Amazon has released a 51-languages parallel database called MASSIVE to the public domain. The same dataset is also available in Huggingface at https://huggingface.co/datasets/qanastek/MASSIVE . The dataset typically consists of sentences from 51 languages structured in a JSON format. The JSON structure contains the following headings (features). ['id', 'locale', 'partition', 'scenario', 'intent', 'utt', 'annot_utt', 'tokens', 'ner_tags', 'worker_id', 'slot_method', 'judgments'] From those headings, we are interested only in the following subset {'locale', 'partition', 'utt', 'tokens'}. 'locale' represents the language-country pair, 'partition' represents where the sentence is coming from amidst {'train', 'test', 'validation'}, 'utt' represents the actual sentence, and finally 'tokens' represents the split tokens from the sentence. We are going to build a language classifier the covers all the languages with roman letters. There is already a classifier built on this dataset for all the 51 languages using transformers, which appears to be SOTA. https://huggingface.co/qanastek/51-languages-classifier. Our goal is not to compete with transformers, rather we are going to use this exercise to learn and overcome the challenges in dealing with multilingual datasets.

Task 1

Let's construct a dataset ourselves for with a subset of languages that are roman-script based. The following are the locales that we want to consider in our dataset [27 languages]. af-ZA, da-DK, de-DE, en-US, es-ES, fr-FR, fi-FI, hu-HU, is-IS, it-IT, jv-ID, lv-LV, ms-MY, nb-NO, nl-NL, pl-PL, pt-PT, ro-RO, ru-RU, sl-SL, sv-SE, sq-AL, sw-KE, tl-PH, tr-TR, vi-VN, cy-GB Programmatically, extract the utterances "utt" from the dataset for each of the above languages. You can choose between your tokenization vs the preexisting tokens. By the end of this step, you should have 27 files (one for each language) with one sentence per line. Typically, all the 27 files will end up have the same number of lines as the dataset is a parallel-corpus. Besides simple English like characters, you may encounter other characters and characters with accents. You may choose to deaccent the characters if accents are not useful in your method. Choose wisely

```
pip install datasets
```

```
Collecting datasets
  Downloading datasets-3.0.0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from datasets) (3.16.1)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.10/dist-packages (from datasets) (1.26.4)
Collecting pyarrow>=15.0.0 (from datasets)
  Downloading pyarrow-17.0.0-cp310-cp310-
manylinux_2_28_x86_64.whl.metadata (3.3 kB)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in
/usr/local/lib/python3.10/dist-packages (from datasets) (2.1.4)
Requirement already satisfied: requests>=2.32.2 in
/usr/local/lib/python3.10/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in
/usr/local/lib/python3.10/dist-packages (from datasets) (4.66.5)
```

```
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl.metadata (7.2
kB)
Requirement already satisfied: fsspec<=2024.6.1,>=2023.1.0 in
/usr/local/lib/python3.10/dist-packages (from
fsspec[http]<=2024.6.1,>=2023.1.0->datasets) (2024.6.1)
Requirement already satisfied: aiohttp in
/usr/local/lib/python3.10/dist-packages (from datasets) (3.10.5)
Requirement already satisfied: huggingface-hub>=0.22.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (0.24.7)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from datasets) (24.1)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.10/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(2.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.3.1)
Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(6.1.0)
Requirement already satisfied: yarl<2.0,>=1.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.11.1)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.22.0-
>datasets) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
```

```
>datasets) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (2024.8.30)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2024.2)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2024.1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas->datasets) (1.16.0)
Downloading datasets-3.0.0-py3-none-any.whl (474 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 474.3/474.3 kB 15.4 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 4.4 MB/s eta
0:00:00
anylinux_2_28_x86_64.whl (39.9 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 39.9/39.9 MB 12.0 MB/s eta
0:00:00
ultiprocess-0.70.16-py310-none-any.whl (134 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 7.0 MB/s eta
0:00:00
anylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 5.2 MB/s eta
0:00:00
ultiprocess, datasets
  Attempting uninstall: pyarrow
    Found existing installation: pyarrow 14.0.2
    Uninstalling pyarrow-14.0.2:
      Successfully uninstalled pyarrow-14.0.2
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
cudf-cu12 24.4.1 requires pyarrow<15.0.0a0,>=14.0.1, but you have
pyarrow 17.0.0 which is incompatible.
Successfully installed datasets-3.0.0 dill-0.3.8 multiprocess-0.70.16
pyarrow-17.0.0 xxhash-3.5.0
```

```
import os
from datasets import load_dataset
import unicodedata
```

Train dataset

```
locales = ['af-ZA', 'da-DK', 'de-DE', 'en-US', 'es-ES', 'fr-FR', 'fi-
FI', 'hu-HU',
           'is-IS', 'it-IT', 'jv-ID', 'lv-LV', 'ms-MY', 'nb-NO', 'nl-
NL', 'pl-PL',
           'pt-PT', 'ro-RO', 'ru-RU', 'sl-SL', 'sv-SE', 'sq-AL', 'sw-
KE', 'tl-PH',
           'tr-TR', 'vi-VN', 'cy-GB']

data=load_dataset('qanastek/MASSIVE')

output_directory='all_language_files'
os.makedirs(output_directory,exist_ok=True)

def extract_sentence(locale,data):
  output_file=os.path.join(output_directory,f"{locale}.txt")
  local_data=data.filter(lambda x:x['locale']==locale)
  with open(output_file,'w',encoding='utf-8') as file:
    for ex in local_data['train']:
      utt=ex['utt']
      file.write(utt + '\n')
  print(f"finished extracting sentences for {locale}.")


for locale in locales:
  extract_sentence(locale,data)

print("Extraction complete for all 27 language")
```

