# Тестовое Тинькофф

Выполнил Чайников Константин

Тестовое задание по NLP

- * Опционально, будет плюсом

1) Сделать классификатор интентов

- Для обучения использовать датасет https://github.com/PolyAI-LDN/task-specific-datasets/banking_data/ (https://github.com/PolyAI-LDN/task-specific-datasets/banking_data/)
- Сделать небольшой отчет в свободной форме для оценки качества решения (tf-idf baseline на тесте выбивает от 0.9 f1-score если не получится улучшить - не критично, но выше бейзлайна будет большим плюсом)
- Цель: написать более менее адекватную архитектуру, построить читаемый отчет
- * Реализовать Self-adjusting Dice Loss из https://www.aclweb.org/anthology/2020.acl-main.45.pdf (https://www.aclweb.org/anthology/2020.acl-main.45.pdf) Сравнить с Cross Entropy
- * Реализовать механизм семплинга батчей, чтобы компенсировать несбалансированность классов в датасете Сравнить с обычным семплингом

2) Обернуть классификатор в REST сервис

- Метод POST /classify
- На вход подается текст примера
- В ответ возвращается строковый тег интента
- Сервис должен быть завернут в docker контейнер

In [1]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [2]:

```
!git clone https://github.com/PolyAI-LDN/task-specific-datasets
# ссылка на тестовоые данные с лейблами
# https://github.com/congyingxia/IncrementalFSTC/blob/88b17da85024e015372bb5171b271dfcfe6b6a1d/data/banking77/split/total_test.txt
!unzip /content/drive/MyDrive/data_for_tinkoff/total_test.txt.zip
```

```
Cloning into 'task-specific-datasets'...
remote: Enumerating objects: 103, done.
remote: Counting objects: 100% (103/103), done.
remote: Compressing objects: 100% (58/58), done.
remote: Total 103 (delta 58), reused 77 (delta 45), pack-reused 0
Receiving objects: 100% (103/103), 1001.92 KiB | 3.06 MiB/s, done.
Resolving deltas: 100% (58/58), done.
Archive:  /content/drive/MyDrive/data_for_tinkoff/total_test.txt.zip
 extracting: total_test.txt
```

In [3]:

```
!pip install transformers
!pip install sentencepiece
```

```
Collecting transformers
  Downloading https://files.pythonhosted.org/packages/00/92/6153f4912b84ee
1ab53ab45663d23e7cf3704161cb5ef18b0c07e207cef2/transformers-4.7.0-py3-none
-any.whl (2.5MB)
     |████████████████████████████████| 2.5MB 10.0MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-p
ackages (from transformers) (2.23.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-pac
kages (from transformers) (3.13)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-p
ackages (from transformers) (3.0.12)
Collecting huggingface-hub==0.0.8
  Downloading https://files.pythonhosted.org/packages/a1/88/7b1e45720ecf59
c6c6737ff332f41c955963090a18e72acbcbeac6b25e86/huggingface_hub-0.0.8-py3-n
one-any.whl
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python
3.7/dist-packages (from transformers) (2019.12.20)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dis
t-packages (from transformers) (1.19.5)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-
packages (from transformers) (20.9)
Collecting sacremoses
  Downloading https://files.pythonhosted.org/packages/75/ee/67241dc87f2660
93c533a2d4d3d69438e57d7a90abb216fa076e7d475d4a/sacremoses-0.0.45-py3-none-
any.whl (895kB)
     |████████████████████████████████| 901kB 34.4MB/s
Collecting tokenizers<0.11,>=0.10.1
  Downloading https://files.pythonhosted.org/packages/d4/e2/df3543e8ffdab6
8f5acc73f613de9c2b155ac47f162e725dcac87c521c11/tokenizers-0.10.3-cp37-cp37
m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux20
10_x86_64.whl (3.3MB)
     |████████████████████████████████| 3.3MB 35.7MB/s
Requirement already satisfied: importlib-metadata; python_version < "3.8"
in /usr/local/lib/python3.7/dist-packages (from transformers) (4.5.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist
-packages (from transformers) (4.41.1)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/di
st-packages (from requests->transformers) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (1.2
4.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.7/dist-packages (from requests->transformers) (2021.5.30)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python
3.7/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.
7/dist-packages (from packaging->transformers) (2.4.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-pac
kages (from sacremoses->transformers) (1.0.1)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packag
es (from sacremoses->transformers) (1.15.0)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-pack
ages (from sacremoses->transformers) (7.1.2)
Requirement already satisfied: typing-extensions>=3.6.4; python_version <
"3.8" in /usr/local/lib/python3.7/dist-packages (from importlib-metadata;
python_version < "3.8"->transformers) (3.7.4.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata; python_version < "3.8"->transformers)
(3.4.1)
Installing collected packages: huggingface-hub, sacremoses, tokenizers, tr
ansformers
```

```
Successfully installed huggingface-hub-0.0.8 sacremoses-0.0.45 tokenizers-
0.10.3 transformers-4.7.0
Collecting sentencepiece
  Downloading https://files.pythonhosted.org/packages/ac/aa/1437691b0c7c83
086ebb79ce2da16e00bef024f24fec2a5161c35476f499/sentencepiece-0.1.96-cp37-c
p37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2MB)
     |████████████████████████████████| 1.2MB 14.7MB/s
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.96
```

# Обязательно! Перезапускаем ядро

среда выполнения -> перезапустить среду выполнения

In [1]:

```
!apt-get install swig
!pip install jamspell
!tar -xvzf /content/drive/MyDrive/data_for_tinkoff/en.tar.gz
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  swig3.0
Suggested packages:
  swig-doc swig-examples swig3.0-examples swig3.0-doc
The following NEW packages will be installed:
  swig swig3.0
0 upgraded, 2 newly installed, 0 to remove and 39 not upgraded.
Need to get 1,100 kB of archives.
After this operation, 5,822 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/universe amd64 swig3.0 amd64
3.0.12-1 [1,094 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/universe amd64 swig amd64 3.
0.12-1 [6,460 B]
Fetched 1,100 kB in 0s (9,784 kB/s)
Selecting previously unselected package swig3.0.
(Reading database ... 160772 files and directories currently installed.)
Preparing to unpack .../swig3.0_3.0.12-1_amd64.deb ...
Unpacking swig3.0 (3.0.12-1) ...
Selecting previously unselected package swig.
Preparing to unpack .../swig_3.0.12-1_amd64.deb ...
Unpacking swig (3.0.12-1) ...
Setting up swig3.0 (3.0.12-1) ...
Setting up swig (3.0.12-1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Collecting jamspell
  Downloading https://files.pythonhosted.org/packages/02/35/2ad708256367dc
f3443766ca588a856d3334abc8dbbd760ff19e36149308/jamspell-0.0.12.tar.gz (174
kB)
     |████████████████████████████████| 184kB 12.2MB/s
Building wheels for collected packages: jamspell
  Building wheel for jamspell (setup.py) ... done
  Created wheel for jamspell: filename=jamspell-0.0.12-cp37-cp37m-linux_x8
6_64.whl size=1347422 sha256=4baffab422b783f8e12ada1ba833960928341faf15bdb
ad51d9e9b3acef29709
  Stored in directory: /root/.cache/pip/wheels/ed/f1/0e/4a173a979c77db7fdb
3590ab530f2e8334fa3cdedc079ac932
Successfully built jamspell
Installing collected packages: jamspell
Successfully installed jamspell-0.0.12
en.bin
```

In [2]:

```
import jamspell
corrector = jamspell.TSpellCorrector() # корректор опечаток
corrector.LoadLangModel('en.bin') # предобученную модель взял здесь
#                                 https://github.com/bakwc/JamSpell
```

Out[2]:

```
True
```

# Часть 1 Анализ данных

Импортируем библиотеки и проверяем их версии.

In [3]:

```
print('Версии используемых библиотек')
import numpy as np
print('numpy', np.__version__)
import pandas as pd
print('pandas', pd.__version__)

import seaborn as sns
print('seaborn', sns.__version__)

import matplotlib.pyplot as plt
import matplotlib
plt.style.use('ggplot')
print('matplotlib', matplotlib.__version__)

import random

import torch
from transformers import BertTokenizer, BertConfig
from transformers import AdamW, BertForSequenceClassification
import torch.nn.functional as F
import torch.nn as nn
import torch.optim as optim
print('torch', torch.__version__)

from sklearn.metrics import f1_score
import sklearn
print('sklearn', sklearn.__version__)

import transformers
print('transformers', transformers.__version__)

import jamspell
#print('jamspell', jamspell.__version__)

#import sentencepiece
#print('sentencepiece', sentencepiece.__version__)

#import swig
#print('swig', swig.__version__)

#import collections
#print('collections', collections.__version__)

import re
print('re', re.__version__)

import nltk
print('nltk', nltk.__version__)

import IPython
print('IPython', IPython.__version__)
from IPython.display import clear_output
from collections import Counter
import re

import tqdm

print('tqdm', tqdm.__version__)
#from tqdm.notebook import tqdm, trange
```

```
USE_CUDA = torch.cuda.is_available()
device = torch.device("cuda" if USE_CUDA else "cpu")
```

```
Версии используемых библиотек
numpy 1.19.5
pandas 1.1.5
seaborn 0.11.1
matplotlib 3.2.2
torch 1.9.0+cu102
sklearn 0.22.2.post1
transformers 4.7.0
re 2.2.1
nltk 3.2.5
IPython 5.5.0
tqdm 4.41.1
```

Будем сидировать важные моменты

In [4]:

```
def set_seed():
    random.seed(42)
    np.random.seed(42)
    torch.random.manual_seed(42)
    torch.cuda.random.manual_seed(42)
    torch.cuda.random.manual_seed_all(42)
```

In [5]:

```
df_train = pd.read_csv('/content/task-specific-datasets/banking_data/train.csv')
df_test = pd.read_csv('/content/total_test.txt', sep='\t', names= ['category', 'text'])
```

In [6]:

```
df_train.sample(3)
```

Out[6]:

| | text | category |
|---|---|---|
| **4752** | how do i get a refund for a direct debit payme... | direct_debit_payment_not_recognised |
| **334** | How are exchange rates determined? | exchange_rate |
| **6394** | I was charged two times for the same thing. | transaction_charged_twice |

