Самостоятельно обучить классификатор текстов на примере 20newsgroups
На примере 20 newsgroups попробовать разные параметры для сверток для классификации текстов

In [141]:

```
import pandas as pd
import numpy as np
from sklearn.datasets import fetch_20newsgroups
```

возьмем только 4 категории из 20 доступных в наборе данных

In [142]:

```
categories = ['alt.atheism', 'soc.religion.christian',
              'comp.graphics', 'sci.med']
```

In [143]:

```
train = fetch_20newsgroups(subset='train',
   categories=categories, shuffle=True, random_state=42)
```

In [144]:

```
train.target_names
```

Out[144]:

```
['alt.atheism', 'comp.graphics', 'sci.med', 'soc.religion.christian']
```

In [145]:

```
print(len(train.data))
len(train.filenames)
```

```
2257
```

Out[145]:

```
2257
```

In [146]:

```
print(train.target_names[train.target[6]])
train.target[:10]
```

```
soc.religion.christian
```

Out[146]:

```
array([1, 1, 3, 3, 3, 3, 3, 2, 2, 2], dtype=int64)
```

In [147]:

```
for i in train.target[:10]:
    print(train.target_names[i])
```

```
comp.graphics
comp.graphics
soc.religion.christian
soc.religion.christian
soc.religion.christian
soc.religion.christian
soc.religion.christian
sci.med
sci.med
sci.med
```

Выведем первые строки первого загруженного файла

In [ ]:

```

```

In [148]:

```
columns=['data', 'target']
```

In [149]:

```
df = pd.DataFrame(columns= columns)
```

In [150]:

```
# a = {'data': "\n".join(train.data[0].split("\n")),
#      'target': train.target[1]}
# df = df.append(a, ignore_index=True)
# df.iloc[0,0]
```

In [151]:

```
"\n".join(train.data[2].split("\n"))
```

Out[151]:

```
"From: djohnson@cs.ucsd.edu (Darin Johnson)\nSubject: Re: harrassed at work,
could use some prayers\nOrganization: =CSE Dept., U.C. San Diego\nLines: 63
\n\n(Well, I'll email also, but this may apply to other people, so\nI'll pos
t also.)\n\n>I've been working at this company for eight years in various\n>
engineering jobs.  I'm female.  Yesterday I counted and realized that\n>on s
even different occasions I've been sexually harrassed at this\n>company.\n\n
>I dreaded coming back to work today.  What if my boss comes in to ask\n>me
some kind of question...\n\nYour boss should be the person bring these probl
ems to.  If he/she\ndoes not seem to take any action, keep going up higher a
nd higher.\nSexual harrassment does not need to be tolerated, and it can be
an\nenormous emotional support to discuss this with someone and know that\nt
hey are trying to do something about it.  If you feel you can not\ndiscuss t
his with your boss, perhaps your company has a personnel\ndepartment that ca
n work for you while preserving your privacy.  Most\ncompanies will want to
deal with this problem because constant anxiety\ndoes seriously affect how e
ffectively employees do their jobs.\n\nIt is unclear from your letter if you
have done this or not.  It is\nnot inconceivable that management remains ign
orant of employee\nproblems/strife even after eight years (it's a miracle if
they do\nnotice).  Perhaps your manager did not bring to the attention of\nh
igher ups?  If the company indeed does seem to want to ignore the\nentire pr
oblem, there may be a state agency willing to fight with\nyou.  (check with
a lawyer, a women's resource center, etc to find out)\n\nYou may also want t
o discuss this with your paster, priest, husband,\netc.  That is, someone yo
u know will not be judgemental and that is\nsupportive, comforting, etc.  Th
is will bring a lot of healing.\n\n>So I returned at 11:25, only to find tha
t ever single\n>person had already left for lunch.  They left at 11:15 or s
o.  No one\n>could be bothered to call me at the other building, even though
my\n>number was posted.\n\nThis happens to a lot of people.  Honest.  I beli
eve it may seem\nto be due to gross insensitivity because of the feelings yo
u are\ngoing through.  People in offices tend to be more insensitive while\n
working than they normally are (maybe it's the hustle or stress or...)\nI've
had this happen to me a lot, often because they didn't realize\nmy car was b
roken, etc.  Then they will come back and wonder why I\ndidn't want to go (t
his would tend to make me stop being angry at\nbeing ignored and make me lau
gh).  Once, we went off without our\nboss, who was paying for the lunch :-)
\n\n>For this\n>reason I hope good Mr. Moderator allows me this latest indul
gence.\n\nWell, if you can't turn to the computer for support, what would\nw
e do?  (signs of the computer age :-)\n\nIn closing, please don't let the ha
teful actions of a single person\nharm you.  They are doing it because they
are still the playground\nbully and enjoy seeing the hurt they cause.  And y
ou should not\naccept the opinions of an imbecile that you are worthless - m
uch\nwiser people hold you in great esteem.\n-- \nDarin Johnson\ndjohnson@uc
sd.edu\n  - Luxury!  In MY day, we had to make do with 5 bytes of swap...\n"
```

