



## Deep Learning School

### ✓ Глубокое обучение. Часть 2

#### Домашнее задание по теме "Механизм внимания"

Это домашнее задание проходит в формате peer-review. Это означает, что его будут проверять ваши однокурсники. Поэтому пишите разборчивый код, добавляйте комментарии и пишите выводы после проделанной работы.

В этом задании вы будете решать задачу классификации математических задач по темам (многоклассовая классификация) с помощью Transformer.

В качестве датасета возьмем датасет математических задач по разным темам. Нам необходим следующий файл:

[Файл с классами](#)

**Hint:** не перезаписывайте модели, которые вы получите на каждом из этапов этого дз. Они ещё понадобятся.

```
import re
import copy
import torch
import pandas as pd
import numpy as np
import torch.nn as nn
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

from tqdm.notebook import tqdm
from typing import Union, List
from functools import partial
from transformers import AutoTokenizer, AutoModel
from torch.utils.data import Dataset, DataLoader
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

### ✓ Задание 1 (2 балла)

Напишите кастомный класс для модели трансформера для задачи классификации, использующей в качестве backbone какую-то из моделей huggingface.

Т.е. конструктор класса должен принимать на вход название модели и подгружать её из huggingface, а затем использовать в качестве backbone (достаточно возможности использовать в качестве backbone те модели, которые упомянуты в последующих пунктах)

```
### This is just an interface example. You may change it if you want.
class TransformerClassificationModel(nn.Module):
    def __init__(self,
                 base_transformer_model: Union[str, nn.Module],
                 dropout_rate=0.1,
                 num_labels=4
                 ):
        super().__init__()
        self.backbone = AutoModel.from_pretrained(pretrained_model_name_or_path=base_transformer_model)
        self.dropout = nn.Dropout(dropout_rate)
        self.classifier = nn.Linear(self.backbone.config.hidden_size, num_labels)

    def forward(self, inputs, attention_mask):
        # YOUR CODE: propagate inputs through the model. Return dict with logits
```

```

outputs = self.backbone(input_ids=inputs, attention_mask=attention_mask)
pooled_output = self.dropout(outputs.pooler_output) # Dropout для регуляризации
logits = self.classifier(pooled_output)
return logits

```

Чтобы изменить содержимое ячейки, дважды нажмите на нее (или выберите "Ввод")

```

model_1 = TransformerClassificationModel(base_transformer_model='cointegrated/rubert-tiny2')
model_1.classifier

```

↗ /usr/local/lib/python3.11/dist-packages/huggingface\_hub/utils/\_auth.py:94: UserWarning:  
The secret `HF\_TOKEN` does not exist in your Colab secrets.  
To authenticate with the Hugging Face Hub, create a token in your settings tab (<https://huggingface.co/settings/tokens>), set it as :  
You will be able to reuse this secret in all of your notebooks.  
Please note that authentication is recommended but still optional to access public models or datasets.

```

warnings.warn(
config.json: 100% 693/693 [00:00<00:00, 12.1kB/s]
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For better p
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back
model.safetensors: 100% 118M/118M [00:01<00:00, 116MB/s]
Linear(in_features=312, out_features=4, bias=True)

```

```

# проверка
sum_res = 0
for name, param in model_1.named_parameters():
    if "backbone" in name and param.requires_grad: # Проверяем только параметры backbone
        sum_res += 1
sum_res

```

↗ 55

## ✓ Задание 2 (1 балл)

Напишите функцию заморозки backbone у модели (если необходимо, возвращайте из функции модель)

```

def freeze_backbone_function(model: TransformerClassificationModel):
    model_freeze = copy.deepcopy(model)
    for param in model_freeze.backbone.parameters():
        param.requires_grad = False
    return model_freeze

```

```

model_1_freze = freeze_backbone_function(model_1)

```

```

# проверка
sum_res = 0
for name, param in model_1_freze.named_parameters():
    if "backbone" in name and param.requires_grad: # Проверяем только параметры backbone
        sum_res += 1
sum_res

```

↗ 0

## ✓ Задание 3 (2 балла)

Напишите функцию, которая будет использована для тренировки (дообучения) трансформера (TransformerClassificationModel).  
Функция должна поддерживать обучение с замороженным и размороженным backbone.

**функция тренировки ниже**

## ✓ скачивание датасета

```
import requests
```

```
file_url = 'https://docs.google.com/spreadsheets/d/13YIbphbWc62sfa-bCh8MLQWKizaXbQK9/export?format=xlsx'
r = requests.get(file_url, allow_redirects=True)
open('data.xlsx', 'wb').write(r.content)
data = pd.read_excel('./data.xlsx')
data.shape
```

```
(5273, 3)
```

```
data.head()
```

	Unnamed: 0	problem_text	topic
0	0	To prove that the sum of the numbers of the ex...	number_theory
1	1	( b) Will the statement of the previous challe...	number_theory
2	2	The quadratic three-member graph with the coef...	polynoms
3	3	Can you draw on the surface of Rubik's cube a ...	combinatorics
4	4	Dima, who came from Vrunlandia, said that ther...	graphs

## ▼ препроцессинг

```
import string
import nltk
nltk.download('stopwords')
nltk.download('punkt_tab')
```

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
stopWords = set(stopwords.words('english'))
nltk.download('wordnet')
wnl = nltk.WordNetLemmatizer()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
def preproc_nltk(text):
    text = str(text)
    text = re.sub(r'[0-9]+', '', text.lower())
    prcessed_text = text.lower().translate(
        str.maketrans('', '', string.punctuation))
    return ' '.join([wnl.lemmatize(word) for word in word_tokenize(prcessed_text.lower()) if word not in stopWords])
data['process_text'] = data['problem_text'].apply(preproc_nltk)
topic_2_idx = {value: key for key, value in enumerate(data.topic.unique())}
data['label'] = data['topic'].apply(lambda x: topic_2_idx[x])
```

```
tokenizer = AutoTokenizer.from_pretrained('cointegrated/rubert-tiny2')
data['text_tok_1'] = data['process_text'].apply(lambda x: tokenizer(str(x)).input_ids)
data['len_tok_1'] = data['text_tok_1'].apply(lambda x: len(x))
```

```
tokenizer = AutoTokenizer.from_pretrained('tbs17/MathBert')
data['text_tok_2'] = data['process_text'].apply(lambda x: tokenizer(str(x)).input_ids)
data['len_tok_2'] = data['text_tok_1'].apply(lambda x: len(x))
```

```
tokenizer_config.json: 100% 401/401 [00:00<00:00, 23.8kB/s]
vocab.txt: 100% 1.08M/1.08M [00:00<00:00, 7.77MB/s]
tokenizer.json: 100% 1.74M/1.74M [00:00<00:00, 8.58MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 13.1kB/s]
tokenizer_config.json: 100% 28.0/28.0 [00:00<00:00, 2.70kB/s]
config.json: 100% 569/569 [00:00<00:00, 65.3kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 3.49MB/s]
tokenizer.json: 100% 466k/466k [00:00<00:00, 41.5MB/s]
```

```
data.head()
```

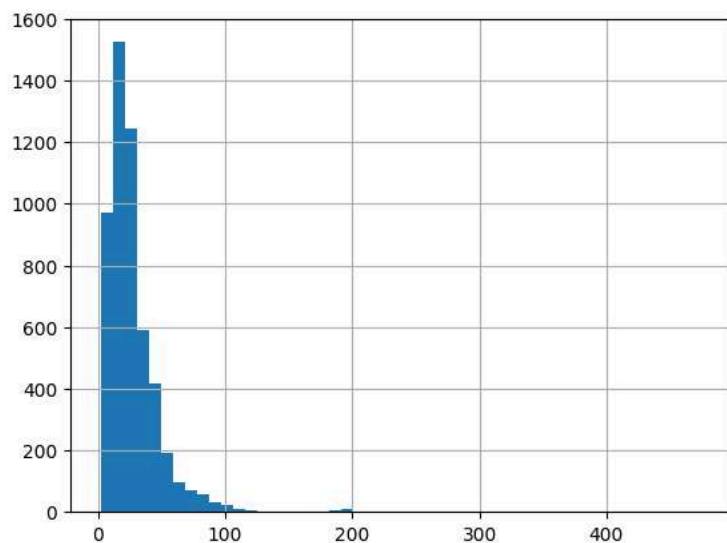


	Unnamed: 0	problem_text	topic	process_text	label	text_tok_1	len_tok_1	text_tok_2	len_tok_2
0	0	To prove that the sum of the numbers of the ex...	number_theory	prove sum number exact square equal	0	[2, 11544, 9009, 1503, 14568, 3511, 10553, 3]	8	[101, 6011, 7680, 2193, 6635, 2675, 5020, 102]	8
1	1	( b) Will the statement of the previous challe...	number_theory	b statement previous challenge remain true pet...	0	[2, 69, 10670, 4244, 11591, 7937, 6607, 10501,...]	17	[101, 1038, 4861, 3025, 4119, 3961, 2995, 9004...	17
2	2	The quadratic three-member graph with the coef...	polynoms	quadratic threemember graph coefficient two po...	1	[2, 3184, 25885, 1267, 1177, 18577, 1955, 2360...	24	[101, 17718, 23671, 2093, 4168, 21784, 10629, ...]	24
3	3	Can you draw on the surface of Rubik's cube a ...	combinatorics	draw surface rubiks cube closed path pass squa...	2	[2, 8128, 3919, 2643, 18229, 533, 1093, 1207. ...]	17	[101, 4009, 3302, 14548, 5480, 2015, 14291, 27...	17

```
data['len_tok_1'].hist(bins=50)
```

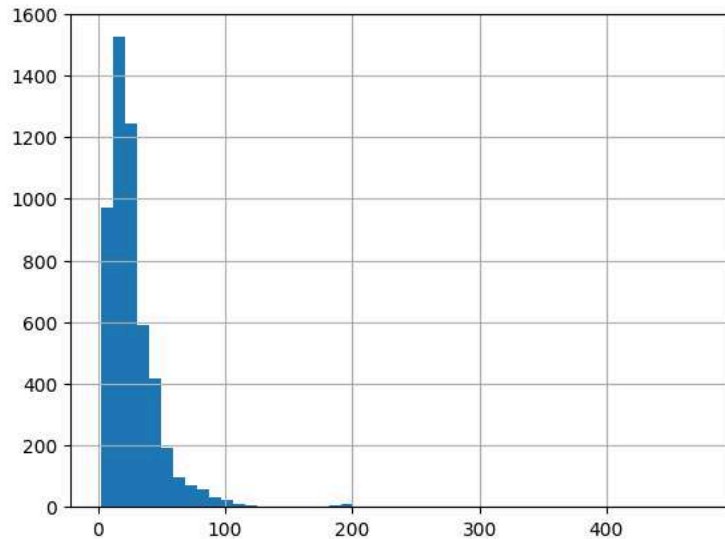


<Axes: >



```
data['len_tok_2'].hist(bins=50)
```

&lt;Axes: &gt;



в токенизаторе можем брать длину = 128

```
max_len = 128
```

```
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)
```

#### ▼ dataloader

```
class TextDataset(Dataset):
    def __init__(self, sentences):
        self.data = sentences

    def __getitem__(self, idx: int):
        processed_text = self.data.iloc[idx]['process_text']
        train_sample = {
            "text": processed_text,
            "label": int(self.data.iloc[idx]['label'])
        }

        return train_sample

    def __len__(self) -> int:
        return len(self.data)

def collate_fn_with_padding(batch, tokenizer, max_len=max_len):
    texts = [value['text'] for value in batch]
    new_batch = tokenizer(texts, max_length=max_len, truncation=True, padding='longest', return_tensors='pt')
    new_batch['labels'] = torch.LongTensor([value['label'] for value in batch])

    return new_batch

train_dataset = TextDataset(train_data)
eval_dataset = TextDataset(test_data)
```

#### ▼ тренировка

```
def train_transformer(
    base_transformer_model,
    train_dataloader,
    val_dataloader,
    optimizer_class=torch.optim.AdamW,
    learning_rate=5e-5,
```

```

epochs=3,
freeze_backbone=True,
device="cuda" if torch.cuda.is_available() else "cpu",
scheduler=False
):
    """Дообучаем трансформер.

Args:
    base_transformer_model: Модель трансформера для дообучения.
    train_dataloader: DataLoader для обучающей выборки.
    val_dataloader: DataLoader для валидационной выборки.
    optimizer_class: Класс оптимизатора (по умолчанию AdamW).
    learning_rate: Скорость обучения (по умолчанию 5e-5).
    epochs: Количество эпох обучения (по умолчанию 3).
    freeze_backbone: Замораживать ли backbone (по умолчанию True).
    device: Устройство для обучения ("cuda" или "cpu").
    scheduler: шедулер

Returns:
    Дообученная модель, метрики
    """

print(f'device = {device}')
metrics = {}
model = TransformerClassificationModel(
    base_transformer_model=base_transformer_model,
    num_labels=data.topic.nunique()
)
if freeze_backbone:
    model = freeze_backbone_function(model)
model.to(device)
optimizer = optimizer_class(model.parameters(), lr=learning_rate)
if scheduler:
    scheduler = torch.optim.lr_scheduler.ExponentialLR(optimizer, gamma=0.9) # ExponentialLR / StepLR

losses_train, losses_val = [], []
f1_train, f1_val = [], []

# обучение
for epoch in tqdm(range(epochs), leave=False):
    epoch_losses_train, epoch_losses_val = [], []
    predictions, target = [], []

    model.train()
    for batch in tqdm(train_dataloader, leave=False):
        inputs, attention_mask, labels = batch['input_ids'], batch['attention_mask'], batch['labels']
        inputs, attention_mask, labels = inputs.to(device), attention_mask.to(device), labels.to(device)

        optimizer.zero_grad()
        logits = model(inputs, attention_mask)
        loss = torch.nn.CrossEntropyLoss()(logits, labels)
        loss.backward()
        optimizer.step()

        epoch_losses_train.append(loss)
        predictions.append(logits.argmax(dim=1))
        target.append(labels)
    predictions = torch.cat(predictions)
    target = torch.cat(target)
    epoch_f1_train = f1_score(target.detach().cpu(), predictions.detach().cpu(), average='macro')
    f1_train.append(epoch_f1_train)
    epoch_loss_train = sum(epoch_losses_train) / len(epoch_losses_train)
    losses_train.append(epoch_loss_train)
    if scheduler:
        scheduler.step()

# валидация
predictions, target = [], []
model.eval()
with torch.no_grad():
    for batch in tqdm(eval_dataloader, leave=False):
        inputs, attention_mask, labels = batch['input_ids'], batch['attention_mask'], batch['labels']
        inputs, attention_mask, labels = inputs.to(device), attention_mask.to(device), labels.to(device)
        logits = model(inputs, attention_mask)
        loss = torch.nn.CrossEntropyLoss()(logits, labels)

        epoch_losses_val.append(loss)
        predictions.append(logits.argmax(dim=1))
        target.append(labels)
    predictions = torch.cat(predictions)
    target = torch.cat(target)
    epoch_f1_val = f1_score(target.detach().cpu(), predictions.detach().cpu(), average='macro')

```

```

    fl_val.append(epoch_fl_val)
    epoch_loss_val = sum(epoch_losses_val) / len(epoch_losses_val)
    losses_val.append(epoch_loss_val)
    print(f'эпоха {epoch}, fl_train = {epoch_fl_train}, fl_val = {epoch_fl_val}')
    print(f'эпоха {epoch}, loss_train = {epoch_loss_train}, loss_val = {epoch_loss_val}')
    print(f'lr = {optimizer.param_groups[0]["lr"]}')
    print()

    metrics['losses_train'] = losses_train
    metrics['losses_val'] = losses_val
    metrics['fl_train'] = fl_train
    metrics['fl_val'] = fl_val
    return model, metrics

```

## ▼ графики

```

def plot(metrics):
    losses_train = [metric.detach().cpu().numpy() for metric in metrics['losses_train']]
    losses_val = [metric.detach().cpu().numpy() for metric in metrics['losses_val']]

    plt.plot(np.arange(len(losses_train)), losses_train, label='Train')
    plt.plot(np.arange(len(losses_val)), losses_val, label='Validation')
    plt.title('Лосс')
    plt.xlabel("эпоха")
    plt.grid()
    plt.show()

    plt.plot(np.arange(len(metrics['fl_train'])), metrics['fl_train'], label='Train')
    plt.plot(np.arange(len(metrics['fl_val'])), metrics['fl_val'], label='Validation')
    plt.title('F1')
    plt.xlabel("эпоха")
    plt.grid()
    plt.show()