{"model_id":"5362964b3b314cfebe7f0e98cc021d4e","version_major":2,"vers
ion_minor":0}

{"model_id":"0e1141050f7245d4b670b92e7cff27e4","version_major":2,"vers
ion_minor":0}

{"model_id":"5e874fd8c15f45728c283ebaee54b472","version_major":2,"vers
ion_minor":0}

```
finished extracting sentences for af-ZA.
```

{"model_id":"8b03f4a0190c4e6dbffff2a0eecc72b2","version_major":2,"vers
ion_minor":0}

{"model_id":"12da62f230f941f2a3e51fca4ad21105","version_major":2,"vers
ion_minor":0}

{"model_id":"c87e56e2eb22440aabc0ca23df4da299","version_major":2,"vers
ion_minor":0}

```
finished extracting sentences for da-DK.
```

{"model_id":"720fcbb899624adbb45ee896b4b2801d","version_major":2,"vers
ion_minor":0}

{"model_id":"f6328b17307e4ddebe8276972c35a5b0","version_major":2,"version_minor":0}

{"model_id":"59325978826445edaeea8f98b85b377e","version_major":2,"version_minor":0}

finished extracting sentences for de-DE.

{"model_id":"b3b7ca0c2e704f2a8ff9cc66c4cb5ece","version_major":2,"version_minor":0}

{"model_id":"5722e28bbecb44079133943b1559a01f","version_major":2,"version_minor":0}

{"model_id":"26885ae9631d4319a7ab9a4a425731f1","version_major":2,"version_minor":0}

finished extracting sentences for en-US.

{"model_id":"27f4d590ece24f2b88dbae7684150c89","version_major":2,"version_minor":0}

{"model_id":"79bcf5cfa386494cb343bce55b4adc8e","version_major":2,"version_minor":0}

{"model_id":"16a84a6c8a2b48f4b4389f51303db750","version_major":2,"version_minor":0}

finished extracting sentences for es-ES.

{"model_id":"a14d598ee3eb480b9d9168ad2088ed7d","version_major":2,"version_minor":0}

{"model_id":"b203ec8380ef49a4b51c094da67b1ee4","version_major":2,"version_minor":0}

{"model_id":"c1ed554fb0484c0b8162b84174e17cdf","version_major":2,"version_minor":0}

finished extracting sentences for fr-FR.

{"model_id":"cc6ba088e0464168b6b7e846db2f1c17","version_major":2,"version_minor":0}

{"model_id":"622a036f0f5243299cac8a0350d863c1","version_major":2,"version_minor":0}

{"model_id":"83a6792a628a40208b00093220fcf7d5","version_major":2,"version_minor":0}

finished extracting sentences for fi-FI.

{"model_id":"fd502277aa5949ac9460002ec12e25e2","version_major":2,"version_minor":0}

{"model_id":"0f5fb4a1421242c3b2a27be71165a92c","version_major":2,"version_minor":0}

{"model_id":"1d271d605ec849f7b657e5e8b8fd60c0","version_major":2,"version_minor":0}

finished extracting sentences for hu-HU.

{"model_id":"233bfc7dc36e4ae987ba18ada16ac125","version_major":2,"version_minor":0}

{"model_id":"01f57233004049188e84e6e749b818fa","version_major":2,"version_minor":0}

{"model_id":"1c7afe5382654e628e7c19ba6dddea1c","version_major":2,"version_minor":0}

finished extracting sentences for is-IS.

{"model_id":"6913c49f6c1145daa82ecfaa8b89d9df","version_major":2,"version_minor":0}

{"model_id":"1d861e009f3d4bfba352dbdcbbae1f5d","version_major":2,"version_minor":0}

{"model_id":"cf2eeeceeb5a4e7ab9a18b05230fd8ae","version_major":2,"version_minor":0}

finished extracting sentences for it-IT.

{"model_id":"edafe5f27ddb42dc8d4ef8adcf4b1eaa","version_major":2,"version_minor":0}

{"model_id":"082113c23d5345d5af18f84af96c9706","version_major":2,"version_minor":0}

{"model_id":"27ff5d3eb8a54cd884acdbd917bc04f8","version_major":2,"version_minor":0}

finished extracting sentences for jv-ID.

