In [1]:
```
import sys

import pandas as pd
import torch

from hmlstm.utils import plot_z, get_boundaries, evaluate_z, get_z, stack_z

try:
    from google.colab import drive

    IN_COLAB = True

    drive.mount('/content/drive')
    path = "/content/drive/My Drive/Colab Notebooks/"

    # for python imports from google drive
    sys.path.append(path)
except:
    IN_COLAB = False
    path = "./"

from utils.datasets import Characters as TextDataset
from utils import HMLSTMTrainer as Trainer, print_cuda_info, load_model
from hmlstm import HMLSTMNetwork

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [2]:
```
use_cuda = torch.cuda.is_available()
# use_cuda = False

if use_cuda:
    print_cuda_info()
    torch.cuda.empty_cache()
    # torch.set_num_threads(1)

device = torch.device('cuda' if use_cuda else 'cpu')
```

```
cuda device: 0 / name: GeForce GTX 1080 Ti / cuda-capability: (6, 1) / memory: 1
1.0 GB
```

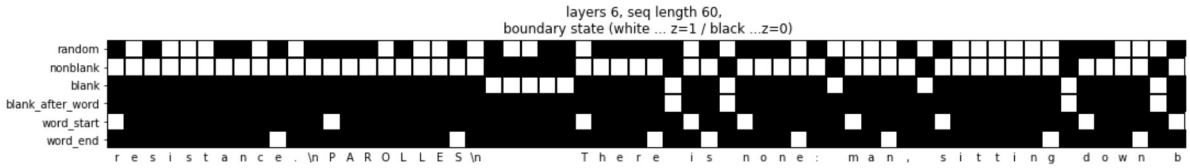
In [3]:
```
seq_len = 60

project_name = "shakespeare"

path_data = path + "projects/" + project_name + "/data/"
path_states = path + "projects/" + project_name + "/states/"
dataset = TextDataset(path_data + "data.txt", seq_length=seq_len)
```

In [4]:
```
print(f"dataset length: {len(dataset)}")

i = 100
x, y, i = dataset.get_sample(i, decode=False)
print(f"sample: \"{dataset.decode(x)}\"")
```

```
dataset length: 87128
sample: "resistance.
PAROLLES
     There is none: man, sitting down b"
```

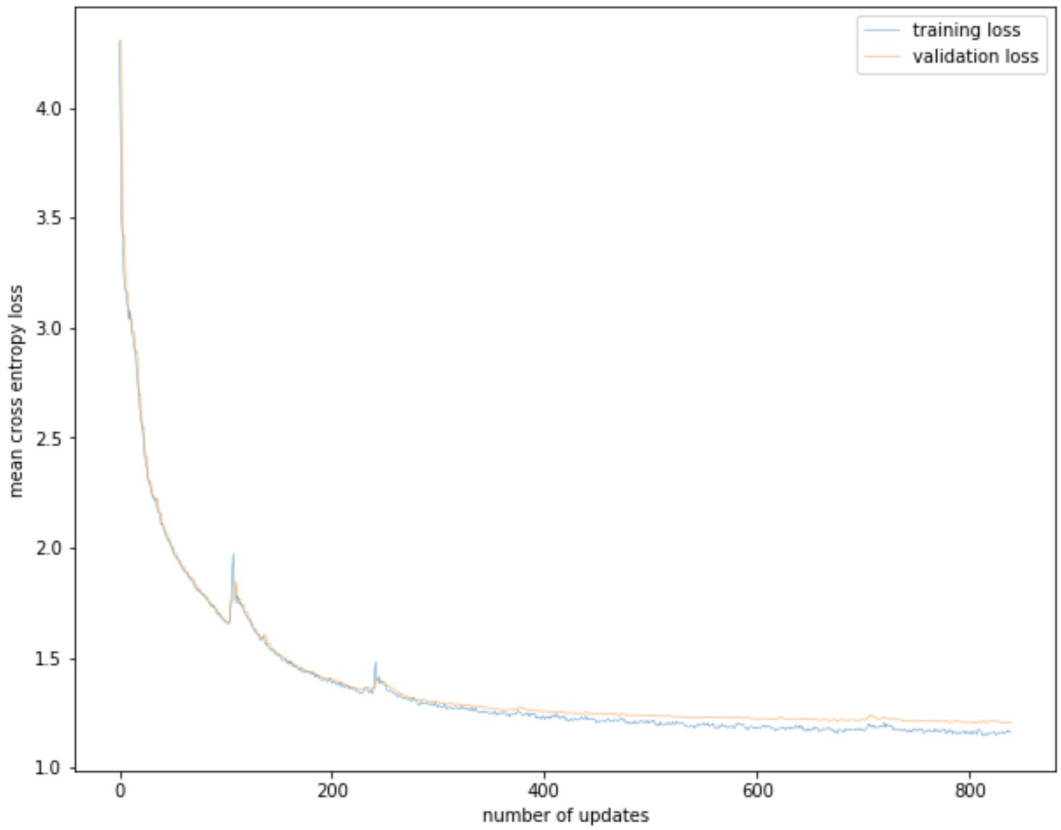
In [5]:
```
metrics = {
    "word_end": r"[a-zA-Z]\b",
    "word_start": r"\b[a-zA-Z]",
#     "blank_before_word": r" \b",
    "blank_after_word": r"\b ",
    "blank": r" ",
    "nonblank": r"[^ ]",
    "random": "random_0.5" # comparison with random boundary
}
metrics = get_boundaries(metrics, dataset.text, dataset.seq_length, device=device)
zb = stack_z(metrics, i, dataset.seq_length)