In [7]:

```
df_test.sample(3)
```

Out[7]:

| | category | text |
|---|---|---|
| **2844** | activate_my_card | How can I activate my card so I can start usin... |
| **665** | pending_top_up | Hi I'm a new customer and tried topping up for... |
| **2247** | receiving_money | I get paid in GBP. Should I configure this and... |

In [8]:

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10003 entries, 0 to 10002
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   text      10003 non-null  object
 1   category  10003 non-null  object
dtypes: object(2)
memory usage: 156.4+ KB
```

In [10]:

```
labels_dict = {}
weight = [] # пригодятся для лосса в будущем
i = 0
for cat, j in df_train["category"].value_counts().items():
    labels_dict[cat] = i
    weight.append(187/j)
    i+=1
label_to_text = {j:i for i,j in labels_dict.items()}

def f(text):
    return labels_dict[text]

X_train = df_train.text
X_test  = df_test.text
y_train = df_train.category.apply(f)
y_test = df_test.category.apply(f)
```

In [11]:

```
label_to_text
```

Out[11]:

```
{0: 'card_payment_fee_charged',
 1: 'direct_debit_payment_not_recognised',
 2: 'balance_not_updated_after_cheque_or_cash_deposit',
 3: 'wrong_amount_of_cash_received',
 4: 'cash_withdrawal_charge',
 5: 'transaction_charged_twice',
 6: 'declined_cash_withdrawal',
 7: 'transfer_fee_charged',
 8: 'transfer_not_received_by_recipient',
 9: 'balance_not_updated_after_bank_transfer',
 10: 'request_refund',
 11: 'card_payment_not_recognised',
 12: 'card_payment_wrong_exchange_rate',
 13: 'extra_charge_on_statement',
 14: 'wrong_exchange_rate_for_cash_withdrawal',
 15: 'Refund_not_showing_up',
 16: 'reverted_card_payment?',
 17: 'cash_withdrawal_not_recognised',
 18: 'activate_my_card',
 19: 'pending_card_payment',
 20: 'cancel_transfer',
 21: 'beneficiary_not_allowed',
 22: 'card_arrival',
 23: 'declined_card_payment',
 24: 'pending_top_up',
 25: 'pending_transfer',
 26: 'top_up_reverted',
 27: 'top_up_failed',
 28: 'pending_cash_withdrawal',
 29: 'card_linking',
 30: 'failed_transfer',
 31: 'visa_or_mastercard',
 32: 'declined_transfer',
 33: 'card_about_to_expire',
 34: 'country_support',
 35: 'getting_spare_card',
 36: 'supported_cards_and_currencies',
 37: 'transfer_timing',
 38: 'automatic_top_up',
 39: 'verify_top_up',
 40: 'apple_pay_or_google_pay',
 41: 'fiat_currency_support',
 42: 'change_pin',
 43: 'exchange_charge',
 44: 'disposable_card_limits',
 45: 'why_verify_identity',
 46: 'lost_or_stolen_phone',
 47: 'edit_personal_details',
 48: 'order_physical_card',
 49: 'exchange_via_app',
 50: 'pin_blocked',
 51: 'top_up_by_card_charge',
 52: 'top_up_by_cash_or_cheque',
 53: 'verify_source_of_funds',
 54: 'transfer_into_account',
 55: 'card_not_working',
 56: 'exchange_rate',
 57: 'card_delivery_estimate',
 58: 'top_up_by_bank_transfer_charge',
```

```
 59: 'age_limit',
 60: 'terminate_account',
 61: 'get_physical_card',
 62: 'passcode_forgotten',
 63: 'verify_my_identity',
 64: 'topping_up_by_card',
 65: 'unable_to_verify_identity',
 66: 'getting_virtual_card',
 67: 'top_up_limits',
 68: 'get_disposable_virtual_card',
 69: 'receiving_money',
 70: 'atm_support',
 71: 'compromised_card',
 72: 'lost_or_stolen_card',
 73: 'card_swallowed',
 74: 'card_acceptance',
 75: 'virtual_card_not_working',
 76: 'contactless_not_working'}
```

In [12]:

```
import json

# Serialize data into file:
json.dump( label_to_text, open( "label_to_text.json", 'w' ) )

# Read data from file:
label_to_text_1 = json.load( open( "label_to_text.json" ) )
```

In [13]:

```
label_to_text_1
```

Out[13]:

```
{'0': 'card_payment_fee_charged',
 '1': 'direct_debit_payment_not_recognised',
 '10': 'request_refund',
 '11': 'card_payment_not_recognised',
 '12': 'card_payment_wrong_exchange_rate',
 '13': 'extra_charge_on_statement',
 '14': 'wrong_exchange_rate_for_cash_withdrawal',
 '15': 'Refund_not_showing_up',
 '16': 'reverted_card_payment?',
 '17': 'cash_withdrawal_not_recognised',
 '18': 'activate_my_card',
 '19': 'pending_card_payment',
 '2': 'balance_not_updated_after_cheque_or_cash_deposit',
 '20': 'cancel_transfer',
 '21': 'beneficiary_not_allowed',
 '22': 'card_arrival',
 '23': 'declined_card_payment',
 '24': 'pending_top_up',
 '25': 'pending_transfer',
 '26': 'top_up_reverted',
 '27': 'top_up_failed',
 '28': 'pending_cash_withdrawal',
 '29': 'card_linking',
 '3': 'wrong_amount_of_cash_received',
 '30': 'failed_transfer',
 '31': 'visa_or_mastercard',
 '32': 'declined_transfer',
 '33': 'card_about_to_expire',
 '34': 'country_support',
 '35': 'getting_spare_card',
 '36': 'supported_cards_and_currencies',
 '37': 'transfer_timing',
 '38': 'automatic_top_up',
 '39': 'verify_top_up',
 '4': 'cash_withdrawal_charge',
 '40': 'apple_pay_or_google_pay',
 '41': 'fiat_currency_support',
 '42': 'change_pin',
 '43': 'exchange_charge',
 '44': 'disposable_card_limits',
 '45': 'why_verify_identity',
 '46': 'lost_or_stolen_phone',
 '47': 'edit_personal_details',
 '48': 'order_physical_card',
 '49': 'exchange_via_app',
 '5': 'transaction_charged_twice',
 '50': 'pin_blocked',
 '51': 'top_up_by_card_charge',
 '52': 'top_up_by_cash_or_cheque',
 '53': 'verify_source_of_funds',
 '54': 'transfer_into_account',
 '55': 'card_not_working',
 '56': 'exchange_rate',
 '57': 'card_delivery_estimate',
 '58': 'top_up_by_bank_transfer_charge',
 '59': 'age_limit',
 '6': 'declined_cash_withdrawal',
 '60': 'terminate_account',
 '61': 'get_physical_card',
```

```
 '62': 'passcode_forgotten',
 '63': 'verify_my_identity',
 '64': 'topping_up_by_card',
 '65': 'unable_to_verify_identity',
 '66': 'getting_virtual_card',
 '67': 'top_up_limits',
 '68': 'get_disposable_virtual_card',
 '69': 'receiving_money',
 '7': 'transfer_fee_charged',
 '70': 'atm_support',
 '71': 'compromised_card',
 '72': 'lost_or_stolen_card',
 '73': 'card_swallowed',
 '74': 'card_acceptance',
 '75': 'virtual_card_not_working',
 '76': 'contactless_not_working',
 '8': 'transfer_not_received_by_recipient',
 '9': 'balance_not_updated_after_bank_transfer'}
```

In [14]:

```
for i in label_to_text.keys():
    if not label_to_text_1[str(i)] == label_to_text[i]:
        print('Error!')
```

Количество классов с тестовом и тренировачном наборе

In [15]:

```
df_train.category.nunique()
```

Out[15]:

```
77
```

In [16]:

```
df_test.category.nunique()
```

Out[16]:

```
77
```

Посмотрим сколько объектов каждого класса

In [ ]:

```
df_train.category.value_counts()
```

Out[ ]:

```
card_payment_fee_charged                          187
direct_debit_payment_not_recognised               182
balance_not_updated_after_cheque_or_cash_deposit  181
wrong_amount_of_cash_received                     180
cash_withdrawal_charge                            177
                                                 ...
lost_or_stolen_card                                82
card_swallowed                                     61
card_acceptance                                    59
virtual_card_not_working                           41
contactless_not_working                            35
Name: category, Length: 77, dtype: int64
```

Явный дисбаланс классов

In [ ]:

```
df_test.category.value_counts()
```

Out[ ]:

```
disposable_card_limits         40
exchange_charge                40
card_payment_not_recognised    40
passcode_forgotten             40
fiat_currency_support          40
                               ..
card_delivery_estimate         40
transfer_fee_charged           40
pending_top_up                 40
why_verify_identity            40
transfer_timing                40
Name: category, Length: 77, dtype: int64
```

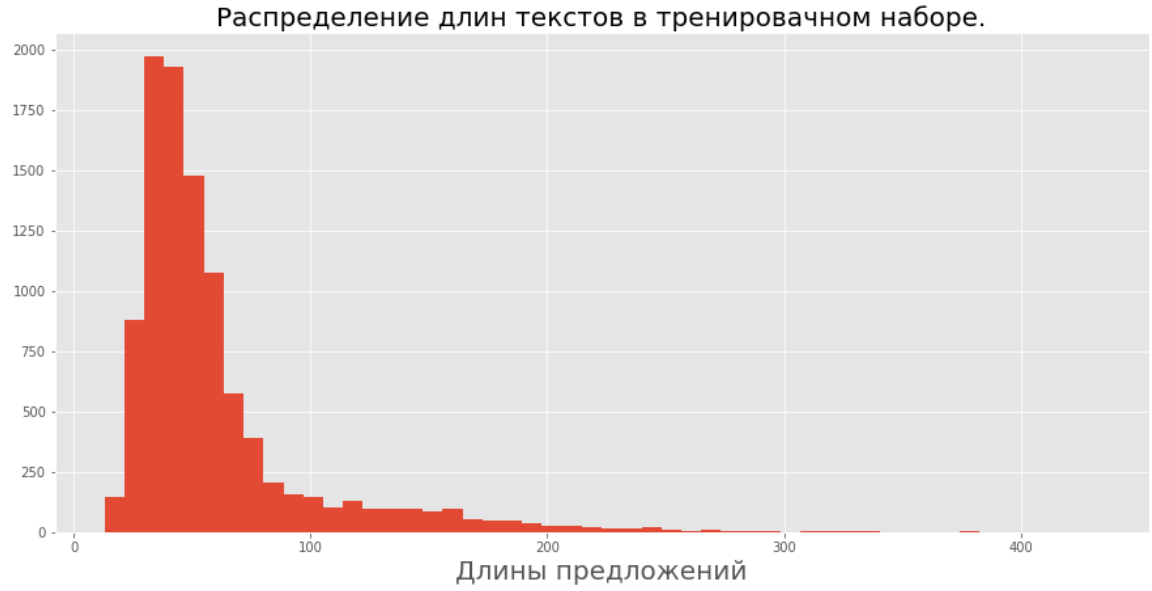
In [ ]:

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10003 entries, 0 to 10002
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   text      10003 non-null  object
 1   category  10003 non-null  object
dtypes: object(2)
memory usage: 156.4+ KB
```