In [152]:

```
for i in range(len(train.data)):
    a = {'data': "\n".join(train.data[i].split("\n")),
    'target': train.target[i]}
    df = df.append(a, ignore_index=True)
```

In [153]:

```
df.head()
```

Out[153]:

	data	target
0	From: sd345@city.ac.uk (Michael Collier)\nSubj...	1
1	From: ani@ms.uky.edu (Aniruddha B. Deglurkar)\...	1
2	From: djohnson@cs.ucsd.edu (Darin Johnson)\nSu...	3
3	From: s0612596@let.rug.nl (M.M. Zwart)\nSubjec...	3
4	From: stanly@grok11.columbiasc.ncr.com (stanly...	3

In [154]:

```
df = df.astype({"target": np.int64})
```

In [155]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2257 entries, 0 to 2256
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   data    2257 non-null   object
 1   target  2257 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 35.4+ KB
```

In [156]:

```
from string import punctuation
from stop_words import get_stop_words
from pymorphy2 import MorphAnalyzer
import re
```

In [157]:

```
sw = set(get_stop_words("ru"))
exclude = set(punctuation)
morpher = MorphAnalyzer()

def preprocess_text(txt):
    txt = str(txt)
    txt = "".join(c for c in txt if c not in exclude)
    txt = txt.lower()
    txt = re.sub("не\s*", "не", txt)
    txt = [morpher.parse(word)[0].normal_form for word in txt.split() if word not in sw]
    return " ".join(txt)

df['data'] = df.data.apply(preprocess_text)
# df_val['text'] = df_val['text'].apply(preprocess_text)
# df_test['text'] = df_test['text'].apply(preprocess_text)
```

In [158]:

```
dfn = df.head(5)
```

In [159]:

```
num = []
```

In [160]:

```
for i in df['data']:
#     print(i, '\n')
    num.append(len(i))

print(max(num))
```

```
53709
```

In [161]:

```
tr = int(len(train.data)* 0.6)
tv = int(len(train.data)* 0.2)
print(tr, tv)
```

```
1354 451
```

In [162]:

```
df['target'].unique()
```

Out[162]:

```
array([1, 3, 2, 0], dtype=int64)
```

In [163]:

```
df_train = df.loc[:tr]
df_test = df.loc[tr:tv+tr]
df_val = df.loc[tv+tr:]

print(df_train.shape)
print(df_test.shape)
print(df_val.shape)
```

```
(1355, 2)
(452, 2)
(452, 2)
```

In [164]:

```
train_corpus = " ".join(df_train["data"])
train_corpus = train_corpus.lower()
```

In [165]:

```
import nltk
from nltk.tokenize import word_tokenize
nltk.download("punkt")

tokens = word_tokenize(train_corpus)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\VoronkovSergey\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [166]:

```
tokens_filtered = [word for word in tokens if word.isalnum()]
```

In [167]:

```
max_words = 2000
max_len = 1000
num_classes = 4
```

In [168]:

```
from nltk.probability import FreqDist
dist = FreqDist(tokens_filtered)
tokens_filtered_top = [pair[0] for pair in dist.most_common(max_words-1)]
```

In [169]:

```
tokens_filtered_top[10:]
```

Out[169]:

```
['for',
 'you',
 'not',
 'this',
 'be',
 'are',
 'from',
 'have',
 'as',
 'with',
 'on',
 'but',
 'or',
 'if',
 'was',
 'can',
 'we',
 'by']
```

In [170]:

```
vocabulary = {v: k for k, v in dict(enumerate(tokens_filtered_top, 1)).items()}
```

In [171]:

```
import numpy as np
def text_to_sequence(text, maxlen):
    result = []
    tokens = word_tokenize(text.lower())
    tokens_filtered = [word for word in tokens if word.isalnum()]
    for word in tokens_filtered:
        if word in vocabulary:
            result.append(vocabulary[word])
    padding = [0]*(maxlen-len(result))
    return padding + result[-maxlen:]
```

