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

BATCH_SIZE = 128
CUDA = torch.cuda.is_available()
LR = 0.1
EPOCHS= 500
```

In [2]:

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

```
Collecting bpemb
  Downloading bpemb-0.3.3-py3-none-any.whl (19 kB)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-pack
ages (from bpemb) (1.20.3)
Requirement already satisfied: requests in /opt/conda/lib/python3.7/site-p
ackages (from bpemb) (2.26.0)
Requirement already satisfied: gensim in /opt/conda/lib/python3.7/site-pac
kages (from bpemb) (4.0.1)
Requirement already satisfied: sentencepiece in /opt/conda/lib/python3.7/s
ite-packages (from bpemb) (0.1.96)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packa
ges (from bpemb) (4.62.3)
Requirement already satisfied: smart-open>=1.8.1 in /opt/conda/lib/python
3.7/site-packages (from gensim->bpemb) (5.2.1)
Requirement already satisfied: scipy>=0.18.1 in /opt/conda/lib/python3.7/s
ite-packages (from gensim->bpemb) (1.7.3)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python
3.7/site-packages (from requests->bpemb) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/pyt
hon3.7/site-packages (from requests->bpemb) (1.26.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/li
b/python3.7/site-packages (from requests->bpemb) (2.0.9)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.7/si
te-packages (from requests->bpemb) (3.1)
Installing collected packages: bpemb
Successfully installed bpemb-0.3.3
WARNING: Running pip as the 'root' user can result in broken permissions a
nd conflicting behaviour with the system package manager. It is recommende
d to use a virtual environment instead: https://pip.pypa.io/warnings/venv
Cloning into 'PoS-Tagging'...
remote: Enumerating objects: 64, done.
remote: Counting objects: 100% (64/64), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 64 (delta 19), reused 63 (delta 18), pack-reused 0
Unpacking objects: 100% (64/64), 596.65 KiB | 1.55 MiB/s, done.
```

In [3]:

```
from model.lstm import LSTMTagger
from model import train_model
from dataset import TWITADS
from lstm_model_grid_eval import mk_from_key
```

```
downloading https://nlp.h-its.org/bpemb/it/it.wiki.bpe.vs1000.model

100%|██████████| 251184/251184 [00:00<00:00, 455712.47B/s]

downloading https://nlp.h-its.org/bpemb/it/it.wiki.bpe.vs1000.d100.w2v.bi
n.tar.gz

100%|██████████| 375705/375705 [00:00<00:00, 673800.67B/s]
```

1/? [00:00<00:00, 26.07it/s]

In [4]:

```
def do_train(key):
    model, dl_tr, dl_val = mk_from_key(key,['resampled_train','resampled_validation'])
    loss, acc = train_model(model,dl_tr,dl_val,cuda=CUDA,lr=LR,epochs=EPOCHS,show_plots
=False)
    torch.save(model.state_dict(),out_dir/f"{key}.pth")
    with (out_dir/f"{key}.csv").open("w+") as f:
        f.write(",".join(map(str,loss)))
        f.write("\n")
        f.write(",".join(map(str,acc)))
        f.write("\n")
    return model, loss, acc
```

In [5]:

```
top_1 = {
    'accuracy': 'mono_1_32_1_bow_last',
  'alpha': 'bi_2_128_1__last',
  'combined': 'bi_1_64_1__last',
  'explained': 'bi_1_32_1__last',
  'f1': 'mono_1_128_1_#ow_last'
}
top_1_all = {'accuracy': 'mono_1_64_1__all',
  'alpha': 'mono_2_16_1_eow_all',
  'combined': 'mono_1_128_1_eow_all',
  'explained': 'mono_2_32_1__all',
  'f1': 'mono_1_128_1_eow_all'
}
```

Top all tagging modes

accuracy

In [6]:

```
_,loss,acc = do_train(top_1['accuracy'])
plt.figure(figsize=(16,6))
plt.subplot(121)
plt.plot(loss)
plt.subplot(122)
plt.plot(acc)
```

Loss:1.348015809059143

Accurancy:0.7068858742713928: 100%

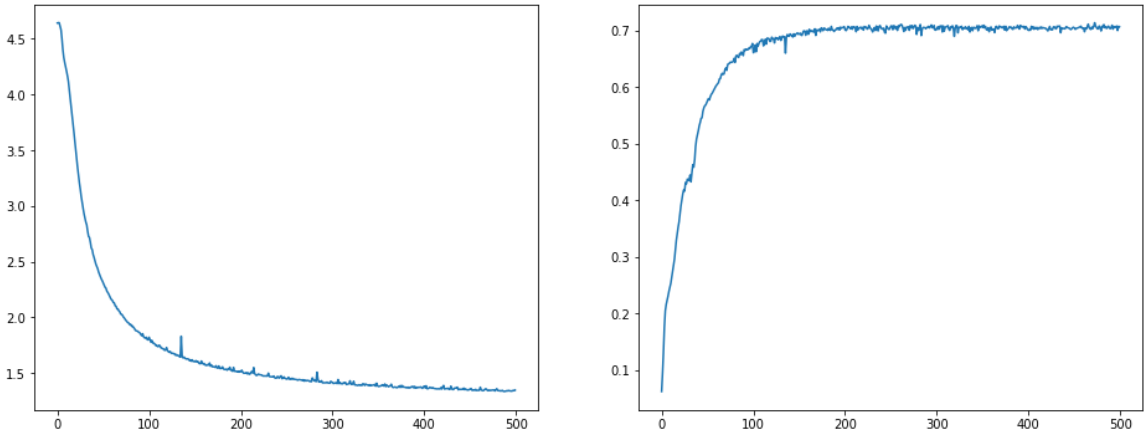
500/500

[03:20<00:00,

2.55it/s]

Out[6]:

```
[<matplotlib.lines.Line2D at 0x7f0e1c4b1290>]
```



alpha

In [7]:

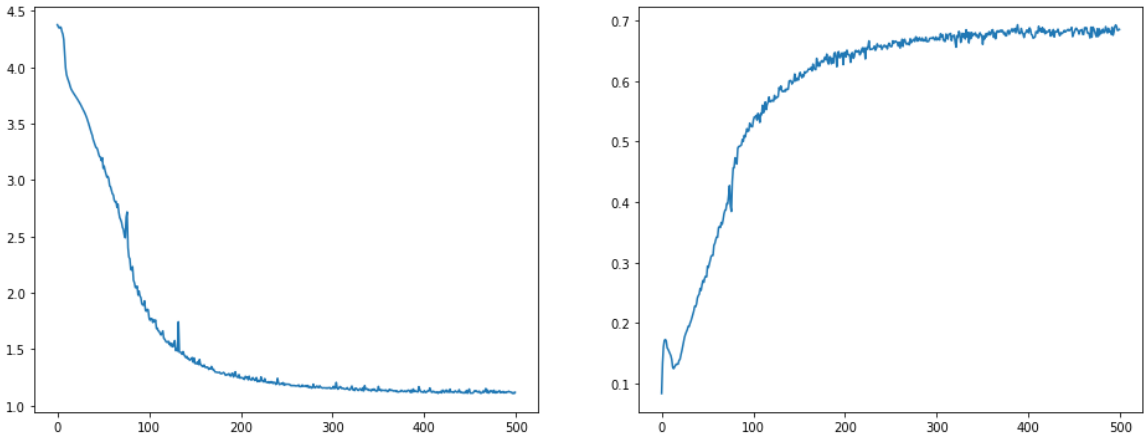
```
_,loss,acc = do_train(top_1['alpha'])
plt.figure(figsize=(16,6))
plt.subplot(121)
plt.plot(loss)
plt.subplot(122)
plt.plot(acc)
```

Loss:1.1182380437850952 500/500

Accurancy:0.6853842735290527: 100% [13:51<00:00, 1.66s/it]

Out[7]:

```
[<matplotlib.lines.Line2D at 0x7f0df5bf0b10>]
```



combined

In [8]:

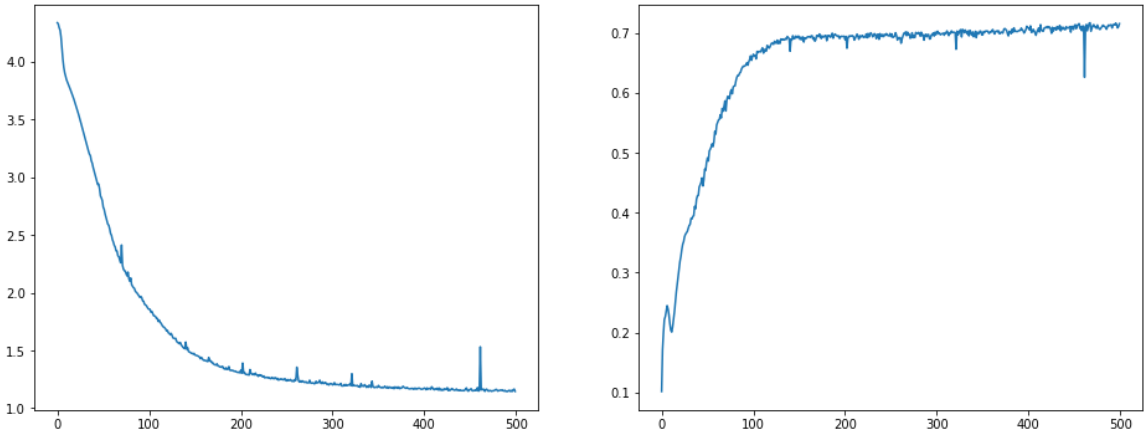
```
_,loss,acc = do_train(top_1['combined'])
plt.figure(figsize=(16,6))
plt.subplot(121)
plt.plot(loss)
plt.subplot(122)
plt.plot(acc)
```

Loss:1.1461727619171143 500/500

Accurancy:0.7153265476226807: 100% [04:44<00:00, 1.76it/s]

Out[8]:

```
[<matplotlib.lines.Line2D at 0x7f0dec12dbd0>]
```



f1

In [9]:

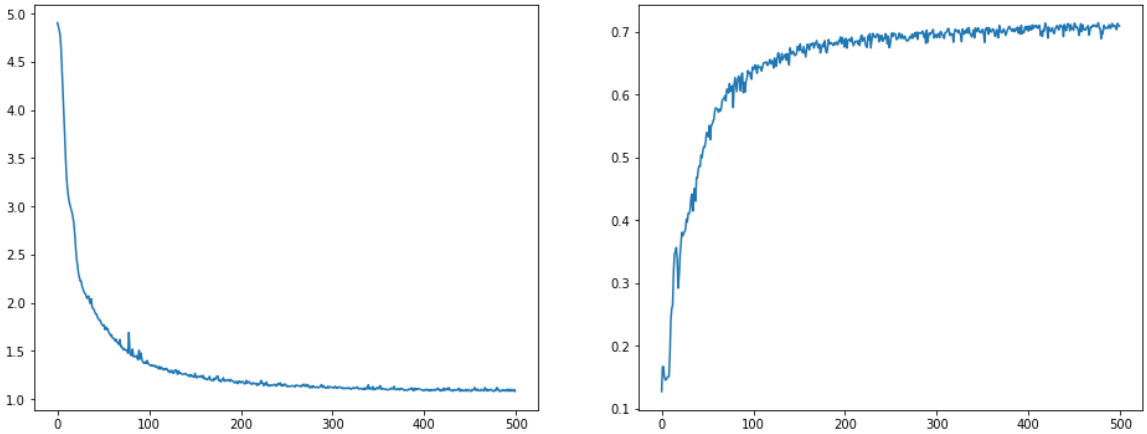
```
_,loss,acc  = do_train(top_1['f1'])
plt.figure(figsize=(16,6))
plt.subplot(121)
plt.plot(loss)
plt.subplot(122)
plt.plot(acc)
```

Loss:1.0824989318847655 500/500

Accurancy:0.7093736529350281: 100% [07:25<00:00, 1.14it/s]

Out[9]:

```
[<matplotlib.lines.Line2D at 0x7f0d8eb5ec90>]
```



Top only tag all (tokens) mode

accuracy

In [10]:

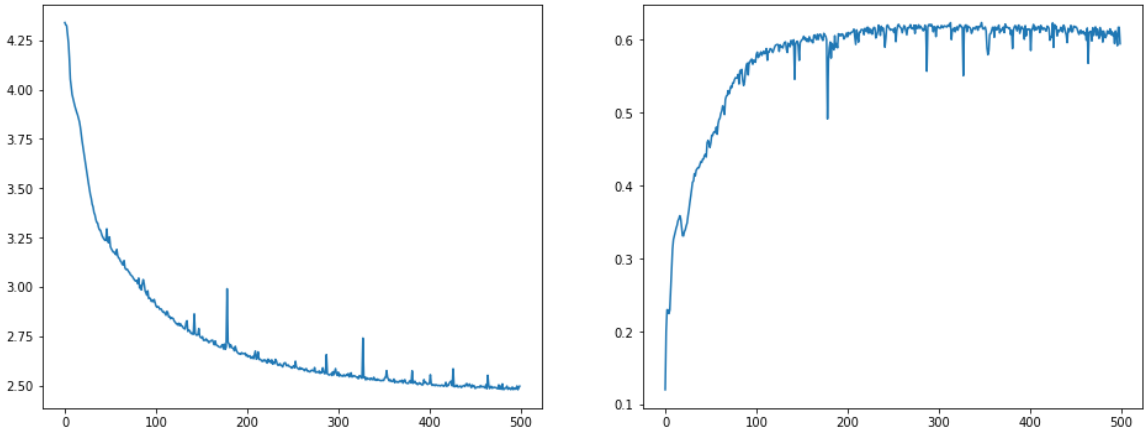
```
_,loss,acc = do_train(top_1_all['accuracy'])
plt.figure(figsize=(16,6))
plt.subplot(121)
plt.plot(loss)
plt.subplot(122)
plt.plot(acc)
```

Loss:2.4961227416992187 500/500

Accurancy:0.5940973162651062: 100% [02:54<00:00, 2.89it/s]

Out[10]:

```
[<matplotlib.lines.Line2D at 0x7f0d8e79a8d0>]
```



alpha

In [11]:

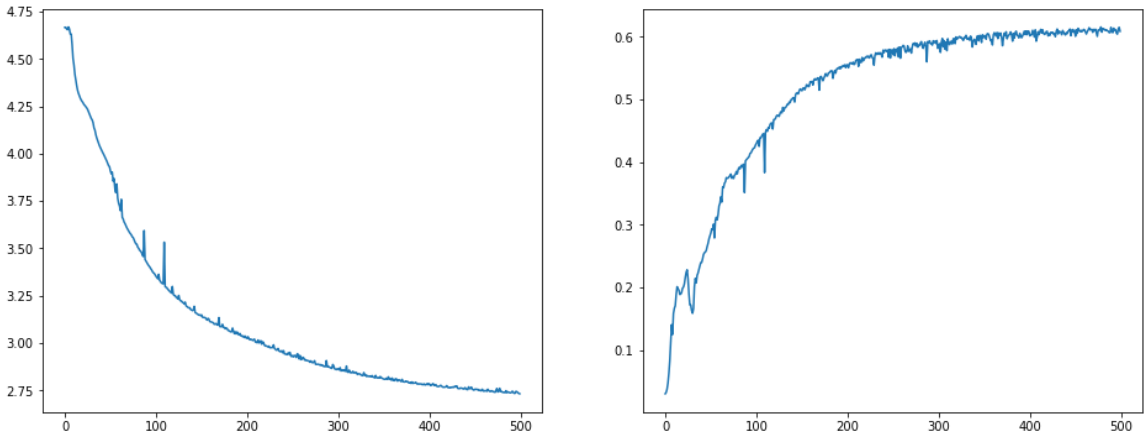
```
_,loss,acc = do_train(top_1_all['alpha'])
plt.figure(figsize=(16,6))
plt.subplot(121)
plt.plot(loss)
plt.subplot(122)
plt.plot(acc)
```

Loss:2.7331411838531494 500/500

Accurancy:0.6088253259658813: 100% [05:59<00:00, 1.42it/s]

Out[11]:

```
[<matplotlib.lines.Line2D at 0x7f0d79507090>]
```



combined

In [12]:

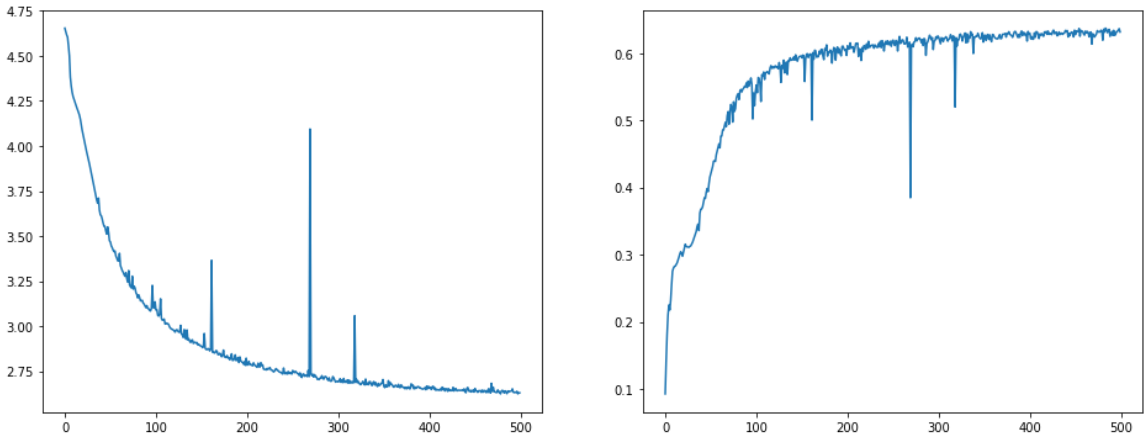
```
_,loss,acc = do_train(top_1_all['combined'])
plt.figure(figsize=(16,6))
plt.subplot(121)
plt.plot(loss)
plt.subplot(122)
plt.plot(acc)
```

Loss:2.630258893966675 500/500

Accurancy:0.6316718459129333: 100% [05:46<00:00, 1.45it/s]

Out[12]:

```
[<matplotlib.lines.Line2D at 0x7f0d779204d0>]
```



f1

In [13]:

```
_,loss,acc = do_train(top_1_all['f1'])
plt.figure(figsize=(16,6))
plt.subplot(121)
plt.plot(loss)
plt.subplot(122)
plt.plot(acc)
```

Loss:2.631148433685303 500/500

Accurancy:0.6420882344245911: 100% [05:47<00:00, 1.46it/s]

Out[13]:

```
[<matplotlib.lines.Line2D at 0x7f0d78fb3390>]
```