{"model_id":"3c7002ee70da4d4f82ff880087aa0198","version_major":2,"version_minor":0}

{"model_id":"1e3a1fbebce4454ebf5dc784d7e299ff","version_major":2,"version_minor":0}

{"model_id":"6d85c4c55b674430b57d8b12a2b92d5c","version_major":2,"version_minor":0}

```
finished extracting sentences for lv-LV.

{"model_id":"5f45d5e20e7f4a88935a4f4947618516","version_major":2,"vers
ion_minor":0}

{"model_id":"278291b4c59c43bb8cf40193fa3489fd","version_major":2,"vers
ion_minor":0}

{"model_id":"0fa8ea483c6c4cf88c334e83fd66d319","version_major":2,"vers
ion_minor":0}

finished extracting sentences for ms-MY.

{"model_id":"669e37055c3c430195f0007f7ccf3071","version_major":2,"vers
ion_minor":0}

{"model_id":"cd308f25c1c44ea8a293a25a3eb87a3e","version_major":2,"vers
ion_minor":0}

{"model_id":"d1b3b3a7b3974789a6c578a802abc4e8","version_major":2,"vers
ion_minor":0}

finished extracting sentences for nb-NO.

{"model_id":"211baae78b8b42808fd6c229454fb5b4","version_major":2,"vers
ion_minor":0}

{"model_id":"24abdb00c8a64f66a009b1763f97dc9c","version_major":2,"vers
ion_minor":0}

{"model_id":"d9d998d29e9f4614a8ef6e76b1316f4a","version_major":2,"vers
ion_minor":0}

finished extracting sentences for nl-NL.

{"model_id":"67e41b421fdc4b33828db61d0f4fb7c5","version_major":2,"vers
ion_minor":0}

{"model_id":"6b28fc627e9a4440a56438907bc3dd5a","version_major":2,"vers
ion_minor":0}

{"model_id":"4dcd3849734a40eb88a9c0078b6011dd","version_major":2,"vers
ion_minor":0}

finished extracting sentences for pl-PL.

{"model_id":"030939a9f318451fb4b55e6402aafaaf","version_major":2,"vers
ion_minor":0}

{"model_id":"029883767ebb41819ec4d4999c0acfb0","version_major":2,"vers
ion_minor":0}
```

{"model_id":"5ab976b66fe744e69189fddf6e3f92a0","version_major":2,"version_minor":0}

finished extracting sentences for pt-PT.

{"model_id":"94594c0ecfeb49919cb9f8dfae6ee609","version_major":2,"version_minor":0}

{"model_id":"2cddbba2e442406d8bbdd69697a6ce0f","version_major":2,"version_minor":0}

{"model_id":"8363be056dc74d77a7e1c6acc1ac0070","version_major":2,"version_minor":0}

finished extracting sentences for ro-RO.

{"model_id":"a73b08b5c93140dfa82e30eea1746e14","version_major":2,"version_minor":0}

{"model_id":"9160a51bbfb34b95bd15872f8c4ffc59","version_major":2,"version_minor":0}

{"model_id":"441ca129b1484a35b0bda463e97c6d98","version_major":2,"version_minor":0}

finished extracting sentences for ru-RU.

{"model_id":"ac11077de7934fa097dfcf4451be20a7","version_major":2,"version_minor":0}

{"model_id":"ebeedb065dc64df2ab5a64d3d21613b4","version_major":2,"version_minor":0}

{"model_id":"9a3f92b21dd140e0acc686cbc3c3f81f","version_major":2,"version_minor":0}

finished extracting sentences for sl-SL.

{"model_id":"a4c847fd997d4501816a96d96c23171b","version_major":2,"version_minor":0}

{"model_id":"cd56c5e06688401581fe720d41af9e0e","version_major":2,"version_minor":0}

{"model_id":"aaf4919389c3462a819131822f78baa5","version_major":2,"version_minor":0}

finished extracting sentences for sv-SE.

{"model_id":"36320a8ca42b4a86b002904f5284773e","version_major":2,"version_minor":0}

{"model_id":"a589675fdf00472cae9ec3be60b1d1a6","version_major":2,"version_minor":0}

{"model_id":"af676b4019e740899070b721f739e330","version_major":2,"version_minor":0}

finished extracting sentences for sq-AL.

{"model_id":"cd704b442e8c4b7aa20dc2632e6290b6","version_major":2,"version_minor":0}

{"model_id":"b712e6fe81fa42239dc567df0a39f429","version_major":2,"version_minor":0}

{"model_id":"c0363cfa0d1545fc897e27f6ff3b7b41","version_major":2,"version_minor":0}

finished extracting sentences for sw-KE.

{"model_id":"ce1655cdf4544e1c87ffa08a49ca38de","version_major":2,"version_minor":0}

{"model_id":"05f84d4b16e64568b4cb2e473835c448","version_major":2,"version_minor":0}

{"model_id":"c42bb55287d646e79346c2865beb171b","version_major":2,"version_minor":0}

finished extracting sentences for tl-PH.

{"model_id":"84a839838e7143b6be6bbc5642d5f572","version_major":2,"version_minor":0}

{"model_id":"93c84ec434544ce49c85c17731d5226d","version_major":2,"version_minor":0}

{"model_id":"9380986eeedd4f39b692a6d9c8b91978","version_major":2,"version_minor":0}

finished extracting sentences for tr-TR.

