -> None:
        self.form_input = {'username': 'MichaelScott', 'password': 'Password123'}

    def test_form_contains_required_fields(self):
        form = LogInForm()
        self.assertIn('username', form.fields)
        self.assertIn('password', form.fields)
        password_field = form.fields['password']
        self.assertTrue(isinstance(password_field.widget, forms.PasswordInput))

    def test_form_accepts_valid_input(self):
        form = LogInForm(data=self.form_input)
        self.assertTrue(form.is_valid())
```

```
    def test_form_rejects_blank_username(self):
        self.form_input['username'] = ''
        form = LogInForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['username'], ['This field is required.'])

    def test_form_rejects_blank_password(self):
        self.form_input['password'] = ''
        form = LogInForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['password'], ['This field is required.'])

    def test_form_accepts_incorrect_username(self):
        self.form_input['username'] = 'DwightSchrute'
        form = LogInForm(data=self.form_input)
        self.assertTrue(form.is_valid())

    def test_form_accepts_incorrect_password(self):
        self.form_input['password'] = 'WrongPassword123'
        form = LogInForm(data=self.form_input)
        self.assertTrue(form.is_valid())

    def test_can_authenticate_valid_user(self):
        form = LogInForm(data=self.form_input)
        user = form.get_user()
        self.assertEqual(user.username, 'MichaelScott')

    def test_invalid_credentials_do_not_authenticate(self):
        self.form_input['password'] = 'WrongPassword123'
        form = LogInForm(data=self.form_input)
        user = form.get_user()
        self.assertEqual(user, None)

    def test_blank_password_do_not_authenticate(self):
        self.form_input['password'] = ''
        form = LogInForm(data=self.form_input)
        user = form.get_user()
        self.assertEqual(user, None)

    def test_blank_username_do_not_authenticate(self):
        self.form_input['username'] = ''
        form = LogInForm(data=self.form_input)
        user = form.get_user()
        self.assertEqual(user, None)

```

Translator/translate/tests/forms/test_sign_up_form.py

```
from django.test import TestCase

from django import forms
from translate.forms import SignUpForm

"""Tests of the sign up form. The structure of the tests was inspired by the Clucker project from 5CCS2SEG."""


class TestSignUpForm(TestCase):
    fixtures = ['translate/tests/fixtures/default_user.json']

    def setUp(self):
        self.form_input = {
            'username': 'JimHalpert',
            'email': 'jimhalpert@dundermifflin.com',
            'password1': 'DwightIsTheBest123',
            'password2': 'DwightIsTheBest123'
        }

    def test_form_contains_required_fields(self):
        form = SignUpForm()
        self.assertIn('username', form.fields)
        self.assertIn('email', form.fields)
        self.assertIn('password1', form.fields)
        self.assertIn('password2', form.fields)
        password1_field = form.fields['password1']
        password2_field = form.fields['password2']
```

```
        self.assertTrue(isinstance(password1_field.widget, forms.PasswordInput))
        self.assertTrue(isinstance(password2_field.widget, forms.PasswordInput))

    def test_form_accepts_valid_input(self):
        form = SignUpForm(data=self.form_input)
        self.assertTrue(form.is_valid())

    def test_form_rejects_blank_username(self):
        self.form_input['username'] = ''
        form = SignUpForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['username'], ['This field is required.'])

    def test_form_rejects_blank_email(self):
        self.form_input['email'] = ''
        form = SignUpForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['email'], ['This field is required.'])

    def test_form_rejects_blank_password1(self):
        self.form_input['password1'] = ''
        self.form_input['password2'] = ''
        form = SignUpForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['password1'], ['This field is required.'])

    def test_form_rejects_blank_password2(self):
        self.form_input['password2'] = ''
        form = SignUpForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['password2'], ['This field is required.'])

    def test_form_rejects_passwords_that_do_not_match(self):
        self.form_input['password2'] = 'WrongPassword123'
        form = SignUpForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['password2'], ['The two password fields didn’t match.'])

    def test_form_rejects_username_that_is_already_taken(self):
        self.form_input['username'] = 'MichaelScott'
        form = SignUpForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['username'], ['A user with that username already exists.'])

    def test_form_rejects_email_that_is_already_taken(self):
        self.form_input['email'] = 'michaelscott@dundermifflin.com'
        form = SignUpForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['email'], ['User with this Email already exists.'])

