In [1]:

```
import os
from pathlib import Path

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt

import torch
from torch.utils.data import DataLoader
from torch.nn.utils.rnn import pad_sequence
from torch import nn

from tqdm.auto import tqdm,trange

from transformers import AdamW


BATCH_SIZE = 64
```

In [2]:

```
!rm -rf PoS-Tagging
!git clone https://github.com/Janluke0/PoS-Tagging/
os.chdir('PoS-Tagging')
out_dir = Path('/kaggle/working/')
out_dir.mkdir(exist_ok=True)
```

```
Cloning into 'PoS-Tagging'...
remote: Enumerating objects: 86, done.
remote: Counting objects: 100% (86/86), done.
remote: Compressing objects: 100% (55/55), done.
remote: Total 86 (delta 30), reused 80 (delta 24), pack-reused 0
Unpacking objects: 100% (86/86), 15.25 MiB | 8.80 MiB/s, done.
```

In [3]:

```
from model.transformers.italian import ItELECTRACasedPos,ItELECTRAXXLCasedPos
from dataset import TWITADS
```

Common

this part should be adde to the repo, it's time to go lighting

In [4]:

```
def train_model(model, dl_train, dl_test, cuda=False, lr=0.001, epochs=10, show_plots=Fa
lse, save_dir=None):
    loss_function = nn.NLLLoss()
    optimizer = AdamW(model.parameters(), lr=lr,weight_decay=0.01)

    if cuda:
        model = model.cuda()
    if save_dir is not None:
        save_dir.mkdir(exist_ok=True)

    losses = []
    accuracies = []
    best_acc = 0
    best_loss = float('inf')
    pbar = trange(epochs)
    for epoch in pbar:
        model.train()
        for sample in tqdm(iter(dl_train), desc=f"Training {epoch}° epoch", leave=False)
:
            x,m,y = sample['input_ids'],sample['attention_mask'], sample['labels']
            if cuda:
                x, m, y = x.cuda(), m.cuda(), y.cuda()
            optimizer.zero_grad()

            tag_scores =  model(input_ids=x,attention_mask=m)
            loss = loss_function(tag_scores.transpose(1, 2),y)

            loss.backward()
            optimizer.step()

        acc = []
        los = []
        ## evaluation
        model.eval()
        with torch.no_grad():
            for sample in tqdm(iter(dl_test), desc=f"Eval {epoch}° epoch", leave=False):
                x,m,y = sample['input_ids'],sample['attention_mask'], sample['labels']
                if cuda:
                    x, m, y = x.cuda(), m.cuda(), y.cuda()

                tag_scores =  model(input_ids=x,attention_mask=m)
                if hasattr(tag_scores,'logits'):
                    tag_scores = tag_scores.logits

                loss = loss_function(tag_scores.transpose(1, 2),y)
                los.append(loss.cpu().item())

                acc.append(((tag_scores.argmax(2))==y)[m==1].float())

        acc = torch.cat(acc).mean().item()
        los = np.array(los).mean()

        losses.append(los)
        accuracies.append(acc)
        #show epoch results
        pbar.set_description(f"Loss:{los}\tAccurancy:{acc}")
        if show_plots:
            plt.subplot(121)
            plt.title("Test loss")
            plt.plot(losses)

            plt.subplot(122)
            plt.title("Test accuracy")
            plt.plot(accuracies)
        if save_dir is not None and acc >= best_acc:
            torch.save(model.state_dict(),save_dir/f"model_best_acc.pth")
        if save_dir is not None and loss <= best_loss:
            torch.save(model.state_dict(),save_dir/f"model_best_loss.pth")

        best_acc = max(acc,best_acc)
        best_loss = max(los,best_loss)

    return losses,accuracies

def show_pred(model, ds, i):
    REVTAG = {v:k for k,v in ds._TAGS.items()}
    model.cpu()
    sample = ds.collate([ds[i]])
    x,m,y = sample['input_ids'],sample['attention_mask'], sample['labels']
    with torch.no_grad():
        pred = model(input_ids=x,attention_mask=m)
    tkns = ds.tokenizer.convert_ids_to_tokens(x[0,1:-1])
    return list(zip(tkns,[REVTAG[v.item()] for v in pred[0].argmax(1)[1:-1]],[REVTAG[v.i
tem()] for v in y[0][1:-1]]))
```