print("reference boundaries")
plot_z(zb, list(dataset.decode(x)), list(metrics.keys()))
```

```
reference boundaries
```



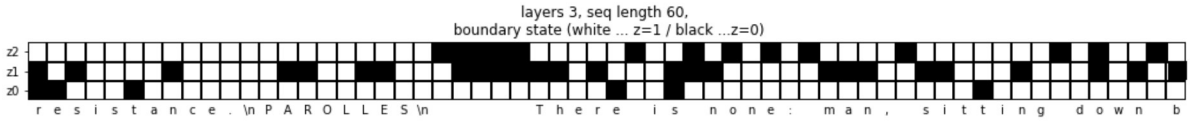
In [9]:
```
# best val loss

input_size = 1
embedding_size_input = 128
hidden_sizes = [114, 204, 182]

embedding_size_output = 256
linear_sizes = [128]
output_size = dataset.vocab_len()

model = HMLSTMNetwork(input_size, embedding_size_input, hidden_sizes, embedding_siz
e_output, linear_sizes, output_size,
                      layer_norm=True)
trainer = Trainer(model, None, None, None, device=device)
trainer.load_state(path=path_states + "20200308_231327/" + "complete.pt")

trainer.plot_loss()
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

In [10]:
```
z, text = get_z(x, model, dataset, device)
plot_z(z, list(text))
```