    def test_form_rejects_common_passwords(self):
        self.form_input['password1'] = 'password'
        self.form_input['password2'] = 'password'
        form = SignUpForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['password2'], ['This password is too common.'])

    def test_form_rejects_passwords_that_are_too_short(self):
        self.form_input['password1'] = 'short'
        self.form_input['password2'] = 'short'
        form = SignUpForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['password2'],
                         ['This password is too short. It must contain at least 8 characters.'])

    def test_form_rejects_passwords_similar_to_username(self):
        self.form_input['password1'] = 'JimHalpert'
        self.form_input['password2'] = 'JimHalpert'
        form = SignUpForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['password2'], ['The password is too similar to the username.'])

    def test_form_rejects_passwords_similar_to_email(self):
        self.form_input['password1'] = 'dundermifflin.com'
        self.form_input['password2'] = 'dundermifflin.com'
        form = SignUpForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['password2'], ['The password is too similar to the email.'])

```

Translator/translate/tests/forms/test_translator_form.py

```python
from django.test import TestCase

from translate.forms import TranslatorForm

"""Tests of the translator form."""


class TestTranslatorForm(TestCase):
    def setUp(self):
        self.form_input = {
            'text': 'Hello World',
            'language': 'English'
        }

    def test_form_contains_required_fields(self):
        form = TranslatorForm()
        self.assertIn('text', form.fields)
        self.assertIn('language', form.fields)

    def test_form_accepts_valid_input(self):
        form = TranslatorForm(data=self.form_input)
        self.assertTrue(form.is_valid())

    def test_form_rejects_blank_text(self):
        self.form_input['text'] = ''
        form = TranslatorForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['text'], ['This field is required.', 'Please enter some text.'])

    def test_form_rejects_blank_language(self):
        self.form_input['language'] = ''
        form = TranslatorForm(data=self.form_input)
        self.assertFalse(form.is_valid())
        self.assertEqual(form.errors['language'], ['This field is required.'])

```

Translator/translate/tests/helpers.py

```python
class LogInTester:
    def _is_logged_in(self):
        return '_auth_user_id' in self.client.session.keys()

```

Translator/translate/tests/models/__init__.py

```python

```

Translator/translate/tests/models/test_translation_model.py

```python
from django.core.exceptions import ValidationError
from django.test import TestCase
from translate.models import User, Translation

"""Tests of the translation model."""


class TestTranslationModel(TestCase):
    fixtures = ['translate/tests/fixtures/default_user.json']

    def setUp(self):
        self.user = User.objects.get(username='MichaelScott')
        self.translation = Translation.objects.create(
            user=self.user,
            text='Hello',
            translated_text='Cześć',
            input_language='English'
        )
        self.translation.save()
```

```
    def test_translation_is_valid(self):
        self._assert_translation_is_valid()

    def test_original_text_cannot_be_blank(self):
        self.translation.text = ''
        self._assert_translation_is_invalid()

    def test_original_text_can_be_255_characters_long(self):
        self.translation.text = 'x' * 255
        self._assert_translation_is_valid()

    def test_original_text_cannot_be_more_than_255_characters_long(self):
        self.translation.text = 'x' * 256
        self._assert_translation_is_invalid()

    def test_translated_text_cannot_be_blank(self):
        self.translation.translated_text = ''
        self._assert_translation_is_invalid()

    def test_translated_text_can_be_255_characters_long(self):
        self.translation.translated_text = 'x' * 255
        self._assert_translation_is_valid()

    def test_translated_text_cannot_be_more_than_255_characters_long(self):
        self.translation.translated_text = 'x' * 256
        self._assert_translation_is_invalid()

    def test_user_can_be_null(self):
        self.translation.user = None
        self._assert_translation_is_valid()

    def test_input_language_cannot_be_blank(self):
        self.translation.input_language = ''
        self._assert_translation_is_invalid()

    def test_input_language_can_only_be_english_or_polish(self):
        self.translation.input_language = 'German'
        self._assert_translation_is_invalid()

    def _assert_translation_is_valid(self):
        try:
            self.translation.full_clean()
        except ValidationError:  # pragma: no cover
            self.fail('Test translation should be valid')

    def _assert_translation_is_invalid(self):
        with self.assertRaises(ValidationError):
            self.translation.full_clean()

```

Translator/translate/tests/models/test_user_model.py

```
from django.core.exceptions import ValidationError
from django.test import TestCase
from translate.models import User