```

## ▼ Задание 4 (1 балл)

Проверьте вашу функцию из предыдущего пункта, дообучив двумя способами *cointegrated/rubert-tiny2* из huggingface.

```

model_name = 'cointegrated/rubert-tiny2'
tokenizer = AutoTokenizer.from_pretrained(model_name)
collate_fn_with_padding = partial(collate_fn_with_padding, tokenizer=tokenizer)
batch_size = 128
train_dataloader = DataLoader(
    train_dataset, shuffle=True, collate_fn=collate_fn_with_padding, batch_size=batch_size)

eval_dataloader = DataLoader(
    eval_dataset, shuffle=False, collate_fn=collate_fn_with_padding, batch_size=batch_size)
print(len(train_dataloader), len(eval_dataloader))

list(train_dataloader)[0]['labels']

```

```

33 9
tensor([2, 0, 0, 0, 0, 2, 5, 0, 2, 0, 0, 1, 5, 0, 3, 0, 1, 0, 5, 0, 0, 5, 4, 0,
        0, 0, 2, 0, 0, 1, 6, 0, 6, 3, 2, 2, 1, 3, 3, 0, 3, 0, 0, 2, 2, 2, 0, 2,
        0, 1, 0, 0, 4, 3, 0, 0, 6, 0, 2, 0, 0, 2, 6, 0, 6, 0, 2, 6, 0, 0, 0, 1,
        3, 2, 2, 0, 0, 0, 0, 2, 0, 0, 0, 2, 6, 3, 0, 1, 3, 0, 6, 2, 6, 1, 0, 0,
        3, 4, 0, 0, 4, 6, 6, 0, 0, 4, 2, 0, 0, 2, 6, 0, 6, 0, 0, 0, 2, 4, 3, 0,
        2, 1, 0, 3, 4, 4, 0, 4])


```

## ▼ rubert\_tiny\_finetuned\_with\_freezed\_backbone

```

rubert_tiny_finetuned_with_freezed_backbone, metrics_1_1 = train_transformer(
    base_transformer_model=model_name,
    freeze_backbone=True,
    train_dataloader=train_dataloader,
    val_dataloader=eval_dataloader,
    epochs=20,
    learning_rate=5e-5,
    scheduler=True
)

```

 device = cuda

```
эпоха 0, f1_train = 0.12125748743953937, f1_val = 0.11346669692151931
эпоха 0, loss_train = 1.903627634048462, loss_val = 1.8943402767181396
lr = 4e-05
```

```
эпоха 1, f1_train = 0.1252060669017729, f1_val = 0.12313533742759439
эпоха 1, loss_train = 1.8852275609970093, loss_val = 1.878797173500061
lr = 3.2000000000000005e-05
```

```
эпоха 2, f1_train = 0.12573446963343596, f1_val = 0.12683726910585238
эпоха 2, loss_train = 1.8712801933288574, loss_val = 1.866904377937317
lr = 2.5600000000000006e-05
```

```
эпоха 3, f1_train = 0.12608499881371113, f1_val = 0.128160980311367
эпоха 3, loss_train = 1.8596609830856323, loss_val = 1.8575717210769653
lr = 2.0480000000000007e-05
```

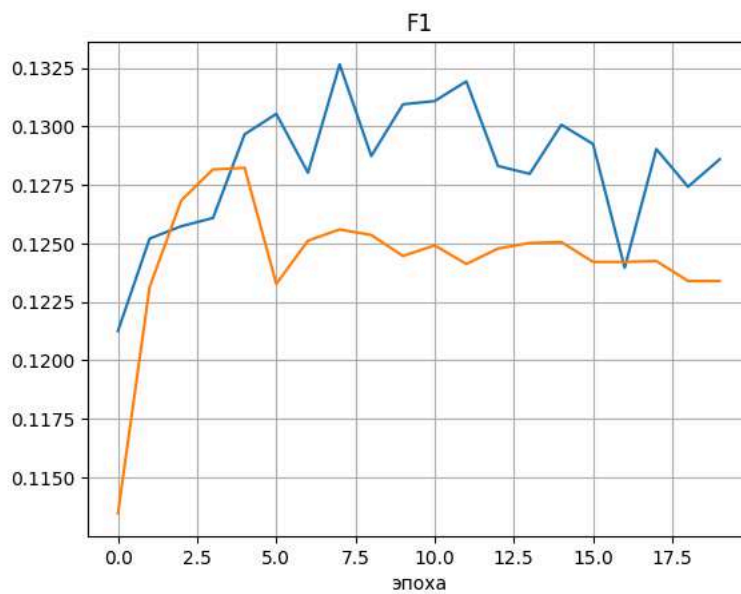
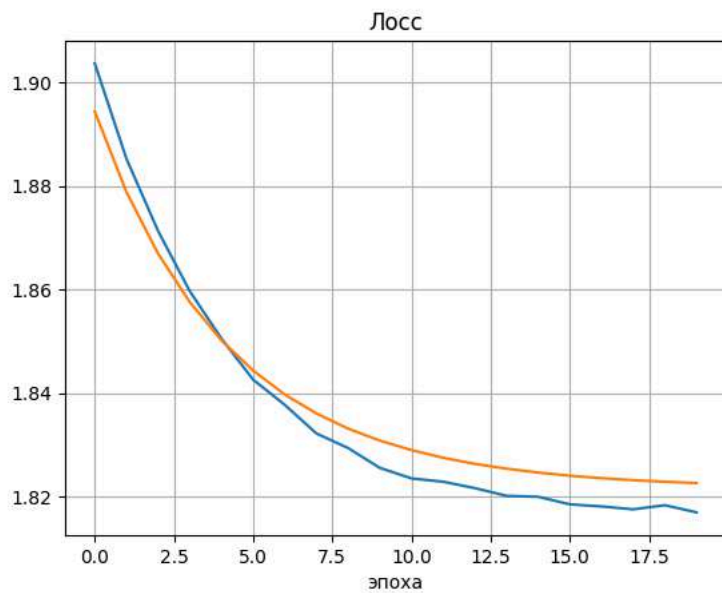
```
эпоха 4, f1_train = 0.12966277334283763, f1_val = 0.12822927643133122
эпоха 4, loss_train = 1.8505208492279053, loss_val = 1.8501697778701782
lr = 1.6384000000000008e-05
```

```
эпоха 5, f1_train = 0.1305346604774878, f1_val = 0.12326951764409909
эпоха 5, loss_train = 1.8426016569137573, loss_val = 1.8443557024002075
lr = 1.3107200000000007e-05
```

```
эпоха 6, f1_train = 0.12802150661887984, f1_val = 0.12511874925408759
эпоха 6, loss_train = 1.837729573249817, loss_val = 1.839719533920288
lr = 1.0485760000000006e-05
```


```
plot(metrics_1_1)
```





#### ▼ rubert\_tiny\_full\_finetuned

```
rubert_tiny_full_finetuned, metrics_1_2 = train_transformer(
    base_transformer_model=model_name,
    freeze_backbone=False,
    train_dataloader=train_dataloader,
    val_dataloader=eval_dataloader,
    epochs=20,
    learning_rate=5e-5, # 5e-5
    scheduler=True
)
```

 device = cuda

```
эпоха 0, f1_train = 0.13369511975172127, f1_val = 0.2323925489130467
эпоха 0, loss_train = 1.6017566919326782, loss_val = 1.4029275178909302
lr = 4.5e-05
```

```
эпоха 1, f1_train = 0.30925995106183907, f1_val = 0.3500783336399774
эпоха 1, loss_train = 1.2906755208969116, loss_val = 1.2121472358703613
lr = 4.05e-05
```

```
эпоха 2, f1_train = 0.38481522593670586, f1_val = 0.3726688736479996
эпоха 2, loss_train = 1.128773808479309, loss_val = 1.1313072443008423
lr = 3.6450000000000005e-05
```

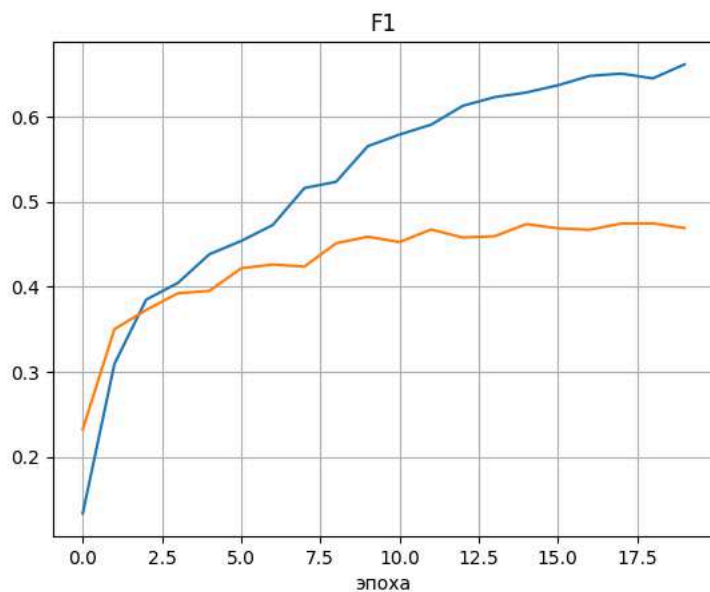
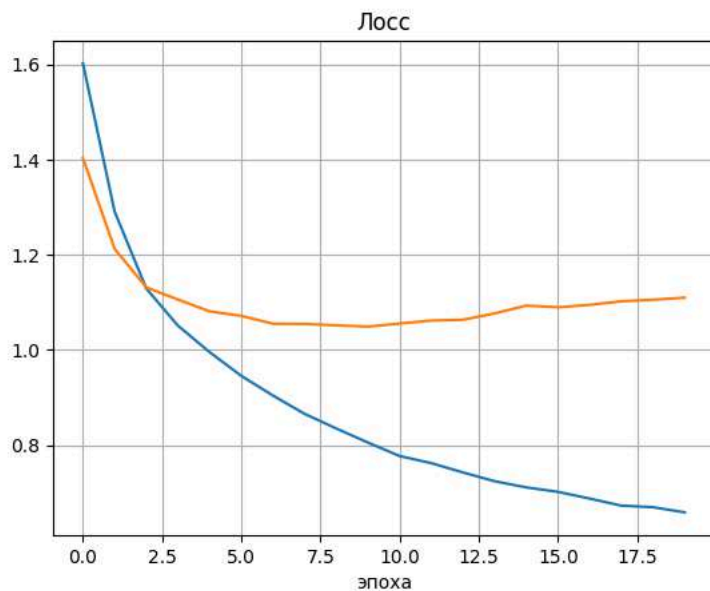
```
эпоха 3, f1_train = 0.40439998423053575, f1_val = 0.39226495150221846
эпоха 3, loss_train = 1.0508453845977783, loss_val = 1.1055879592895508
lr = 3.280500000000001e-05
```

```
эпоха 4, f1_train = 0.4382297995284513, f1_val = 0.3951986668547013
эпоха 4, loss_train = 0.9951379299163818, loss_val = 1.0808290243148804
lr = 2.952450000000001e-05
```

```
эпоха 5, f1_train = 0.45377291268789044, f1_val = 0.4216675961287553
эпоха 5, loss_train = 0.9453039169311523, loss_val = 1.071263313293457
lr = 2.657205000000001e-05
```

```
эпоха 6, f1_train = 0.4725588422839954, f1_val = 0.4261841672349728
эпоха 6, loss_train = 0.9038732647895813, loss_val = 1.05483877658844
lr = 2.391484500000001e-05
```

```
plot(metrics_1_2)
```



без заморозки слоев результат получается лучше

### ✓ Задание 5 (1 балл)

Обучите *tbs17/MathBert* (с замороженным backbone и без заморозки), проанализируйте результаты. Сравните скоры с первым заданием. Получилось лучше или нет? Почему?

### YOUR CODE IS HERE (probably, similar on the previous step)

```
model_name = 'tbs17/MathBert'
tokenizer = AutoTokenizer.from_pretrained(model_name)
collate_fn_with_padding = partial(collate_fn_with_padding, tokenizer=tokenizer)
batch_size = 128
train_dataloader = DataLoader(
    train_dataset, shuffle=True, collate_fn=collate_fn_with_padding, batch_size=batch_size)

eval_dataloader = DataLoader(
    eval_dataset, shuffle=False, collate_fn=collate_fn_with_padding, batch_size=batch_size)
len(train_dataloader), len(eval_dataloader)

list(train_dataloader)[0]['labels']
```




```
tensor([6, 0, 0, 1, 6, 3, 3, 0, 0, 0, 4, 0, 3, 0, 2, 2, 2, 2, 0, 0, 2, 0, 4, 6,
        4, 2, 0, 0, 5, 4, 6, 0, 6, 2, 3, 0, 2, 3, 0, 0, 0, 0, 0, 2, 2, 4, 6, 2,
```

```
2, 0, 4, 0, 0, 1, 3, 1, 6, 0, 0, 3, 5, 0, 0, 6, 0, 0, 3, 2, 2, 0, 0, 0,
1, 2, 0, 4, 1, 6, 0, 0, 0, 2, 0, 0, 1, 0, 4, 0, 0, 3, 1, 4, 0, 4, 1, 0,
0, 6, 0, 2, 0, 0, 5, 6, 2, 4, 2, 0, 2, 1, 0, 5, 6, 6, 0, 2, 0, 3, 0, 0,
3, 2, 0, 2, 2, 4, 0, 3])
```

## MathBert\_finetuned\_with\_freezed\_backbone

```
MathBert_finetuned_with_freezed_backbone, metrics_2_1 = train_transformer(
    base_transformer_model=model_name,
    freeze_backbone=True,
    train_dataloader=train_dataloader,
    val_dataloader=eval_dataloader,
    epochs=20,
    learning_rate=1e-5,
    scheduler=True
)
```

 device = cuda

```
эпоха 0, f1_train = 0.07935787973316485, f1_val = 0.05954778648568083
эпоха 0, loss_train = 2.001835823059082, loss_val = 1.9821518659591675
lr = 9e-06
```

```
эпоха 1, f1_train = 0.08114705679003935, f1_val = 0.06113144080472205
эпоха 1, loss_train = 1.9443285465240479, loss_val = 1.92418372631073
lr = 8.1e-06
```

```
эпоха 2, f1_train = 0.09797051663807517, f1_val = 0.07190081440698792
эпоха 2, loss_train = 1.893043875694275, loss_val = 1.87656831741333
lr = 7.2900000000000005e-06
```

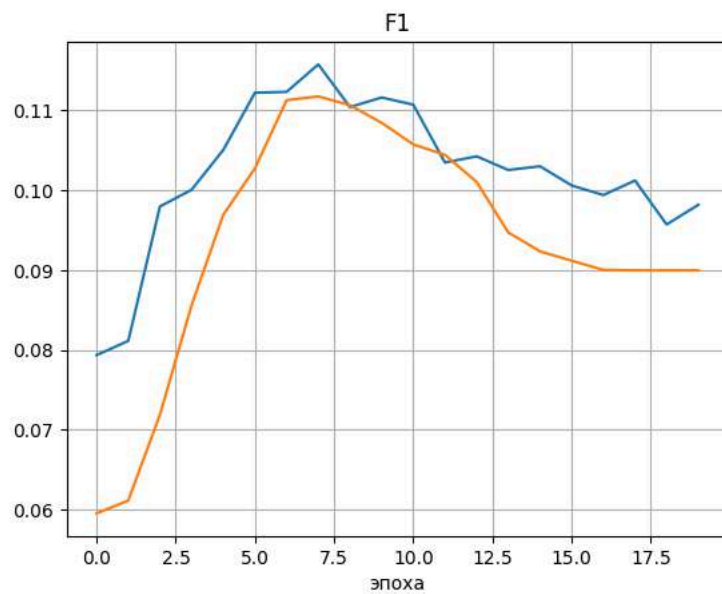
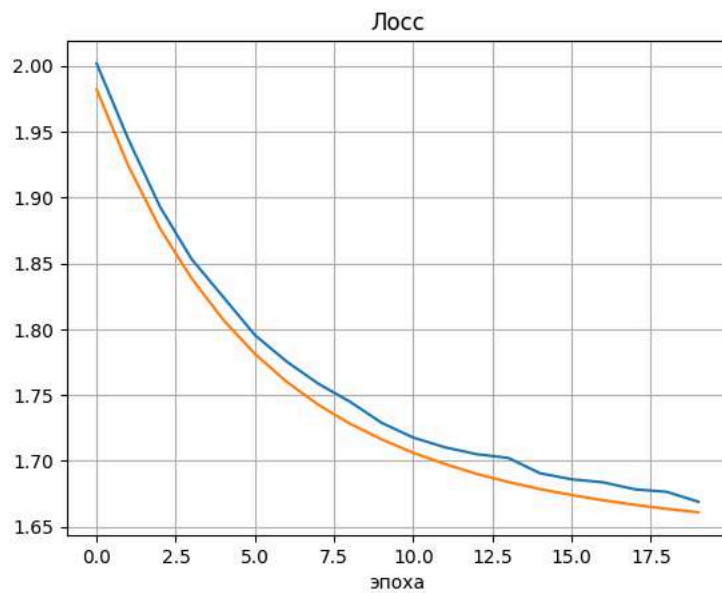
```
эпоха 3, f1_train = 0.1000691099162796, f1_val = 0.0856060813507622
эпоха 3, loss_train = 1.853242158897705, loss_val = 1.838584542274475
lr = 6.561e-06
```

```
эпоха 4, f1_train = 0.10505432806613789, f1_val = 0.09693435088023587
эпоха 4, loss_train = 1.824408888168335, loss_val = 1.80713951587677
lr = 5.904900000000001e-06
```

```
эпоха 5, f1_train = 0.11222235175174701, f1_val = 0.10275130920374807
эпоха 5, loss_train = 1.7954471111297607, loss_val = 1.7812801599502563
lr = 5.314410000000001e-06
```


```
эпоха 6, f1_train = 0.11232215482011812, f1_val = 0.11128332636140147
эпоха 6, loss_train = 1.7754337787628174, loss_val = 1.7600562572479248
lr = 4.782969000000001e-06
```

```
plot(metrics_2_1)
```