In [5]:

```
def collate_fn(batch):
    input_ids, token_type_ids, attention_mask, labels = [[] for _ in range(4)]
    for sample in batch:
        input_ids.append(sample['input_ids'])
        token_type_ids.append(sample['token_type_ids'])
        attention_mask.append(sample['attention_mask'])
        labels.append(sample['labels'])
    d = {
        'input_ids':pad_sequence(input_ids,batch_first=True),
        'token_type_ids': pad_sequence(token_type_ids,batch_first=True),
        'labels':pad_sequence(labels, padding_value=-100,batch_first=True),
    }
    d['attention_mask'] = (d['labels']!=-100).float()*torch.ones(d['labels'].shape)
    return d
```

In [6]:

```
def tokenize_and_align_labels(tokenizer, tokens, tags):
    tokens = list(tokens)
    tokenized_inputs = tokenizer(tokens, truncation=True, is_split_into_words=True)

    word_ids = tokenized_inputs.word_ids(batch_index=0)  # Map tokens to their respectiv
e word.
    previous_word_idx = None
    label_ids = []
    for word_idx in word_ids:                             # Set the special tokens to -10
0
        if word_idx is None:
            label_ids.append(-100)
        elif word_idx != previous_word_idx:               # Only label the first token of
a given word.
            label_ids.append(tags[word_idx])

    tokenized_inputs["labels"] = label_ids

    return {k:torch.tensor(v) for k,v in tokenized_inputs.items()}
```

mc4 model

In [7]:

```
mc4_model = ItELECTRACasedPos(23)
```

```
Some weights of the model checkpoint at dbmdz/electra-base-italian-mc4-cased-discriminato
r were not used when initializing ElectraForTokenClassification: ['discriminator_predicti
ons.dense.weight', 'discriminator_predictions.dense_prediction.weight', 'discriminator_pr
edictions.dense.bias', 'discriminator_predictions.dense_prediction.bias']
- This IS expected if you are initializing ElectraForTokenClassification from the checkpo
int of a model trained on another task or with another architecture (e.g. initializing a
BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing ElectraForTokenClassification from the che
ckpoint of a model that you expect to be exactly identical (initializing a BertForSequenc
eClassification model from a BertForSequenceClassification model).
Some weights of ElectraForTokenClassification were not initialized from the model checkpo
int at dbmdz/electra-base-italian-mc4-cased-discriminator and are newly initialized: ['cl
assifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predi
ctions and inference.
```

In [8]:

```
tokenizer = ItELECTRACasedPos.tokenizer()
ds_train = TWITADS('resampled_train',
                 lambda w:[w],
                 transform=lambda a,b: tokenize_and_align_labels(tokenizer,a,b)
)
ds_val = TWITADS('resampled_validation',
                 lambda w:[w],
                 transform=lambda a,b: tokenize_and_align_labels(tokenizer,a,b)
)
```

In [9]:

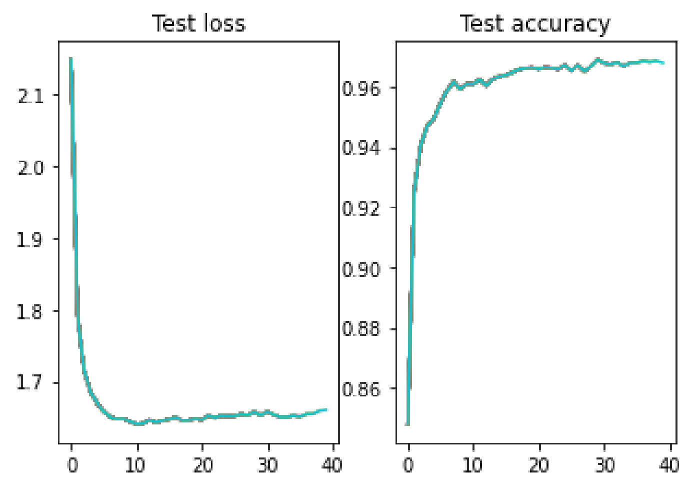
```
dl_train = DataLoader(ds_train, shuffle=True, batch_size=BATCH_SIZE, collate_fn=collate_f
n)
dl_val   = DataLoader(ds_val, shuffle=True, batch_size=BATCH_SIZE, collate_fn=collate_fn)
```