In [172]:

```
tokens = word_tokenize(df_train["data"][0].lower())
```

In [173]:

```
tokens_filtered = [word for word in tokens if word.isalnum()]
```

In [174]:

```
tokens_filtered[:5]
```

Out[174]:

```
['from', 'sd345cityacuk', 'michael', 'collier', 'subject']
```

In [175]:

```
result = []
```

In [176]:

```
x_train = np.asarray([text_to_sequence(text, max_len) for text in df_train["data"]], dtype=
x_test = np.asarray([text_to_sequence(text, max_len) for text in df_test["data"]], dtype=np
x_val = np.asarray([text_to_sequence(text, max_len) for text in df_val["data"]], dtype=np.i
```

In [177]:

```
x_train[1]
```

Out[177]:

```
array([  0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
```

```
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,   17,  383,   31,
         150,    4,  632,    4,   37,   70,    2,  457,    2, 1506,   34,
        1232,  622,    9,   18,    4,  140,    9,  366,   46,    2,    1,
          26,  150,   56, 1636,    2,    1,  140,    9,   18,    4,    7,
           1, 1588,  488,    1,    6,    4,    2,    4,  377,  116, 1367,
           1,  732,    7,   14, 1588,   55,    6, 1368,    4,   82,    3,
          15,    1,    6,    4,  292,    2, 1367,   11,    1, 1930,    2,
           1,    6,  196,   90,    2, 1853,   41,  140,    6,   14,    1,
         732,   51,    3,   15,   90,  116, 1104, 1931, 1853,   28,    1,
          24,    4,    6, 1368,    1,   10,    6, 1407,   19,    4,  388,
         322,   10,    6,   67,   54,  343,   64,   78,   26,   15,  259,
          23,    6,   30,   54, 1006,   11,  277,   14,   54,  150,   42,
          15,  720,  215,    3,   96,    6,  208,    3, 1455,  908])
```

In [178]:

```python
import random
import torch
import torch.nn as nn

seed = 0

random.seed(seed)
np.random.seed(seed)
torch.manual_seed(seed)
torch.cuda.manual_seed(seed)
torch.backends.cudnn.deterministic = True
```

In [179]:

```
from torch.utils.data import DataLoader, Dataset

class DataWrapper(Dataset):
    def __init__(self, data, target=None, transform=None):
        self.data = torch.from_numpy(data).long()
        if target is not None:
            self.target = torch.from_numpy(target).long()
        self.transform = transform

    def __getitem__(self, index):
        x = self.data[index]
        y = self.target[index] if self.target is not None else None

        if self.transform:
            x = self.transform(x)

        return x, y

    def __len__(self):
        return len(self.data)
```

In [252]:

```
class Net(nn.Module):
    def __init__(self, vocab_size=20, embedding_dim = 128, out_channel = 128, num_classes =
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.conv = nn.Conv1d(embedding_dim, out_channel, kernel_size=3)
        self.relu = nn.ReLU()
        self.linear = nn.Linear(out_channel, num_classes)

    def forward(self, x):
        output = self.embedding(x)
        #                       B  F  L
        output = output.permute(0, 2, 1)
        output = self.conv(output)
        output = self.relu(output)
        output = torch.max(output, axis=2).values
#         print(output.shape)
        output = self.linear(output)
        print(output.shape)

        return output
```

In [253]:

```
# result_loss = pd.DataFrame(columns= ['epochs', 'batch', 'lr','optimizer' , 'criterion', '
# result_loss
```

In [254]:

```
# Training
epochs = 20
batch_size = 128
print_batch_n = 100
learning_rate = 0.01
criterion = nn.BCEWithLogitsLoss()
# criterion = nn.CrossEntropyLoss()
# optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate, momentum=0.9)
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

opt = str(optimizer).split()[0]
```