#### MathBert\_full\_finetuned

```
MathBert_full_finetuned, metrics_2_2 = train_transformer(
    base_transformer_model=model_name,
    freeze_backbone=False,
    train_dataloader=train_dataloader,
    val_dataloader=eval_dataloader,
    epochs=20,
    learning_rate=1e-5,
    scheduler=True
)
```

 device = cuda

```
эпоха 0, f1_train = 0.2151761964887533, f1_val = 0.3033423394117194
эпоха 0, loss_train = 1.445900797843933, loss_val = 1.2637038230895996
lr = 9e-06
```

```
эпоха 1, f1_train = 0.38009631746501954, f1_val = 0.39074853588953706
эпоха 1, loss_train = 1.1386117935180664, loss_val = 1.1272931098937988
lr = 8.1e-06
```

```
эпоха 2, f1_train = 0.42409443821212933, f1_val = 0.4267041671678567
эпоха 2, loss_train = 1.0249509811401367, loss_val = 1.0912660360336304
lr = 7.2900000000000005e-06
```

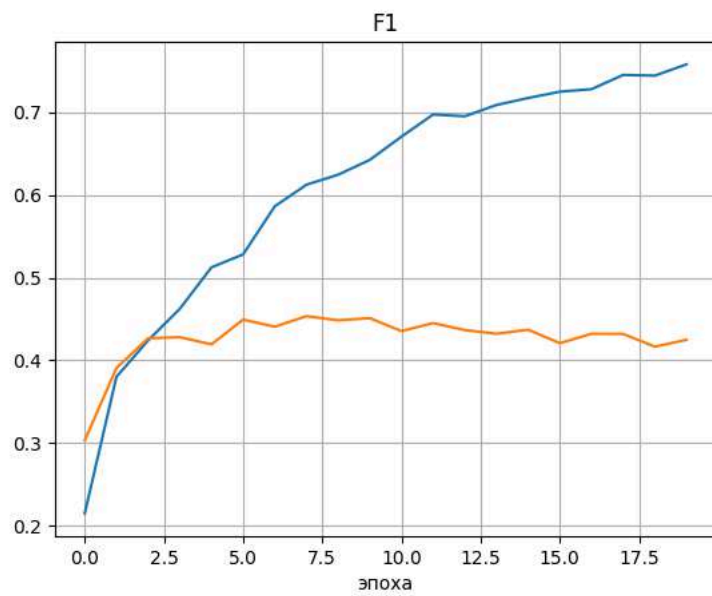
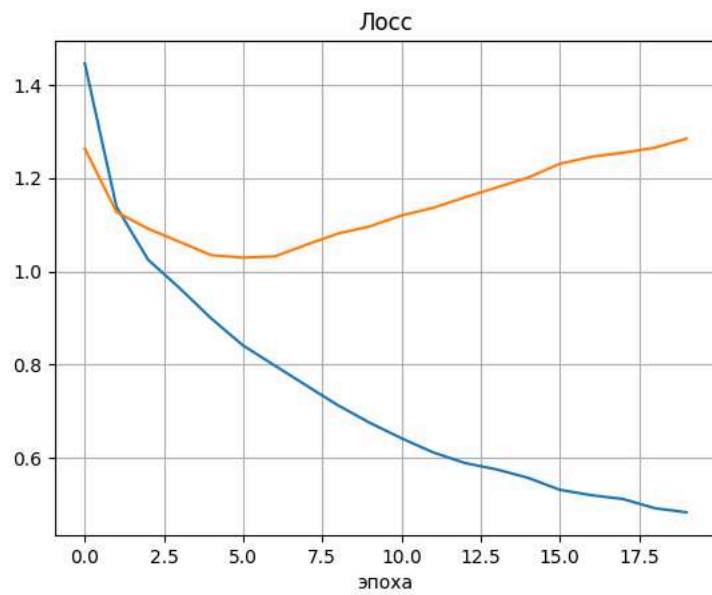
```
эпоха 3, f1_train = 0.46224329909605055, f1_val = 0.42800742974548206
эпоха 3, loss_train = 0.9636439085006714, loss_val = 1.0629396438598633
lr = 6.561e-06
```

```
эпоха 4, f1_train = 0.5123576273878895, f1_val = 0.4193311879329878
эпоха 4, loss_train = 0.8985554575920105, loss_val = 1.0345178842544556
lr = 5.904900000000001e-06
```

```
эпоха 5, f1_train = 0.5280143651173196, f1_val = 0.4492891085570126
эпоха 5, loss_train = 0.8408647775650024, loss_val = 1.029391884803772
lr = 5.314410000000001e-06
```

```
эпоха 6, f1_train = 0.5861686643976213, f1_val = 0.44062033879350126
эпоха 6, loss_train = 0.7979336977005005, loss_val = 1.0319782495498657
lr = 4.782969000000001e-06
```

```
plot(metrics_2_2)
```



Вывод в конце

✓ Задание 6 (1 балл)

Напишите функцию для отрисовки карт внимания первого слоя для моделей из задания



```
def draw_first_layer_attention_maps(attention_head_ids: List[int],
                                    text: str,
                                    model: TransformerClassificationModel,
                                    ):
    """
    Визуализирует карты внимания для указанных голов внимания первого слоя модели.

    Args:
        attention_head_ids: Список индексов голов внимания для визуализации
        text: Текст для анализа
        model: Модель TransformerClassificationModel
        model_name: Имя модели для токенизатора (если не указано, берется из model.backbone)
    """

    model.to('cpu')
    # Получаем токенизатор
    model_name = model.backbone.name_or_path
    tokenizer = AutoTokenizer.from_pretrained(model_name)

    # Токенизируем текст
    inputs = tokenizer(text, return_tensors="pt")
    input_ids = inputs["input_ids"]
    attention_mask = inputs["attention_mask"]

    # Получаем выходы модели
    with torch.no_grad():
        outputs = model.backbone(**inputs, output_attentions=True)
        output_label = model(inputs["input_ids"], inputs["attention_mask"]).argmax(dim=1)
        print(f'output_label = {output_label}')

    # Берем внимания первого слоя (индекс 0)
    # attentions имеет размерность [layers, batch, heads, seq_len, seq_len]
    attentions = outputs.attentions
    # print(len(attentions)) # layers
    # print(attentions[0].shape) # batch, heads, seq_len, seq_len
    first_layer_attention = attentions[0][0] # [heads, seq_len, seq_len]

    # Получаем токены для подписей
    tokens = tokenizer.convert_ids_to_tokens(input_ids[0])

    n_heads = len(attention_head_ids)
    n_rows = 4
    n_cols = int(n_heads / n_rows)

    fig = plt.figure(figsize=(12 * n_cols, 12 * n_rows))

    for i in range(n_heads):

        ax = fig.add_subplot(n_rows, n_cols, i+1)
        ax.set_title(f"Attention Head {i}")
        _attention = first_layer_attention.squeeze(0)[i].cpu().detach().numpy()

        cax = ax.matshow(_attention, cmap='viridis')

        ax.tick_params(labelsize=12)
        ticks = tokens

        ax.set_xticks(np.arange(len(ticks)))
        ax.set_yticks(np.arange(len(ticks)))

        ax.set_xticklabels(ticks,
                           rotation=45)
        ax.set_yticklabels(ticks)

        ax.xaxis.set_major_formatter(ticker.FixedFormatter(ticks))
        ax.yaxis.set_major_formatter(ticker.FixedFormatter(ticks))
```

## ✓ Задание 7 (1 балл)

Проведите инференс для всех моделей **ДО ДООБУЧЕНИЯ** на 2-3 текстах из датасета. Посмотрите на головы Attention первого слоя в каждой модели на выбранных текстах (отрисуйте их отдельно).

Попробуйте их проинтерпретировать. Какие связи улавливают карты внимания? (если в модели много голов Attention, то проинтерпретируйте наиболее интересные)

```

model_1_1 = TransformerClassificationModel(
    base_transformer_model='cointegrated/rubert-tiny2',
    num_labels=data.topic.nunique()
)
model_1_2 = freeze_backbone_function(model_1_1)

model_2_1 = TransformerClassificationModel(
    base_transformer_model='tbs17/MathBert',
    num_labels=data.topic.nunique()
)
model_2_2 = freeze_backbone_function(model_2_1)

model_1_1.backbone.config.num_attention_heads, model_2_1.backbone.config.num_attention_heads

↗ (12, 12)

```

## ▼ ТЕКСТ 1

```
test_data.iloc[4]
```

```

↗

```

	4151
Unnamed: 0	4151
problem_text	A graph of the function $y = ax^2 + bx + c$ is sh...
topic	polynoms
process_text	graph function ax bx c shown coordinate plane ...
label	1
text_tok_1	[2, 23607, 5471, 68, 981, 69, 981, 70, 5741, 2...
len_tok_1	23
text_tok_2	[101, 10629, 3853, 22260, 1038, 2595, 1039, 34...
len_tok_2	23

dtype: object

```

text=test_data.iloc[4].process_text
text

```

```

↗ 'graph function ax bx c shown coordinate plane see figure coordinate plane plot graph function cx bx'

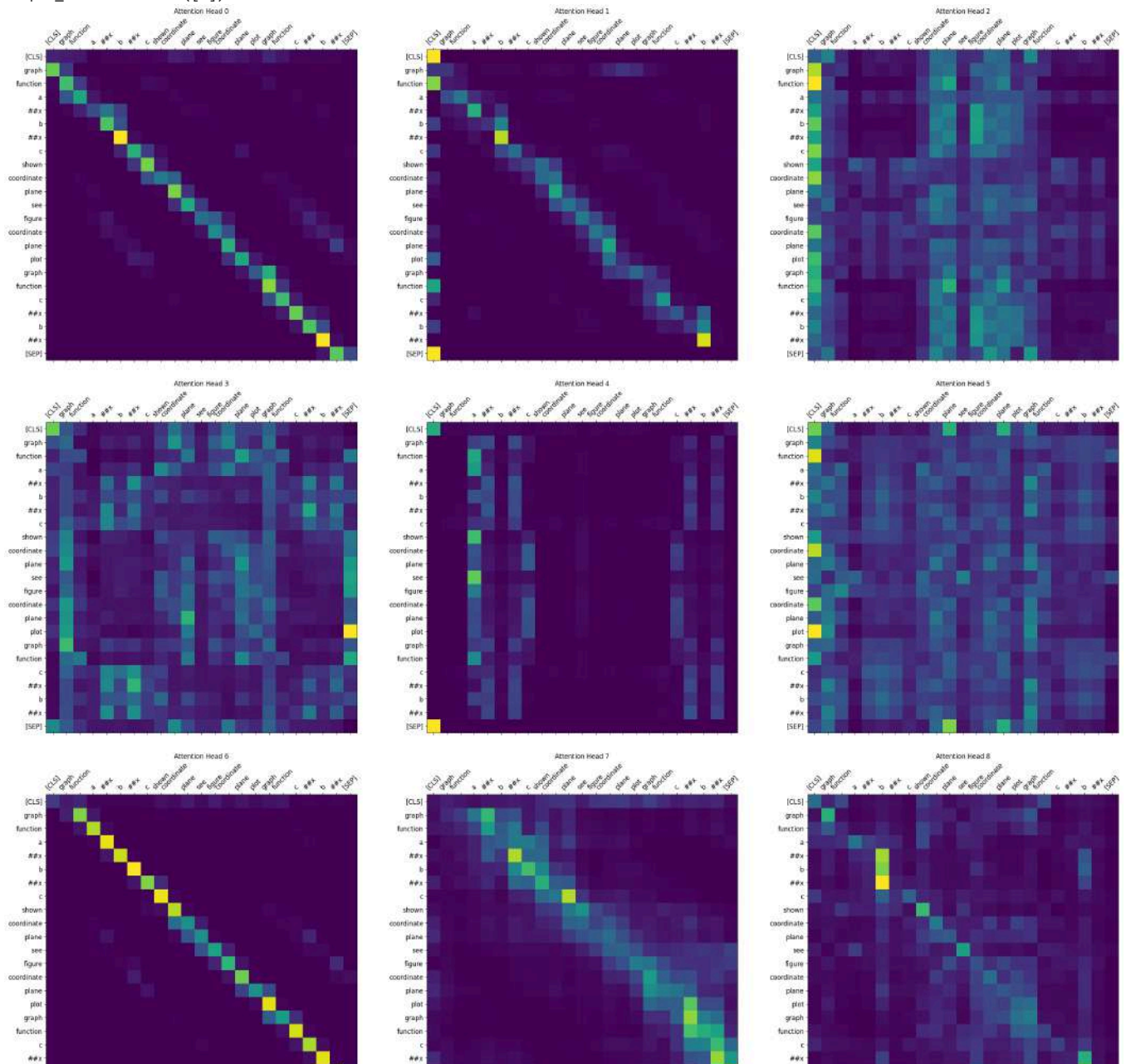
```

```

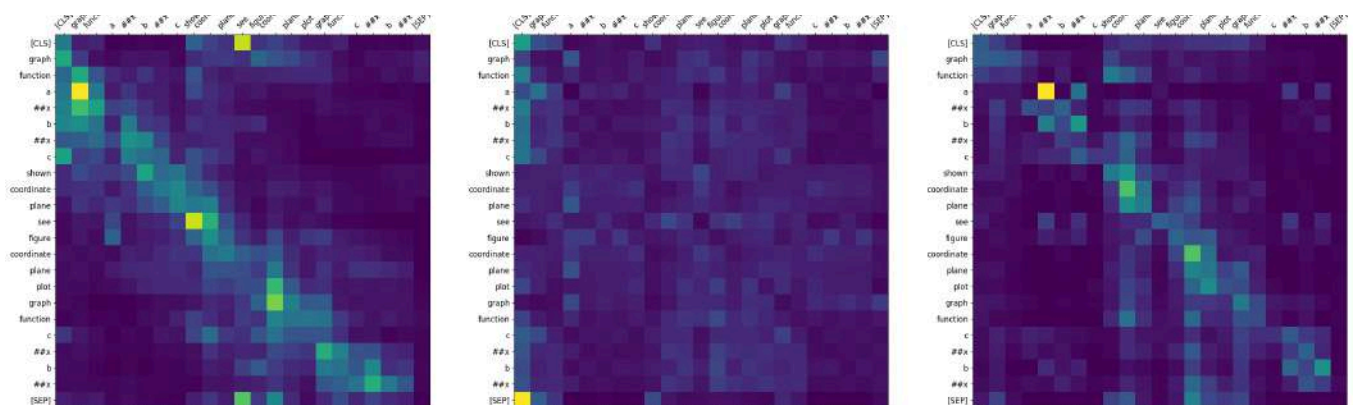
# подаем текст после препроцессинга
draw_first_layer_attention_maps(attention_head_ids=range(0, model_1_1.backbone.config.num_attention_heads),
                                text=text,
                                model=model_1_1)