{"model_id":"d45af88ee87b4b498a4668cc678bfed0","version_major":2,"version_minor":0}

{"model_id":"e6e675cb581e4cd3885ba0ab15de8cba","version_major":2,"version_minor":0}

{"model_id":"e346cc7f0f1d42f1a9299bfdb4e7289c","version_major":2,"version_minor":0}

finished extracting sentences for vi-VN.

```
{"model_id":"3492b67ef48b4810a43b9df860bef090","version_major":2,"vers
ion_minor":0}

{"model_id":"27414c5aa53641d3bbdddc924652c130","version_major":2,"vers
ion_minor":0}

{"model_id":"b1f1b775ede54253b647b260f4323f74","version_major":2,"vers
ion_minor":0}

finished extracting sentences for cy-GB.
Extraction complete for all 27 language

!zip -r all_language_files.zip /content/all_language_files

  adding: content/all_language_files/ (stored 0%)
  adding: content/all_language_files/jv-ID.txt (deflated 71%)
  adding: content/all_language_files/sw-KE.txt (deflated 73%)
  adding: content/all_language_files/vi-VN.txt (deflated 76%)
  adding: content/all_language_files/nl-NL.txt (deflated 72%)
  adding: content/all_language_files/da-DK.txt (deflated 71%)
  adding: content/all_language_files/it-IT.txt (deflated 74%)
  adding: content/all_language_files/af-ZA.txt (deflated 71%)
  adding: content/all_language_files/ms-MY.txt (deflated 74%)
  adding: content/all_language_files/ro-RO.txt (deflated 71%)
  adding: content/all_language_files/sq-AL.txt (deflated 73%)
  adding: content/all_language_files/es-ES.txt (deflated 73%)
  adding: content/all_language_files/pl-PL.txt (deflated 72%)
  adding: content/all_language_files/fi-FI.txt (deflated 72%)
  adding: content/all_language_files/de-DE.txt (deflated 72%)
  adding: content/all_language_files/lv-LV.txt (deflated 72%)
  adding: content/all_language_files/pt-PT.txt (deflated 73%)
  adding: content/all_language_files/sl-SL.txt (deflated 71%)
  adding: content/all_language_files/tr-TR.txt (deflated 72%)
  adding: content/all_language_files/cy-GB.txt (deflated 72%)
  adding: content/all_language_files/nb-NO.txt (deflated 71%)
  adding: content/all_language_files/is-IS.txt (deflated 73%)
  adding: content/all_language_files/tl-PH.txt (deflated 75%)
  adding: content/all_language_files/ru-RU.txt (deflated 80%)
  adding: content/all_language_files/sv-SE.txt (deflated 72%)
  adding: content/all_language_files/fr-FR.txt (deflated 74%)
  adding: content/all_language_files/hu-HU.txt (deflated 71%)
  adding: content/all_language_files/en-US.txt (deflated 72%)
```

Test dataset

```
locales = ['af-ZA', 'da-DK', 'de-DE', 'en-US', 'es-ES', 'fr-FR', 'fi-
FI', 'hu-HU',
           'is-IS', 'it-IT', 'jv-ID', 'lv-LV', 'ms-MY', 'nb-NO', 'nl-
NL', 'pl-PL',
           'pt-PT', 'ro-RO', 'ru-RU', 'sl-SL', 'sv-SE', 'sq-AL', 'sw-
```

```
KE', 'tl-PH',
           'tr-TR', 'vi-VN', 'cy-GB']

data=load_dataset('qanastek/MASSIVE')

output_directory='test_dataset'
os.makedirs(output_directory,exist_ok=True)

def extract_sentence(locale,data):
  output_file=os.path.join(output_directory,f"{locale}.txt")
  local_data=data.filter(lambda x:x['locale']==locale)
  with open(output_file,'w',encoding='utf-8') as file:
    for ex in local_data['test']:
      utt=ex['utt']
      file.write(utt + '\n')
  print(f"finished extracting sentences for {locale}.")


for locale in locales:
  extract_sentence(locale,data)

print("Extraction complete for all 27 language")
```

```
finished extracting sentences for af-ZA.
finished extracting sentences for da-DK.
finished extracting sentences for de-DE.
finished extracting sentences for en-US.
finished extracting sentences for es-ES.
finished extracting sentences for fr-FR.
finished extracting sentences for fi-FI.
finished extracting sentences for hu-HU.
finished extracting sentences for is-IS.
finished extracting sentences for it-IT.
finished extracting sentences for jv-ID.
finished extracting sentences for lv-LV.
finished extracting sentences for ms-MY.
finished extracting sentences for nb-NO.
finished extracting sentences for nl-NL.
finished extracting sentences for pl-PL.
finished extracting sentences for pt-PT.
finished extracting sentences for ro-RO.
finished extracting sentences for ru-RU.
finished extracting sentences for sl-SL.
finished extracting sentences for sv-SE.
finished extracting sentences for sq-AL.
finished extracting sentences for sw-KE.
finished extracting sentences for tl-PH.
finished extracting sentences for tr-TR.
finished extracting sentences for vi-VN.
```

```
finished extracting sentences for cy-GB.
Extraction complete for all 27 language
```

```
!zip -r test_dataset.zip /content/test_dataset
```

```
  adding: content/test_dataset/ (stored 0%)
  adding: content/test_dataset/jv-ID.txt (deflated 67%)
  adding: content/test_dataset/sw-KE.txt (deflated 70%)
  adding: content/test_dataset/vi-VN.txt (deflated 73%)
  adding: content/test_dataset/nl-NL.txt (deflated 69%)
  adding: content/test_dataset/da-DK.txt (deflated 68%)
  adding: content/test_dataset/it-IT.txt (deflated 70%)
  adding: content/test_dataset/af-ZA.txt (deflated 68%)
  adding: content/test_dataset/ms-MY.txt (deflated 71%)
  adding: content/test_dataset/ro-RO.txt (deflated 67%)
  adding: content/test_dataset/sq-AL.txt (deflated 70%)
  adding: content/test_dataset/es-ES.txt (deflated 70%)
  adding: content/test_dataset/pl-PL.txt (deflated 68%)
  adding: content/test_dataset/fi-FI.txt (deflated 69%)
  adding: content/test_dataset/de-DE.txt (deflated 69%)
  adding: content/test_dataset/lv-LV.txt (deflated 69%)
  adding: content/test_dataset/pt-PT.txt (deflated 69%)
  adding: content/test_dataset/sl-SL.txt (deflated 67%)
  adding: content/test_dataset/tr-TR.txt (deflated 68%)
  adding: content/test_dataset/cy-GB.txt (deflated 68%)
  adding: content/test_dataset/nb-NO.txt (deflated 68%)
  adding: content/test_dataset/is-IS.txt (deflated 70%)
  adding: content/test_dataset/tl-PH.txt (deflated 72%)
  adding: content/test_dataset/ru-RU.txt (deflated 78%)
  adding: content/test_dataset/sv-SE.txt (deflated 68%)
  adding: content/test_dataset/fr-FR.txt (deflated 71%)
  adding: content/test_dataset/hu-HU.txt (deflated 68%)
  adding: content/test_dataset/en-US.txt (deflated 68%)
```