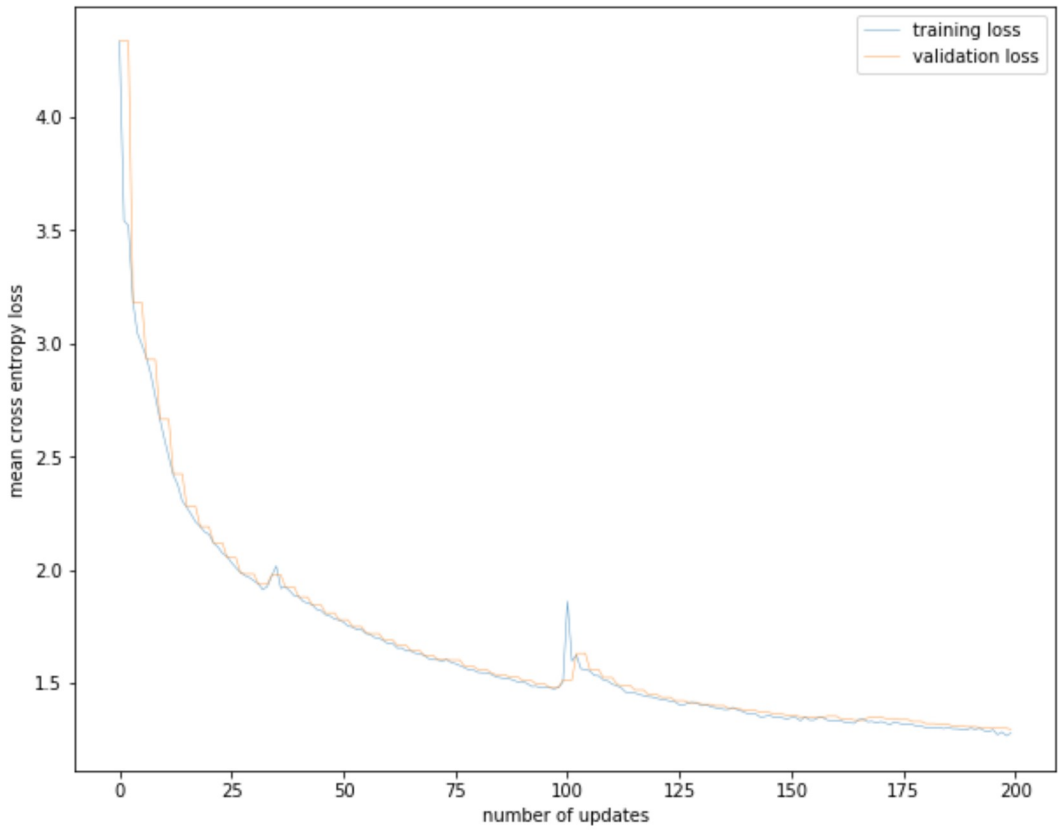
In [11]:
```
# this takes some time
results, metrics = evaluate_z(model, dataset, metrics, batch_size=40000, device=dev
ice, build_metrics=False)
results = { key: values.cpu().tolist() for key, values in results.items()}

data = pd.DataFrame(results)
data.index = ["layer_" + str(l) for l in range(len(data))]
data.head()
```

Out[11]:

	word_end_f1_score	word_end_accuracy	word_end_cross_entropy	word_start_f1_score	word_start_acc
layer_0	0.341895	0.437055	0.251533	0.277958	0.1
layer_1	0.409628	0.623231	0.430225	0.085733	0.0
layer_2	0.260545	0.207724	0.328076	0.274459	0.1