In [ ]:

```
plt.figure(figsize=(15,7))
plt.title('Распределение длин текстов в тренировачном наборе.', fontsize=20)
df_train.text.apply(len).hist(bins=50)
plt.xlabel('Длины предложений', fontsize=20)
plt.show()
```

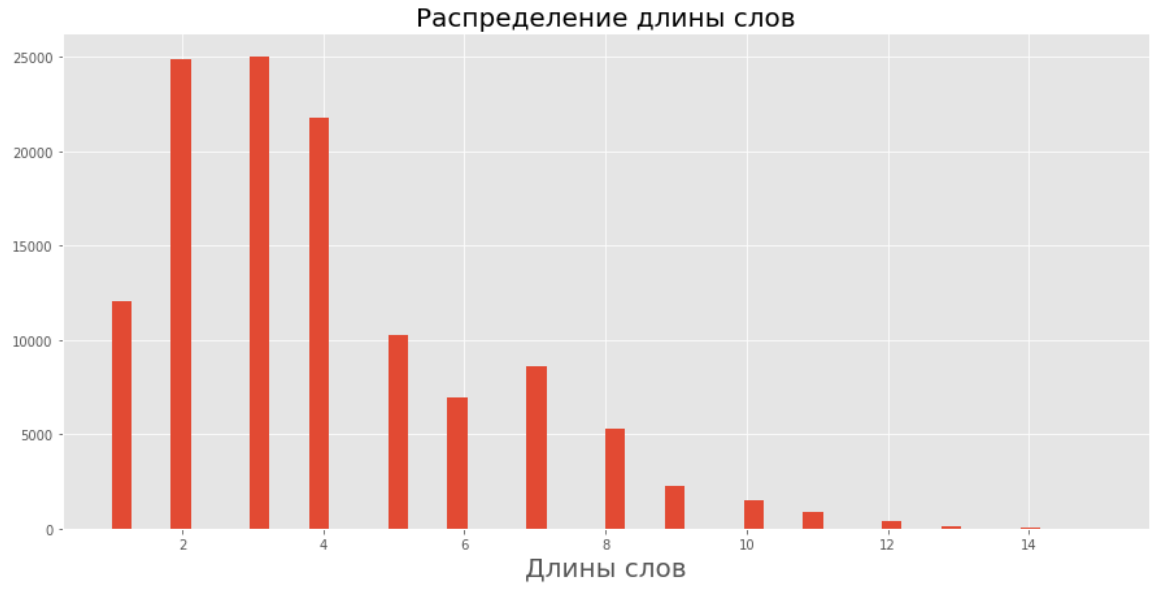



In [ ]:

```
dlin = list(map(len, ' '.join(df_train.text.str.lower().str.replace(r'[^a-z0-9\' ]',r'
 ')).split()))
```

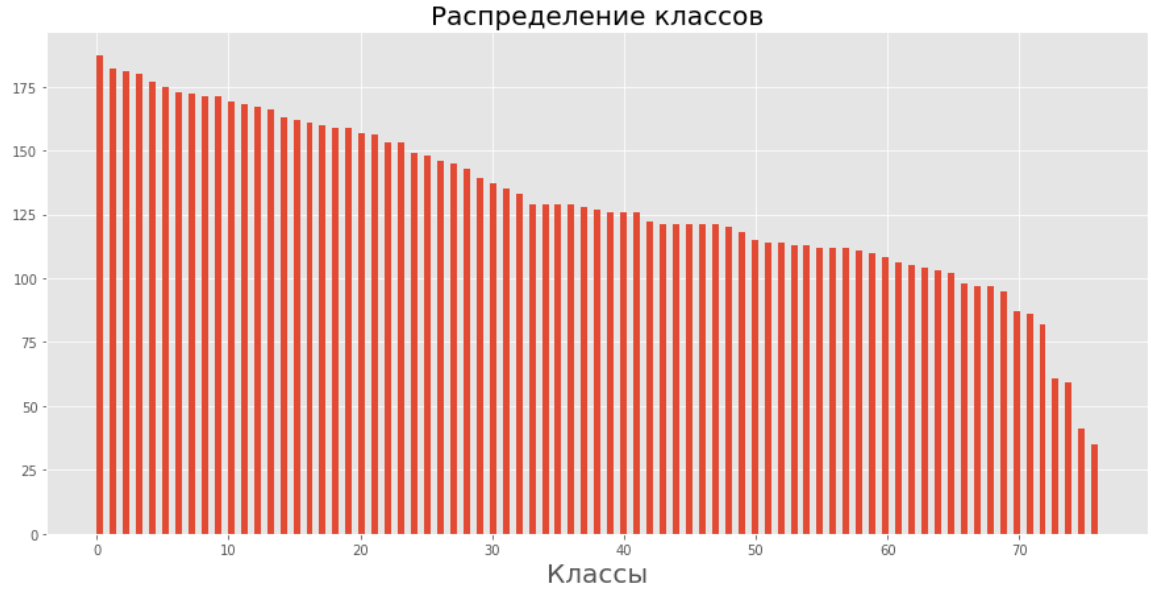
In [ ]:

```
plt.figure(figsize=(15,7))
plt.title('Распределение длины слов', fontsize=20)
plt.hist(dlin, bins = 50)
plt.xlabel('Длины слов', fontsize=20)
plt.show()
```



In [ ]:

```
plt.figure(figsize=(15,7))
plt.title('Распределение классов', fontsize=20)
y_train.hist(bins=y_train.nunique() * 2 - 1)
plt.xlabel('Классы', fontsize=20)
plt.show()
```



In [ ]:

```
from collections import Counter

Counter(' '.join(df_train.text.str.lower())).most_common()
```

Out[ ]:

```
[(' ', 120135),
 ('e', 53193),
 ('t', 43481),
 ('a', 41713),
 ('i', 34323),
 ('o', 33876),
 ('n', 33161),
 ('r', 25958),
 ('s', 23801),
 ('h', 23244),
 ('d', 21243),
 ('c', 19857),
 ('m', 16059),
 ('y', 14749),
 ('w', 13238),
 ('u', 13225),
 ('l', 12432),
 ('p', 11470),
 ('g', 9912),
 ('f', 8429),
 ('?', 6505),
 ('b', 4869),
 ('.', 4375),
 ('v', 4012),
 ('k', 3680),
 ("'", 2607),
 (',', 1564),
 ('x', 1345),
 ('-', 534),
 ('!', 383),
 ('j', 334),
 ('q', 322),
 ('0', 176),
 ('z', 173),
 ('1', 153),
 ('$', 87),
 ('2', 58),
 ('"', 42),
 ('£', 30),
 ('5', 28),
 ('3', 27),
 ('€', 21),
 ('8', 20),
 (';', 14),
 ('\n', 13),
 ('(', 10),
 ('/', 9),
 (')', 6),
 (':', 6),
 ('4', 4),
 ('6', 3),
 ('\xa0', 2),
 ('&', 2),
 ('|', 1),
 ('>', 1),
 ('…', 1),
 ('7', 1),
 ('%', 1)]
```

In [ ]:

```
Counter(' '.join(df_train.text.str.lower().str.replace(r'[^a-z0-9\' ]',r'')).split()).m
ost_common()
```