"""Tests of the user model. The structure of the tests was inspired by the clucker project from 5CCS2SEG."""


class TestUserModel(TestCase):
    fixtures = ['translate/tests/fixtures/default_user.json', 'translate/tests/fixtures/other_users.json']

    def setUp(self):
        self.user = User.objects.get(username='MichaelScott')

    def test_user_is_valid(self):
        self._assert_user_is_valid()

    def test_username_cannot_be_blank(self):
        self.user.username = ''
        self._assert_user_is_invalid()

    def test_username_must_be_unique(self):
        dwight = User.objects.get(username='DwightSchrute')
        self.user.username = dwight.username

```

```
        self._assert_user_is_invalid()

    def test_email_must_be_unique(self):
        dwight = User.objects.get(username='DwightSchrute')
        self.user.email = dwight.email
        self._assert_user_is_invalid()

    def test_username_can_be_150_characters_long(self):
        self.user.username = 'x' * 150
        self.test_user_is_valid()

    def test_username_cannot_be_more_than_150_characters_long(self):
        self.user.username = 'x' * 151
        self._assert_user_is_invalid()

    def test_email_must_contain_an_at(self):
        self.user.email = 'michaelscottdundermifflin.com'
        self._assert_user_is_invalid()

    def test_email_must_contain_a_dot_in_the_domain(self):
        self.user.email = 'michaelscott@dundermifflincom'
        self._assert_user_is_invalid()

    def test_email_can_be_254_characters_long(self):
        self.user.email = 'x' * (254 - len("@dundermifflin.com")) + "@dundermifflin.com"
        self._assert_user_is_valid()

    def test_email_cannot_be_longer_than_254_characters_long(self):
        self.user.email = 'x' * (255 - len("@dundermifflin.com")) + "@dundermifflin.com"
        self._assert_user_is_invalid()

    def _assert_user_is_valid(self):
        try:
            self.user.full_clean()
        except ValidationError:  # pragma: no cover
            self.fail('Test user should be valid')

    def _assert_user_is_invalid(self):
        with self.assertRaises(ValidationError):
            self.user.full_clean()

```

Translator/translate/tests/views/__init__.py

```

```

Translator/translate/tests/views/test_change_password_view.py

```
"""Tests of the change password view."""
from django.test import TestCase
from django.urls import reverse

from translate.models import User


class HomeViewTestCase(TestCase):
    """Tests of the change password view."""

    fixtures = ['translate/tests/fixtures/default_user.json']

    def setUp(self):
        self.url = reverse('password_change')
        self.user = User.objects.get(username='MichaelScott')
        self.form = {"old_password": "Password123", "new_password1": "ThisIsMyPassword",
                     "new_password2": "ThisIsMyPassword"}

    def test_change_password_url(self):
        self.assertEqual(self.url, '/password_change/')

    def test_get_change_password(self):
        self.client.login(username=self.user.username, password='Password123')
        response = self.client.get(self.url)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'password_change.html')

```

```
    def test_post_change_password(self):
        self.client.login(username=self.user.username, password='Password123')
        response = self.client.post(self.url, self.form)
        self.assertEqual(response.status_code, 302)
        self.assertRedirects(response, '/')

    def test_post_change_password_with_wrong_old_password(self):
        self.client.login(username=self.user.username, password='Password123')
        self.form['old_password'] = 'WrongPassword'
        response = self.client.post(self.url, self.form)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'password_change.html')

    def test_post_change_password_with_blank_old_password(self):
        self.client.login(username=self.user.username, password='Password123')
        self.form['old_password'] = ''
        response = self.client.post(self.url, self.form)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'password_change.html')

    def test_post_change_password_with_blank_new_password(self):
        self.client.login(username=self.user.username, password='Password123')
        self.form['new_password1'] = ''
        response = self.client.post(self.url, self.form)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'password_change.html')

    def test_post_change_password_with_blank_confirm_password(self):
        self.client.login(username=self.user.username, password='Password123')
        self.form['new_password2'] = ''
        response = self.client.post(self.url, self.form)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'password_change.html')

    def test_post_change_password_with_mismatched_passwords(self):
        self.client.login(username=self.user.username, password='Password123')
        self.form['new_password2'] = 'ThatIsNotMyPassword'
        response = self.client.post(self.url, self.form)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'password_change.html')

```

Translator/translate/tests/views/test_home_view.py