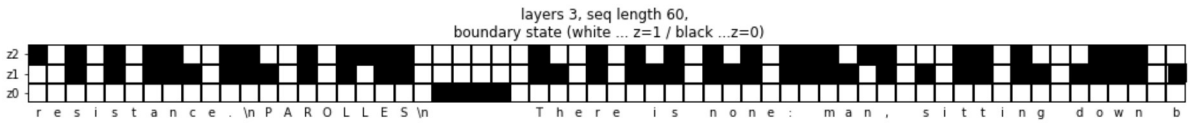
In [17]:
```
# 2nd best val loss
input_size = 1
embedding_size_input = 128
hidden_sizes = [256, 128, 64]

embedding_size_output = 256
linear_sizes = [128]
output_size = dataset.vocab_len()

model = HMLSTMNetwork(input_size, embedding_size_input, hidden_sizes, embedding_siz
e_output, linear_sizes, output_size,
                      layer_norm=True)
trainer = Trainer(model, None, None, None, device=device)
trainer.load_state(path=path_states + "20200308_213204/" + "checkpoint.pt")

trainer.plot_loss()
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

In [18]:
```
z, text = get_z(x, model, dataset, device)
plot_z(z, list(text))
```



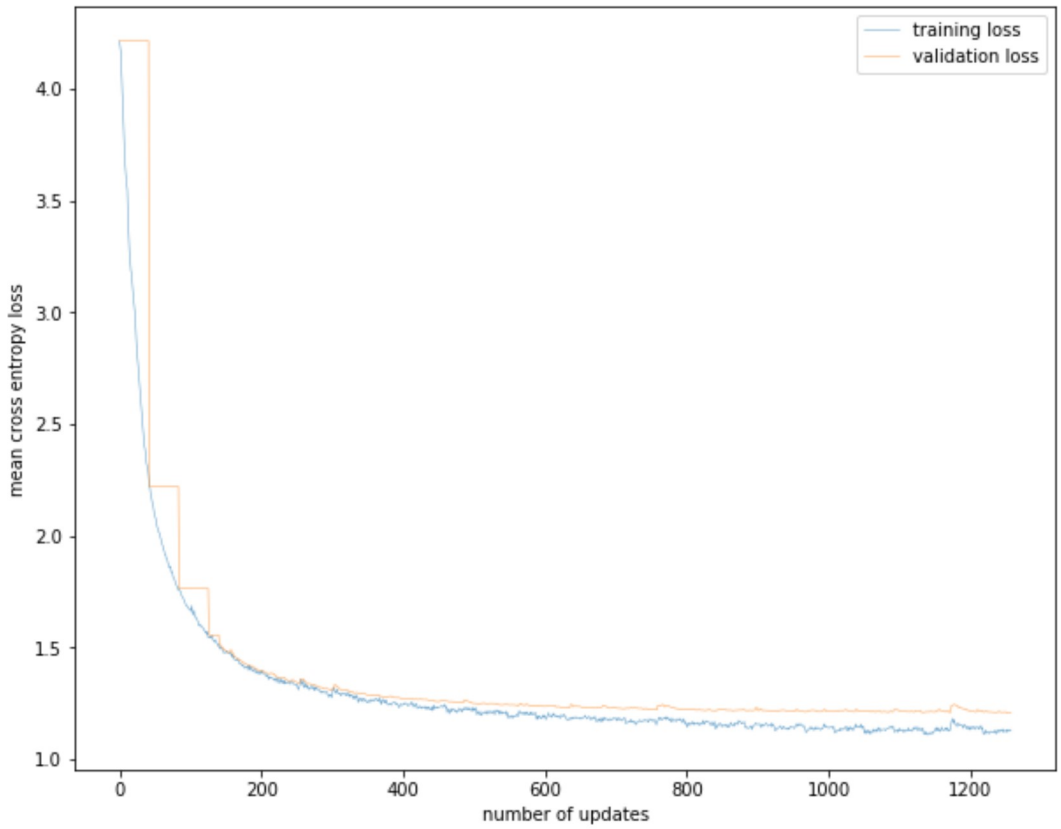
In [19]:
```
# this takes some time
results, metrics = evaluate_z(model, dataset, metrics, batch_size=40000, device=dev
ice, build_metrics=False)
results = { key: values.cpu().tolist() for key, values in results.items()}

data = pd.DataFrame(results)
data.index = ["layer_" + str(l) for l in range(len(data))]
data.head()
```

Out[19]:

	word_end_f1_score	word_end_accuracy	word_end_cross_entropy	word_start_f1_score	word_start_acc
layer_0	0.212748	0.410294	1.017694	0.167157	0.0
layer_1	0.227303	0.529731	1.138739	0.025567	0.0
layer_2	0.299610	0.229767	0.038346	0.287009	0.1

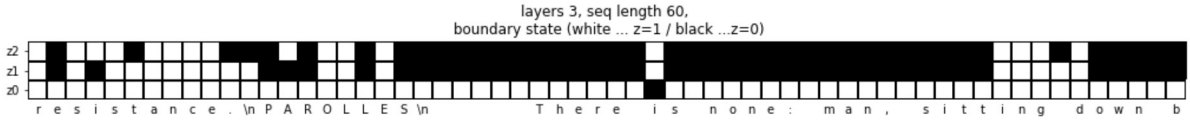
In [6]:
```
input_size = 1
embedding_size_input = 32
hidden_sizes = [256, 256, 256]