Out[ ]:

```
[('i', 8312),
 ('my', 5684),
 ('to', 4038),
 ('a', 3565),
 ('the', 3498),
 ('card', 2672),
 ('is', 2376),
 ('it', 1849),
 ('do', 1848),
 ('can', 1842),
 ('for', 1581),
 ('how', 1520),
 ('what', 1375),
 ('why', 1365),
 ('account', 1348),
 ('you', 1216),
 ('and', 1215),
 ('money', 1130),
 ('was', 1090),
 ('transfer', 1081),
 ('me', 1037),
 ('have', 1000),
 ('that', 997),
 ('in', 983),
 ('up', 969),
 ('not', 967),
 ('on', 878),
 ('there', 876),
 ('this', 825),
 ('get', 808),
 ('payment', 745),
 ('an', 722),
 ('but', 713),
 ('need', 698),
 ('cash', 690),
 ('of', 681),
 ('from', 665),
 ('be', 645),
 ('top', 614),
 ('with', 607),
 ('exchange', 549),
 ('are', 529),
 ('charged', 528),
 ('please', 514),
 ('when', 484),
 ('atm', 474),
 ('app', 469),
 ('am', 467),
 ('been', 457),
 ('fee', 456),
 ('will', 440),
 ('use', 427),
 ('pending', 423),
 ('if', 410),
 ('has', 409),
 ('where', 396),
 ('topup', 394),
 ('did', 388),
 ('help', 368),
```

```
    ('long', 361),
    ('made', 356),
    ('transaction', 354),
    ('still', 351),
    ('would', 347),
    ('pin', 343),
    ('take', 338),
    ('new', 334),
    ('rate', 333),
    ('make', 322),
    ("didn't", 314),
    ('wrong', 313),
    ('charge', 309),
    ('does', 304),
    ("don't", 303),
    ('know', 297),
    ('as', 294),
    ('through', 292),
    ('declined', 291),
    ('withdrawal', 288),
    ('cards', 287),
    ('refund', 287),
    ('want', 283),
    ("i'm", 276),
    ('out', 268),
    ('see', 268),
    ('like', 263),
    ('extra', 263),
    ('tell', 262),
    ('just', 261),
    ('yet', 260),
    ('at', 257),
    ('go', 256),
    ('tried', 256),
    ('one', 255),
    ('got', 253),
    ('using', 240),
    ('now', 237),
    ('some', 235),
    ('virtual', 227),
    ('work', 226),
    ("can't", 224),
    ('something', 222),
    ('identity', 219),
    ('any', 216),
    ('showing', 215),
    ('disposable', 215),
    ('received', 213),
    ("it's", 213),
    ('change', 212),
    ('going', 207),
    ('statement', 207),
    ('currencies', 205),
    ('about', 204),
    ('your', 202),
    ('amount', 202),
    ('should', 201),
    ('currency', 195),
    ('so', 192),
    ('funds', 190),
    ('check', 189),
```

```
    ('working', 189),
    ("hasn't", 186),
    ('pay', 179),
    ('bank', 178),
    ('ago', 175),
    ('think', 171),
    ('someone', 169),
    ('back', 168),
    ('possible', 165),
    ('getting', 164),
    ("what's", 163),
    ("i've", 163),
    ('or', 162),
    ("isn't", 158),
    ('into', 158),
    ('verify', 158),
    ('by', 157),
    ('find', 152),
    ('time', 152),
    ('cancel', 150),
    ('transfers', 149),
    ('more', 148),
    ('debit', 143),
    ('purchase', 141),
    ('much', 140),
    ('receive', 138),
    ('all', 137),
    ('sent', 135),
    ('used', 128),
    ('fees', 127),
    ('able', 125),
    ('withdraw', 125),
    ('times', 125),
    ('could', 123),
    ('days', 121),
    ('balance', 121),
    ('order', 120),
    ('which', 116),
    ('verification', 114),
    ('add', 113),
    ('before', 113),
    ('mastercard', 112),
    ('after', 110),
    ('phone', 108),
    ('than', 108),
    ('bought', 107),
    ('lost', 106),
    ('show', 106),
    ('direct', 106),
    ('reason', 105),
    ('another', 105),
    ('cheque', 105),
    ('way', 104),
    ('right', 104),
    ('being', 103),
    ('today', 102),
    ('item', 102),
    ('us', 102),
    ('activate', 101),
    ('come', 100),
    ('visa', 100),
```

```
    ('deposit', 99),
    ('details', 99),
    ('while', 98),
    ("haven't", 98),
    ('they', 98),
    ('sure', 97),
    ('trying', 96),
    ('many', 95),
    ('stolen', 94),
    ('payments', 94),
    ('had', 93),
    ('transactions', 93),
    ('shows', 92),
    ('credit', 91),
    ('already', 91),
    ('gone', 89),
    ('couple', 89),
    ('limit', 88),
    ('went', 87),
    ('1', 85),
    ('correct', 85),
    ('charges', 85),
    ('problem', 84),
    ('different', 84),
    ('because', 84),
    ('transferred', 84),
    ('physical', 82),
    ('number', 81),
    ('waiting', 80),
    ('here', 79),
    ('explain', 79),
    ('hi', 78),
    ('accept', 78),
    ('cost', 78),
    ("doesn't", 77),
    ('reverted', 77),
    ('process', 76),
    ('keep', 76),
    ('cancelled', 76),
    ('deposited', 76),
    ('morning', 75),
    ('give', 75),
    ("won't", 73),
    ('seems', 72),
    ('same', 71),
    ('noticed', 71),
    ('let', 71),
    ('put', 70),
    ('foreign', 70),
    ('earlier', 70),
    ('online', 70),
    ('topping', 70),
    ('twice', 70),
    ('apple', 70),
    ('send', 69),
    ('thought', 69),
    ('says', 69),
    ('only', 69),
    ('week', 68),
    ('recognize', 68),
    ('beneficiary', 68),
```

```
    ('passcode', 67),
    ('gbp', 66),
    ('last', 65),
    ('no', 65),
    ('additional', 65),
    ('seller', 65),
    ('two', 64),
    ('available', 64),
    ('few', 64),
    ('information', 63),
    ('hello', 63),
    ('complete', 63),
    ('happened', 62),
    ('never', 62),
    ('auto', 62),
    ('yesterday', 62),
    ('failed', 62),
    ('friend', 62),
    ('since', 61),
    ('requested', 61),
    ('uk', 61),
    ('error', 61),
    ('incorrect', 60),
    ('country', 60),
    ('checked', 60),
    ('really', 60),
    ("wasn't", 60),
    ('free', 60),
    ('access', 59),
    ('rates', 59),
    ('day', 59),
    ('who', 59),
    ('arrived', 58),
    ('wait', 57),
    ('code', 57),
    ('done', 56),
    ('countries', 56),
    ('too', 56),
    ('google', 55),
    ('abroad', 54),
    ('applied', 54),
    ('issue', 54),
    ('multiple', 54),
    ('found', 53),
    ('purchased', 53),
    ('machine', 53),
    ('other', 52),
    ('making', 52),
    ('seem', 52),
    ('support', 52),
    ('china', 52),
    ('soon', 51),
    ('atms', 51),
    ('paid', 51),
    ('came', 50),
    ('message', 50),
    ('its', 49),
    ('set', 49),
    ('though', 49),
    ('international', 49),
    ("there's", 49),
```

```
    ('wanted', 49),
    ('weeks', 48),
    ('over', 48),
    ('steps', 48),
    ('recently', 48),
    ('transferring', 48),
    ('unable', 48),
    ('saw', 47),
    ('delete', 47),
    ('source', 46),
    ('changed', 46),
    ('service', 46),
    ('keeps', 46),
    ('these', 46),
    ('taking', 46),
    ('express', 46),
    ('them', 45),
    ('several', 45),
    ('allowed', 45),
    ('everything', 45),
    ('were', 45),
    ('first', 45),
    ('without', 45),
    ('delivered', 44),
    ('look', 44),
    ('rejected', 44),
    ('american', 44),
    ('open', 44),
    ('link', 43),
    ("i'd", 43),
    ('pound', 43),
    ('accepted', 43),
    ('name', 43),
    ('may', 42),
    ('anything', 42),
    ('fix', 42),
    ('double', 42),
    ('second', 42),
    ('having', 41),
    ('completed', 41),
    ('mine', 41),
    ('exchanges', 40),
    ('even', 40),
    ('missing', 40),
    ('ordered', 39),
    ('shown', 39),
    ('doing', 39),
    ('mistake', 39),
    ('added', 39),
    ('age', 39),
    ('topped', 39),
    ('salary', 39),
    ('needs', 38),
    ('old', 37),
    ('exchanging', 37),
    ('recent', 37),
    ('immediately', 37),
    ('withdrew', 37),
    ('fiat', 37),
    ('cannot', 37),
    ('track', 36),
```

```
    ('delivery', 36),
    ('arrive', 36),
    ('update', 36),
    ('expect', 36),
    ('asked', 36),
    ('id', 36),
    ('hold', 35),
    ('option', 35),
    ('automatically', 35),
    ('wallet', 35),
    ('topups', 35),
    ('gave', 35),
    ('verified', 35),
    ('updated', 35),
    ('longer', 34),
    ('pounds', 34),
    ('off', 34),
    ('mean', 34),
    ('took', 34),
    ('worked', 34),
    ('blocked', 34),
    ('daughter', 34),
    ('else', 34),
    ('expire', 34),
    ('system', 33),
    ('buy', 33),
    ('very', 33),
    ('understand', 33),
    ('actually', 33),
    ('supported', 33),
    ('certain', 33),
    ('unblock', 33),
    ('supposed', 32),
    ('appears', 32),
    ('withdrawing', 32),
    ('place', 32),
    ('eu', 32),
    ('expires', 32),
    ('status', 31),
    ('believe', 31),
    ('remember', 31),
    ('happening', 31),
    ('dispute', 31),
    ('looks', 30),
    ('seen', 30),
    ('store', 30),
    ('verifying', 30),
    ('duplicate', 30),
    ('close', 29),
    ('less', 29),
    ('100', 29),
    ('stop', 29),
    ('european', 29),
    ('both', 29),
    ('europe', 29),
    ('merchant', 29),
    ('returned', 29),
    ('looking', 28),
    ('offer', 28),
    ('paying', 28),
    ('must', 28),
```

```
    ('needed', 28),
    ('seeing', 28),
    ('withdrawals', 28),
    ('request', 28),
    ('might', 28),
    ('limits', 28),
    ('reset', 28),
    ('address', 28),
    ('contactless', 28),
    ('personal', 28),
    ("shouldn't", 27),
    ('again', 27),
    ('figure', 27),
    ('once', 27),
    ('anywhere', 27),
    ('assist', 27),
    ('strange', 27),
    ('live', 27),
    ('ups', 27),
    ('restrictions', 27),
    ('usd', 27),
    ('hours', 27),
    ('disappeared', 27),
    ('date', 26),
    ('attempted', 26),
    ('crypto', 26),
    ('watch', 26),
    ('until', 25),
    ('reactivate', 25),
    ('thing', 25),
    ('freeze', 25),
    ('20', 25),
    ('password', 25),
    ('2', 24),
    ('coming', 24),
    ('appear', 24),
    ('between', 24),
    ('traveling', 24),
    ('receiving', 24),
    ('definitely', 24),
    ('allow', 24),
    ('adding', 24),
    ('services', 24),
    ('entered', 24),
    ('swift', 24),
    ('urgent', 24),
    ('checking', 24),
    ('friends', 24),
    ('refunded', 24),
    ('info', 23),
    ("couldn't", 23),
    ('return', 23),
    ('quickly', 23),
    ('restaurant', 23),
    ('anymore', 23),
    ("wouldn't", 23),
    ('we', 23),
    ('didnt', 23),
    ('sepa', 23),
    ('prove', 23),
    ('activating', 23),
```

```
    ('gotten', 22),
    ('wondering', 22),
    ('holiday', 22),
    ('purchases', 22),
    ('then', 22),
    ('provide', 22),
    ('holding', 22),
    ('flat', 22),
    ('customer', 22),
    ('try', 22),
    ('suddenly', 22),
    ('instead', 22),
    ('urgently', 22),
    ('moved', 22),
    ('past', 22),