```

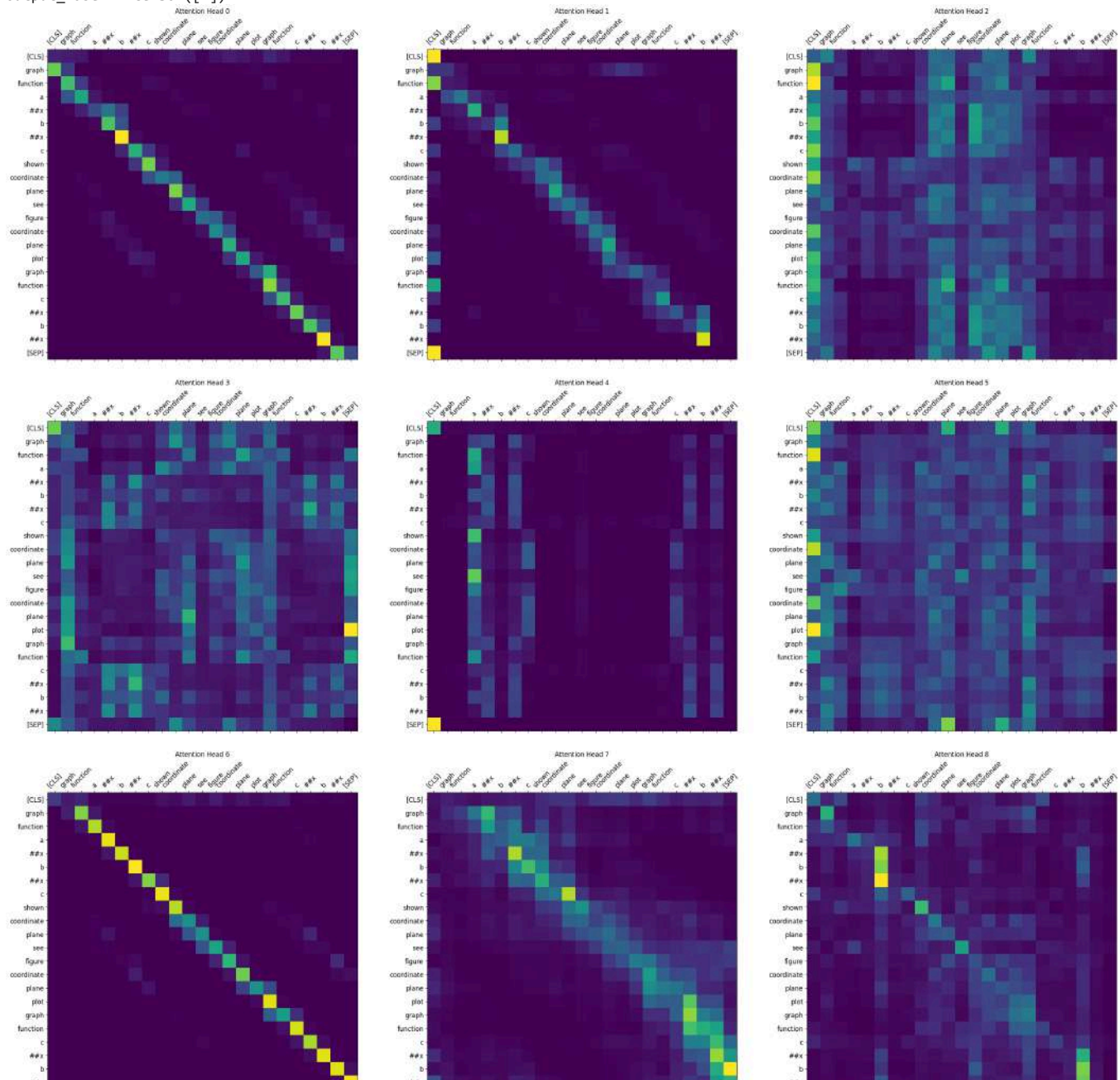
→ BertSdpaSelfAttention is used but `torch.nn.functional.scaled\_dot\_product\_attention` does not support non-absolute `position\_embedding`  
 output\_label = tensor([4])



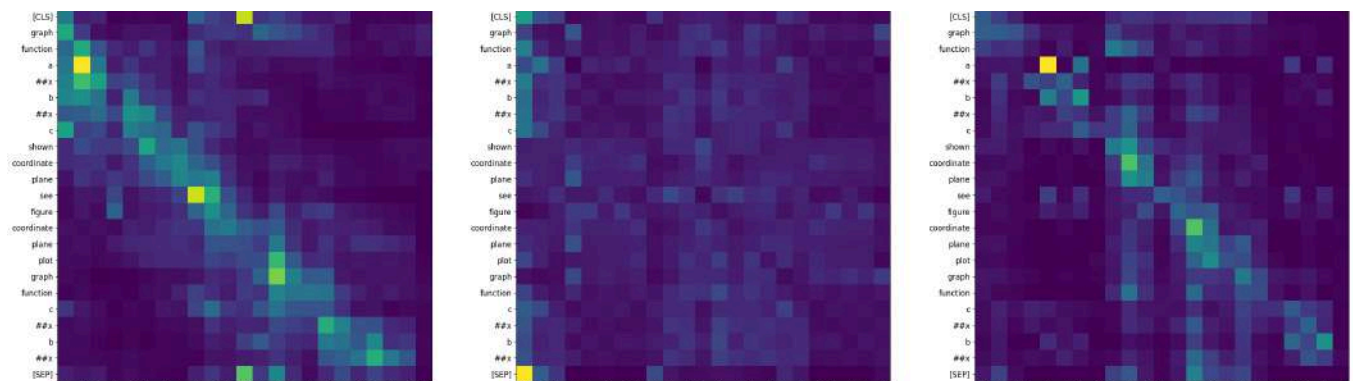
```
draw_first_layer_attention_maps(attention_head_ids=range(0, model_1_2.backbone.config.num_attention_heads),
                                text=text,
                                model=model_1_2)
```



```
output_label = tensor([2])
```

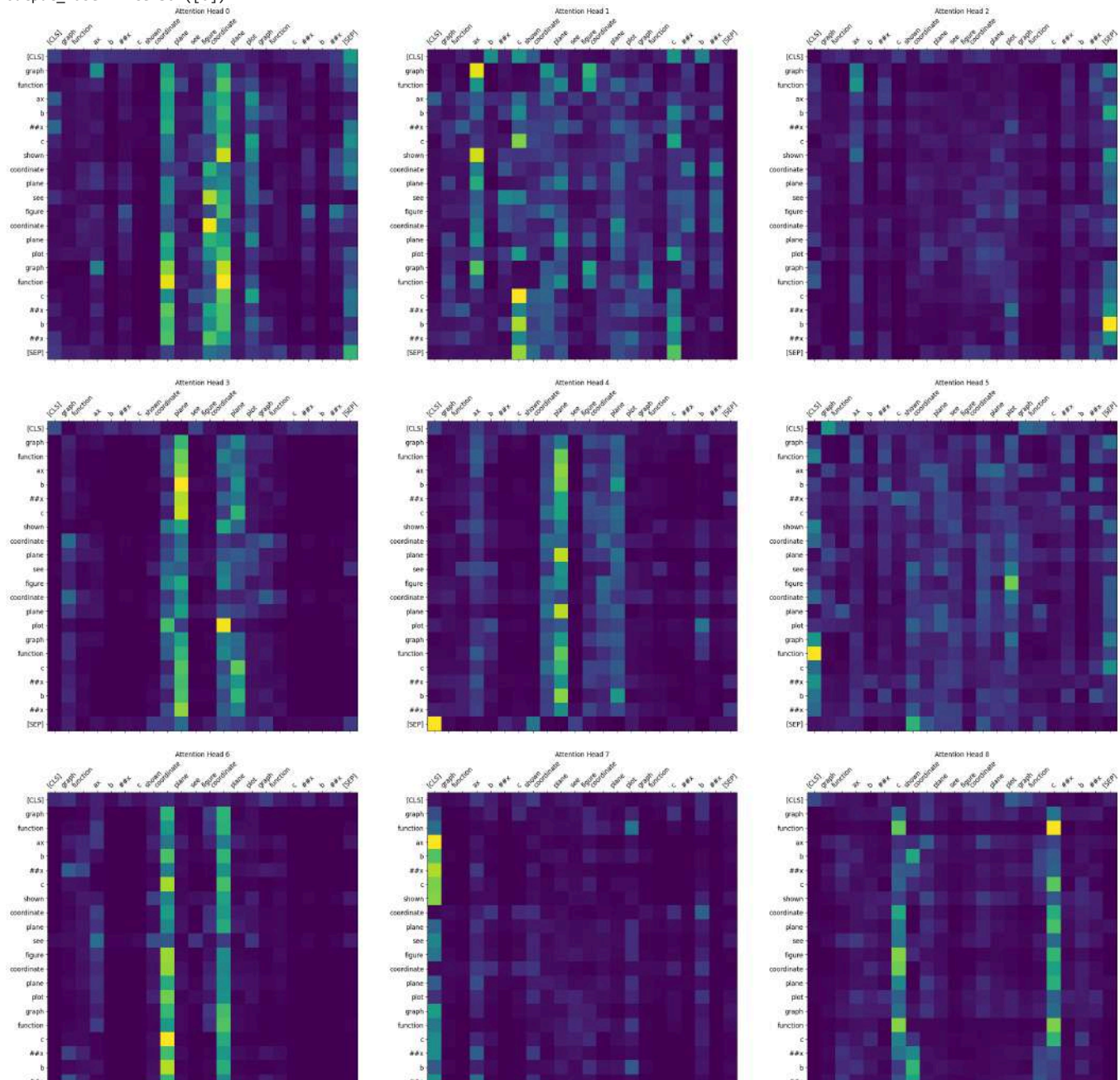


```
draw_first_layer_attention_maps(attention_head_ids=range(0, model_2_1.backbone.config.num_attention_heads),
                                text=text,
                                model=model_2_1)
```

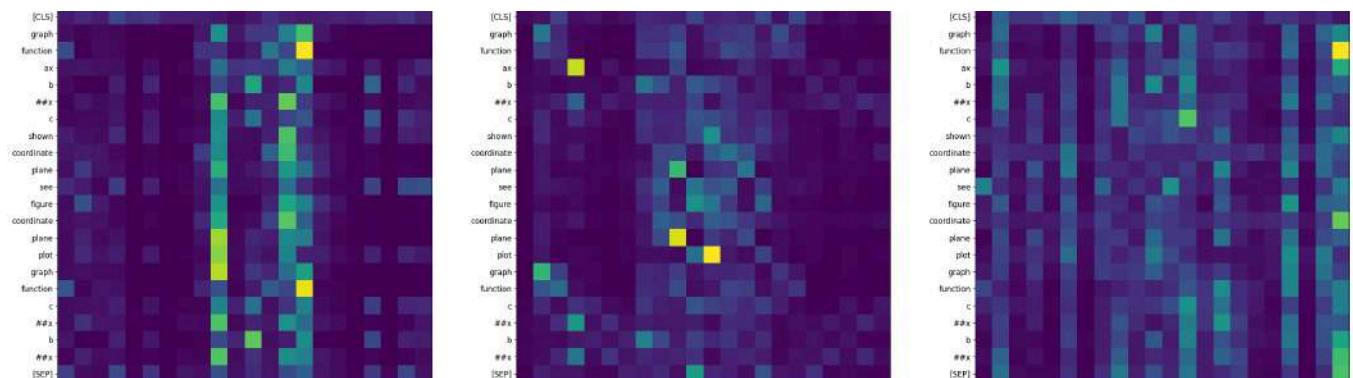




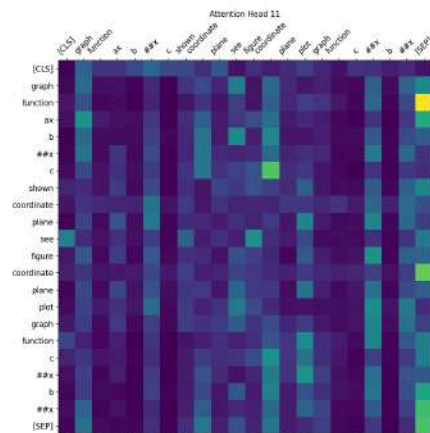
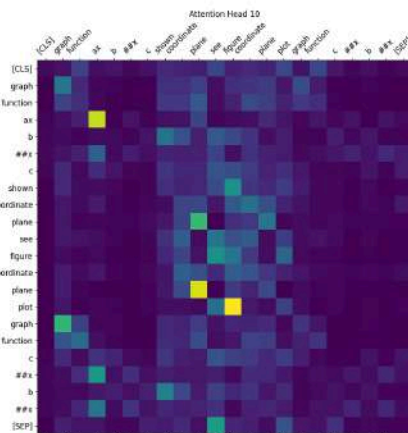
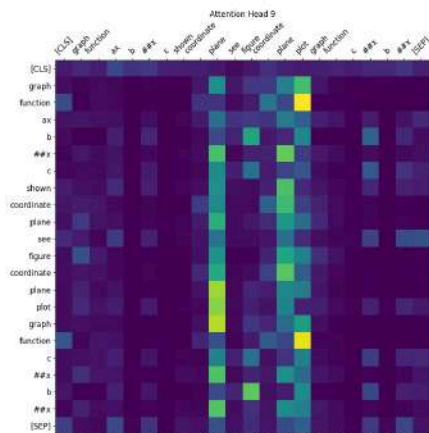
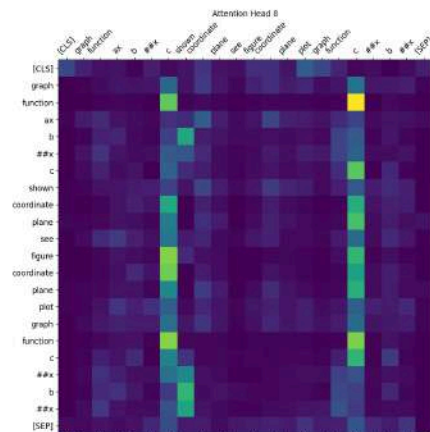
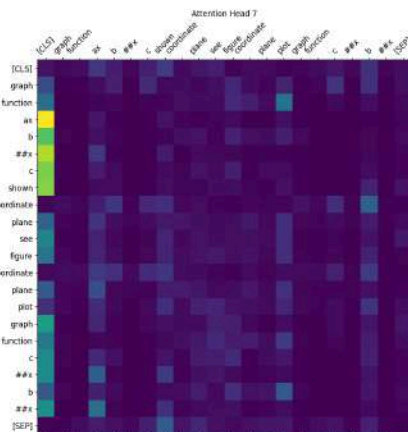
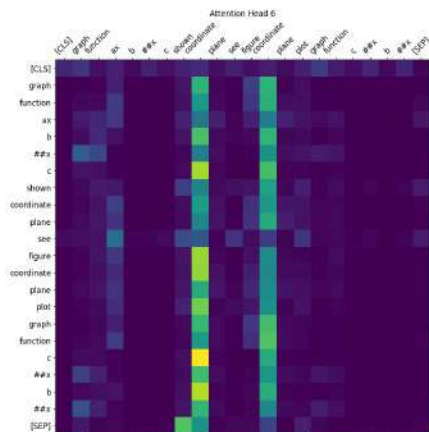
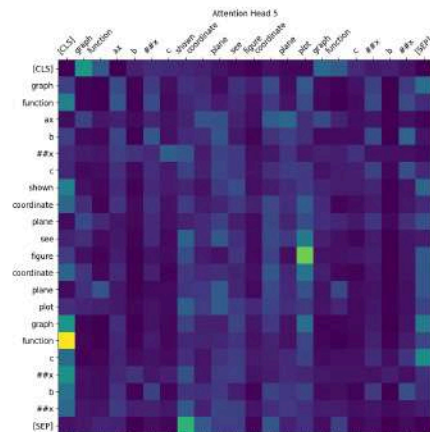
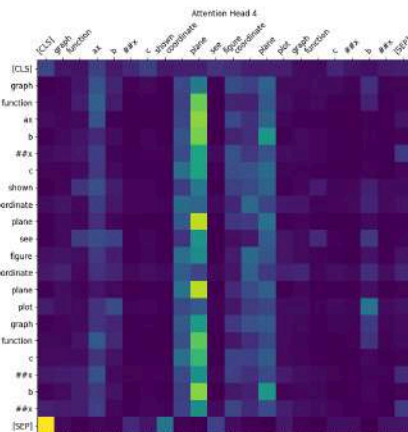
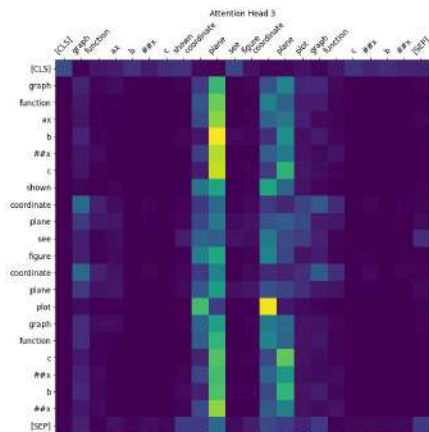
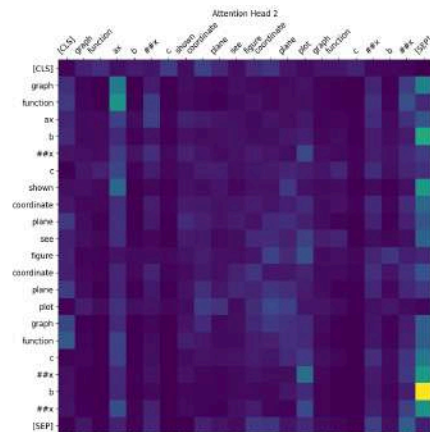
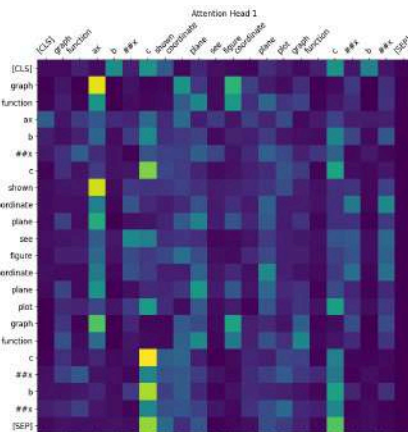
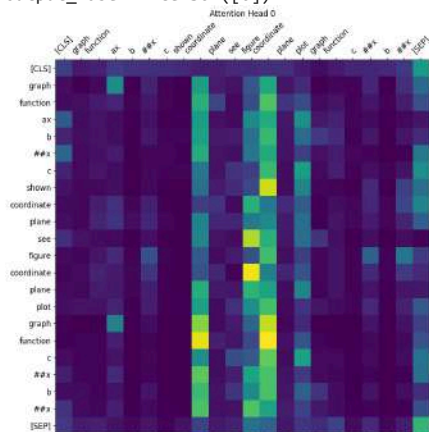
```
output_label = tensor([6])
```



```
draw_first_layer_attention_maps(attention_head_ids=range(0, model_2_2.backbone.config.num_attention_heads),
                                text=text,
                                model=model_2_2)
```




```
output_label = tensor([6])
```