In [10]:

```
torch.manual_seed(42)
train_model(mc4_model,dl_train,dl_val,cuda=torch.cuda.is_available(), lr=2e-5, epochs=40
, show_plots=True, save_dir=out_dir/"mc4_model")
```

Out[10]:

```
([2.1494271993637084,
  1.7923119068145752,
  1.7144807696342468,
  1.6845481157302857,
  1.6687353491783141
  1.6687555491785141,
  1.6575248479843139,
  1.6495205521583558,
  1.646745765209198,
  1.6473207473754883,
  1.643509840965271,
  1.6396408677101135,
  1.6404513835906982,
  1.64548100233078,
  1.6422146677970886,
  1.6446046948432922,
  1.6464330911636353,
  1.6493518114089967,
  1.6446463823318482,
  1.645085048675537,
  1.6472050666809082,
  1.6457035899162293,
  1.651600968837738,
  1.648873221874237,
  1.6514630794525147,
  1.6510429978370667,
  1.6511090755462647,
  1.6538692712783813,
  1.6521823167800904,
  1.6569250464439391,
  1.6521586775779724,
  1.6570473194122315,
  1.6530860304832458,
  1.6495904922485352,
  1.6495405912399292,
  1.651939034461975,
  1.6502442955970764,
  1.653980541229248,
  1.6541460633277894,
  1.6584193110466003,
  1.6597591519355774],
 [0.847879946231842,
  0.9255728125572205,
  0.9404793381690979,
  0.9472215175628662,
  0.9493811130523682,
  0.9544904232025146,
  0.9585989117622375,
  0.9616539478302002,
  0.9592309594154358,
  0.9608638286590576,
  0.9608638286590576,
  0.9624440670013428,
  0.9603371024131775,
  0.9624966979026794,
  0.9633395075798035,
  0.9637608528137207,
  0.9647616744041443,
  0.965657114982605,
  0.9661838412284851,
  0.9663418531417847,
  0.965867817401886,
  0.9664472341537476,
  0.9662365317344666,
  0.9658151268959045,
  0.9671319723129272,
  0.9652884006500244,
  0.9670792818069458,
  0.965025007724762,
  0.9667105674743652,
  0.9690281748771667,
  0.9678167104721069,
  0.9673953056335449,
  0.9680801033973694,
  0.9668159484863281,
  0.9678694009780884,
  0.9679747223854065,
  0.9685541391372681
  0.9685541391372681,
  0.968238115310669,
  0.9686068296432495,
  0.967922031879425])
```



Uncased model

In [11]:

```
xxl_model =ItELECTRAXXLCasedPos(23)
```

```
Some weights of the model checkpoint at dbmdz/electra-base-italian-xxl-cased-discriminato
r were not used when initializing ElectraForTokenClassification: ['discriminator_predicti
ons.dense.weight', 'discriminator_predictions.dense_prediction.weight', 'discriminator_pr
edictions.dense.bias', 'discriminator_predictions.dense_prediction.bias']
- This IS expected if you are initializing ElectraForTokenClassification from the checkpo
int of a model trained on another task or with another architecture (e.g. initializing a
BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing ElectraForTokenClassification from the che
ckpoint of a model that you expect to be exactly identical (initializing a BertForSequenc
eClassification model from a BertForSequenceClassification model).
Some weights of ElectraForTokenClassification were not initialized from the model checkpo
int at dbmdz/electra-base-italian-xxl-cased-discriminator and are newly initialized: ['cl
assifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predi
ctions and inference.
```

In [12]:

```
tokenizer = ItELECTRAXXLCasedPos.tokenizer()
ds_train = TWITADS('resampled_train',
                 lambda w:[w],
                 transform=lambda a,b: tokenize_and_align_labels(tokenizer,a,b)
)
ds_val = TWITADS('resampled_validation',
                 lambda w:[w],
                 transform=lambda a,b: tokenize_and_align_labels(tokenizer,a,b)
)
```

In [13]:

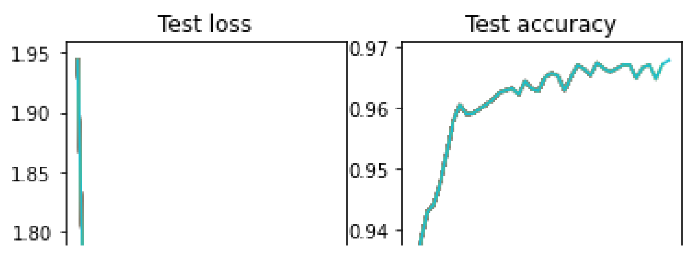
```
dl_train = DataLoader(ds_train, shuffle=True, batch_size=BATCH_SIZE, collate_fn=collate_f
n)
dl_val   = DataLoader(ds_val, shuffle=True, batch_size=BATCH_SIZE, collate_fn=collate_fn)
```

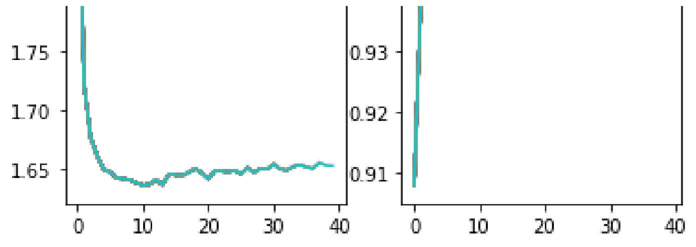
In [14]:

```
torch.manual_seed(42)
train_model(xxl_model,dl_train,dl_val,cuda=torch.cuda.is_available(), lr=2e-5, epochs=40
, show_plots=True,save_dir=out_dir/"xxl_model")
```

Out[14]:

```
([1.944074845314026,
  1.721768605709076,
  1.6772701621055603,
  1.6614771962165833,
  1.6502413988113402,
  1.647764301300049,
  1.6428612351417542,
  1.641972041130066,
  1.6412869453430177,
  1.6386998057365418,
  1.636702871322632,
  1.637635338306427,
  1.641200637817383,
  1.637195885181427,
  1.6462345957756042,
  1.6452369809150695,
  1.6448238253593446,
  1.6476357221603393,
  1.6508209466934205,
  1.647178316116333,
  1.6420470118522643,
  1.6488866686820984,
  1.6487650513648986,
  1.6479997396469117,
  1.6493016242980958,
  1.6461843609809876,
  1.6514342427253723,
  1.647558343410492,
  1.6509463787078857,
  1.6502332091331482,
  1.6545352458953857,
  1.650391435623169,
  1.649366593360901,
  1.6525370240211488,
  1.6540325045585633,
  1.6521865010261536,
  1.6508914709091187,
  1.6557843804359436,
  1.6536250948905944,
  1.6531776309013366],
 [0.9078219532966614,
  0.9378983378410339,
  0.9429023265838623,
  0.9440611004829407,
  0.9476428627967834,
  0.9526468515396118,
  0.9579668045043945,
  0.960495114326477,
  0.9590203166007996,
  0.9591256380081177,
  0.9598103761672974,
  0.9606004953384399,
  0.9613379240036011,
  0.9625493884086609,
  0.9629181027412415,
  0.963286817073822,
  0.9622860550880432,
  0.9644456505775452,
  0.9631814956665039,
  0.9628127813339233,
  0.965025007724762,
  0.9657624363899231,
  0.9653410911560059,
  0.9629707932472229,
  0.9650776982307434,
  0.9670792818069458,
  0.9664472341537476,
  0.9653937220573425,
  0.9675006866455078,
  0.9664998650550842,
  0.9659731388092041,
  0.9664998650550842,
  0.9671319723129272,
  0.9671319723129272,
  0.9649196863174438,
  0.9667105674743652,
  0.9671319723129272,
  0.9648143649101257,
  0.9672372937202454,
  0.967922031879425])
```





In [15]:

```
#remove repo from saved output
!rm -rf /kagle/working/PoS-Tagging
```

```
huggingface/tokenizers: The current process just got forked, after parallelism has alread
y been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:
 - Avoid using `tokenizers` before the fork if possible
 - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
```