embedding_size_output = 128
linear_sizes = []
output_size = dataset.vocab_len()

model = HMLSTMNetwork(input_size, embedding_size_input, hidden_sizes, embedding_siz
e_output, linear_sizes, output_size,
                      layer_norm=True)
trainer = Trainer(model, None, None, None, device=device)
trainer.load_state(path=path_states + "20200309_174119/" + "checkpoint.pt")

trainer.plot_loss()
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

In [7]:
```
z, text = get_z(x, model, dataset, device)
plot_z(z, list(text))
```



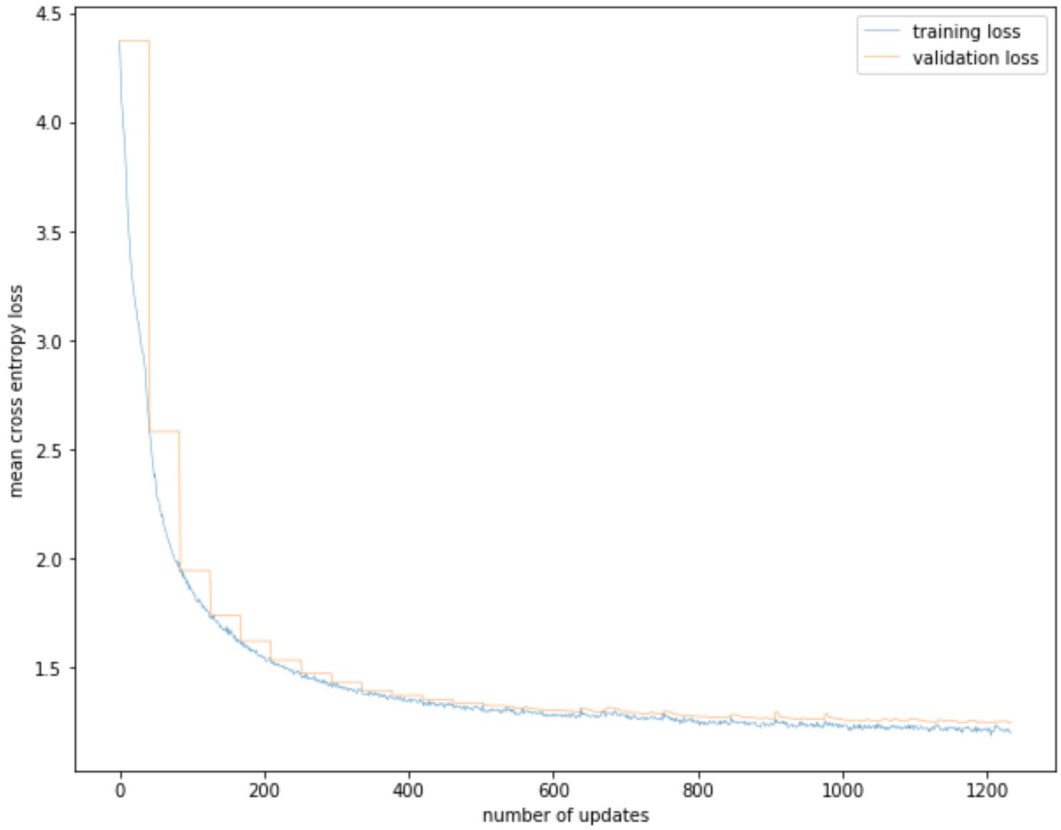
In [9]:
```
# this takes some time
results, metrics = evaluate_z(model, dataset, metrics, batch_size=1000, device=devi
ce, build_metrics=False)
results = { key: values.cpu().tolist() for key, values in results.items()}

data = pd.DataFrame(results)
data.index = ["layer_" + str(l) for l in range(len(data))]
data.head()
```

Out[9]:

	word_end_f1_score	word_end_accuracy	word_end_cross_entropy	word_start_f1_score	word_start_acc
layer_0	0.267679	0.677286	1.256718	0.058115	0.0
layer_1	0.314568	0.665593	1.052243	0.073979	0.0
layer_2	0.279764	0.242547	0.241985	0.262203	0.1