In [255]:

```
model = Net(vocab_size=max_words)

print(model)
print("Parameters:", sum([param.nelement() for param in model.parameters()]))

model.train()
#model = model.cuda()

# optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
optimizer = optimizer
criterion = criterion




train_dataset = DataWrapper(x_train, df_train['target'].values)
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True,drop_last=True

val_dataset = DataWrapper(x_val, df_val['target'].values)
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=True,drop_last=True)

loss_history = []

for epoch in range(1, epochs + 1):
    print(f"Train epoch {epoch}/{epochs}")
    for i, (data, target) in enumerate(train_loader):
        optimizer.zero_grad()

        # data = data.cuda()
        # target = target.cuda()

        # compute output
        output = model(data)

        # compute gradient and do SGD step
#         print(target.float().view(-1, 1))
        loss = criterion(output, target.float().view(-1, 1))
        loss.backward()

        optimizer.step()

        if i%print_batch_n == 0:
            loss = loss.float().item()
            print("Step {}: loss={}".format(i, loss))
            loss_history.append(loss)
            result_loss = result_loss.append({'epochs': epochs, 'batch': batch_size,'optimi
                                              'lr': learning_rate, 'loss': loss},
                                              ignore_index= True)
```

```
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
Train epoch 20/20
torch.Size([128, 1])
```

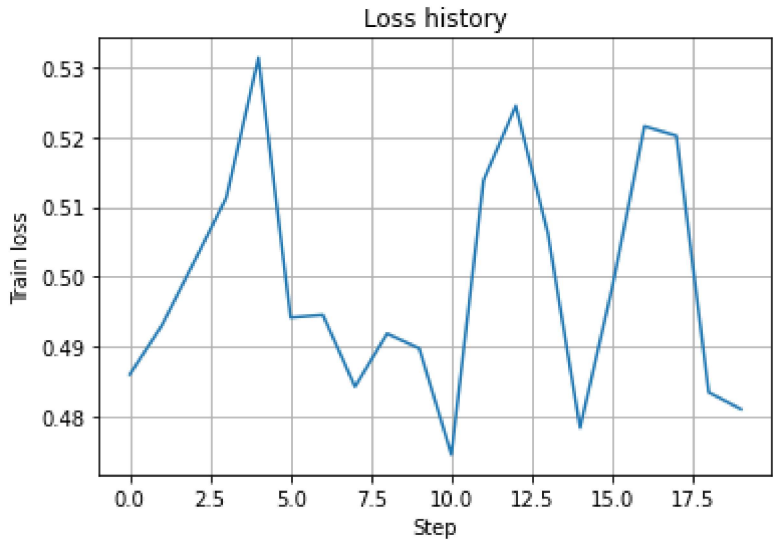
```
Step 0: loss=0.48104915022850037
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
torch.Size([128, 1])
```

In [256]:

```
import matplotlib.pyplot as plt
%pylab inline
plt.title('Loss history')
plt.grid(True)
plt.ylabel('Train loss')
plt.xlabel('Step')
plt.plot(loss_history);
```

```
Populating the interactive namespace from numpy and matplotlib

c:\program files\python37\lib\site-packages\IPython\core\magics\pylab.py:16
0: UserWarning: pylab import has clobbered these variables: ['seed', 'rando
m']
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```



In [139]:

```
result_loss.shape
```

Out[139]:

```
(80, 6)
```

In [110]:

```
# result_loss.to_csv('loss.csv')
```

In [135]:

```
result_loss = pd.read_csv('loss.csv', index_col=0)
```

In [131]:

```
print(result_loss.head(60))
```

```
    epochs  batch     lr optimizer            criterion      loss
0       20    128  0.010      Adam  BCEWithLogitsLoss() -0.248587
1       20    128  0.010      Adam  BCEWithLogitsLoss() -0.602081
2       20    128  0.010      Adam  BCEWithLogitsLoss() -0.459542
3       20    128  0.010      Adam  BCEWithLogitsLoss() -0.499464
4       20    128  0.010      Adam  BCEWithLogitsLoss() -0.476288
5       20    128  0.010      Adam  BCEWithLogitsLoss() -0.394077
6       20    128  0.010      Adam  BCEWithLogitsLoss() -0.833189
7       20    128  0.010      Adam  BCEWithLogitsLoss() -0.407315
8       20    128  0.010      Adam  BCEWithLogitsLoss() -0.643042
9       20    128  0.010      Adam  BCEWithLogitsLoss() -0.413099
10      20    128  0.010      Adam  BCEWithLogitsLoss() -0.713040
11      20    128  0.010      Adam  BCEWithLogitsLoss() -0.729515
12      20    128  0.010      Adam  BCEWithLogitsLoss() -0.455549
13      20    128  0.010      Adam  BCEWithLogitsLoss() -0.553259
14      20    128  0.010      Adam  BCEWithLogitsLoss() -0.382475
15      20    128  0.010      Adam  BCEWithLogitsLoss() -0.627800
16      20    128  0.010      Adam  BCEWithLogitsLoss() -0.702067
17      20    128  0.010      Adam  BCEWithLogitsLoss() -0.531590
18      20    128  0.010      Adam  BCEWithLogitsLoss() -0.588165
19      20    128  0.010      Adam  BCEWithLogitsLoss() -0.668297
20      20    512  0.001      Adam  BCEWithLogitsLoss()  1.192050
21      20    512  0.001      Adam  BCEWithLogitsLoss()  1.199037
22      20    512  0.001      Adam  BCEWithLogitsLoss()  1.184106
23      20    512  0.001      Adam  BCEWithLogitsLoss()  1.171876
24      20    512  0.001      Adam  BCEWithLogitsLoss()  1.183618
25      20    512  0.001      Adam  BCEWithLogitsLoss()  1.177779
26      20    512  0.001      Adam  BCEWithLogitsLoss()  1.169545
27      20    512  0.001      Adam  BCEWithLogitsLoss()  1.194365
28      20    512  0.001      Adam  BCEWithLogitsLoss()  1.187519
29      20    512  0.001      Adam  BCEWithLogitsLoss()  1.172828
30      20    512  0.001      Adam  BCEWithLogitsLoss()  1.201288
31      20    512  0.001      Adam  BCEWithLogitsLoss()  1.175898
32      20    512  0.001      Adam  BCEWithLogitsLoss()  1.181113
33      20    512  0.001      Adam  BCEWithLogitsLoss()  1.216401
34      20    512  0.001      Adam  BCEWithLogitsLoss()  1.192154
35      20    512  0.001      Adam  BCEWithLogitsLoss()  1.173951
36      20    512  0.001      Adam  BCEWithLogitsLoss()  1.209743
37      20    512  0.001      Adam  BCEWithLogitsLoss()  1.199711
38      20    512  0.001      Adam  BCEWithLogitsLoss()  1.184537
39      20    512  0.001      Adam  BCEWithLogitsLoss()  1.223843
40      10    512  0.010       SGD   CrossEntropyLoss() -0.000000
41      10    512  0.010       SGD   CrossEntropyLoss() -0.000000
42      10    512  0.010       SGD   CrossEntropyLoss() -0.000000
43      10    512  0.010       SGD   CrossEntropyLoss() -0.000000
44      10    512  0.010       SGD   CrossEntropyLoss() -0.000000
45      10    512  0.010       SGD   CrossEntropyLoss() -0.000000
46      10    512  0.010       SGD   CrossEntropyLoss() -0.000000
47      10    512  0.010       SGD   CrossEntropyLoss() -0.000000
48      10    512  0.010       SGD   CrossEntropyLoss() -0.000000
49      10    512  0.010       SGD   CrossEntropyLoss() -0.000000
50      10    512  0.001       SGD  BCEWithLogitsLoss()  2.058223
51      10    512  0.001       SGD  BCEWithLogitsLoss()  2.076523
52      10    512  0.001       SGD  BCEWithLogitsLoss()  2.116362
53      10    512  0.001       SGD  BCEWithLogitsLoss()  2.077231
54      10    512  0.001       SGD  BCEWithLogitsLoss()  2.103926
```

```
55      10    512  0.001       SGD  BCEWithLogitsLoss()  2.034565
56      10    512  0.001       SGD  BCEWithLogitsLoss()  2.014136
57      10    512  0.001       SGD  BCEWithLogitsLoss()  2.107410
58      10    512  0.001       SGD  BCEWithLogitsLoss()  2.092659
59      10    512  0.001       SGD  BCEWithLogitsLoss()  2.085810
```

In [123]:

```
loss_1 = result_loss.head(20)
loss_2 = result_loss.iloc[20:40]
loss_2 = loss_2.reset_index(drop=True)
loss_3 = result_loss.iloc[40:50]
loss_3 = loss_3.reset_index(drop=True)
loss_4 = result_loss.iloc[50:60]
loss_4 = loss_4.reset_index(drop=True)
loss_5 = result_loss.iloc[60:]
loss_5 = loss_5.reset_index(drop=True)
loss_4.head(20)
```

Out[123]:

	epochs	batch	lr	optimizer	criterion	loss
0	10	512	0.001	SGD	BCEWithLogitsLoss()	2.058223
1	10	512	0.001	SGD	BCEWithLogitsLoss()	2.076523
2	10	512	0.001	SGD	BCEWithLogitsLoss()	2.116362
3	10	512	0.001	SGD	BCEWithLogitsLoss()	2.077231
4	10	512	0.001	SGD	BCEWithLogitsLoss()	2.103926
5	10	512	0.001	SGD	BCEWithLogitsLoss()	2.034565
6	10	512	0.001	SGD	BCEWithLogitsLoss()	2.014136
7	10	512	0.001	SGD	BCEWithLogitsLoss()	2.107410
8	10	512	0.001	SGD	BCEWithLogitsLoss()	2.092659
9	10	512	0.001	SGD	BCEWithLogitsLoss()	2.085810

In [124]:

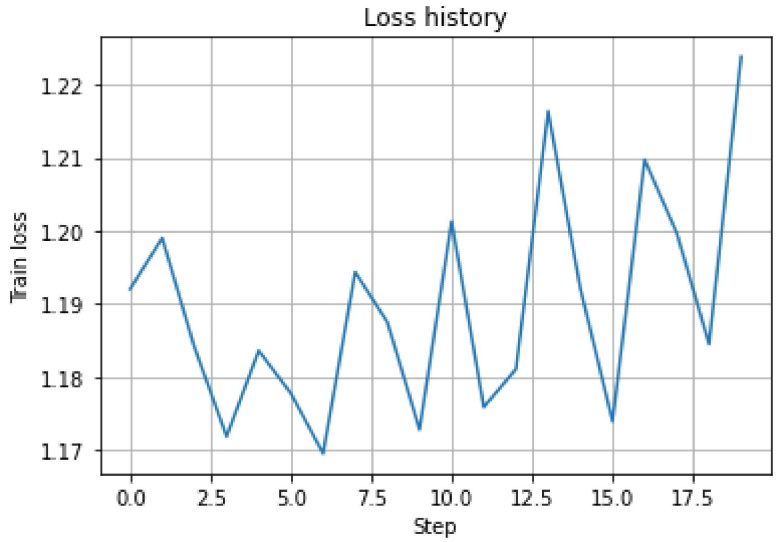
```
import matplotlib.pyplot as plt
%pylab inline
plt.title('Loss history')
plt.grid(True)
plt.ylabel('Train loss')
plt.xlabel('Step')
plt.plot(loss_1['loss']);
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [125]:

```
import matplotlib.pyplot as plt
%pylab inline
plt.title('Loss history')
plt.grid(True)
plt.ylabel('Train loss')
plt.xlabel('Step')
plt.plot(loss_2['loss']);
```

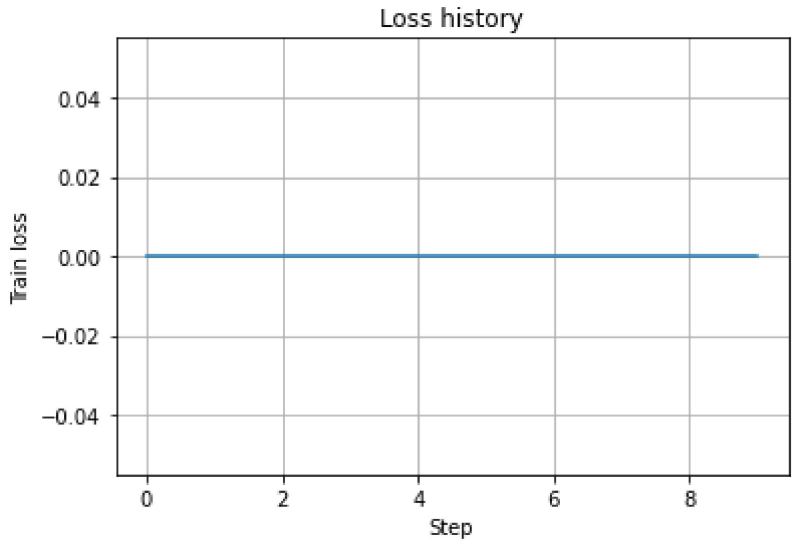
```
Populating the interactive namespace from numpy and matplotlib
```



In [126]:

```
import matplotlib.pyplot as plt
%pylab inline
plt.title('Loss history')
plt.grid(True)
plt.ylabel('Train loss')
plt.xlabel('Step')
plt.plot(loss_3['loss']);
```

```
Populating the interactive namespace from numpy and matplotlib
```



In [127]:

```
import matplotlib.pyplot as plt
%pylab inline
plt.title('Loss history')
plt.grid(True)
plt.ylabel('Train loss')
plt.xlabel('Step')
plt.plot(loss_4['loss']);
```

```
Populating the interactive namespace from numpy and matplotlib
```