    ('decline', 22),
    ('okay', 21),
    ("that's", 21),
    ('things', 21),
    ('nothing', 21),
    ('taken', 21),
    ('broken', 21),
    ('choose', 21),
    ('asap', 21),
    ('aud', 21),
    ('stuck', 21),
    ('hour', 21),
    ('rent', 21),
    ('recipient', 21),
    ('guys', 20),
    ('eur', 20),
    ('saying', 20),
    ('forgot', 20),
    ('30', 20),
    ('revert', 20),
    ('current', 19),
    ('also', 19),
    ('trouble', 19),
    ('sending', 19),
    ('accounts', 19),
    ('happens', 19),
    ('maximum', 18),
    ('given', 18),
    ('company', 18),
    ('processed', 18),
    ('kind', 18),
    ('fine', 18),
    ('happen', 18),
    ('stole', 18),
    ('person', 18),
    ('denied', 18),
    ('edit', 18),
    ('remove', 18),
    ('declining', 18),
    ('people', 17),
    ('authorize', 17),
    ('withdrawn', 17),
    ('fast', 17),
    ('children', 17),
    ('enough', 17),
    ('good', 17),
```

```
    ('approved', 17),
    ('10', 17),
    ('shopping', 17),
    ('month', 17),
    ('discount', 17),
    ('worried', 16),
    ('home', 16),
    ('overcharged', 16),
    ('away', 16),
    ('dont', 16),
    ('assistance', 16),
    ('said', 16),
    ('vacation', 16),
    ('required', 16),
    ('he', 16),
    ('myself', 16),
    ('large', 16),
    ('full', 16),
    ('france', 16),
    ('within', 16),
    ('identification', 16),
    ('activation', 16),
    ('expected', 15),
    ('next', 15),
    ('mail', 15),
    ('hey', 15),
    ('left', 15),
    ('actual', 15),
    ('deal', 15),
    ('type', 15),
    ('usually', 15),
    ('items', 15),
    ('normally', 15),
    ('bit', 15),
    ('works', 15),
    ('well', 15),
    ('via', 15),
    ('most', 15),
    ('reversed', 15),
    ('machines', 15),
    ('post', 15),
    ('always', 15),
    ('failing', 15),
    ('contacted', 15),
    ('receiver', 15),
    ('point', 14),
    ('interested', 14),
    ('specific', 14),
    ('saturday', 14),
    ('during', 14),
    ('exchanged', 14),
    ('listed', 14),
    ('cause', 14),
    ('clear', 14),
    ('types', 14),
    ('start', 14),
    ('own', 14),
    ('whats', 14),
    ('checks', 14),
    ('her', 14),
    ('im', 14),
```

```
    ('move', 14),
    ('per', 14),
    ('buying', 14),
    ('landlord', 14),
    ('recognise', 14),
    ('documents', 14),
    ('fail', 14),
    ('statements', 14),
    ('tracking', 13),
    ('policy', 13),
    ('interbank', 13),
    ('overseas', 13),
    ('looked', 13),
    ('under', 13),
    ('end', 13),
    ('every', 13),
    ('problems', 13),
    ('procedure', 13),
    ('kids', 13),
    ('locations', 13),
    ('resolve', 13),
    ('create', 13),
    ('reasons', 13),
    ('charging', 13),
    ('she', 13),
    ('reach', 13),
    ('however', 13),
    ('somebody', 13),
    ('unauthorized', 13),
    ('married', 13),
    ('hotel', 13),
    ('far', 13),
    ('told', 13),
    ('mortgage', 13),
    ('activated', 13),
    ('located', 12),
    ('apply', 12),
    ('bad', 12),
    ('product', 12),
    ('correctly', 12),
    ('odd', 12),
    ('pretty', 12),
    ('due', 12),
    ('normal', 12),
    ('list', 12),
    ('ok', 12),
    ('autotop', 12),
    ('places', 12),
    ('cant', 12),
    ('months', 12),
    ('important', 12),
    ('tomorrow', 12),
    ('canceled', 12),
    ('deducted', 12),
    ('5', 12),
    ('happy', 12),
    ('frequently', 12),
    ('500', 12),
    ('issued', 11),
    ('jacket', 11),
    ('their', 11),
```

```
    ('maybe', 11),
    ('idea', 11),
    ('dollar', 11),
    ('goes', 11),
    ('withdrawl', 11),
    ('options', 11),
    ('takes', 11),
    ('function', 11),
    ('accidentally', 11),
    ('stopped', 11),
    ('report', 11),
    ('permission', 11),
    ('aware', 11),
    ('reflect', 11),
    ('gas', 11),
    ('quite', 11),
    ('reverse', 11),
    ('kept', 11),
    ('shop', 11),
    ('business', 11),
    ('credited', 11),
    ('compromised', 11),
    ('notice', 11),
    ('expiration', 11),
    ('expired', 11),
    ('replacement', 10),
    ('progress', 10),
    ('those', 10),
    ('russian', 10),
    ('states', 10),
    ('cleared', 10),
    ('forever', 10),
    ('deliver', 10),
    ('frozen', 10),
    ('changing', 10),
    ('replace', 10),
    ('spain', 10),
    ('tries', 10),
    ('advise', 10),
    ('issues', 10),
    ('contact', 10),
    ('businesses', 10),
    ('fraudulent', 10),
    ('identify', 10),
    ('directly', 10),
    ('mugged', 10),
    ('down', 10),
    ('terminate', 10),
    ('frequent', 10),
    ('outside', 10),
    ('expecting', 9),
    ('travel', 9),
    ('official', 9),
    ('bill', 9),
    ('high', 9),
    ('higher', 9),
    ('convert', 9),
    ('placed', 9),
    ('typically', 9),
    ('concerned', 9),
    ('currently', 9),
```

```
    ('say', 9),
    ('security', 9),
    ('unblocked', 9),
    ("aren't", 9),
    ('giving', 9),
    ('showed', 9),
    ('frame', 9),
    ('prefer', 9),
    ('form', 9),
    ('notting', 9),
    ('hill', 9),
    ('duplicated', 9),
    ('standard', 9),
    ('expiring', 9),
    ("card's", 8),
    ('gets', 8),
    ('putting', 8),
    ('calculated', 8),
    ('ruble', 8),
    ('recall', 8),
    ('posted', 8),
    ('reflected', 8),
    ('acceptable', 8),
    ('ones', 8),
    ('low', 8),
    ('automatic', 8),
    ('feature', 8),
    ('lot', 8),
    ('necessary', 8),
    ('child', 8),
    ('18', 8),
    ('locked', 8),
    ('somewhere', 8),
    ('confused', 8),
    ('matter', 8),
    ('impression', 8),
    ('talk', 8),
    ('realize', 8),
    ('wish', 8),
    ('side', 8),
    ('family', 8),
    ('trace', 8),
    ('proof', 8),
    ('history', 8),
    ('modify', 8),
    ('require', 8),
    ('letting', 8),
    ('area', 8),
    ('fraud', 8),
    ('town', 8),
    ("atm's", 8),
    ('mind', 8),
    ('alright', 8),
    ('swallowed', 8),
    ('clearly', 8),
    ('previously', 7),
    ('reported', 7),
    ('travelling', 7),
    ('confirm', 7),
    ('regards', 7),
    ('small', 7),
```

```
    ('hidden', 7),
    ('causing', 7),
    ('price', 7),
    ('accurate', 7),
    ('removed', 7),
    ('solve', 7),
    ('future', 7),
    ('ordering', 7),
    ('intervals', 7),
    ('often', 7),
    ('locate', 7),
    ('wont', 7),
    ('block', 7),
    ('location', 7),
    ('regarding', 7),
    ('years', 7),
    ('log', 7),
    ('ate', 7),
    ('car', 7),
    ('faster', 7),
    ('minutes', 7),
    ('typo', 7),
    ('ask', 7),
    ('house', 7),
    ('appeared', 7),
    ('methods', 7),
    ("where's", 7),
    ('offered', 7),
    ('unsuccessful', 7),
    ('depositing', 7),
    ('unknown', 7),
    ('although', 7),
    ('passed', 7),
    ('proving', 7),
    ('accepting', 7),
    ('inform', 7),
    ('appearing', 7),
    ('five', 7),
    ('satisfied', 7),
    ('comes', 7),
    ('costs', 7),
    ('discounts', 7),
    ('play', 7),
    ('pulled', 6),
    ('purchasing', 6),
    ('feel', 6),
    ('question', 6),
    ('incorrectly', 6),
    ('each', 6),
    ('associated', 6),
    ('previous', 6),
    ('random', 6),
    ('city', 6),
    ('isnt', 6),
    ('hit', 6),
    ('onto', 6),
    ('heard', 6),
    ('quick', 6),
    ('united', 6),
    ('finding', 6),
    ('little', 6),
```

```
    ('provided', 6),
    ('attempts', 6),
    ('entering', 6),
    ('sort', 6),
    ('finish', 6),
    ('fixed', 6),
    ('half', 6),
    ('resolved', 6),
    ('standing', 6),
    ('rather', 6),
    ('thanks', 6),
    ('around', 6),
    ('attempting', 6),
    ('confirmed', 6),
    ('device', 6),
    ('accepts', 6),
    ('cheques', 6),
    ('perform', 6),
    ('unfamiliar', 6),
    ('weird', 6),
    ('results', 6),
    ('photos', 6),
    ('closest', 6),
    ('forgotten', 6),
    ('visit', 6),
    ('rejecting', 6),
    ('refused', 6),
    ('ive', 6),
    ('single', 6),
    ('local', 6),
    ('lesser', 6),
    ('employer', 6),
    ('alternatives', 6),
    ('operate', 6),
    ('linked', 5),
    ('enter', 5),
    ('thinking', 5),
    ('such', 5),
    ('hope', 5),
    ('terrible', 5),
    ('thank', 5),
    ("you're", 5),
    ('guess', 5),
    ('possibly', 5),
    ('unsure', 5),
    ('trip', 5),
    ('stays', 5),
    ('changes', 5),
    ('properly', 5),
    ('shipped', 5),
    ('policies', 5),
    ('estimated', 5),
    ('broke', 5),
    ('grocery', 5),
    ('deactivate', 5),
    ('x', 5),
    ('pick', 5),
    ('activity', 5),
    ('realized', 5),
    ('minimum', 5),
    ('requirements', 5),
```

```
    ('fund', 5),
    ('son', 5),
    ('older', 5),
    ('completing', 5),
    ('stay', 5),
    ('finished', 5),
    ('instant', 5),
    ('suppose', 5),
    ('according', 5),
    ('messed', 5),
    ('submitted', 5),
    ('extremely', 5),
    ('thru', 5),
    ('cancelling', 5),
    ('later', 5),
    ('upset', 5),
    ('max', 5),
    ('receipt', 5),
    ('needing', 5),
    ('numbers', 5),
    ('usa', 5),
    ('obtain', 5),
    ('submit', 5),
    ('increased', 5),
    ('almost', 5),
    ('unusual', 5),
    ('suspicious', 5),
    ('living', 5),
    ('ahead', 5),
    ('images', 5),
    ('readable', 5),
    ('resident', 5),
    ('switzerland', 5),
    ('economic', 5),
    ('active', 5),
    ('refunds', 5),
    ('attempt', 5),
    ('groceries', 5),
    ('2018', 5),
    ('dissatisfied', 5),
    ('inside', 5),
    ('exactly', 5),
    ('austria', 5),
    ('unexpected', 5),
    ('manage', 5),
    ('configure', 5),
    ('spare', 5),
    ('real', 5),
    ('known', 5),
    ('copy', 5),
    ('package', 4),
    ('recieve', 4),
    ('mailed', 4),
    ('finally', 4),
    ('wondered', 4),
    ('yours', 4),
    ('best', 4),
    ('basis', 4),
    ('determine', 4),
    ('determined', 4),
    ('disappointed', 4),
```

```
 ('rubles', 4),
 ('hoping', 4),
 ('better', 4),
 ('apparently', 4),
 ('banks', 4),
 ('spend', 4),
 ('nowhere', 4),
 ('billing', 4),
 ('frustrated', 4),
 ("someone's", 4),
 ('hacked', 4),
 ('case', 4),
 ('3', 4),
 ('latest', 4),
 ('following', 4),
 ('appreciate', 4),
 ('limitations', 4),
 ('euros', 4),
 ('length', 4),
 ('chose', 4),
 ('along', 4),
 ('run', 4),
 ('reaches', 4),
 ('setup', 4),
 ('ways', 4),
 ('turn', 4),
 ...]
```