```
"""Tests of the home view."""
from django.test import TestCase
from django.urls import reverse

from translate.models import User, Translation


class HomeViewTestCase(TestCase):
    """Tests of the home view."""

    fixtures = ['translate/tests/fixtures/default_user.json', 'translate/tests/fixtures/default_translation.json']

    def setUp(self):
        self.url = reverse('home')
        self.user = User.objects.get(username='MichaelScott')
        self.form = {'text': 'Hello world!', 'language': 'English'}

    def test_home_url(self):
        self.assertEqual(self.url, '/')

    def test_get_home(self):
        response = self.client.get(self.url)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'home.html')

    def test_get_home_with_translation(self):
        response = self.client.get(self.url, {'translation': 1})
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'home.html')
        self.assertInHTML('Hello world!', response.content.decode('utf-8'))

    def test_post_home(self):
```

```
        count_before = Translation.objects.count()
        response = self.client.post(self.url, self.form)
        count_after = Translation.objects.count()
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'home.html')
        self.assertInHTML('Hello world!', response.content.decode('utf-8'))
        self.assertInHTML('Witaj świecie! Witaj świecie!', response.content.decode('utf-8'))
        self.assertEqual(count_after, count_before + 1)
        translation = Translation.objects.last()
        self.assertEqual(translation.text, 'Hello world!')
        self.assertEqual(translation.translated_text, 'Witaj świecie! Witaj świecie!')
        self.assertEqual(translation.user, None)

    def test_post_home_with_an_existing_translation(self):
        self.client.login(username=self.user.username, password='Password123')
        count_before = Translation.objects.count()
        response = self.client.post(self.url, self.form)
        count_after = Translation.objects.count()
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'home.html')
        self.assertInHTML('Hello world!', response.content.decode('utf-8'))
        self.assertInHTML('Witaj świecie! Witaj świecie!', response.content.decode('utf-8'))
        self.assertEqual(count_after, count_before)
        translation = Translation.objects.last()
        self.assertEqual(translation.text, 'Hello world!')
        self.assertEqual(translation.translated_text, 'Witaj świecie! Witaj świecie!')
        self.assertEqual(translation.user, self.user)

    def test_post_home_with_polish(self):
        count_before = Translation.objects.count()
        response = self.client.post(self.url, {'text': 'Witaj świecie!', 'language': 'Polish'})
        count_after = Translation.objects.count()
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'home.html')
        self.assertInHTML('Witaj świecie!', response.content.decode('utf-8'))
        self.assertInHTML('Hello world!', response.content.decode('utf-8'))
        self.assertEqual(count_after, count_before + 1)
        translation = Translation.objects.last()
        self.assertEqual(translation.text, 'Witaj świecie!')
        self.assertEqual(translation.translated_text, 'Hello world!')
        self.assertEqual(translation.user, None)

    def test_post_home_with_user(self):
        self.client.login(username=self.user.username, password='Password123')
        count_before = Translation.objects.count()
        response = self.client.post(self.url, {'text': 'Hello there!', 'language': 'English'})
        count_after = Translation.objects.count()
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'home.html')
        self.assertInHTML('Hello there!', response.content.decode('utf-8'))
        self.assertInHTML('Witajcie tutaj!', response.content.decode('utf-8'))
        self.assertEqual(count_after, count_before + 1)
        translation = Translation.objects.last()
        self.assertEqual(translation.text, 'Hello there!')
        self.assertEqual(translation.translated_text, 'Witajcie tutaj!')
        self.assertEqual(translation.user, self.user)

    def test_home_greets_guest(self):
        response = self.client.get(self.url)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'home.html')
        self.assertInHTML('Welcome, guest!', response.content.decode('utf-8'))

    def test_home_greets_user(self):
        self.client.login(username=self.user.username, password='Password123')
        response = self.client.get(self.url)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'home.html')
        self.assertInHTML('Welcome, MichaelScott!', response.content.decode('utf-8'))

```

Translator/translate/tests/views/test_log_in_view.py

```
from django.test import TestCase
from django.urls import reverse

from translate.models import User
```

```
from translate.tests.helpers import LogInTester

"""Tests of the log in view. The structure of the tests was inspired by the Clucker project from 5CCS2SEG."""