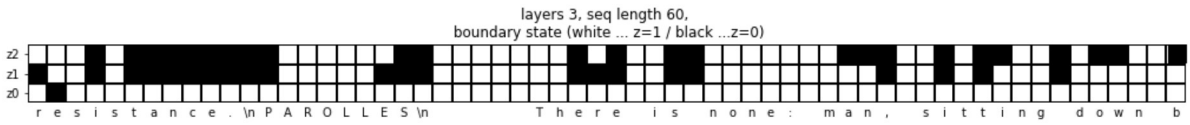
In [10]:
```
input_size = 1
embedding_size_input = 32
hidden_sizes = [160, 128, 96]

embedding_size_output = 128
linear_sizes = [64]
output_size = dataset.vocab_len()

model = HMLSTMNetwork(input_size, embedding_size_input, hidden_sizes, embedding_siz
e_output, linear_sizes, output_size,
                      layer_norm=True)

trainer = Trainer(model, None, None, None, device=device)
trainer.load_state(path=path_states + "20200309_210936/" + "checkpoint.pt")

trainer.plot_loss()
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

In [11]:
```
z, text = get_z(x, model, dataset, device)
plot_z(z, list(text))
```



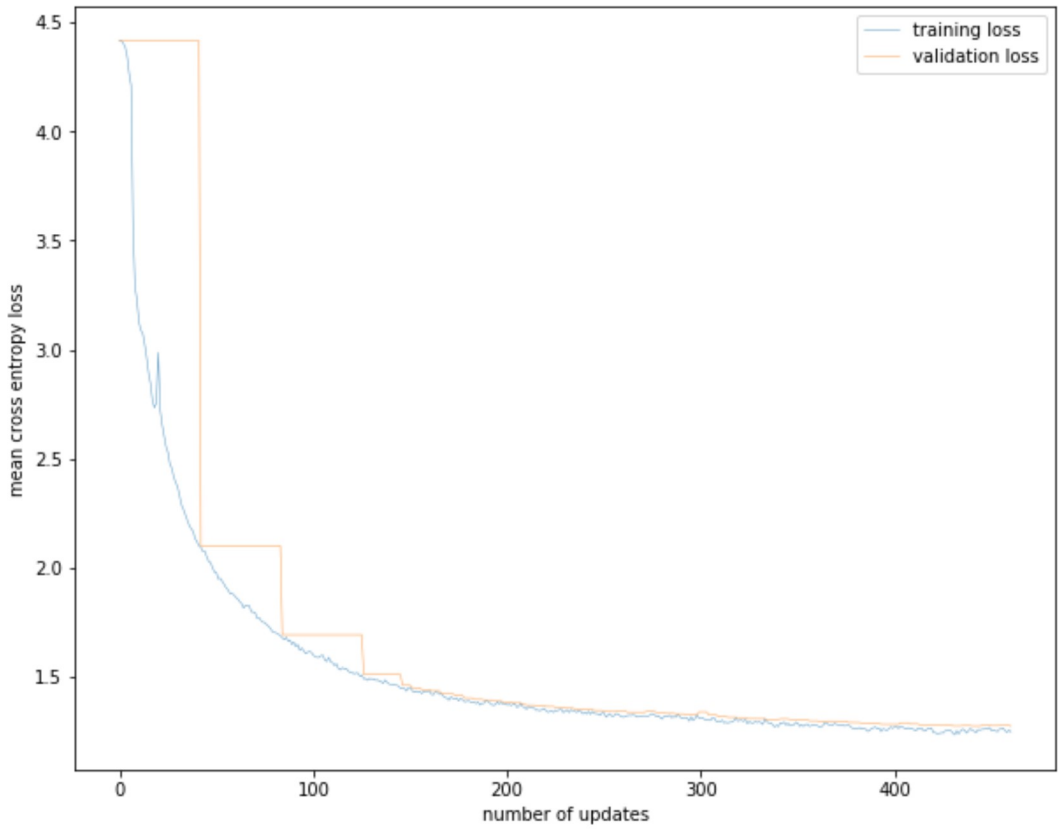
In [12]:
```
# this takes some time
results, metrics = evaluate_z(model, dataset, metrics, batch_size=1000, device=devi
ce, build_metrics=False)
results = { key: values.cpu().tolist() for key, values in results.items()}