# Часть 2 Baseline Классический ML (Случайный лес и логрегрессия)

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression


from sklearn.metrics import f1_score
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [ ]:

```
def train_classic(X_train, X_test, y_train, y_test):
    clf_l = LogisticRegression(random_state=42).fit(X_train, y_train)
    y_pred_l = clf_l.predict(X_test)
    f1_l = f1_score(y_test, y_pred_l, average='weighted')

    clf_r = RandomForestClassifier(random_state=42).fit(X_train.toarray(), y_train)
    y_pred_r = clf_r.predict(X_test.toarray())
    f1_r = f1_score(y_test, y_pred_r, average='weighted')

    print(f'F1 для логрегрессии    : {round(f1_l, 6)}')
    print(f'F1 для случайного леса : {round(f1_r, 6)}')
```

In [ ]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

In [ ]:

```
def count_vs_tfidf(X_train, X_test, y_train, y_test, param_tfidf= {}, param_count =
{}):
    set_seed() # для повторяемости экспериментов
    vectorizer = TfidfVectorizer(param_tfidf)
    X_train_tfidf = vectorizer.fit_transform(X_train)
    X_test_tfidf = vectorizer.transform(X_test)
    print('Tfidf')
    train_classic(X_train_tfidf, X_test_tfidf, y_train, y_test)
    print('---')
    vectorizer = CountVectorizer(param_count)
    X_train_count = vectorizer.fit_transform(X_train)
    X_test_count = vectorizer.transform(X_test)
    print('Count')
    train_classic(X_train_count, X_test_count, y_train, y_test)
```

Посмотрим как работаю алгоритмы с дефолтными параметрами и с сырым текстом

In [ ]:

```
count_vs_tfidf(X_train, X_test, y_train, y_test, param_tfidf={'lowercase':False}, param
_count={'lowercase':False})
```

```
Tfidf
F1 для логрегрессии    : 0.877251
F1 для случайного леса : 0.865825
---
Count
F1 для логрегрессии    : 0.889721
F1 для случайного леса : 0.866377
```

Возьмём нижний регистр

In [ ]:

```
count_vs_tfidf(X_train, X_test, y_train, y_test, param_tfidf={'lowercase':True}, param_
count={'lowercase':True})
```

```
Tfidf
F1 для логрегрессии    : 0.877251
F1 для случайного леса : 0.865825
---
Count
F1 для логрегрессии    : 0.889721
F1 для случайного леса : 0.866377
```

Удалим все не буквенные символы

In [ ]:

```
count_vs_tfidf(X_train.str.replace(r'[^a-z0-9\' ]',r' '),
               X_test.str.replace(r'[^a-z0-9\' ]',r' '),
               y_train, y_test, param_tfidf={'lowercase':True}, param_count={'lowercas
e':True})
```

```
Tfidf
F1 для логрегрессии    : 0.843179
F1 для случайного леса : 0.832573
---
Count
F1 для логрегрессии    : 0.86314
F1 для случайного леса : 0.832788
```

Удалим стопслова

In [ ]:

```
count_vs_tfidf(X_train, X_test,
               y_train, y_test,
               param_tfidf={'lowercase':True, 'stop_words':'English'},
               param_count={'lowercase':True, 'stop_words':'English'})
```

```
Tfidf
F1 для логрегрессии    : 0.877251
F1 для случайного леса : 0.865825
---
Count
F1 для логрегрессии    : 0.889721
F1 для случайного леса : 0.866377
```

Попробуем убрать редковстречающиеся символы и заменить все конструкции по типу you're -> you are

Заменим также знаки € и £ на $ и на всякий случай ещё обработаем корректировщиком ошибок

In [ ]:

```
import jamspell
corrector = jamspell.TSpellCorrector() # корректор опечаток
corrector.LoadLangModel('en.bin') # предобученную модель взял здесь
#                                 https://github.com/bakwc/JamSpell
```

Out[ ]:

```
True
```

Обрабатываем текст автокорректировщиком до применения регулярных выражений

In [17]:

```
X_train1 = X_train.apply(corrector.FixFragment)
X_test1 = X_test.apply(corrector.FixFragment)
```

In [18]:

```
X_train1 = X_train1.str.lower().str.replace("can't", 'can not').str.replace("didn't",
'did not').str.replace("hasn't", 'has not')\
.str.replace("won't", 'will not').str.replace("wasn't", 'was not').str.replace("isn't",
'is not').str.replace("doesn't", 'does not')\
.str.replace("haven't", 'have not').str.replace("don't", 'do not').str.replace("should
n't", 'should not')\
.str.replace("aren't", 'are not').str.replace("wouldn't", 'would not').str.replace("cou
ldn't", 'could not')\
.str.replace("i'm", 'i am').str.replace("i'd", 'i would').str.replace("it's", 'it is')\
.str.replace("what's", 'what is').str.replace("i'll", 'i will')\
.str.replace("there's", 'there is').str.replace("where's", 'where is').str.replace("tha
t's", 'that is')\
.str.replace("you're", 'you are').str.replace("they're", 'they are').str.replace("'ve",
' have').str.replace("\"", '\'')\
.str.replace('€','$').str.replace('£','$').str.replace(r'[^a-z0-9\'\?\!\.\,\- ]',r' ').
str.replace(r'\s{1,}',r' ')\
.str.replace(r'([^a-z0-9 ])',r' \1 ').str.strip()


X_test1 = X_test1.str.lower().str.replace("can't", 'can not').str.replace("didn't", 'di
d not').str.replace("hasn't", 'has not')\
.str.replace("won't", 'will not').str.replace("wasn't", 'was not').str.replace("isn't",
'is not').str.replace("doesn't", 'does not')\
.str.replace("haven't", 'have not').str.replace("don't", 'do not').str.replace("should
n't", 'should not')\
.str.replace("aren't", 'are not').str.replace("wouldn't", 'would not').str.replace("cou
ldn't", 'could not')\
.str.replace("i'm", 'i am').str.replace("i'd", 'i would').str.replace("it's", 'it is')\
.str.replace("what's", 'what is').str.replace("i'll", 'i will')\
.str.replace("there's", 'there is').str.replace("where's", 'where is').str.replace("tha
t's", 'that is')\
.str.replace("you're", 'you are').str.replace("they're", 'they are').str.replace("'ve",
' have').str.replace("\"", '\'')\
.str.replace('€','$').str.replace('£','$').str.replace(r'[^a-z0-9\'\?\!\.\,\- ]',r' ').
str.replace(r'\s{1,}',r' ')\
.str.replace(r'([^a-z0-9 ])',r' \1 ').str.strip()
```

И обрабатываем после, на случай если регулярки задели что-то важное

In [19]:

```
X_train1 = X_train1.apply(corrector.FixFragment)
X_test1 = X_test1.apply(corrector.FixFragment)
```

In [20]:

```
X_train1.to_csv('trainn.csv', index=False)
X_test1.to_csv('testt.csv', index=False)
```

In [ ]:

```
count_vs_tfidf(X_train1, X_test1,
               y_train, y_test,
               param_tfidf={'lowercase':True, },
               param_count={'lowercase':True, })
```

```
Tfidf
F1 для логрегрессии    : 0.879228
F1 для случайного леса : 0.862191
---
Count
F1 для логрегрессии    : 0.890145
F1 для случайного леса : 0.861347
```

Следующий этап, попробуем нормализовать (с помощью лем) текст через nltk и токенизировать его

In [ ]:

```
import nltk
nltk.download('punkt')
from nltk.corpus import brown
nltk.download('brown')
from nltk import wordnet, pos_tag
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
from nltk import WordNetLemmatizer
text = set([str(i).lower() for i in brown.words()])
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Unzipping corpora/brown.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
```

In [ ]:

```
def get_wordnet_pos(treebank_tag):
    my_switch = {
        'J': wordnet.wordnet.ADJ,
        'V': wordnet.wordnet.VERB,
        'N': wordnet.wordnet.NOUN,
        'R': wordnet.wordnet.ADV,
    }
    for key, item in my_switch.items():
        if treebank_tag.startswith(key):
            return item
    return wordnet.wordnet.NOUN

def my_lemmatizer(sent):
    lemmatizer = WordNetLemmatizer()
    tokenized_sent = sent.split()
    pos_tagged = [(word, get_wordnet_pos(tag))
                  for word, tag in pos_tag(tokenized_sent)]
    return ' '.join([lemmatizer.lemmatize(word, tag)
                     for word, tag in pos_tagged if word.isdigit() or lemmatizer.lemmati
ze(word, tag) in text] )
```

In [ ]:

```
count_vs_tfidf(X_train1.apply(my_lemmatizer), X_test1.apply(my_lemmatizer),
               y_train, y_test,
               param_tfidf={'lowercase':True,  'tokenizer': nltk.word_tokenize},
               param_count={'lowercase':True, 'tokenizer': nltk.word_tokenize})
```

```
Tfidf
F1 для логрегрессии    : 0.847969
F1 для случайного леса : 0.842592
---
Count
F1 для логрегрессии    : 0.861613
F1 для случайного леса : 0.841239
```

Итого можно сказать, что классический мл неплохо себя показал, достигнув 0.89 без настройки моделей

Теперь настало время нейронок

# Часть 3 Нейронка

(неудачная попытка со своей архитектурой)

Идея состоит в том чтобы обрабатывать текст посимвольно, по каждой отдельной букве

В качестве данных возьмём датасет на котором классические алгоритмы ML показали наилучшие метрики.