## ▼ текст 2


```
test_data.iloc[5]
```



	2851
<b>Unnamed: 0</b>	2851
<b>problem_text</b>	Calculate in natural numbers the equation: x3 ...
<b>topic</b>	number_theory
<b>process_text</b>	calculate natural number equation x xy
<b>label</b>	0
<b>text_tok_1</b>	[2, 24896, 2081, 2606, 1503, 19042, 91, 91, 58...
<b>len_tok_1</b>	10
<b>text_tok_2</b>	[101, 18422, 3019, 2193, 8522, 1060, 1060, 210...
<b>len_tok_2</b>	10

**dtype:** object

```
text=test_data.iloc[5].process_text
text
```

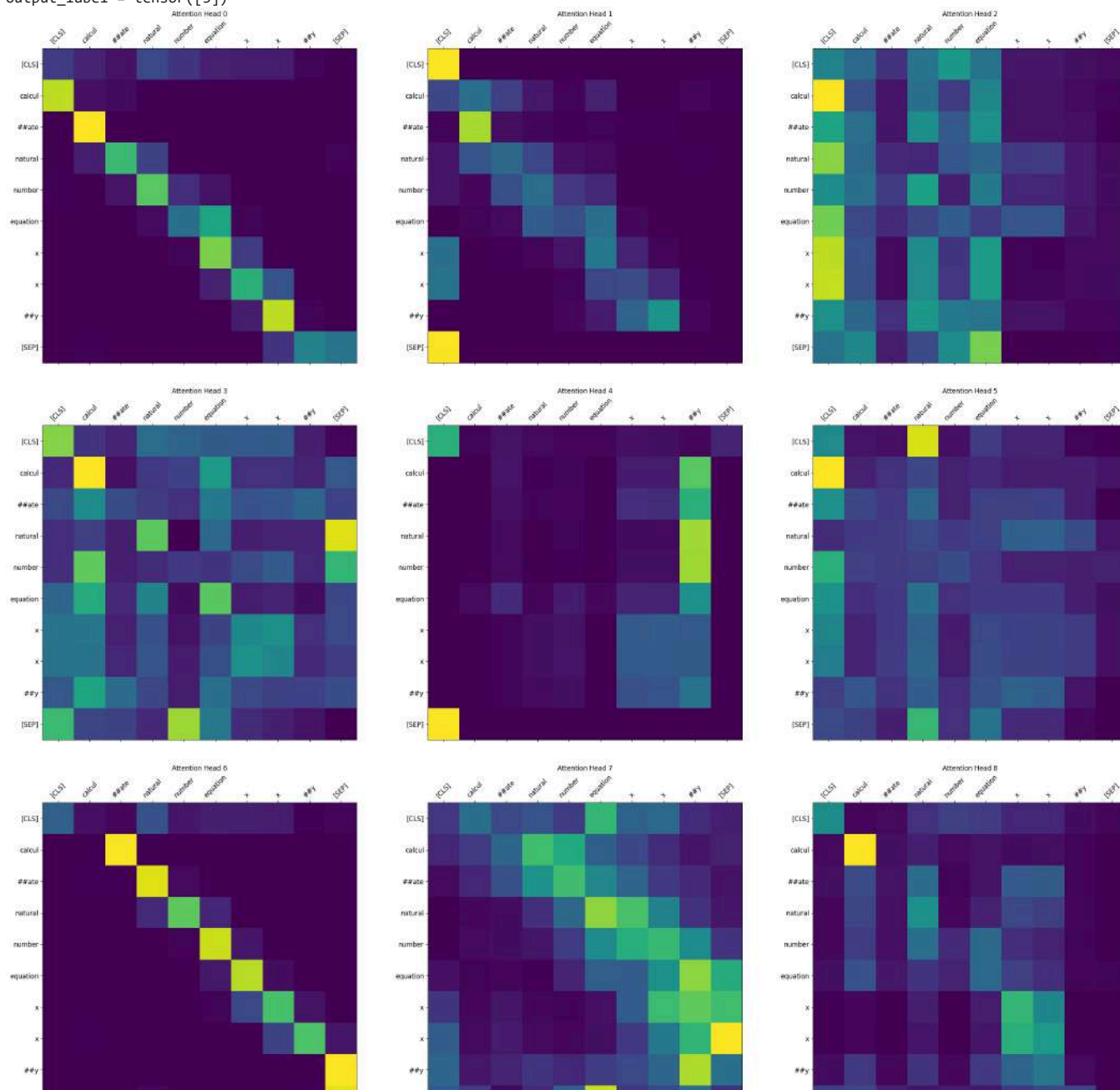


```
'calculate natural number equation x xy'
```

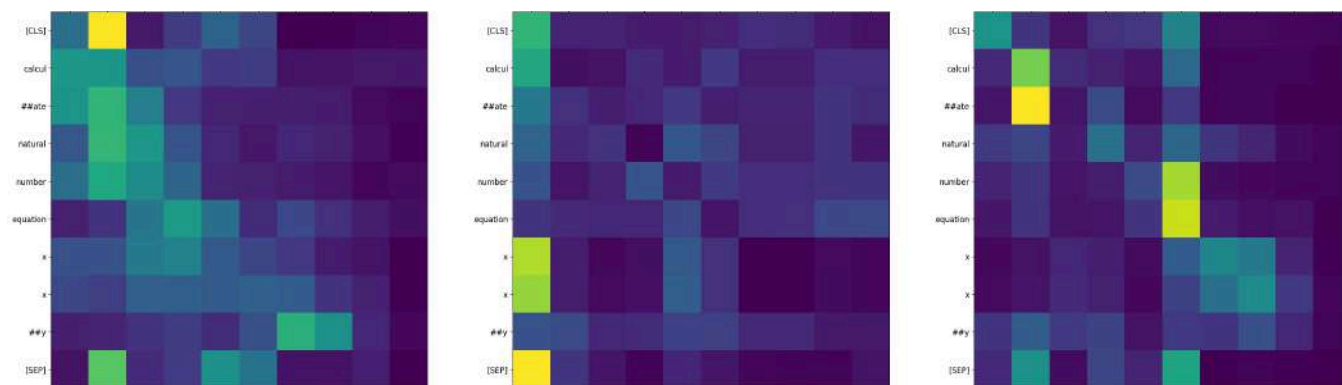
```
# подаем текст после препроцессинга
draw_first_layer_attention_maps(attention_head_ids=range(0, model_1_1.backbone.config.num_attention_heads),
                                text=text,
                                model=model_1_1)
```



```
output_label = tensor([3])
```

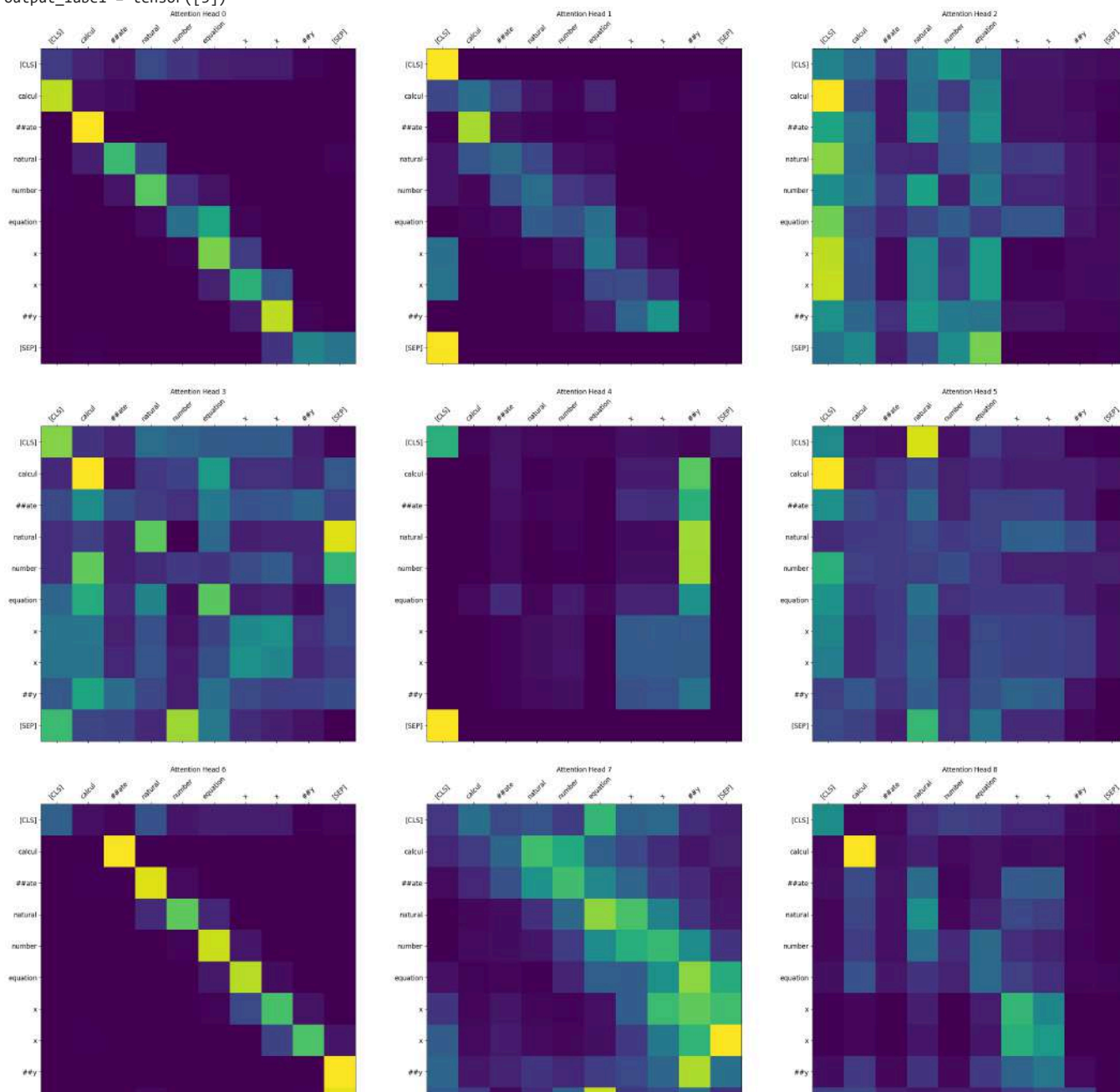


```
draw_first_layer_attention_maps(attention_head_ids=range(0, model_1_2.backbone.config.num_attention_heads),
                                text=text,
                                model=model_1_2)
```