Validation Dataset

```
locales = ['af-ZA', 'da-DK', 'de-DE', 'en-US', 'es-ES', 'fr-FR', 'fi-
FI', 'hu-HU',
           'is-IS', 'it-IT', 'jv-ID', 'lv-LV', 'ms-MY', 'nb-NO', 'nl-
NL', 'pl-PL',
           'pt-PT', 'ro-RO', 'ru-RU', 'sl-SL', 'sv-SE', 'sq-AL', 'sw-
KE', 'tl-PH',
           'tr-TR', 'vi-VN', 'cy-GB']

data=load_dataset('qanastek/MASSIVE')

output_directory='validation_dataset'
os.makedirs(output_directory,exist_ok=True)

def extract_sentence(locale,data):
```

```
  output_file=os.path.join(output_directory,f"{locale}.txt")
  local_data=data.filter(lambda x:x['locale']==locale)
  with open(output_file,'w',encoding='utf-8') as file:
    for ex in local_data['validation']:
      utt=ex['utt']
      file.write(utt + '\n')
  print(f"finished extracting sentences for {locale}.")


for locale in locales:
  extract_sentence(locale,data)

print("Extraction complete for all 27 language")
```

```
finished extracting sentences for af-ZA.
finished extracting sentences for da-DK.
finished extracting sentences for de-DE.
finished extracting sentences for en-US.
finished extracting sentences for es-ES.
finished extracting sentences for fr-FR.
finished extracting sentences for fi-FI.
finished extracting sentences for hu-HU.
finished extracting sentences for is-IS.
finished extracting sentences for it-IT.
finished extracting sentences for jv-ID.
finished extracting sentences for lv-LV.
finished extracting sentences for ms-MY.
finished extracting sentences for nb-NO.
finished extracting sentences for nl-NL.
finished extracting sentences for pl-PL.
finished extracting sentences for pt-PT.
finished extracting sentences for ro-RO.
finished extracting sentences for ru-RU.
finished extracting sentences for sl-SL.
finished extracting sentences for sv-SE.
finished extracting sentences for sq-AL.
finished extracting sentences for sw-KE.
finished extracting sentences for tl-PH.
finished extracting sentences for tr-TR.
finished extracting sentences for vi-VN.
finished extracting sentences for cy-GB.
Extraction complete for all 27 language
```

```
!zip -r validation_dataset.zip /content/validation_dataset
```

```
  adding: content/validation_dataset/ (stored 0%)
  adding: content/validation_dataset/jv-ID.txt (deflated 67%)
  adding: content/validation_dataset/sw-KE.txt (deflated 69%)
  adding: content/validation_dataset/vi-VN.txt (deflated 72%)
  adding: content/validation_dataset/nl-NL.txt (deflated 68%)
```

```
  adding: content/validation_dataset/da-DK.txt (deflated 67%)
  adding: content/validation_dataset/it-IT.txt (deflated 69%)
  adding: content/validation_dataset/af-ZA.txt (deflated 67%)
  adding: content/validation_dataset/ms-MY.txt (deflated 70%)
  adding: content/validation_dataset/ro-RO.txt (deflated 66%)
  adding: content/validation_dataset/sq-AL.txt (deflated 69%)
  adding: content/validation_dataset/es-ES.txt (deflated 69%)
  adding: content/validation_dataset/pl-PL.txt (deflated 67%)
  adding: content/validation_dataset/fi-FI.txt (deflated 68%)
  adding: content/validation_dataset/de-DE.txt (deflated 68%)
  adding: content/validation_dataset/lv-LV.txt (deflated 68%)
  adding: content/validation_dataset/pt-PT.txt (deflated 68%)
  adding: content/validation_dataset/sl-SL.txt (deflated 67%)
  adding: content/validation_dataset/tr-TR.txt (deflated 67%)
  adding: content/validation_dataset/cy-GB.txt (deflated 67%)
  adding: content/validation_dataset/nb-NO.txt (deflated 67%)
  adding: content/validation_dataset/is-IS.txt (deflated 69%)
  adding: content/validation_dataset/tl-PH.txt (deflated 71%)
  adding: content/validation_dataset/ru-RU.txt (deflated 77%)
  adding: content/validation_dataset/sv-SE.txt (deflated 68%)
  adding: content/validation_dataset/fr-FR.txt (deflated 70%)
  adding: content/validation_dataset/hu-HU.txt (deflated 67%)
  adding: content/validation_dataset/en-US.txt (deflated 67%)
```