data = pd.DataFrame(results)
data.index = ["layer_" + str(l) for l in range(len(data))]
data.head()
```

Out[12]:

	word_end_f1_score	word_end_accuracy	word_end_cross_entropy	word_start_f1_score	word_start_acc
layer_0	0.219309	0.411586	0.984097	0.236908	0.1
layer_1	0.245051	0.304650	0.636345	0.278046	0.1
layer_2	0.286627	0.178675	0.035992	0.283080	0.1

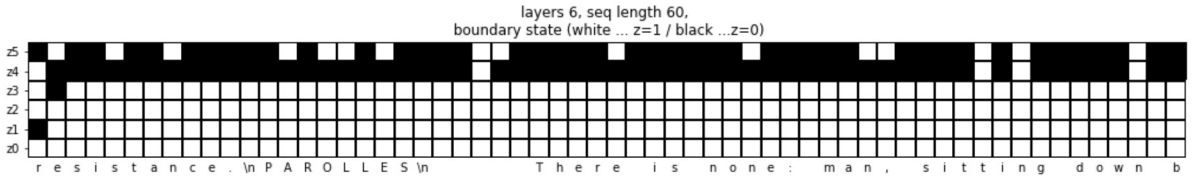
In [18]:
```
input_size = 1
embedding_size_input = 32
hidden_sizes = [160, 120, 100, 80, 60, 40]

embedding_size_output = 128
linear_sizes = []
output_size = dataset.vocab_len()

model = HMLSTMNetwork(input_size, embedding_size_input, hidden_sizes, embedding_siz
e_output, linear_sizes, output_size,
                      layer_norm=True)
trainer = Trainer(model, None, None, None, device=device)
trainer.load_state(path=path_states + "20200309_222751/" + "checkpoint.pt")

trainer.plot_loss()
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

In [19]:
```
z, text = get_z(x, model, dataset, device)
plot_z(z, list(text))
```



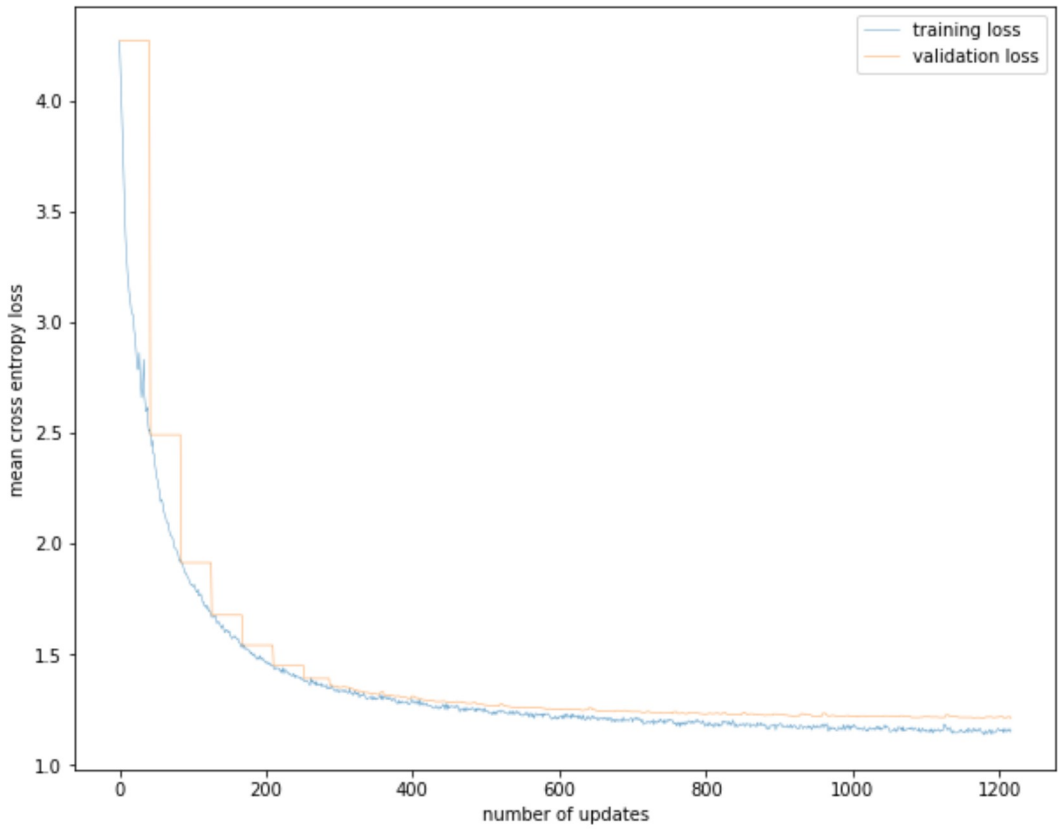
In [20]:
```
# this takes some time
results, metrics = evaluate_z(model, dataset, metrics, batch_size=1000, device=devi
ce, build_metrics=False)
results = { key: values.cpu().tolist() for key, values in results.items()}

data = pd.DataFrame(results)
data.index = ["layer_" + str(l) for l in range(len(data))]
data.head(n=6)
```