X_train1, X_test1

In [ ]:

```
import numpy as np
import torch
import torch.nn as nn
#from pathlib import Path
import torch.nn.functional as F
USE_CUDA = torch.cuda.is_available()
device = torch.device("cuda" if USE_CUDA else "cpu")
```

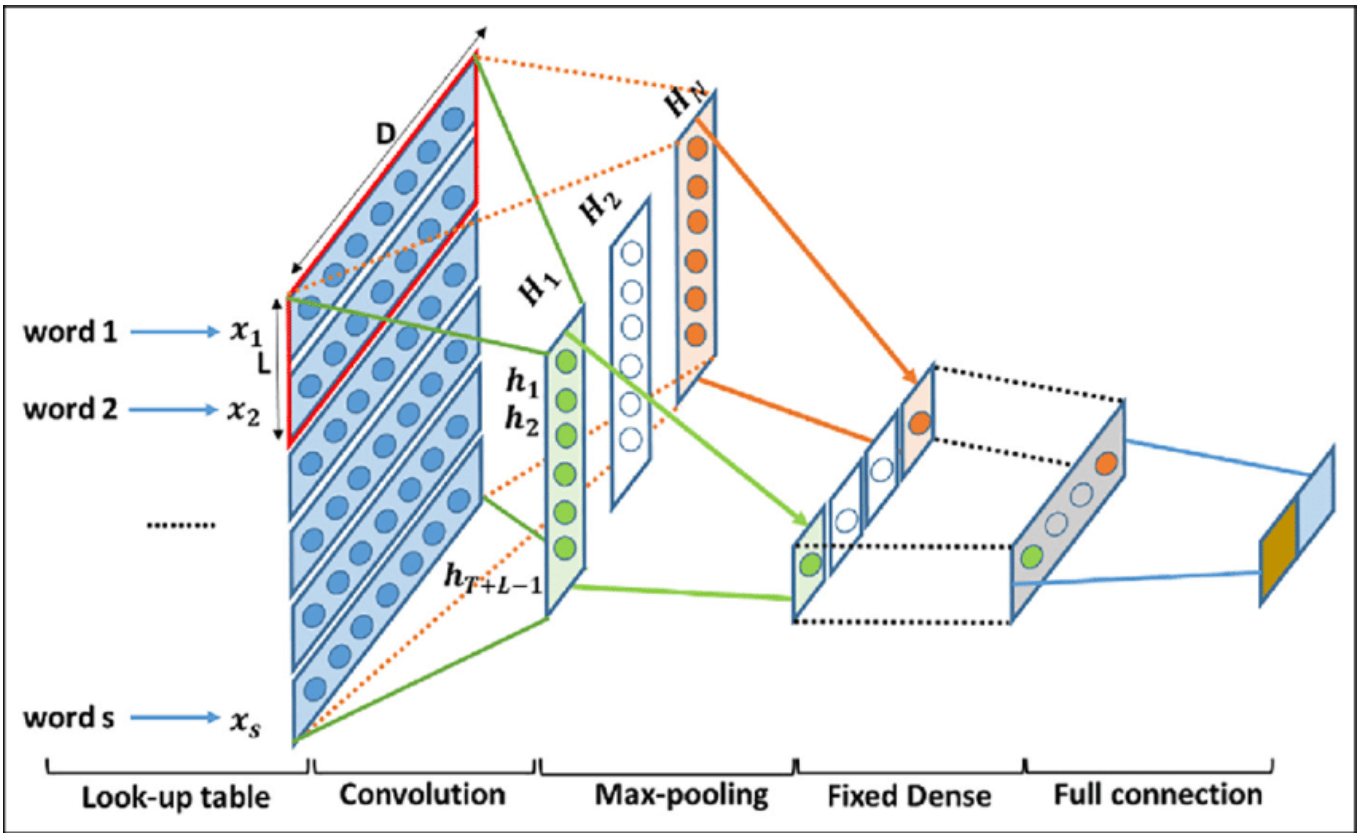
In [ ]:

```
class Vocab:
    def __init__(self, vocab, cur_len=100):
        self.cur_len = cur_len
        self.char2idx = {'<PAD>':0, '<UNK>':1, '<SOS>': 2, '<EOS>':3}         # ...
        y = vocab
        self.char2idx.update({j:i+4 for i,j in enumerate(y)})

    def tokenize(self, sequence, o = 1):
        l = len(sequence)
        return [1] + [self.char2idx[char] for char in sequence] + [3] + [0]*(self.cur_l
en - l)

    def __len__(self):
        return len(self.char2idx)
```

In [ ]:

```
class DataIter:

    def __init__(self, data, y, vocab1, batch_size=10, train = True, maxi = 100):

        self.y = y

        self.max = maxi
        if train:
            self.max = 2000
            #random.shuffle(data)
        self.size = len(data)
        self.vocab1 = vocab1

        self.data = data
        self.lens = [len(i) for i in data]

        self.num = 0
        self.batch_size = batch_size
        self.len = len(data)//batch_size

    def __len__(self):
        return self.max

    def __iter__(self):
        return self

    def __next__(self):
        if self.num <  self.max:
            tokens = np.random.randint(0, self.size, self.batch_size)
            sample1 = []
            target = []
            for i in tokens:
                target.append(self.y[i])
                sample1.append(self.vocab1.tokenize(self.data[i].lower()))
            sample1 = torch.LongTensor(sample1)
            target = torch.LongTensor(target)
            self.num += 1
            return sample1, target
        else:
            self.num = 0
            raise StopIteration
```

Я взял архитектуру сети как на картинке ниже и добавил ещё лстм, в надежде что это добавит в эмбединги информацию о связи между символами


D
$H_n$
$H_2$
$H_1$
word 1
$x_1$
L
word 2
$x_2$
.....
$h_1$
$h_2$
$h_{T+L-1}$
word s
$x_s$
Look-up table
Convolution
Max-pooling
Fixed Dense
Full connection

In [ ]:

```
class CNN(nn.Module):

    def __init__(self, size_voc):
        super().__init__()
        self.emb = nn.Embedding(size_voc, 128)
        #self.lin = nn.Linear(128, 128)#
        self.ls = nn.LSTM(128, 128, num_layers=3, batch_first=True, bidirectional=True)
        self.conv_0 = nn.Conv1d(in_channels = 1, out_channels = 64, kernel_size=(5, 256
))
        self.conv_1 = nn.Conv1d(in_channels = 1, out_channels = 64, kernel_size=(8, 256
))
        self.conv_2 = nn.Conv1d(in_channels = 1, out_channels = 64 , kernel_size=(10, 2
56))
        self.linear = nn.Linear(3*64, 77)#
        self.dropout = nn.Dropout(0.1)

    def forward(self, x):
        #x = self.lin(x)
        x,_ = self.ls(self.emb(x))
        embedded = x.unsqueeze(1)
        conved_0 = F.relu(self.conv_0(embedded.squeeze(3)))
        conved_1 = F.relu(self.conv_1(embedded.squeeze(3)))
        conved_2 = F.relu(self.conv_2(embedded.squeeze(3)))
        #print(conved_0.shape)
        #print(conved_0.squeeze(3).shape)
        pooled_0 = F.max_pool1d(conved_0.squeeze(3), conved_0.shape[2]).squeeze(2)
        pooled_1 = F.max_pool1d(conved_1.squeeze(3), conved_1.shape[2]).squeeze(2)
        pooled_2 = F.max_pool1d(conved_2.squeeze(3), conved_2.shape[2]).squeeze(2)
        out = self.dropout(torch.cat((pooled_0, pooled_1, pooled_2), dim = 1))
        out = self.linear(out)
        return torch.softmax(out, dim = 1)
```

Ограничим данные максимальной длинной предложения в 100 символов

In [ ]:

```
mask = [True if len(i)<100 else False for i in X_train1]
X_train2 = X_train1[mask].str.lower()
v1 = set(' '.join(X_train2.str.lower()))
vocab = Vocab(v1)
XX = DataIter(X_train2.values, y_train[mask].values, vocab)
```

In [ ]:

```
for i, j in XX:
    break
```

In [ ]:

```
model = CNN(len(vocab)).to(device)
```

In [ ]:

```
nn.CrossEntropyLoss()(torch.softmax(model(i.to(device)), dim=1), j.to(device))
```

Out[ ]:

```
tensor(4.3438, device='cuda:0', grad_fn=<NllLossBackward>)
```

In [ ]:

```
def f1_score_my(y_true, y_pred):
    y_t = y_true.cpu()
    y_p = y_pred.cpu()
    return f1_score(y_t, y_p, average='weighted')
```

Проверим что сетка учится на одном баче

In [ ]:

```
epochs = 20
lr = 0.0001
loss_func = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=lr)
for epoch in range(1, epochs+1):


    output = model(i.to(device))


    loss = loss_func(output , j.to(device))
    losses = float(loss)


    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    output = output.argmax(dim=1)
    f1 = f1_score_my(j, output)
    #f1.append(float(loss))

    print('F1: ',float(f1))
    print('Loss: ',float(losses))
    print('')
```

```
F1:  0.0
Loss:  4.3438262939453125

F1:  0.0
Loss:  4.343693733215332

F1:  0.06666666666666667
Loss:  4.343463897705078

F1:  0.02222222222222222
Loss:  4.343306541442871

F1:  0.03333333333333334
Loss:  4.343111038208008

F1:  0.01818181818181818
Loss:  4.342913627624512

F1:  0.019999999999999997
Loss:  4.342686653137207

F1:  0.08888888888888888
Loss:  4.342524528503418

F1:  0.02222222222222222
Loss:  4.3422698974609375

F1:  0.12
Loss:  4.342090606689453

F1:  0.01818181818181818
Loss:  4.341738700866699

F1:  0.019999999999999997
Loss:  4.341422080993652

F1:  0.0
Loss:  4.3411760330200195

F1:  0.0
Loss:  4.340442180633545

F1:  0.03333333333333334
Loss:  4.340092182159424

F1:  0.08
Loss:  4.339285850524902

F1:  0.025
Loss:  4.338679313659668

F1:  0.1
Loss:  4.337846755981445

F1:  0.04
Loss:  4.337375640869141

F1:  0.01818181818181818
Loss:  4.335618019104004
```