Task 2

Build a multinomial Naive Bayes classifier on your 27 language dataset using the 'training' partition of the dataset. Finetune the model with the validation partition. Finally, report the performance metrics for all the three partitions.

```
from google.colab import drive
import zipfile
import os

drive.mount('/content/drive')
zip_path='/content/drive/MyDrive/all_language_files.zip'
with zipfile.ZipFile(zip_path,'r') as zipref:
  zipref.extractall('/content/train_language_dataset')

zip_path='/content/drive/MyDrive/test_dataset.zip'
with zipfile.ZipFile(zip_path,'r') as zipref:
  zipref.extractall('/content/test_language_dataset')

zip_path='/content/drive/MyDrive/validation_dataset.zip'
with zipfile.ZipFile(zip_path,'r') as zipref:
  zipref.extractall('/content/validation_language_dataset')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).

import os

def get_data_from_directory(directory):
  X,y=[],[]
  for filename in os.listdir(directory):
    if filename.endswith(".txt"):
      label=filename.split(".")[0]
      with open(os.path.join(directory,filename),'r',encoding='utf-8')
as file:
        for line in file:
          X.append(line.strip())
          y.append(label)
  return X, y

train_dir='/content/train_language_dataset/all_language_files'
test_dir='/content/test_language_dataset/test_dataset'
validation_dir='/content/validation_language_dataset/validation_datase
t'

X_train,y_train=get_data_from_directory(train_dir)
X_test,y_test=get_data_from_directory(test_dir)
X_validation,y_validation=get_data_from_directory(validation_dir)

# concatenating the training ,test and validation dataset into a
single dataset say X and Y
X=X_train+X_test+X_validation
X1=X
Y=y_train+y_test+y_validation
Y1=Y

import pandas as pd
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score,classification_report
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
label_encoder.fit(Y)
y_train = label_encoder.transform(y_train)
y_validation = label_encoder.transform(y_validation)
y_test = label_encoder.transform(y_test)

tfidf=TfidfVectorizer(max_features=10000)
X_train=tfidf.fit_transform(X_train)
X_test=tfidf.transform(X_test)
X_validation=tfidf.transform(X_validation)
X=tfidf.transform(X)
```

```
# X_train_tran=tfidf.fit_transform(X_train)
#X_test=tfidf.fit_transform(X_test)
#X_validation=tfidf.fit_transform(X_validation)
```

Tune the parameter

```
param_grid={
    'alpha':[0.01,0.1,0.5,1.0,2.0],
    'fit_prior':[True,False]
}

model=MultinomialNB()
grid_search=GridSearchCV(model,param_grid,cv=5,scoring='accuracy')

grid_search.fit(X_train,y_train)
print("Best parameters:",grid_search.best_params_)
```

```
Best parameters: {'alpha': 0.1, 'fit_prior': True}
```

Train the model with the best parameters on the training dataset

```
best_param=grid_search.best_params_
final_model=MultinomialNB(alpha=grid_search.best_params_['alpha'],fit_
prior=grid_search.best_params_['fit_prior'])
final_model.fit(X,Y)
```

```
MultinomialNB(alpha=0.1)
```