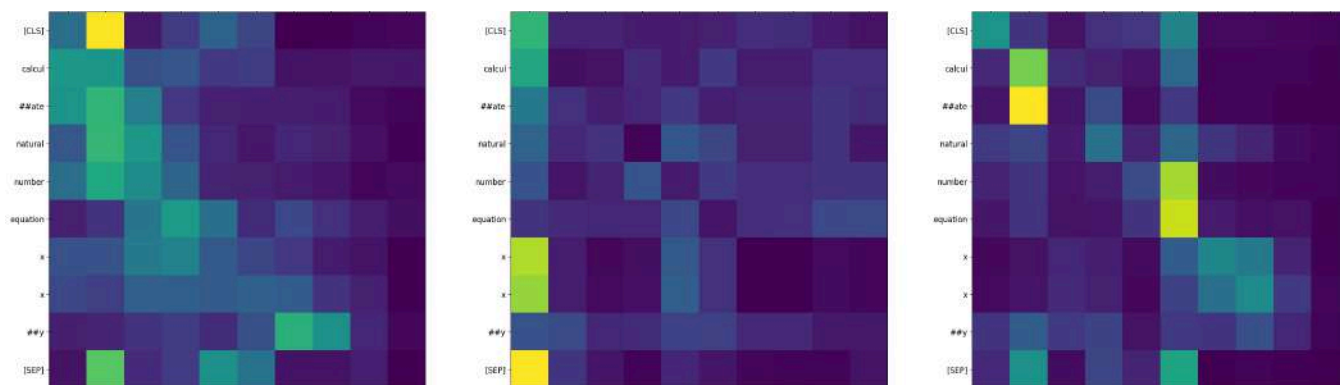




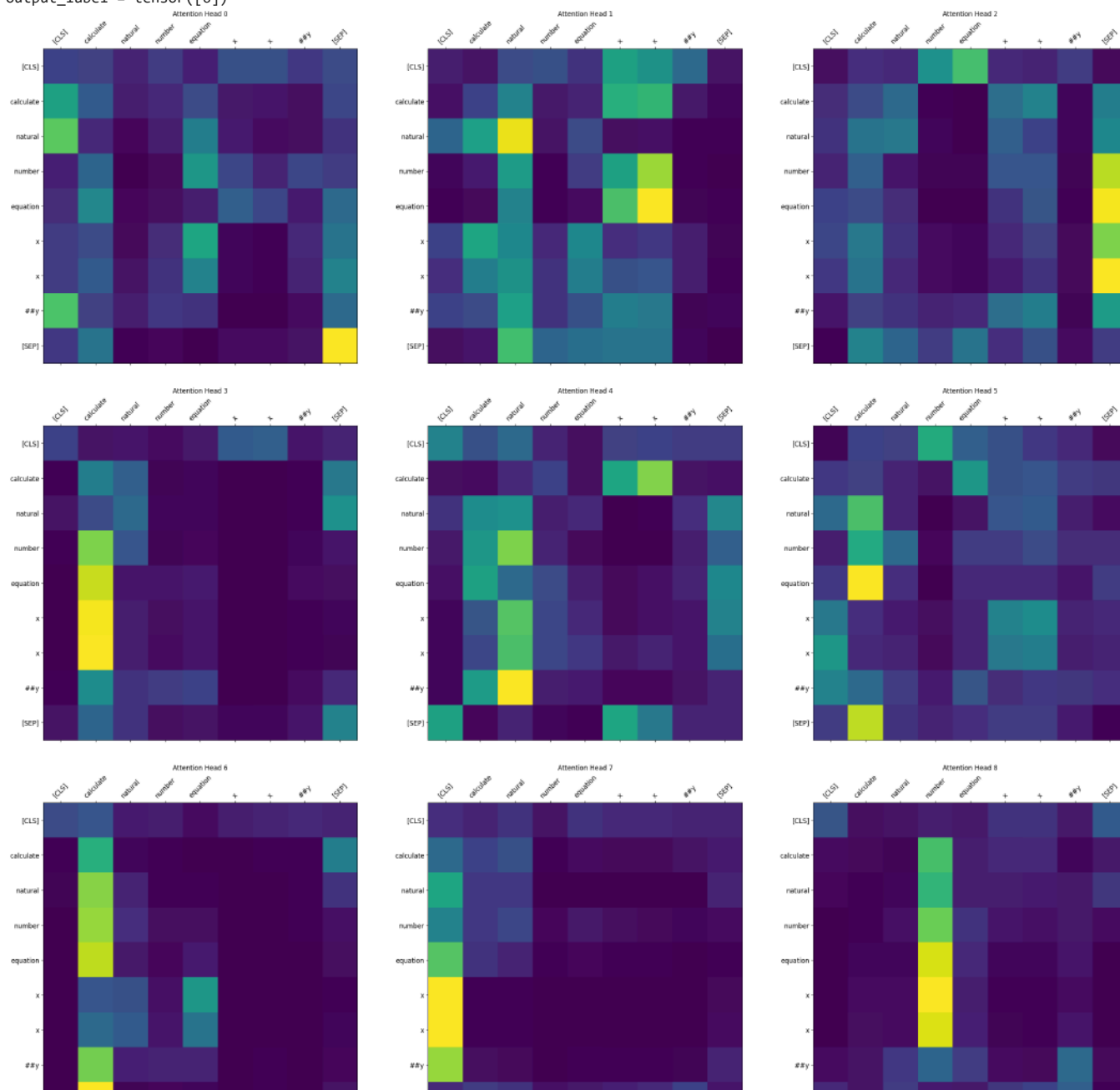
```
output_label = tensor([3])
```



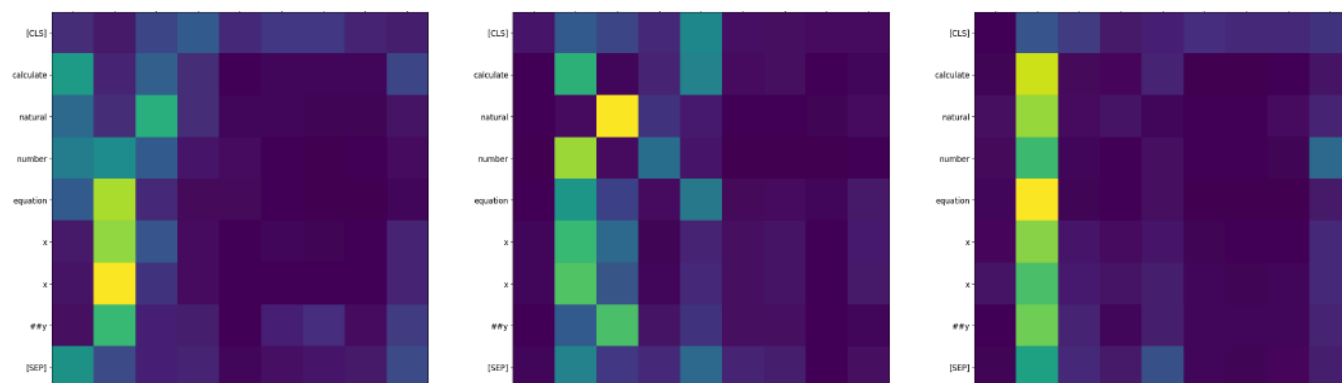
```
draw_first_layer_attention_maps(attention_head_ids=range(0, model_2_1.backbone.config.num_attention_heads),
                                text=text,
                                model=model_2_1)
```



```
output_label = tensor([6])
```

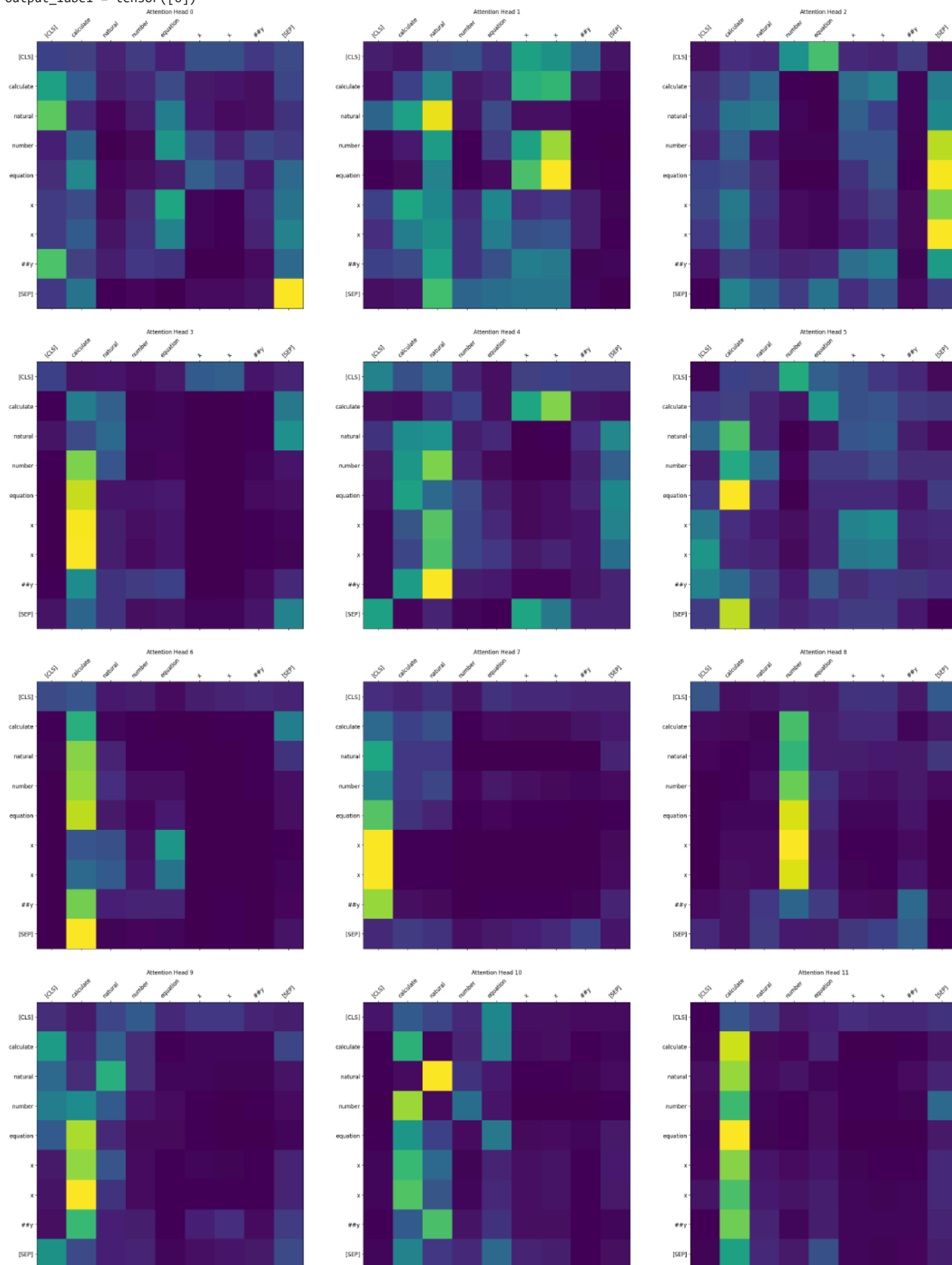


```
draw_first_layer_attention_maps(attention_head_ids=range(0, model_2_2.backbone.config.num_attention_heads),
                                text=text,
                                model=model_2_2)
```



```
draw_first_layer_attention_maps(attention_head_ids=range(0, model_2_2.backbone.config.num_attention_heads),  
                                text=text,  
                                model=model_2_2)
```

```
output_label = tensor([6])
```



Вывод в конце

## ✓ Задание 8 (1 балл)

Сделайте то же самое для дообученных моделей. Изменились ли карты внимания и связи, которые они улавливают? Почему?

### ✓ текст 1

```
test_data.iloc[4]
```

	4151
Unnamed: 0	4151
problem_text	A graph of the function $y = ax^2 + bx + c$ is sh...
topic	polynoms
process_text	graph function ax bx c shown coordinate plane ...
label	1
text_tok_1	[2, 23607, 5471, 68, 981, 69, 981, 70, 5741, 2...
len_tok_1	23
text_tok_2	[101, 10629, 3853, 22260, 1038, 2595, 1039, 34...
len_tok_2	23

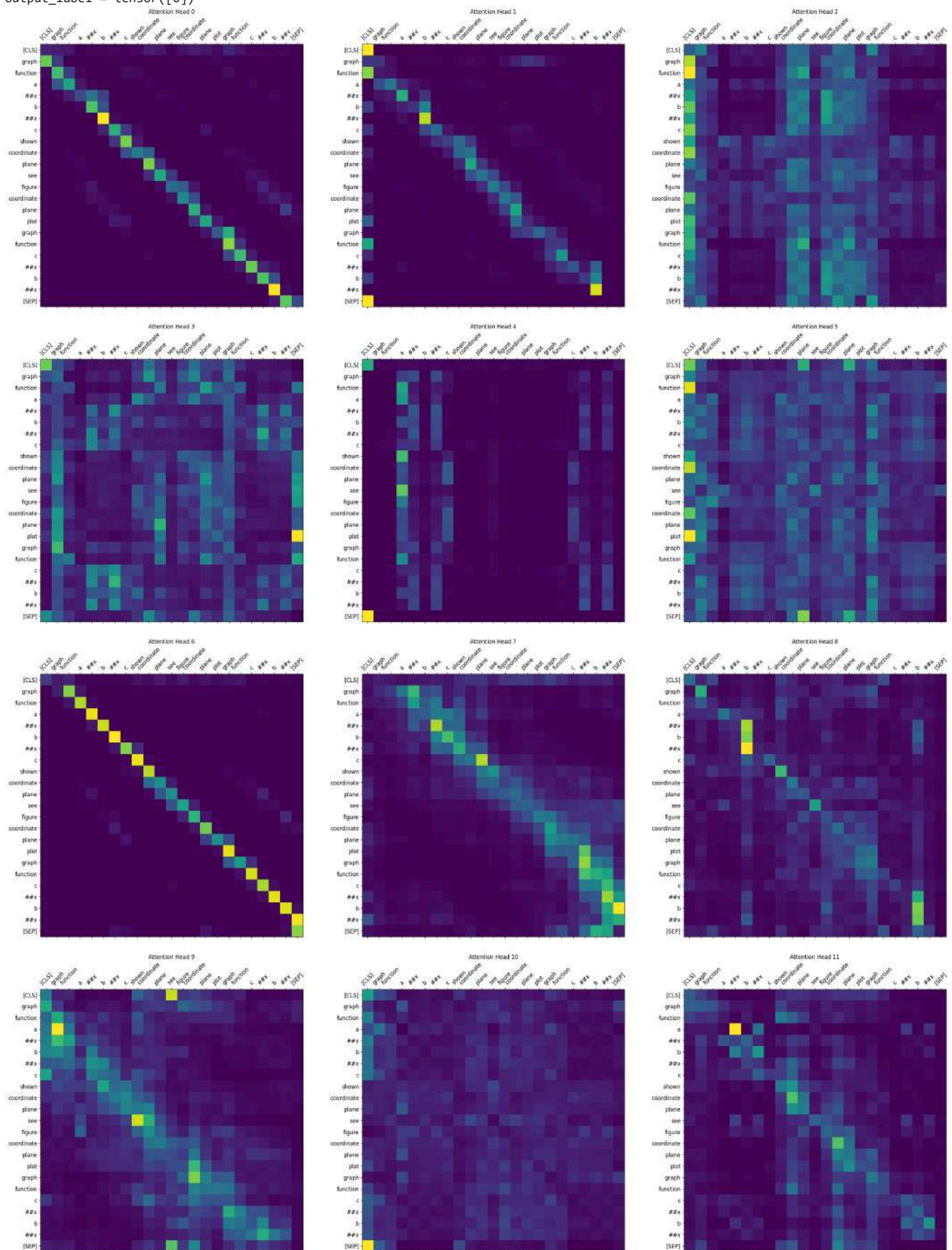
dtype: object

```
text=test_data.iloc[4].process_text
text
```

```
'graph function ax bx c shown coordinate plane see figure coordinate plane plot graph function cx bx'
```

```
# подаем текст после препроцессинга
draw_first_layer_attention_maps(attention_head_ids=range(0, rubert_tiny_finetuned_with_freezed_backbone.backbone.config.num_
                                text=text,
                                model=rubert_tiny_finetuned_with_freezed_backbone)
```

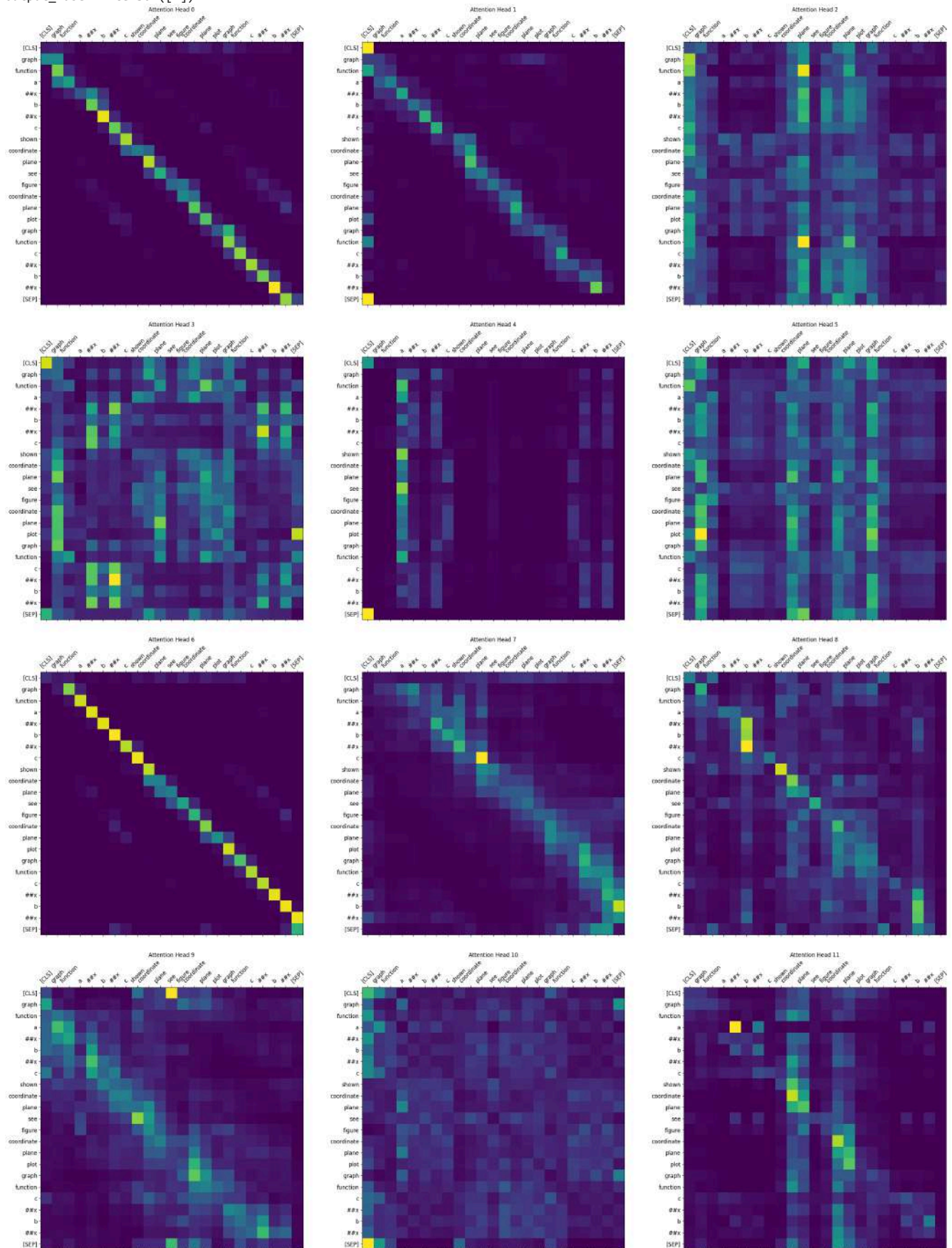
```
output_label = tensor([0])
```



```
draw_first_layer_attention_maps(attention_head_ids=range(0, rubert_tiny_full_finetuned.backbone.config.num_attention_heads),
                                text=text,
                                model=rubert_tiny_full_finetuned)
```



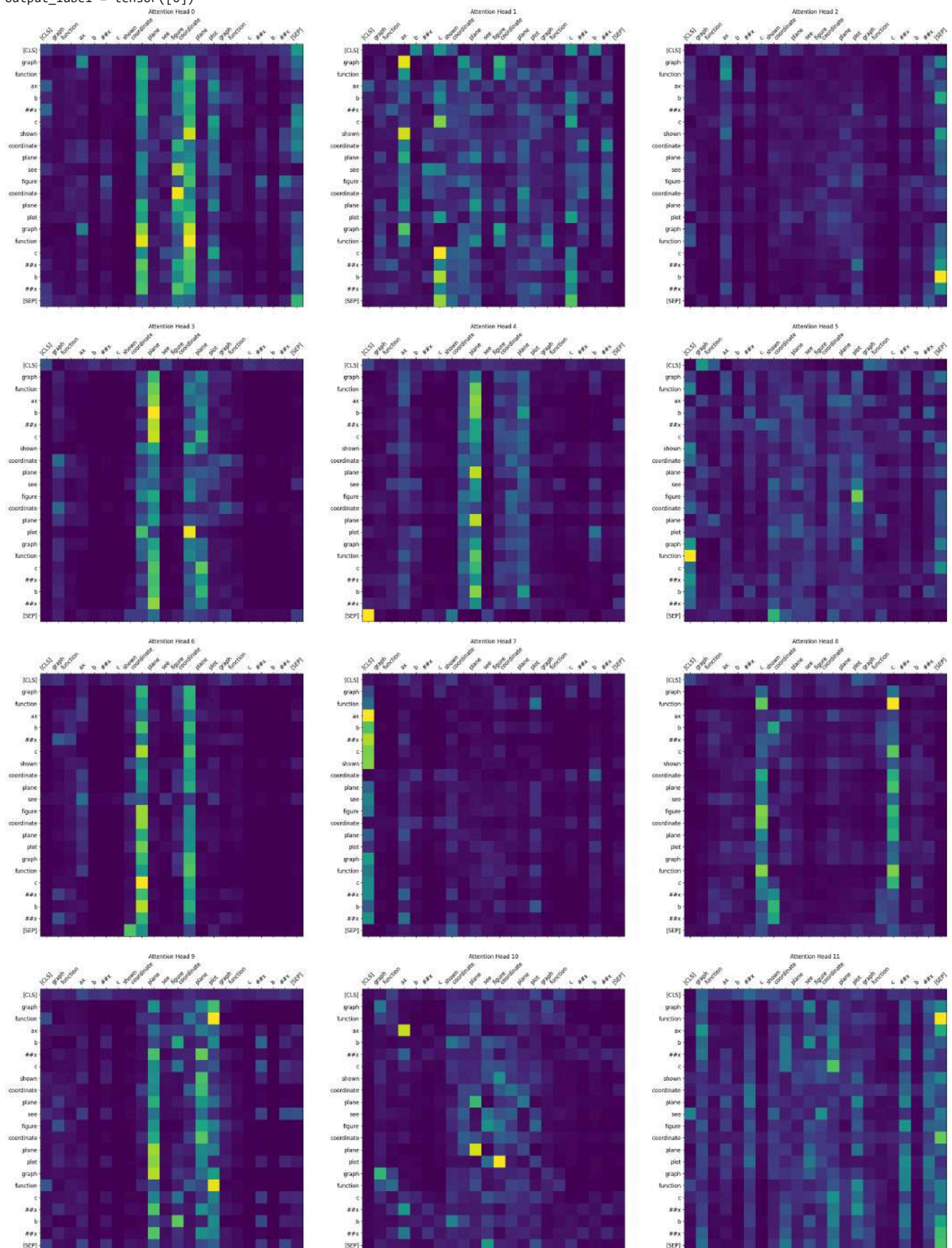
```
output_label = tensor([1])
```





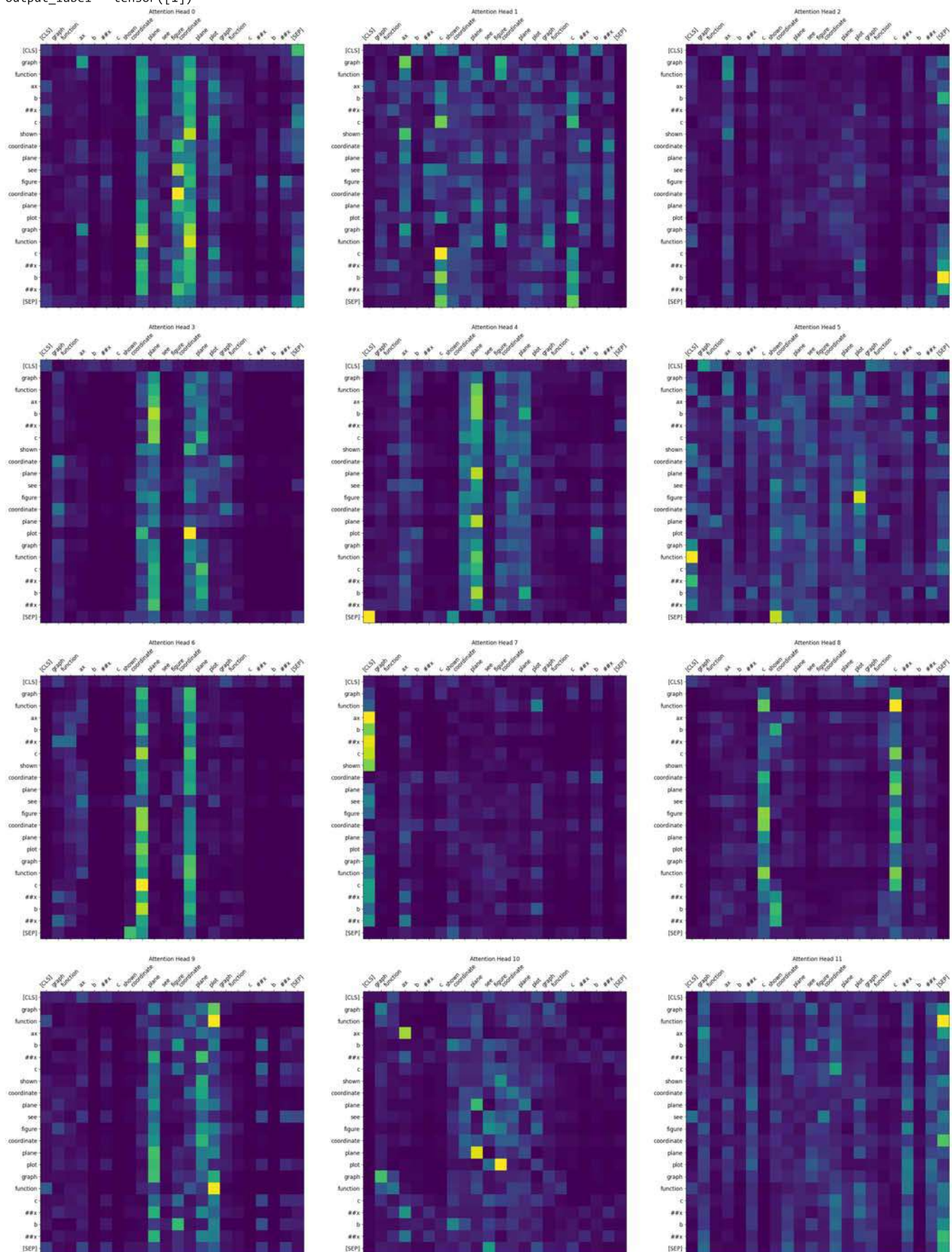
```
draw_first_layer_attention_maps(attention_head_ids=range(0, MathBert_finetuned_with_freezed_backbone.backbone.config.num_att
                                text=text,
                                model=MathBert_finetuned_with_freezed_backbone)
```

→ output\_label = tensor([0])



```
draw_first_layer_attention_maps(attention_head_ids=range(0, MathBert_full_finetuned.backbone.config.num_attention_heads),  
                                text=text,  
                                model=MathBert_full_finetuned)
```

```
output_label = tensor([1])
```



## ▼ текст 2

```
test_data.iloc[5]
```



	2851
<b>Unnamed: 0</b>	2851
<b>problem_text</b>	Calculate in natural numbers the equation: x3 ...
<b>topic</b>	number_theory
<b>process_text</b>	calculate natural number equation x xy
<b>label</b>	0
<b>text_tok_1</b>	[2, 24896, 2081, 2606, 1503, 19042, 91, 91, 58...
<b>len_tok_1</b>	10
<b>text_tok_2</b>	[101, 18422, 3019, 2193, 8522, 1060, 1060, 210...
<b>len_tok_2</b>	10

**dtype:** object

```
text=test_data.iloc[5].process_text
text
```

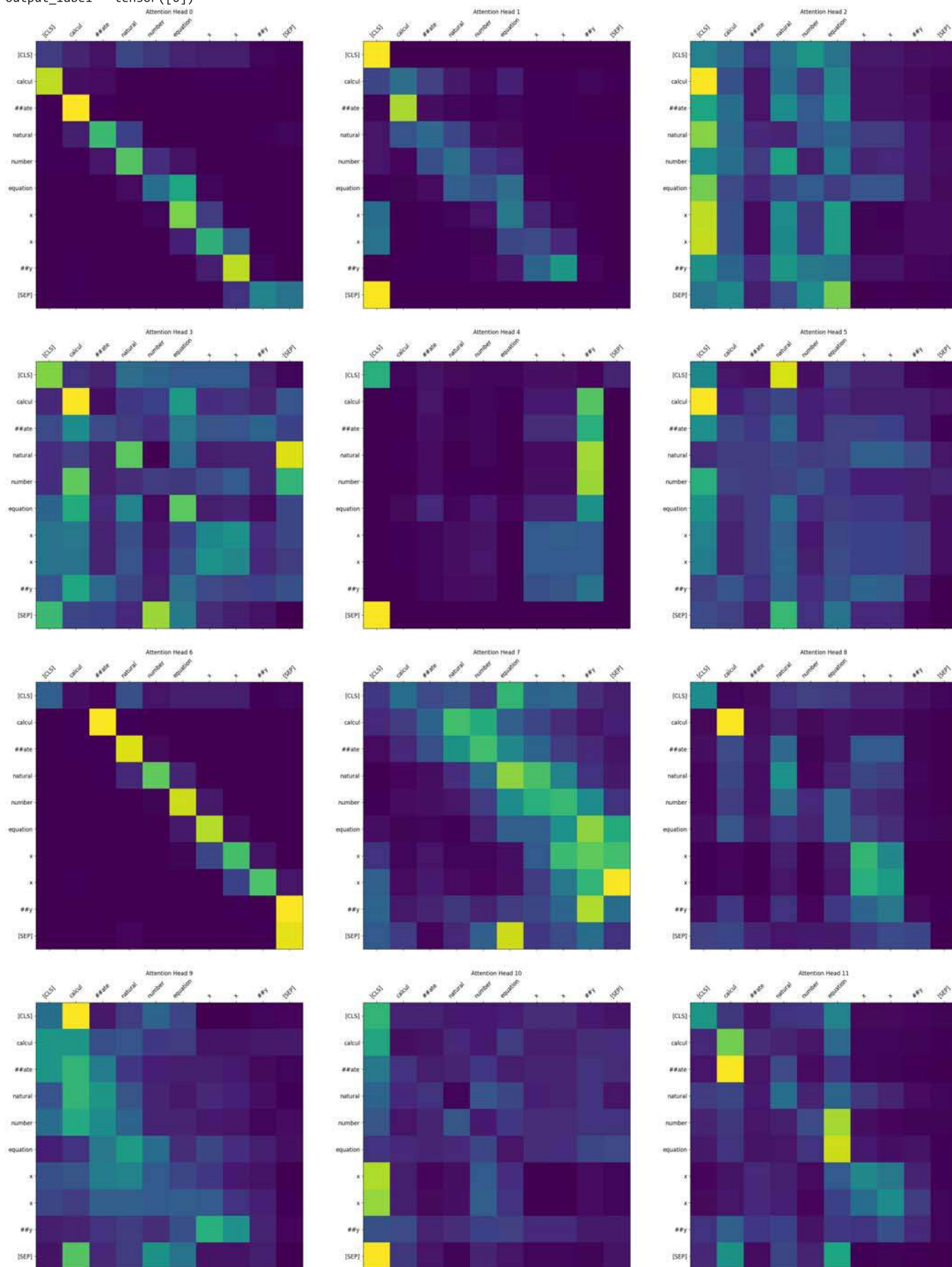


```
"calculate natural number equation x xy"
```

```
# подаем текст после препроцессинга
draw_first_layer_attention_maps(attention_head_ids=range(0, rubert_tiny_finetuned_with_freezed_backbone.backbone.config.num_
                                text=text,
                                model=rubert_tiny_finetuned_with_freezed_backbone)
```

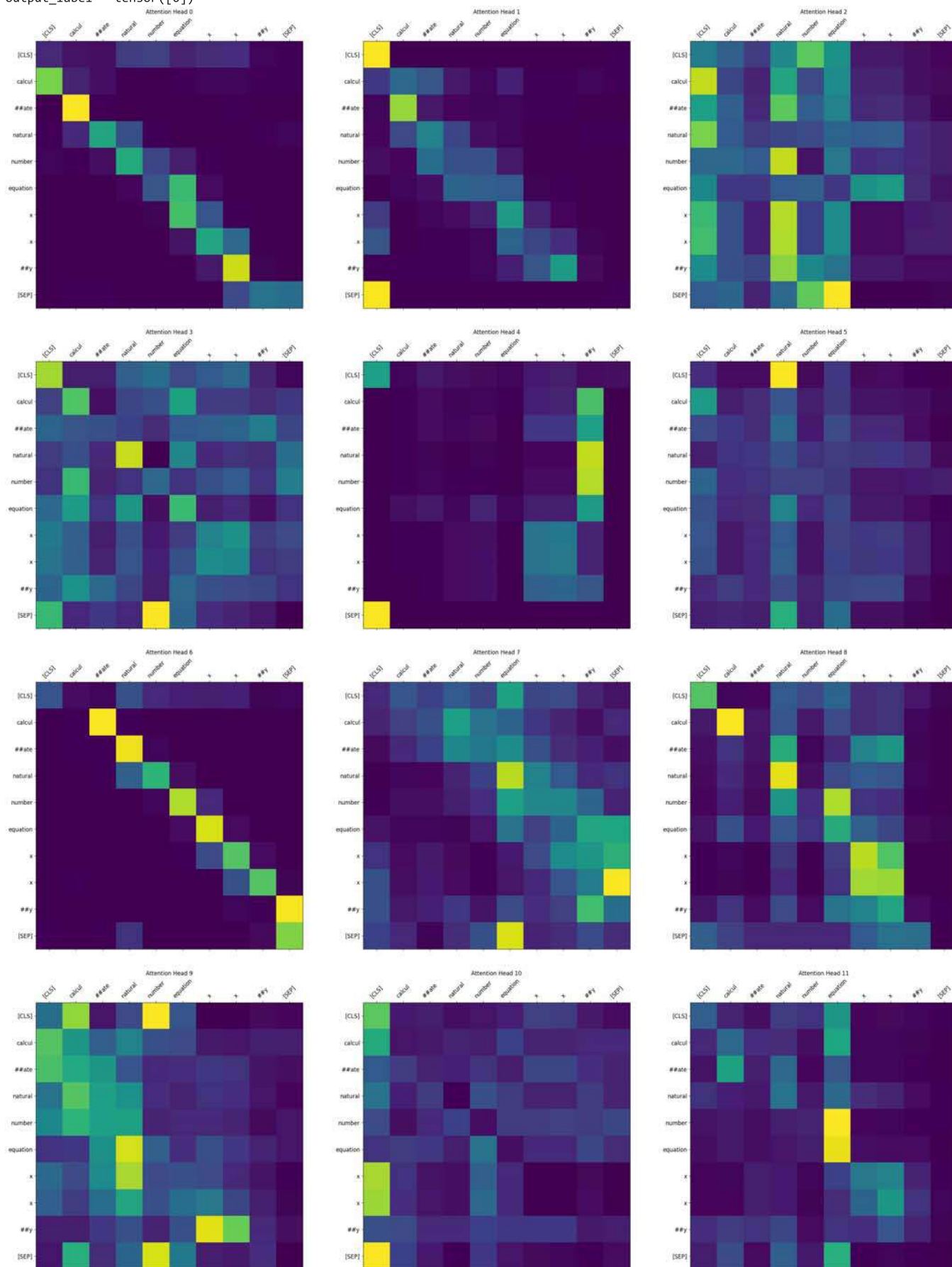


```
output_label = tensor([0])
```



```
draw_first_layer_attention_maps(attention_head_ids=range(0, rubert_tiny_full_finetuned.backbone.config.num_attention_heads),  
                                text=text,  
                                model=rubert_tiny_full_finetuned)
```

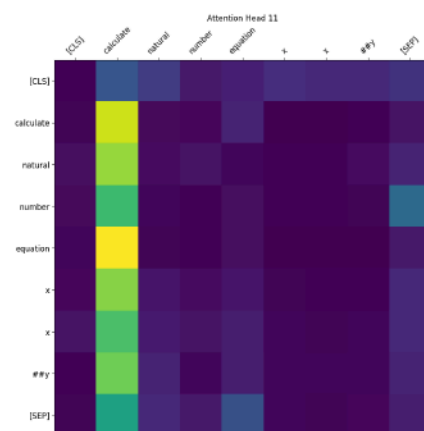
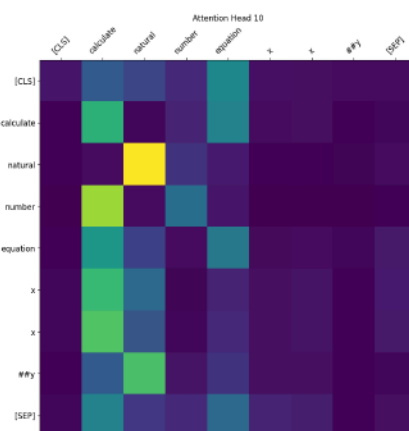
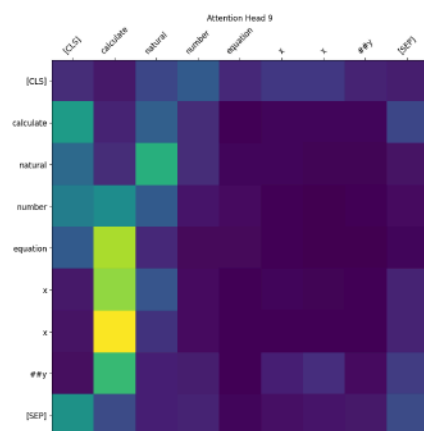
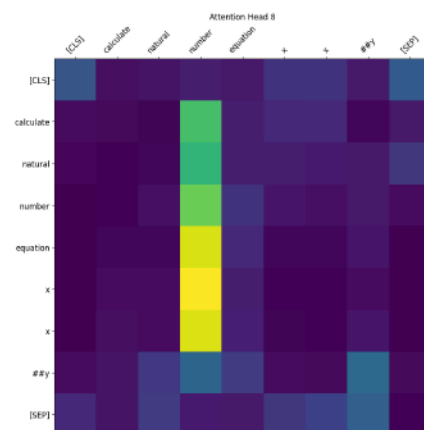
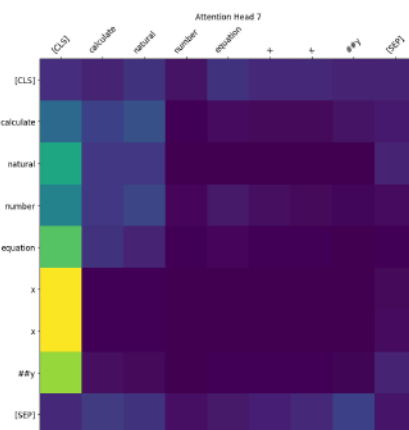
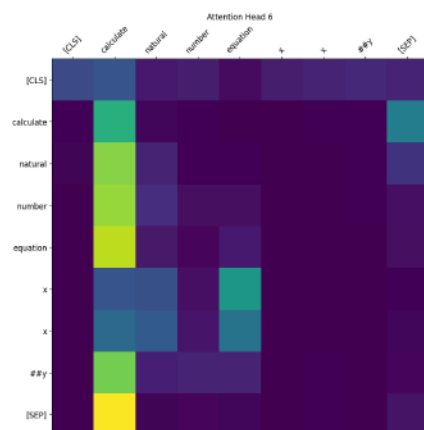
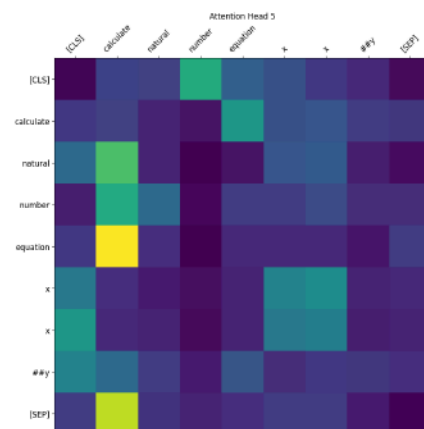
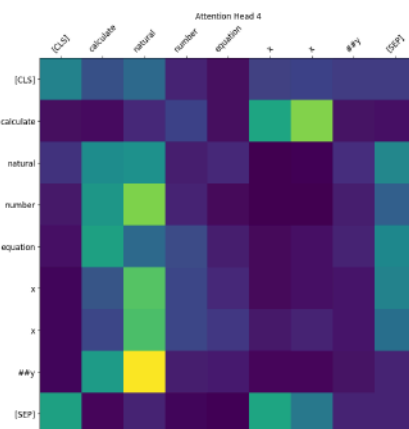
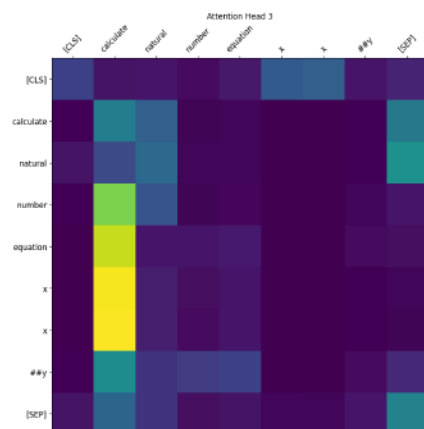
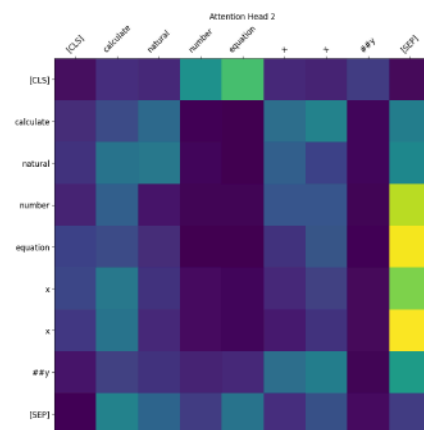
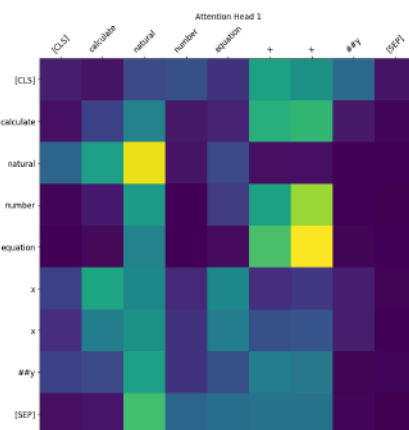
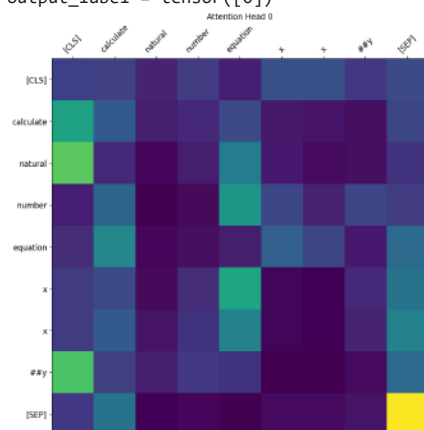
```
output_label = tensor([0])
```





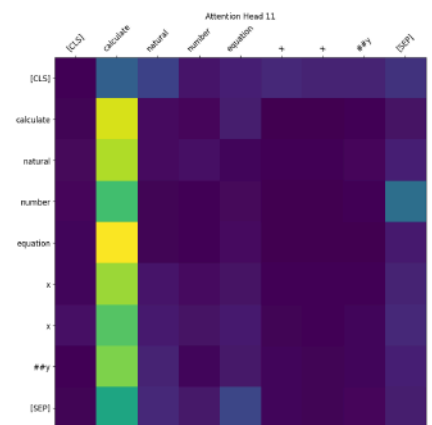
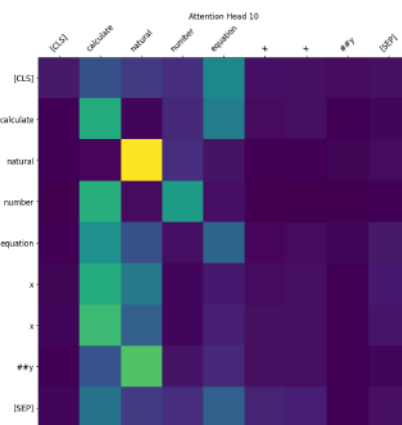
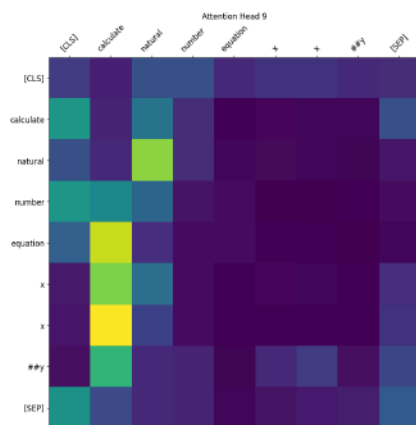
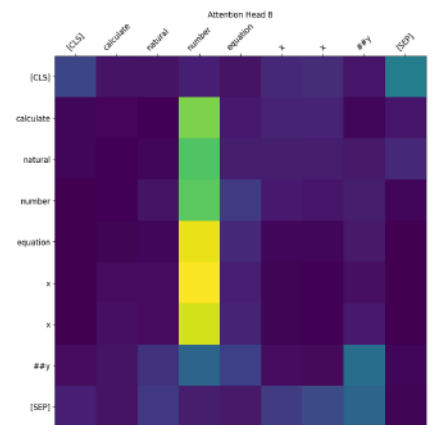
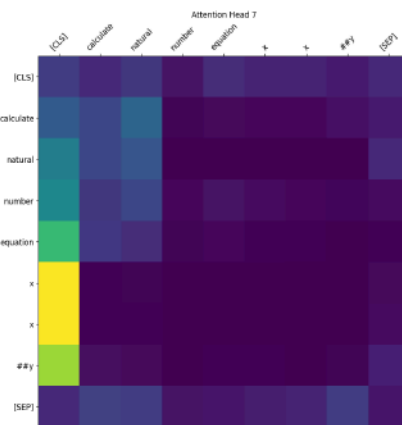
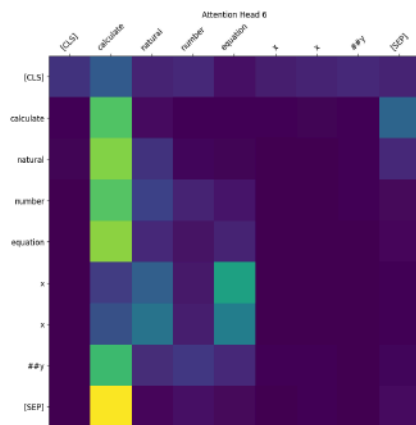
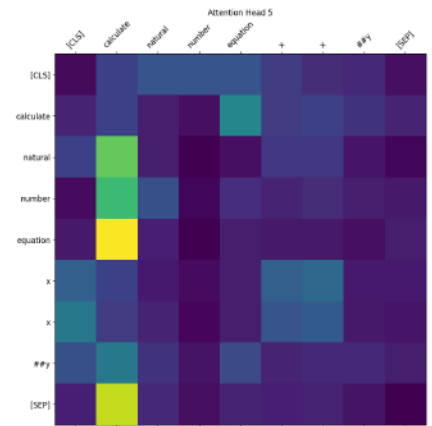
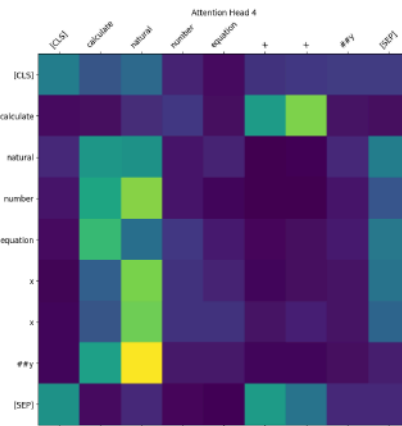
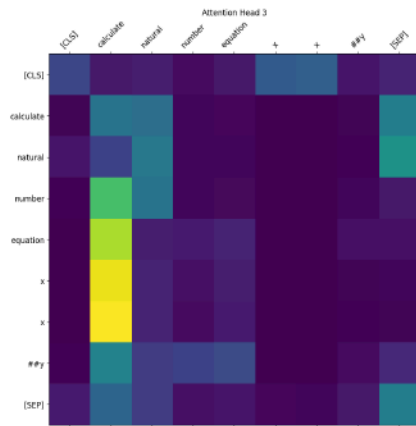
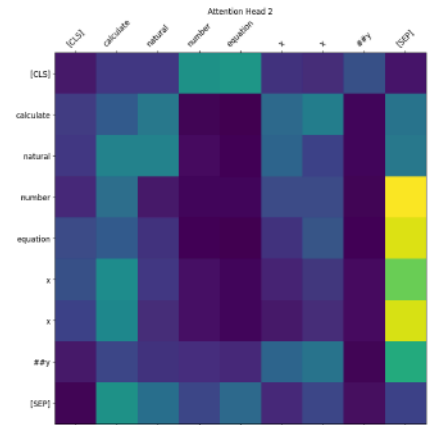
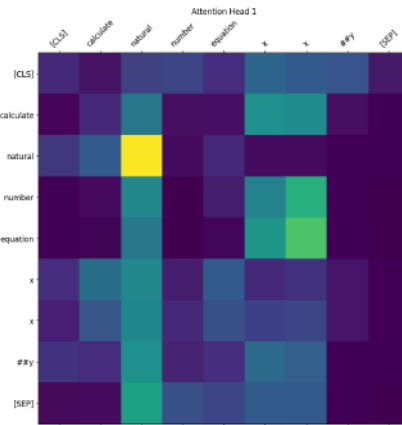
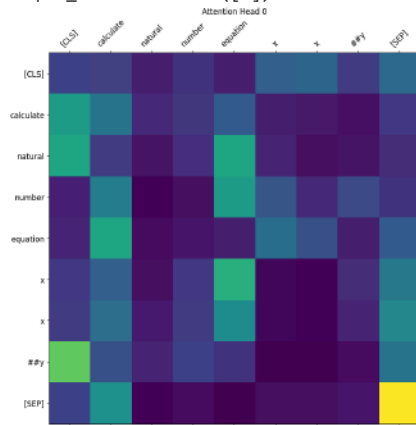
```
draw_first_layer_attention_maps(attention_head_ids=range(0, MathBert_finetuned_with_freezed_backbone.backbone.config.num_att
                                text=text,
                                model=MathBert_finetuned_with_freezed_backbone)
```

output\_label = tensor([0])



```
draw_first_layer_attention_maps(attention_head_ids=range(0, MathBert_full_finetuned.backbone.config.num_attention_heads),
                                text=text,
                                model=MathBert_full_finetuned)
```

→ output\_label = tensor([0])



вывод в конце

## ✓ Вывод

### Задание 5

оцениваем по валидационному датасету:

- при прочих равных условиях метрики в модели math\_bert получились немного хуже по сравнению с rubert\_tiny
- изначально math\_bert была предобучена на математических данных, возможно нужна более тщательная настройка гиперпараметров для улучшения результата
- rubert\_tiny: f1\_train = 0.6448678766048417, f1\_val = 0.47445461744121803 (эпоха 18)
- math\_bert: f1\_train = 0.6422360327444256, f1\_val = 0.4509936372832443 (эпоха 9)

### Задание 7

- **rubert\_tiny**
  - в головах 0,1,6 (менее выражено в головах 7, 8, 9, 11) присутствует сильно выделенная диагональ для первого слоя внимания это нормальное поведение (Каждый token уделяет максимальное внимание самому себе)
  - головы 0, 1, 6 (в меньшей степени головы 7, 8, 9, 11) фокусируются на самовнимании
  - в головах 1, 2, 5, 10 видим вертикальную линию у токена [CLS], это означает что в этих головах один token активно влияет на многие другие (т.е., [CLS] доминирует в контексте).
  - в голове 3 видим вертикальную линию у токена [SEP], это означает что в этих головах один token активно влияет на многие другие (т.е., [SEP] доминирует в контексте).
  - в голове 2 так же отслеживаются некие зависимости между полными словами
  - в голове 4 наоборот отслеживаются некие зависимости между неполными словами (a, b, c, ##x, ##y, x)
- **math\_bert**
  - нет голов, где присутствует диагональ
  - в голове 2 видим вертикальную линию у токена [SEP], это означает что в этих головах один token активно влияет на многие другие (т.е., [SEP] доминирует в контексте).
  - в голове 7 видим вертикальную линию у токена [CLS], это означает что в этих головах один token активно влияет на многие другие (т.е., [CLS] доминирует в контексте).
  - в головах 0, 1, 3, 4, 6, 8, 9, 11 ярко выражены вертикальные линии

### Задание 8

- у моделей с замороженными слоями (rubert\_tiny\_finetuned\_with\_freezed\_backbone, MathBert\_finetuned\_with\_freezed\_backbone) ничего не поменялось
- у моделей с незамороженными слоями прослеживается некая динамика, но все равно связи особо не изменились, т.к. мы смотрим только первый слой attention