class LogInViewTest(TestCase, LogInTester):
    fixtures = ['translate/tests/fixtures/default_user.json']

    def setUp(self):
        self.url = reverse('login')
        self.user = User.objects.get(username='MichaelScott')

    def test_log_in_url(self):
        self.assertEqual(self.url, '/login/')

    def test_unsuccessful_log_in(self):
        form_input = {'username': 'MichaelScott', 'password': 'WrongPassword123'}
        response = self.client.post(self.url, form_input)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'login.html')
        self.assertFalse(self._is_logged_in())

    def test_successful_log_in(self):
        form_input = {'username': 'MichaelScott', 'password': 'Password123'}
        response = self.client.post(self.url, form_input)
        self.assertEqual(response.status_code, 302)
        self.assertTrue(self._is_logged_in())
        self.assertRedirects(response, reverse('home'))

    def test_successful_log_in_with_redirect(self):
        redirect_url = reverse('profile', kwargs={'pk': self.user.id})
        form_input = {'username': 'MichaelScott', 'password': 'Password123', 'next': redirect_url}
        response = self.client.post(self.url, form_input, follow=True)
        self.assertTrue(self._is_logged_in())
        self.assertRedirects(response, redirect_url, status_code=302, target_status_code=200)

    def test_get_log_in(self):
        response = self.client.get(self.url)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'login.html')

    def test_get_log_in_with_redirect(self):
        redirect_url = reverse('profile', kwargs={'pk': self.user.id})
        self.url = reverse('login') + '?next=' + redirect_url
        response = self.client.get(self.url)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'login.html')
        self.assertEqual(response.context['next'], redirect_url)

    def test_get_log_in_redirects_to_home_if_logged_in(self):
        self.client.force_login(self.user)
        response = self.client.get(self.url, follow=True)
        self.assertRedirects(response, reverse('home'), status_code=302, target_status_code=200)
        self.assertTemplateUsed(response, 'home.html')

    def test_post_log_in_redirects_to_home_when_logged_in(self):
        self.client.login(username=self.user.username, password='Password123')
        form_input = {'username': 'wronguser', 'password': 'WrongPassword123'}
        response = self.client.post(self.url, form_input, follow=True)
        redirect_url = reverse('home')
        self.assertRedirects(response, redirect_url, status_code=302, target_status_code=200)
        self.assertTemplateUsed(response, 'home.html')

```

Translator/translate/tests/views/test_logout_view.py

```
"""Tests of the log out view."""
from django.test import TestCase
from django.urls import reverse
from translate.models import User
from translate.tests.helpers import LogInTester


class LogOutViewTestCase(TestCase, LogInTester):
    """Tests of the logout view."""

    fixtures = ['translate/tests/fixtures/default_user.json']
```

```
12
13      def setUp(self):
14          self.url = reverse('logout')
15          self.user = User.objects.get(username='MichaelScott')
16
17      def test_log_out_url(self):
18          self.assertEqual(self.url, '/logout/')
19
20      def test_get_log_out(self):
21          self.client.login(username='MichaelScott', password='Password123')
22          self.assertTrue(self._is_logged_in())
23          response = self.client.get(self.url, follow=True)
24          response_url = reverse('home')
25          self.assertRedirects(response, response_url, status_code=302, target_status_code=200)
26          self.assertTemplateUsed(response, 'home.html')
27          self.assertFalse(self._is_logged_in())
28
29      def test_log_out_redirects_when_not_logged_in(self):
30          self.assertFalse(self._is_logged_in())
31          response = self.client.get(self.url, follow=True)
32          response_url = reverse('login') + '?next=' + self.url
33          self.assertRedirects(response, response_url, status_code=302, target_status_code=200)
34          self.assertTemplateUsed(response, 'login.html')
35          self.assertFalse(self._is_logged_in())
36
```

Translator/translate/tests/views/test_profile_view.py

```
1  """Tests of the show user view."""
2  from django.test import TestCase
3  from django.urls import reverse
4  from translate.models import User, Translation
5
6
7  class ShowUserViewTestCase(TestCase):
8      """Tests of the profile view."""
9
10     fixtures = [
11         'translate/tests/fixtures/default_user.json',
12         'translate/tests/fixtures/other_users.json',
13         'translate/tests/fixtures/default_translation.json',
14         'translate/tests/fixtures/other_translations.json',
15     ]
16
17     def setUp(self):
18         self.user = User.objects.get(username='MichaelScott')
19         self.target_user = User.objects.get(username='DwightSchrute')
20         self.url = reverse('profile', kwargs={'pk': self.user.id})
21
22     def test_show_user_url(self):
23         self.assertEqual(self.url, f'/profile/{self.user.id}/')
24
25     def test_get_show_user(self):
26         self.client.login(username=self.user.username, password='Password123')
27         response = self.client.get(self.url)
28         self.assertEqual(response.status_code, 200)
29         self.assertTemplateUsed(response, 'profile.html')
30
31     def test_get_show_user_redirects_when_not_logged_in(self):
32         redirect_url = reverse('login') + '?next=' + self.url
33         response = self.client.get(self.url)
34         self.assertRedirects(response, redirect_url, status_code=302, target_status_code=200)
35
36     def test_get_show_user_with_valid_id_fails_if_not_self(self):
37         self.client.login(username=self.user.username, password='Password123')
38         url = reverse('profile', kwargs={'pk': self.target_user.id})
39         response = self.client.get(url)
40         self.assertRedirects(response, reverse('home'), status_code=302, target_status_code=200)
41
42     def test_get_show_user_with_valid_id_of_self(self):
43         self.client.login(username=self.user.username, password='Password123')
44         response = self.client.get(self.url)
45         self.assertEqual(response.status_code, 200)
46         self.assertTemplateUsed(response, 'profile.html')
47         self.assertContains(response, "MichaelScott")
48         self.assertContains(response, "michaelscott@dundermifflin.com")
```

```
        translations = Translation.objects.filter(user_id=self.user.id)
        for translation in translations:
            self.assertContains(response, translation.text_display() + ' -> ' +
translation.translated_text_display())

    def test_get_show_user_with_invalid_id(self):
        self.client.login(username=self.user.username, password='Password123')
        url = reverse('profile', kwargs={'pk': self.user.id + 9999})
        response = self.client.get(url)
        self.assertRedirects(response, reverse('home'), status_code=302, target_status_code=200)

```

Translator/translate/tests/views/test_sign_up_view.py

```
from django.contrib.auth.hashers import check_password
from django.test import TestCase
from django.urls import reverse
from translate.forms import SignUpForm
from translate.models import User
from translate.tests.helpers import LogInTester

"""Tests of the sign up view. The structure of the tests was inspired by the clucker project from 5CCS2SEG."""


class SignUpViewTestCase(TestCase, LogInTester):
    fixtures = ['translate/tests/fixtures/default_user.json']

    def setUp(self):
        self.url = reverse('signup')
        self.user = User.objects.get(username='MichaelScott')
        self.form_input = {
            'username': 'DwightSchrute',
            'email': 'dwightschrute@dundermifflin.com',
            'password1': 'JimHalpertIsTheBest123',
            'password2': 'JimHalpertIsTheBest123'
        }

    def test_sign_up_url(self):
        self.assertEqual(self.url, '/signup/')

    def test_get_sign_up(self):
        response = self.client.get(self.url)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'signup.html')
        form = response.context['form']
        self.assertTrue(isinstance(form, SignUpForm))
        self.assertFalse(form.is_bound)

    def test_unsuccessful_sign_up(self):
        self.form_input['username'] = 'BAD USERNAME'
        before_count = User.objects.count()
        response = self.client.post(self.url, self.form_input)
        after_count = User.objects.count()
        self.assertEqual(after_count, before_count)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'signup.html')
        form = response.context['form']
        self.assertTrue(isinstance(form, SignUpForm))
        self.assertTrue(form.is_bound)
        self.assertFalse(self._is_logged_in())

    def test_successful_sign_up(self):
        before_count = User.objects.count()
        response = self.client.post(self.url, self.form_input, follow=True)
        response_url = reverse('home')
        self.assertRedirects(response, response_url, status_code=302, target_status_code=200)
        after_count = User.objects.count()
        self.assertEqual(after_count, before_count + 1)
        self.assertTemplateUsed(response, 'home.html')

        user = User.objects.get(username='DwightSchrute')
        self.assertEqual(user.email, 'dwightschrute@dundermifflin.com')
        is_password_correct = check_password(self.form_input['password1'], user.password)
        self.assertTrue(is_password_correct)
        self.assertTrue(self._is_logged_in())

    def test_get_sign_up_redirects_to_home_when_logged_in(self):
```

```
        self.client.login(username=self.user.username, password='Password123')
        response = self.client.get(self.url, follow=True)
        redirect_url = reverse('home')
        self.assertTemplateUsed(response, 'home.html')
        self.assertRedirects(response, redirect_url, status_code=302, target_status_code=200)

    def test_post_sign_up_redirects_to_home_when_logged_in(self):
        before_count = User.objects.count()
        self.client.login(username=self.user.username, password='Password123')
        response = self.client.post(self.url, self.form_input, follow=True)
        after_count = User.objects.count()
        self.assertEqual(after_count, before_count)
        redirect_url = reverse('home')
        self.assertTemplateUsed(response, 'home.html')
        self.assertRedirects(response, redirect_url, status_code=302, target_status_code=200)

```

Translator/translate/views.py

```
from django.contrib.auth import login
from django.contrib.auth.forms import PasswordChangeForm, PasswordResetForm
from django.core.exceptions import ImproperlyConfigured
from django.http import Http404
from django.shortcuts import render, redirect
from django.contrib.auth.views import LogoutView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views import View
from django.views.generic import DetailView, CreateView
from django.urls import reverse
from django.contrib import messages

from translate.forms import LogInForm, SignUpForm, TranslatorForm
from translate.models import User, Translation
from translator import settings


class LoginProhibitedMixin:
    """The LoginProhibitedMixin is used to prevent logged in users from accessing the login and signup pages.
    It is adapted from the Clucker project from 5CCS2SEG."""
    redirect_when_logged_in_url = None

    def dispatch(self, *args, **kwargs):
        if self.request.user.is_authenticated:
            return self.handle_already_logged_in(*args, **kwargs)
        return super().dispatch(*args, **kwargs)

    def handle_already_logged_in(self, *args, **kwargs):
        url = self.get_redirect_when_logged_in_url()
        return redirect(url)

    def get_redirect_when_logged_in_url(self):
        if self.redirect_when_logged_in_url is None:
            raise ImproperlyConfigured('LoginProhibitedMixin requires either a value for '  # pragma: no cover
                                       'redirect_when_logged_in_url or an implementation '
                                       'for get_redirect_when_logged_in_url().')
        else:
            return self.redirect_when_logged_in_url


class HomeView(View):
    """View that shows the home page."""
    template_name = 'home.html'
    translated_text = ''
    text = ''
    form = TranslatorForm()

    def get(self, request):
        translation_pk = request.GET.get('translation', None)
        if translation_pk:
            translation = Translation.objects.get(pk=translation_pk)
            self.form = TranslatorForm(initial={'text': translation.text, 'language': translation.input_language})

        return self.render()


    def post(self, request):
        self.form = TranslatorForm(request.POST)
```

```
        if self.form.is_valid():
            self.translated_text = self.form.translate()

            # try to find an existing translation
            try:
                translation = Translation.objects.get(
                    text=self.form.cleaned_data['text'],
                    input_language=self.form.cleaned_data['language'],
                    user_id=request.user.pk
                )
                translation.translated_text = self.translated_text
                translation.save()
            except Translation.DoesNotExist:
                # create a new translation
                translation = Translation.objects.create(
                    text=self.form.cleaned_data['text'],
                    translated_text=self.translated_text,
                    user_id=request.user.pk,
                    input_language=self.form.cleaned_data['language']
                )

            translation.save()
            self.text = self.form.cleaned_data['text']

        return self.render()

    def render(self):
        return render(self.request, self.template_name,
                      {'form': self.form, 'text': self.text, 'translations': self.translated_text,
                       'second_language': self.get_second_language()})

    def get_second_language(self):
        try:
            if self.form.cleaned_data['language'] == 'English':
                return 'Polish'
            else:
                return 'English'
        except AttributeError:
            return 'Polish'


class LogInView(LoginProhibitedMixin, View):
    """View that logs in user. The structure was inspired by the Clucker project from 5CCS2SEG."""
    template_name = 'login.html'
    next_page = '/'
    form_class = LogInForm
    http_method_names = ['get', 'post']
    redirect_when_logged_in_url = settings.LOGIN_REDIRECT_URL

    def get(self, request):
        self.next_page = request.GET.get('next') or '/'
        return self.render()

    def post(self, request):
        form = self.form_class(request.POST)
        self.next_page = request.POST.get('next') or settings.LOGIN_REDIRECT_URL
        user = form.get_user()
        if user is not None:
            login(request, user)
            return redirect(self.next_page)
        messages.add_message(request, messages.ERROR, 'The provided credentials were invalid!')
        return self.render()

    def render(self):
        return render(self.request, self.template_name, {'form': self.form_class, 'next': self.next_page})


class LogOutView(LoginRequiredMixin, LogoutView):
    next_page = '/'
    success_url = '/'
    template_name = 'logout.html'
    login_url = '/login/'


class SignUpView(LoginProhibitedMixin, CreateView):
    form_class = SignUpForm
    template_name = 'signup.html'
    redirect_when_logged_in_url = settings.LOGIN_REDIRECT_URL
    success_url = settings.LOGIN_REDIRECT_URL
```

```
    def form_valid(self, form):
        form.save()
        login(self.request, form.instance)
        return redirect(self.get_success_url())

    def get_success_url(self):
        return reverse(self.success_url)


class ProfileView(LoginRequiredMixin, DetailView):
    model = User
    template_name = 'profile.html'
    login_url = '/login/'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        user = self.get_object()
        # can only view your own profile
        if user != self.request.user:
            raise Http404
        translations = Translation.objects.filter(user=user)
        # sort translations by date
        translations = translations.order_by('-updated_at')
        context['translations'] = translations
        return context

    def get(self, request, *args, **kwargs):
        try:
            return super(ProfileView, self).get(request, *args, **kwargs)
        except Http404:
            return redirect('home')


class PasswordChangeView(LoginRequiredMixin, View):
    template_name = 'password_change.html'
    success_url = '/'
    form_class = PasswordChangeForm

    def get(self, request):
        form = self.form_class(request.user)
        return render(request, self.template_name, {'form': form})

    def post(self, request):
        form = self.form_class(request.user, request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect(self.success_url)
        messages.add_message(request, messages.ERROR, 'The provided credentials were invalid!')
        return render(request, self.template_name, {'form': form})

```

Translator/translator/__init__.py

```

```

Translator/translator/asgi.py

```
"""
ASGI config for translator project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/4.1/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'translator.settings')

application = get_asgi_application()
```

Translator/translator/settings.py

```
"""
Django settings for translator project.

Generated by 'django-admin startproject' using Django 4.1.5.

For more information on this file, see
https://docs.djangoproject.com/en/4.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.1/ref/settings/
"""
from os import path
from pathlib import Path
from django.contrib.messages import constants as message_constants

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-2_3%pqhek9kyo-!wx@okx4j%ueq#%fqb91$1$q=j&9a=a$49z@'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'translate',
    'widget_tweaks',
    'django_extensions',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'translator.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

```

```
WSGI_APPLICATION = 'translator.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/4.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/4.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS = [
    path.join(BASE_DIR, 'static')
]

# Default primary key field type
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

# User model for the purpose of authentication
AUTH_USER_MODEL = 'translate.User'

APPEND_SLASH = False

LOGIN_REDIRECT_URL = 'home'

# Message level tags should use Bootstrap terms
MESSAGE_TAGS = {
    message_constants.DEBUG: 'dark',
    message_constants.ERROR: 'danger',
}

```

Translator/translator/urls.py

```
"""translator URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
```

```
    https://docs.djangoproject.com/en/4.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

from translate import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.HomeView.as_view(), name='home'),
    path('accounts/', include('django.contrib.auth.urls')),
    path('signup/', views.SignUpView.as_view(), name='signup'),
    path('login/', views.LogInView.as_view(), name='login'),
    path('profile/<int:pk>/', views.ProfileView.as_view(), name='profile'),
    path('password_change/', views.PasswordChangeView.as_view(), name='password_change'),
    path('logout/', views.LogOutView.as_view(), name='logout'),
]

```

Translator/translator/wsgi.py

```
"""
WSGI config for translator project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/4.1/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'translator.settings')

application = get_wsgi_application()

```

Translator/translator_backend.py

```
from transformers import pipeline

print("Loading models...")
translator_en_pl = pipeline('translation', tokenizer="alirezamsh/small100",
                            model="MikolajDeja/alirezamsh-small100-en-pl-para_crawl-finetune",
                            src_lang="en", tgt_lang="pl")

translator_pl_en = pipeline("translation", model="MikolajDeja/Helsinki-NLP-opus-mt-pl-en-para_crawl-finetune",
                            tokenizer="Helsinki-NLP/opus-mt-pl-en")

print("Models loaded.")


def translateEnglishToPolish(english_sentence):
    return translator_en_pl(english_sentence.split("\n"))


def translatePolishToEnglish(polish_sentence):
    return translator_pl_en(polish_sentence.split("\n"))

```