Set up

In [ ]:
```
### CONFIG (Run on linux backend)
!pip install transformers
# !pip install sentencepiece
# !wget https://bakrianoo.ewr1.vultrobjects.com/aravec/full_grams_cbow_100_twitter.zip
# !unzip "full_grams_cbow_100_twitter.zip"
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting transformers
  Downloading transformers-4.24.0-py3-none-any.whl (5.5 MB)
     |████████████████████████████████| 5.5 MB 15.4 MB/s
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from transformers)
(4.13.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (1.21.
6)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from transformers) (3.8.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from transformers) (2.23.0)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
  Downloading tokenizers-0.13.1-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.6 MB)
     |████████████████████████████████| 7.6 MB 15.4 MB/s
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from transformers) (4.64.1
)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from transformers)
(2022.6.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (from transformers) (6.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from transformers) (2
1.3)
Collecting huggingface-hub<1.0,>=0.10.0
  Downloading huggingface_hub-0.10.1-py3-none-any.whl (163 kB)
     |████████████████████████████████| 163 kB 38.8 MB/s
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-packages (from huggi
ngface-hub<1.0,>=0.10.0->transformers) (4.1.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packagi
ng>=20.0->transformers) (3.0.9)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->tr
ansformers) (3.10.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->tra
nsformers) (2022.9.24)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-package
s (from requests->transformers) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->transform
ers) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->tran
sformers) (3.0.4)
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.10.1 tokenizers-0.13.1 transformers-4.24.0
```

In [ ]:
```
## Imports
from transformers import pipeline
import numpy as np
import pandas as pd
import gensim
import gensim.downloader
import re
import time
```

In [ ]:
```
def clean(text):
  # remove any punctuations in the text
  punc = """،.:!?؟!:.,''!"#$%&'()*+, -./:;<=>?@[\]^_`{|}~"""
  for l in text:
    if l in punc and l != " ":
      text = text.replace(l,"")
  return text
```

In [ ]:
```
LOADED_MODELS = {}
```

Augmenting through w2v (aug_w2v)

In [ ]:
```
def load_w2v(ar_model_name):
  global LOADED_MODELS
  if not ar_model_name in LOADED_MODELS:
    try:
        ar_model = gensim.models.KeyedVectors.load_word2vec_format(ar_model_name,binary=True,unicode_errors='ig
    except:
        ar_model = gensim.models.Word2Vec.load(ar_model_name)
    LOADED_MODELS[ar_model_name] = ar_model
  return LOADED_MODELS[ar_model_name]
```

In [ ]:
```
def w2v(ar_model,sentence):
  l = []
  augs = []
  if len(sentence.split()) > 2:
    for i,token in enumerate(sentence.split()):
          model_to_use = ar_model
          try:
            word_vectors = model_to_use.wv
            if token in word_vectors.key_to_index:
              exist = True
            else:
              exist = False
          except:
            if token in model_to_use:
              exist = True
            else:
              exist = False
          if exist:
            try:
              most_similar = model_to_use.wv.most_similar( token, topn=5 )
            except:
              most_similar = model_to_use.most_similar( token, topn=5 )
            for term, score in most_similar:
                  if term != token:
                      term = term.replace("_"," ")
                      if not term.isalpha():
                        s = sentence.split()
                        s[i] = term
                        aug = " ".join(s)
                        if not clean(aug) in augs:
                          augs.append(clean(aug))
                          aug = " ".join(aug.split())
                          l.append(aug)
  return l
```

```
In [ ]: # text here is a list of sentences or one string sentence
        def aug_w2v(ar_model,text, model_name):
            print(f"Loading {model_name}... ")
            tic = time.perf_counter()
            ar_model = load_w2v(ar_model)
            toc = time.perf_counter()
            print(f"Loading {model_name} done ✅: " + str(round(toc-tic, 3)) + " seconds")
            print(f"Augmenting with {model_name}... ")
            tic = time.perf_counter()
            if isinstance(text, str):
              ret = w2v(ar_model, text)
              toc = time.perf_counter()
              print(f"Augmenting with {model_name} done ✅: " + str(round(toc-tic, 3)) + " seconds")
              return ret
            else:
              all_sentences = []
              for sentence in text:
                sentence = sentence.strip()
                all_sentences.append([sentence,w2v(ar_model,sentence)])
              toc = time.perf_counter()
              print(f"Augmenting with {model_name} done ✅: " + str(round(toc-tic, 3)) + " seconds")
              return all_sentences
```

Augmenting through fill mask (aug_bert)

```
In [ ]: def load_bert(ar_model_name):
          global LOADED_MODELS
          if not ar_model_name in LOADED_MODELS:
            ar_model = pipeline('fill-mask', model= ar_model_name)
            LOADED_MODELS[ar_model_name] = ar_model
          return LOADED_MODELS[ar_model_name]
```

```
In [ ]: # Contextual word embeddings
        def bert(model, sentence):
          l = []
          augs = [sentence.split(),sentence.split(),sentence.split()]
          # key:index , value: list of predicitions
          aug_words = {}
          if len(sentence.split()) > 2:
            for n,token in enumerate(sentence.split()):
                    s = sentence.split()
                    try:
                      s[n] = "<mask>"
                      masked_text = " ".join(s)
                      pred = model(masked_text , top_k = 3)
                    except:
                      s[n] = "[MASK]"
                      masked_text = " ".join(s)
                      pred = model(masked_text , top_k = 3)
                    for i in pred:
                      if isinstance(i, dict):
                        output = i['token_str']
```

```
              if not output == token:
                if not len(output) < 2 and clean(output) == output:
                  output = output.replace("_"," ")
                  ara = re.findall(r'[\u0600-\u06FF]+', output)
                  if len("".join(ara)) == len(output.replace(" ","")):
                     if not n in aug_words:
                        aug_words[n] = [output]
                     else:
                        aug_words[n].append(output)
  for s in range(len(augs)):
      for i in aug_words:
         predicted = aug_words[i]
         if not s + 1 > len(predicted):
            augs[s][i] = predicted[s]
         else:
            augs[s][i] = predicted[len(predicted) - 1]

  return augs
```

In [ ]:
```
def multi_bert(model, sentence):
    l = bert(model, sentence)
    ret = []
    for i in l:
      ret += bert(model, i)
    return ret
```

In [ ]:
```
# text here is a list of sentences or one string sentence
def aug_bert(model, text, model_name):
    print(f"Loading {model_name}... ")
    tic = time.perf_counter()
    model = load_bert(model)
    toc = time.perf_counter()
    print(f"Loading {model_name} done ✅: " + str(round(toc-tic, 3)) + " seconds")
    print(f"Augmenting with {model_name}... ")
    tic = time.perf_counter()
    if isinstance(text, str):
      ret = bert(model, text)
      toc = time.perf_counter()
      print(f"Augmenting with {model_name} done ✅: " + str(round(toc-tic, 3)) + " seconds")
      return ret
    else:
      all_sentences = []
      for sentence in text:
        sentence = sentence.strip()
        all_sentences.append([sentence, bert(model,sentence)])
      toc = time.perf_counter()
      print(f"Augmenting with {model_name} done ✅: " + str(round(toc-tic, 3)) + " seconds")
      return all_sentences
```

Excecution to list

In [ ]:
```
def augment_to_list(sentences):
  print("Beginning Augmentation... \n")
  ret = []
  # Augment sentences by each model
  # ret += aug_bert("aubmindlab/bert-large-arabertv2", sentences, "Arabert")
  # ret += aug_bert("qarib/bert-base-qarib", sentences, "Qarib Bert")
  # ret += aug_bert("xlm-roberta-base", sentences, "XLM-Roberta")
  # ret += aug_bert("moussaKam/AraBART", sentences, "Arabart")
  # ret += aug_bert("CAMeL-Lab/bert-base-arabic-camelbert-mix", sentences, "Camel Bert")
  ret += aug_bert("alger-ia/dziribert", sentences, "Dziri Bert")
  # ret += aug_bert("asafaya/bert-large-arabic", sentences, "Bert Large Arabic")
  # ret += aug_bert("UBC-NLP/ARBERT", sentences, "Arbert")
  # ret += aug_bert("UBC-NLP/MARBERTv2", sentences, "Marbert")
  # ret += aug_bert("aubmindlab/araelectra-base-generator", sentences, "Araelectra")
  # aragpt2_sentences = aug_GPT("aubmindlab/aragpt2-medium", sentences)
  # ret += aug_w2v("full_grams_cbow_100_twitter.mdl", sentences, "Aravec")
  # ret += aug_w2v("cbow_100.bin", 'glove-twitter-25', sentences, "Mazajak (CBOW 100)")
  # back_translation_sentences = aug_back_translate(sentences)
  return ret
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
project_dir = "/content/drive/MyDrive/afrisent-semeval-2023"
lang_code = "dz"
```

```
Mounted at /content/drive
```

In [ ]:
```
df_train = pd.read_csv(f"{project_dir}/SubtaskA/train/{lang_code}_pro_train.csv")
```

In [ ]:
```
sentences_pos = df_train["tweet"][df_train["label"] == 1].tolist()
sentences_neu = df_train["tweet"][df_train["label"] == 0].tolist()
sentences_neg = df_train["tweet"][df_train["label"] == -1].tolist()
```

In [ ]:
```
label_pos = df_train["label"][df_train["label"] == 1].tolist()
```

```
label_neu = df_train["label"][df_train["label"] == 0].tolist()
label_neg = df_train["label"][df_train["label"] == -1].tolist()
```

In [ ]:
```
l_pos = augment_to_list(sentences_pos)
l_neu = augment_to_list(sentences_neu)
l_neg = augment_to_list(sentences_neg)
```

In [ ]:
```
ls_pos = []
for i in l_pos:
  for s in i[1]:
    ls_pos.append(s)
ls_neu = []
for i in l_neu:
  for s in i[1]:
    ls_neu.append(s)
ls_neg = []
for i in l_neg:
  for s in i[1]:
    ls_neu.append(s)
```

In [ ]:
```
df_train = df_train.append(pd.DataFrame({"tweet":ls_pos,"label":[1]*len(ls_pos)}),ignore_index= True).append(pd
```

In [ ]:
```
df_train["label"] = df_train["label"].apply(lambda x: int(x))
```

In [ ]:
```
df_train.to_csv(f"{project_dir}/SubtaskA/train/{lang_code}_pro_train_aug.csv", index=False)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js