Evaluation on training,validation and test dataset

```
y_train_pred = final_model.predict(X_train)
y_train_pred = label_encoder.transform(y_train_pred)
print("Train Accuracy:", accuracy_score(y_train, y_train_pred))
print("Classification Report (Train Set):\n",
classification_report(y_train, y_train_pred))

y_val_pred = final_model.predict(X_validation)
y_val_pred = label_encoder.transform(y_val_pred)
print("Validation Accuracy:", accuracy_score(y_validation,
y_val_pred))
print("Classification Report (Validation Set):\n",
classification_report(y_validation, y_val_pred))

y_test_pred = final_model.predict(X_test)
y_test_pred = label_encoder.transform(y_test_pred)
print("Test Accuracy:", accuracy_score(y_test, y_test_pred))
print("Classification Report (Test Set):\n",
classification_report(y_test, y_test_pred))
```

```
Train Accuracy: 0.9606179916237237
Classification Report (Train Set):
              precision    recall  f1-score   support

           0       0.64      0.97      0.77     11514
           1       0.99      0.97      0.98     11514
           2       0.93      0.91      0.92     11514
           3       0.99      0.97      0.98     11514
           4       0.96      0.98      0.97     11514
           5       0.96      0.96      0.96     11514
           6       0.99      0.94      0.97     11514
           7       0.99      0.98      0.98     11514
           8       0.98      0.92      0.95     11514
           9       0.99      0.96      0.98     11514
          10       0.97      0.97      0.97     11514
          11       0.99      0.97      0.98     11514
          12       0.99      0.96      0.97     11514
          13       0.98      0.98      0.98     11514
          14       0.92      0.91      0.92     11514
          15       0.97      0.95      0.96     11514
          16       0.98      0.95      0.97     11514
          17       0.98      0.96      0.97     11514
          18       0.99      0.96      0.98     11514
          19       1.00      0.96      0.98     11514
          20       0.98      0.96      0.97     11514
          21       0.99      0.97      0.98     11514
          22       0.98      0.96      0.97     11514
          23       0.99      0.98      0.99     11514
          24       0.99      0.99      0.99     11514
          25       0.99      0.95      0.97     11514
          26       1.00      1.00      1.00     11514

    accuracy                           0.96    310878
   macro avg       0.97      0.96      0.96    310878
weighted avg       0.97      0.96      0.96    310878

Validation Accuracy: 0.9561312419157968
Classification Report (Validation Set):
              precision    recall  f1-score   support

           0       0.61      0.97      0.75      2033
           1       0.99      0.97      0.98      2033
           2       0.93      0.90      0.91      2033
           3       0.99      0.96      0.98      2033
           4       0.96      0.97      0.97      2033
           5       0.96      0.96      0.96      2033
           6       0.98      0.94      0.96      2033
           7       0.98      0.97      0.98      2033
           8       0.99      0.92      0.95      2033
           9       0.99      0.96      0.98      2033
```

```
           10       0.98      0.96      0.97      2033
           11       0.98      0.97      0.98      2033
           12       0.98      0.95      0.97      2033
           13       0.98      0.97      0.98      2033
           14       0.92      0.91      0.91      2033
           15       0.97      0.94      0.95      2033
           16       0.97      0.95      0.96      2033
           17       0.97      0.96      0.96      2033
           18       0.98      0.96      0.97      2033
           19       1.00      0.95      0.97      2033
           20       0.98      0.95      0.97      2033
           21       1.00      0.97      0.98      2033
           22       0.96      0.95      0.95      2033
           23       1.00      0.97      0.98      2033
           24       0.99      0.98      0.99      2033
           25       0.99      0.95      0.97      2033
           26       1.00      1.00      1.00      2033

    accuracy                           0.96     54891
   macro avg       0.96      0.96      0.96     54891
weighted avg       0.96      0.96      0.96     54891

Test Accuracy: 0.9564995392164188
Classification Report (Test Set):
              precision    recall  f1-score   support

           0       0.61      0.98      0.75      2974
           1       1.00      0.97      0.99      2974
           2       0.92      0.91      0.91      2974
           3       0.99      0.97      0.98      2974
           4       0.95      0.98      0.96      2974
           5       0.95      0.96      0.96      2974
           6       1.00      0.93      0.96      2974
           7       0.99      0.97      0.98      2974
           8       0.99      0.91      0.95      2974
           9       0.99      0.95      0.97      2974
          10       0.97      0.97      0.97      2974
          11       0.99      0.96      0.98      2974
          12       0.97      0.96      0.96      2974
          13       0.98      0.98      0.98      2974
          14       0.93      0.90      0.91      2974
          15       0.97      0.95      0.96      2974
          16       0.98      0.95      0.96      2974
          17       0.98      0.96      0.97      2974
          18       0.99      0.96      0.97      2974
          19       0.99      0.96      0.97      2974
          20       0.98      0.95      0.97      2974
          21       0.99      0.97      0.98      2974
          22       0.98      0.95      0.97      2974
          23       1.00      0.97      0.98      2974
```

```
          24       1.00      0.98      0.99      2974
          25       0.98      0.95      0.96      2974
          26       1.00      0.99      1.00      2974

    accuracy                           0.96     80298
   macro avg       0.97      0.96      0.96     80298
weighted avg       0.97      0.96      0.96     80298
```