Out[20]:

	word_end_f1_score	word_end_accuracy	word_end_cross_entropy	word_start_f1_score	word_start_acc
layer_0	0.289033	0.733708	1.312585	0.006389	0.0
layer_1	0.136911	0.755929	1.712870	0.018611	0.0
layer_2	0.286560	0.211964	0.113605	0.276549	0.1
layer_3	0.287879	0.168244	0.000044	0.287856	0.1
layer_4	0.288636	0.177961	0.015592	0.285700	0.1
layer_5	0.286681	0.168911	0.012885	0.287159	0.1

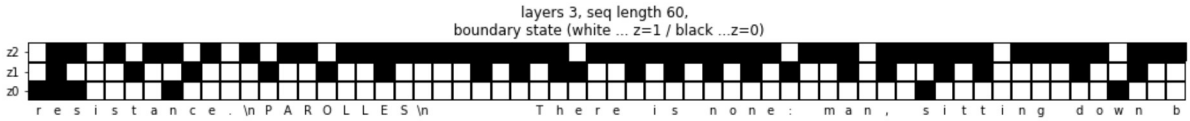
In [21]:
```
input_size = 1
embedding_size_input = 32
hidden_sizes = [256, 128, 128]

embedding_size_output = 256
linear_sizes = [128]
output_size = dataset.vocab_len()

model = HMLSTMNetwork(input_size, embedding_size_input, hidden_sizes, embedding_siz
e_output, linear_sizes, output_size,
                      layer_norm=True)
trainer = Trainer(model, None, None, None, device=device)
trainer.load_state(path=path_states + "20200310_130451/" + "checkpoint.pt")

trainer.plot_loss()
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

In [23]:
```
z, text = get_z(x, model, dataset, device)
plot_z(z, list(text))
```



In [24]:
```
# this takes some time
results, metrics = evaluate_z(model, dataset, metrics, batch_size=40000, device=dev
ice, build_metrics=False)
results = { key: values.cpu().tolist() for key, values in results.items()}

data = pd.DataFrame(results)
data.index = ["layer_" + str(l) for l in range(len(data))]
data.head()
```

Out[24]:

	word_end_f1_score	word_end_accuracy	word_end_cross_entropy	word_start_f1_score	word_start_acc
layer_0	0.263986	0.738137	1.394458	0.009646	0.0
layer_1	0.274264	0.499881	0.847096	0.148914	0.0
layer_2	0.279469	0.242718	0.244227	0.263999	0.1

In [ ]: