In [ ]:
```
#Імпортуємо потрібні бібліотеки
import tensorflow as tf #Для конвертації tensorflow датасету у звичайний датасет
import tensorflow_datasets as tfds #Звідси ми візьмемо потрібний датасет
import spacy #Для безпосередніх навчання
import pandas as pd #Для роботи з датафреймами
from spacy.tokens import DocBin #Для серіалізації
```

```
d:\source\Pythonrepos\Learning\Learning\lib\site-packages\tqdm\auto.py:22: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [ ]:
```
ds_train = tfds.load('ag_news_subset', split='train', shuffle_files=True) #Завантажуємо потрібний датасет tensorflow.org/datasets/catalog/ag_news_subset
assert isinstance(ds_train, tf.data.Dataset) #Вказуємо що завантажений об'єкт має формат tensorflow датасету
df_train = tfds.as_dataframe(ds_train) #Конвертуємо його в датафрейм, з яким зможе працювати spacy
ds_test = tfds.load('ag_news_subset', split='test', shuffle_files=True) #Повторюємо попередні кроки для другої частини даних
assert isinstance(ds_test, tf.data.Dataset)
df_test = tfds.as_dataframe(ds_test)
df_train.head() #Виведемо частину даних, щоб перевірити коректну роботу програми
```

Out[ ]:

	description	label	title
0	b'AMD #39;s new dual-core Opteron chip is desi...	3	b'AMD Debuts Dual-Core Opteron Processor'
1	b'Reuters - Major League Baseball\\Monday anno...	1	b"Wood's Suspension Upheld (Reuters)"
2	b'President Bush #39;s quot;revenue-neutral q...	2	b'Bush reform may have blue states seeing red'
3	b'Britain will run out of leading scientists u...	3	b"Halt science decline in schools'"
4	b'London, England (Sports Network) - England m...	1	b'Gerrard leaves practice'

In [ ]:
```
from sklearn.model_selection import train_test_split #Для розділення датафрейму на два випадкових датафрейма

allDF = pd.concat([df_train, df_test], ignore_index=True) #Зв'язуємо два датафрейма в один
allDF = allDF.sample(frac=0.2).reset_index(drop=True) #Беремо частину даних щоб пришвидшити роботу

trainDF, testDF = train_test_split(allDF, test_size = 0.2) #Розділяємо датафрейм на два, один буде для тренування (80%) інший для тестування (20%)
testDF, validDF = train_test_split(testDF, test_size = 0.2) #Розділяємо тренувальний датафрейм на два, один залишиться для тестування (16%)
#Інший буде для валідації (4%)

print("Train:",len(trainDF), "Test:", len(testDF),"Valid:", len(validDF)) #Виведемо кількість даних в кожному датафреймі
```

```
Train: 20416 Test: 4083 Valid: 1021
```

In [ ]:
```
#Як можна побачити коли ми виводили перші 5 рядків датафрейму у секції 2, текстова інформація зберігається у вигляді байтів.
#Тож після кожного оновлення датасету потрібно буде конвертувати строку байтів у формат, з яким може працювати spacy
def from_byte_to_string(text): #Конвертує строку байтів у звичайну строку
    answer = text.decode("utf-8")
    return answer

def from_int_to_string(number): #Конвертує число у строку
    answer = str(number)
    return answer

def first_process(df): #Конвертує усі байтові строки і числа у датасеті у звичайні строки
    df.description = df.description.apply(from_byte_to_string)
    df.label = df.label.apply(from_int_to_string)

def preprocess(df, embed): #Попередня обробка даних,
    data = tuple(zip(df.description.tolist(), df.label.tolist())) #Об'єднуємо датафрейми у кортежі
    nlp = spacy.load(embed) #Завантажуємо trained pipeline
    docs = [] #Підготовлюємо місце для оброблених даних

    for doc, label in nlp.pipe(data, as_tuples=True): #Проганяючи кортежі через nlp, ми створюємо впорядковані доки, на які можно навішувати ярлики https://spacy.io/usage/processing-pipelines
        doc.cats['World'] = 0 #Вішаємо ярлики
        doc.cats['Sports'] = 0
        doc.cats['Business'] = 0
        doc.cats['Sci/Tech'] = 0

        if label=="0": #Призначаємо конкретний ярлик
            doc.cats['World'] = 1
        elif label=="1":
            doc.cats['Sports'] = 1
        elif label=="2":
            doc.cats['Business']  = 1
        elif label=="3":
            doc.cats['Sci/Tech'] = 1
        else:
            print("Labeling error")

        docs.append(doc) #Додаємо до масиву оброблених даних

    return df, docs
```

In [ ]:
```
!python -m spacy init fill-config ./base_config.cfg ./config.cfg #Підготовлюємо конфігураційний файл
```

```
✔ Auto-filled config with all values
✔ Saved config
config.cfg
You can now add your data and train your pipeline:
python -m spacy train config.cfg --paths.train ./train.spacy --paths.dev ./dev.spacy
```

In [ ]:
```
first_process(trainDF) #Перетворюємо дані у читабельний формат
first_process(testDF)
```

In [ ]:
```
train_data, train_docs = preprocess(trainDF, "en_core_web_sm") #Препроцесимо дані, використовуючи стандартний pipeline https://spacy.io/usage/models
doc_bin = DocBin(docs=train_docs) #Серіалізуємо отримані дані
doc_bin.to_disk("./textcat_train.spacy") #Зберігаємо їх

test_data, test_docs = preprocess(testDF, "en_core_web_sm") #Аналогічно
doc_bin = DocBin(docs=test_docs)
doc_bin.to_disk("./textcat_test.spacy")
```

In [ ]:
```
!python -m spacy train ./config.cfg --verbose --output ./textcat_output --paths.train ./textcat_train.spacy --paths.dev ./textcat_test.spacy #Стартуємо навчання
```

```
ℹ Saving to output directory: textcat_output
ℹ Using CPU

=========================== Initializing pipeline ===========================
✔ Initialized pipeline

============================= Training pipeline =============================
ℹ Pipeline: ['tok2vec', 'textcat']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS TEXTCAT  CATS_SCORE  SCORE
---  ------  ------------  ------------  ----------  ------
  0       0          0.00          0.19        0.00    0.00
  0     200         18.32         43.28       13.02    0.13
  0     400         31.21         40.44        6.86    0.07
  0     600         91.07         37.18       27.95    0.28
  0     800         92.11         36.65       21.90    0.22
  0    1000        106.62         35.65       30.41    0.30
  0    1200        110.54         35.30       27.31    0.27
  0    1400        307.91         32.97       37.01    0.37
  0    1600        680.11         30.64       39.03    0.39
  0    1800       1242.13         30.54       48.02    0.48
  0    2000       4292.16         28.39       53.82    0.54
  1    2200       8205.78         25.05       55.67    0.56
  1    2400      11152.58         23.91       58.88    0.59
  1    2600      19127.65         23.27       62.05    0.62
  1    2800      30728.49         22.55       61.43    0.61
  2    3000      33512.36         22.17       65.79    0.66
  2    3200      42699.74         22.03       67.19    0.67
  2    3400      59277.27         21.25       66.27    0.66
  2    3600      74565.14         21.76       63.47    0.63
  3    3800      86606.77         22.76       64.77    0.65
  3    4000     108286.78         22.28       67.41    0.67
  3    4200     134526.47         21.31       66.98    0.67
  3    4400     167143.91         21.77       66.54    0.67
  4    4600     161786.00         20.39       65.87    0.66
  4    4800     239958.17         21.54       68.72    0.69
  4    5000     295558.09         21.20       67.53    0.68
  4    5200     298042.63         21.35       66.19    0.66
  5    5400     308167.02         21.07       71.06    0.71
  5    5600     360895.58         21.15       69.20    0.69
  5    5800     391194.20         21.40       68.00    0.68
  6    6000     393513.98         20.84       69.85    0.70
  6    6200     474678.63         19.41       71.14    0.71
  6    6400     640969.29         21.04       63.51    0.64
  6    6600     625674.19         20.88       68.42    0.68
  7    6800     759624.40         19.81       70.08    0.70
  7    7000    1110029.47         20.87       70.22    0.70
  8    7800    1235802.71         20.07       71.81    0.72
  8    8000    1357141.65         20.35       71.21    0.71
  8    8200    1245421.49         20.66       62.48    0.62
  9    8400    1942493.49         20.80       72.17    0.72
  9    8600    2019690.56         20.53       61.25    0.61
  9    8800    1861026.42         21.04       65.75    0.66
 10    9000    1857070.89         21.12       68.77    0.69
 10    9200    2159385.48         20.62       66.62    0.67
 10    9400    2294846.52         20.12       71.61    0.72
 10    9600    2339453.03         19.94       67.05    0.67
 11    9800    2873707.06         20.60       63.78    0.64
 11   10000    2765052.42         20.56       71.13    0.71
✔ Saved pipeline to output directory
textcat_output\model-last
```

```
[2022-12-09 20:09:47,388] [DEBUG] Config overrides from CLI: ['paths.train', 'paths.dev']
[2022-12-09 20:09:47,747] [INFO] Set up nlp object from config
[2022-12-09 20:09:47,768] [DEBUG] Loading corpus from path: textcat_test.spacy
[2022-12-09 20:09:47,770] [DEBUG] Loading corpus from path: textcat_train.spacy
[2022-12-09 20:09:47,770] [INFO] Pipeline: ['tok2vec', 'textcat']
[2022-12-09 20:09:47,777] [INFO] Created vocabulary
[2022-12-09 20:09:48,640] [WARNING] [W112] The model specified to use for initial vectors (en_core_web_sm) has no vectors. This is almost certainly a mistake.
[2022-12-09 20:09:48,643] [INFO] Added vectors: en_core_web_sm
[2022-12-09 20:09:48,646] [INFO] Finished initializing nlp object
[2022-12-09 20:10:17,514] [INFO] Initialized pipeline components: ['tok2vec', 'textcat']
[2022-12-09 20:10:17,534] [DEBUG] Loading corpus from path: textcat_test.spacy
[2022-12-09 20:10:17,537] [DEBUG] Loading corpus from path: textcat_train.spacy
[2022-12-09 20:10:17,539] [DEBUG] Removed existing output directory: textcat_output\model-best
[2022-12-09 20:10:17,541] [DEBUG] Removed existing output directory: textcat_output\model-last
```

In [ ]:
```
first_process(validDF)
```

In [ ]:
```
#Підготовуємо валідаційні дані
valid_data, valid_docs = preprocess(validDF, "en_core_web_sm")
doc_bin = DocBin(docs=valid_docs)
doc_bin.to_disk("./textcat_valid.spacy")

#Застосовуємо модель до валідаційних даних
nlp_model = spacy.load("./textcat_output/model-best")
valid_text = valid_data.description.tolist()
valid_cats = valid_data.label.tolist()
```

In [ ]:
```
#Виведемо випадкове передбачення
import random
i=random.randrange(0,100)
doc_valid = nlp_model(valid_text[i])
print("Text: " + valid_text[i] + "\nOriginal category: " + valid_cats[i] + "\nPrediction: " + doc_valid.cats)
```

```
Text: Nicolas Massu beat Taylor Dent of the United States 7-6 (5), 6-1 Friday in the tennis semifinals to move within one victory of giving Chile its first Olympic gold medal in any sport.
Original category: 1
Predicted:
{'World': 0.0036531013902276754, 'Sports': 0.9944242835044861, 'Business': 0.0007581915124319494, 'Sci/Tech': 0.0011644094483926892}
```

In [ ]:
```
!python -m spacy evaluate textcat_output/model-best textcat_valid.spacy #Застосовуємо модель до валідаційних даних
```

```
ℹ Using CPU

================================== Results ==================================

TOK                 100.00
TEXTCAT (macro F)   71.75
SPEED               2938


============================ Textcat F (per label) ============================

               P       R       F
World      73.66   64.96   69.04
Sports     94.42   78.15   85.52
Business   70.16   53.60   60.77
Sci/Tech   83.02   63.08   71.69


======================== Textcat ROC AUC (per label) ========================

           ROC AUC
World         0.90
Sports        0.98
Business      0.86
Sci/Tech      0.92
```

In [ ]:
```
!python -m spacy evaluate textcat_output/model-best textcat_test.spacy #Застосовуємо модель до тестових даних
```

```
ℹ Using CPU

================================== Results ==================================

TOK                 100.00
TEXTCAT (macro F)   72.17
SPEED               2956


============================ Textcat F (per label) ============================

               P       R       F
World      71.53   68.39   69.93
Sports     93.99   77.91   85.20
Business   72.18   58.05   64.35
Sci/Tech   77.52   62.53   69.22


======================== Textcat ROC AUC (per label) ========================

           ROC AUC
World         0.88
Sports        0.96
Business      0.87
Sci/Tech      0.90
```

In [ ]:
```
!python -m spacy evaluate textcat_output/model-best textcat_train.spacy #Застосовуємо модель до тренувальних даних
```

```
ℹ Using CPU

================================== Results ==================================

TOK                 100.00
TEXTCAT (macro F)   74.97
SPEED               3090


============================ Textcat F (per label) ============================

               P       R       F
World      73.18   70.54   71.84
Sports     96.11   83.58   89.41
Business   73.72   57.86   64.83
Sci/Tech   83.76   65.95   73.80


======================== Textcat ROC AUC (per label) ========================

           ROC AUC
World         0.90
Sports        0.98
```

```
Business      0.88
Sci/Tech      0.92
```