Task 3

Convert further your 27 language dataset into language groups, where the grouping is via their respective continent names. It appears that the dataset has Asia, Africa, Europe, and North america. So, you will have four classes now. Collapse the dataset into 4 classes by appending the files into large files. Build a Regularized Discriminant Analysis (RDA) model, which has a hyper-parameter lambda to tradeoff between LDA and QDA. You may use bag-of-words via CountVectorizer or Tfidf Vectorizer to create the feature space of your dataset. It will be a huge feature space, but LDA/QDA can handle large feature spaces, so no worries there. Of course, you may use some clever feature elimination methods such as low frequency pruning, noise removal, etc

```
continent_map = {
    'Asia': ['jv-ID', 'ms-MY', 'tl-PH', 'tr-TR', 'vi-VN'],
    'Africa': ['af-ZA', 'sw-KE'],
    'Europe': ['da-DK', 'de-DE', 'es-ES', 'fr-FR', 'fi-FI', 'hu-HU',
'is-IS', 'it-IT', 'lv-LV', 'nb-NO', 'nl-NL', 'pl-PL', 'pt-PT', 'ro-
RO', 'ru-RU', 'sl-SL', 'sv-SE', 'sq-AL', 'cy-GB'],
    'North America': ['en-US']
}

def map_language_to_continent(label):
    for continent, languages in continent_map.items():
        if label in languages:
            return continent
    return None

y_continent = [map_language_to_continent(label) for label in Y1]

from sklearn.model_selection import train_test_split
X_train, X_temp, y_train, y_temp = train_test_split(X1, y_continent,
test_size=0.3, random_state=42, stratify=y_continent)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42, stratify=y_temp)


vectorizer = TfidfVectorizer(max_features=5000)
```

```
X_train_vec = vectorizer.fit_transform(X_train)
X_val_vec = vectorizer.transform(X_val)
X_test_vec = vectorizer.transform(X_test)

from sklearn.decomposition import TruncatedSVD
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis,
QuadraticDiscriminantAnalysis
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, accuracy_score
import pycountry_convert as pc
import numpy as np
from datasets import load_dataset

# Reduce dimensionality using TruncatedSVD (LSA)
svd = TruncatedSVD(n_components=300)
X_train_svd = svd.fit_transform(X_train_vec)
X_val_svd = svd.transform(X_val_vec)
X_test_svd = svd.transform(X_test_vec)

class RegularizedDiscriminantAnalysis:
  def __init__(self, lda_weight=0.5):
    self.lda_weight = lda_weight
    self.lda = LinearDiscriminantAnalysis()
    self.qda = QuadraticDiscriminantAnalysis()

  def fit(self, X, y):
    self.lda.fit(X, y)
    self.qda.fit(X, y)

  def predict(self, X):
    lda_preds = self.lda.predict_proba(X)
    qda_preds = self.qda.predict_proba(X)

    combined_preds = (self.lda_weight * lda_preds) + ((1 -
self.lda_weight) * qda_preds)
    return np.argmax(combined_preds, axis=1)

  def predict_probability(self, X):
    lda_preds = self.lda.predict_proba(X)
    qda_preds = self.qda.predict_proba(X)
    combined_preds = (self.lda_weight * lda_preds) + ((1 -
self.lda_weight) * qda_preds)
    return combined_preds


lda_weight = 0.7
rda_model = RegularizedDiscriminantAnalysis(0.7)

label_encoder = LabelEncoder()
train_encoded = label_encoder.fit_transform(y_train)
```

```
val_encoded = label_encoder.transform(y_val)
test_encoded = label_encoder.transform(y_test)

rda_model.fit(X_train_svd, train_encoded)

val_predictions_encoded = rda_model.predict(X_val_svd)
val_accuracy = accuracy_score(val_encoded, val_predictions_encoded)
print(f"Validation Accuracy: {val_accuracy * 100:.2f}%")

test_predictions_encoded = rda_model.predict(X_test_svd)
test_accuracy = accuracy_score(test_encoded, test_predictions_encoded)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")

test_predictions =
label_encoder.inverse_transform(test_predictions_encoded)
test_labels = label_encoder.inverse_transform(test_encoded)

print("Classification Report (Test Set):")
print(classification_report(test_labels, test_predictions,
target_names=label_encoder.classes_))


train_predictions_encoded = rda_model.predict(X_train_svd)
train_predictions =
label_encoder.inverse_transform(train_predictions_encoded)
train_accuracy = accuracy_score(train_encoded,
train_predictions_encoded)
print(f"Training Accuracy: {train_accuracy * 100:.2f}%")
```

```
Validation Accuracy: 93.42%
Test Accuracy: 93.37%
Classification Report (Test Set):
               precision    recall  f1-score   support

       Africa       0.95      0.78      0.86      4957
         Asia       0.99      0.80      0.89     12391
       Europe       0.92      0.99      0.96     47085
North America       0.92      0.80      0.86      2478

     accuracy                           0.93     66911
    macro avg       0.95      0.84      0.89     66911
 weighted avg       0.94      0.93      0.93     66911

Training Accuracy: 93.50%
```

Why to use svd?

Reducing the Feature Space: Textual data, when vectorized using techniques like TfidfVectorizer or CountVectorizer, typically results in a very large number of features, especially when dealing with natural language data. This is because every unique word or token becomes a feature, and this can lead to high-dimensional data. In my case, I am working with text data from 27

languages, and when transformed into numerical features via TF-IDF, this results in thousands of features. SVD helps reduce this high-dimensional feature space to a more manageable number of dimensions (e.g., reducing from 5000 features to 300 in my code). It is crucial because classifiers like LDA/QDA can struggle with very high-dimensional data, where there may not be enough samples to support accurate classification.