Вроде прогресс есть, теперь попробуем весь тренировочный набор

In [ ]:

```
from tqdm.notebook import trange, tqdm
```

In [ ]:

```
set_seed()
epochs = 5
lr = 0.001
loss_func = nn.CrossEntropyLoss()
model = CNN(len(vocab)).to(device)
mask = [True if len(i)<100 else False for i in X_train1]
X_train2 = X_train1[mask].str.lower()
v1 = set(' '.join(X_train2.str.lower()))
vocab = Vocab(v1)
XX = DataIter(X_train2.values, y_train[mask].values, vocab)


for epoch in trange(1, epochs+1):
    optimizer = torch.optim.Adam(model.parameters(), lr=lr)
    #lr/=1.2

    #Обучение
    model.train()
    f1 = []

    losses = []
    test_losses = []
    for batch in tqdm(XX, total=len(XX), leave = False):

        sample1, target = batch
        output = model(sample1.to(device))


        loss = loss_func(output , target.to(device))
        losses.append(float(loss))

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        output = output.argmax(dim=1)
        loss = f1_score_my(target, output)
        f1.append(float(loss))
    print(f'Epoch {epoch} | F1 train: {round(sum(f1)/len(f1),3)} | loss train: {round(s
um(losses)/len(losses),3)}' )
    print('')
```

```
Epoch 1 | F1 train: 0.005 | loss train: 4.342
```

```
Epoch 2 | F1 train: 0.021 | loss train: 4.323
```

```
Epoch 3 | F1 train: 0.059 | loss train: 4.286
```

```
Epoch 4 | F1 train: 0.106 | loss train: 4.24
```

```
Epoch 5 | F1 train: 0.198 | loss train: 4.148
```

Опыт со своей архитектурой завершился неудачей, в невошедших экспериментах я не смог добиться качества лучше 0.4 F1

Ничего не остаётся как пойти и взять берт и решить задачу

# Часть 4 Bert

Функции для обучения

In [21]:

```
def step_opt(loss, opt):
    opt.zero_grad()
    loss.backward()
    opt.step()


def pas_batch(model, batch, criterion, train, opt):
    input, mask, labels = batch
    logits = model(input, mask).logits
    loss = criterion(logits, labels)
    y_true = labels.cpu().tolist()
    y_pred = logits.argmax(dim=-1).cpu().tolist()
    losses = loss.item()
    if train:
        step_opt(loss, opt)
    del input, mask, labels
    return losses, y_true, y_pred


def pas_epoch(model, loader, criterion, train = False, opt = None):
    model.train() if train else model.eval()
    losses, y_true, y_pred = 0, [], []

    for batch in loader:


        losses_batch, y_true_batch, y_pred_batch  = pas_batch(model, batch, criterion,
train = train, opt = opt)
        losses += losses_batch
        y_true.extend(y_true_batch)
        y_pred.extend(y_pred_batch)

    f1 = f1_score(y_true, y_pred, average='weighted')
    losses = losses / len(loader)
    return losses, f1


def get_stat(model, loader, opt, criterion, train = True):
    if train:
        return pas_epoch(model, loader, criterion, train = True, opt = opt)
    else:
        with torch.no_grad():
            return pas_epoch(model, loader, criterion, train = False, opt = None)


def show_metrics(loss_train, loss_valid, f1_train, f1_valid):
    fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, sharex=True,
                                   figsize=(12, 6))

    ax1.set_title('Loss')
    ax1.plot(loss_train, 'o-', label='train loss', linewidth=5, markersize=12)
    ax1.plot(loss_valid, 'o-', label='valid loss', linewidth=5, markersize=12)
    ax1.legend()

    ax2.set_title('F1')
    ax2.plot(f1_train, 'o-', label='train f1', linewidth=5, markersize=12)
    ax2.plot(f1_valid, 'o-', label='valid f1', linewidth=5, markersize=12)
    ax2.legend()
    fig.suptitle('Графики лоса и метрики F1')
```

```
    plt.show()


def train(model, loader, loader_val, opt, crit, device=device, n_epochs=20):
    set_seed()
    loss_train, loss_valid = [], []
    f1_train, f1_valid = [], []

    for n in range(1, n_epochs + 1):



        loss_train_epoch, f1_train_epoch = get_stat(model, loader, opt, criterion, trai
n = True)


        loss_train.append(loss_train_epoch)
        f1_train.append(f1_train_epoch)

        loss_valid_epoch, f1_valid_epoch = get_stat(model, loader, opt, criterion, trai
n = False)
        clear_output(True)
        print(f'{n} epoch')
        print(f'    f1_train: {f1_train_epoch:.4f} | loss_train: {loss_train_epoch:.4
f}')
        print(f'    f1_valid: {f1_valid_epoch:.4f} | loss_valid: {loss_valid_epoch:.4
f}')
        loss_valid.append(loss_valid_epoch)
        f1_valid.append(f1_valid_epoch)

        show_metrics(loss_train, loss_valid, f1_train, f1_valid)

        #torch.save(model.state_dict(), f'/content/my_model_epoch_{n}.pth')

    return loss_train, loss_valid, f1_train, f1_valid
```

In [22]:

```
X_train1 = pd.read_csv('/content/trainn.csv').text
X_test1 = pd.read_csv('/content/testt.csv').text
```

Даталоадер с важностью равномерного сэмплинга классов.

In [23]:

```
class DataIter:

    def __init__(self, text, label, tokenizer, device, batch_size=128, train = True, ma
ke_balance = False):
        set_seed()
        self.label = np.array(label)
        self.tokenizer = tokenizer
        self.text = np.array(text)
        self.n = len(self.label)
        self.index = np.arange(self.n)
        self.make_balance = make_balance
        if make_balance:
            self.train = False
            self.d = {}
            for i in range(77):
                self.d[i] = []
            for i, j in enumerate(label):
                self.d[j].append(i)
            self.max = self.n//154
        else:
            self.train = train
            self.max = self.n//batch_size
        if self.train:
            np.random.shuffle(self.index)

        self.device = device
        self.num = 0
        self.batch_size = batch_size


    def __len__(self):
        return self.max

    def __iter__(self):
        return self

    def __next__(self):
        if not self.make_balance and self.num <  self.max:
            ind = self.index[self.batch_size * self.num : self.batch_size * (self.num +
1)]
            tokens = self.tokenizer(self.text[ind].tolist(), return_tensors='pt', paddi
ng=True)
            label = torch.LongTensor(self.label[ind]).to(self.device)
            input = tokens.input_ids.to(self.device)
            mask = tokens.attention_mask.to(self.device)
            self.num += 1
            return input, mask, label
        elif self.make_balance and self.num <  self.max:
            ind = np.array([])
            for i in self.d.keys():
                ind = np.append(ind, np.random.choice(self.d[i], 2))
            ind = ind.astype('int64')
            tokens = self.tokenizer(self.text[ind].tolist(), return_tensors='pt', paddi
ng=True)
            label = torch.LongTensor(self.label[ind]).to(self.device)
            input = tokens.input_ids.to(self.device)
            mask = tokens.attention_mask.to(self.device)
            self.num += 1
            return input, mask, label
```

```
        else:
            self.num = 0
            if not self.make_balance and self.train:
                np.random.shuffle(self.index)
            raise StopIteration
```

Переключатель градиентов берта

In [24]:

```
def switch_grad_bert(model, grad_on = True):
    for param in model.bert.parameters():
            param.requires_grad = grad_on
```

Итак основная идея состоит в том, что берём предобученный берт с классификатором.

Замораживаем веса берта и треним классификатор. Как только классификатор выйдет на плато,

то включаем веса берта и начинаем тренировать их с маленьким шагом.

Потом снова выключаем градиенты берта, треним классификатор, включаем градиенты и треним до убоя.

In [25]:

```
set_seed()
weight2 = [w if w<2 else 2 for w in weight]
weight2 = torch.tensor(weight2).to(device) # на будущее
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
train_loader = DataIter(X_train1, y_train, tokenizer, device, batch_size=64, train = Tr
ue, make_balance = False)
valid_loader = DataIter(X_test1, y_test, tokenizer, device, batch_size=128, train = Fal
se, make_balance = False)
criterion = nn.CrossEntropyLoss(weight = weight2) #
```

In [26]:

```
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=7
7)
model.to(device)
print(1)
```

```
Some weights of the model checkpoint at bert-base-uncased were not used wh
en initializing BertForSequenceClassification: ['cls.seq_relationship.bia
s', 'cls.predictions.transform.LayerNorm.weight', 'cls.seq_relationship.we
ight', 'cls.predictions.transform.LayerNorm.bias', 'cls.predictions.transf
orm.dense.bias', 'cls.predictions.bias', 'cls.predictions.transform.dense.
weight', 'cls.predictions.decoder.weight']
- This IS expected if you are initializing BertForSequenceClassification f
rom the checkpoint of a model trained on another task or with another arch
itecture (e.g. initializing a BertForSequenceClassification model from a B
ertForPreTraining model).
- This IS NOT expected if you are initializing BertForSequenceClassificati
on from the checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a BertForSequence
Classification model).
Some weights of BertForSequenceClassification were not initialized from th
e model checkpoint at bert-base-uncased and are newly initialized: ['class
ifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to u
se it for predictions and inference.
```

```
1
```

In [ ]:

```

```

In [ ]:

```
loss_train, loss_valid, f1_train, f1_valid = [], [], [], []
```

# Эксперимент 4.1

Обычный сэмплер и лосс с весами для корректировки дизбаланса классов

Чтобы компенсировать дизбаланс классов, накидываем весов на лосс.

In [ ]:

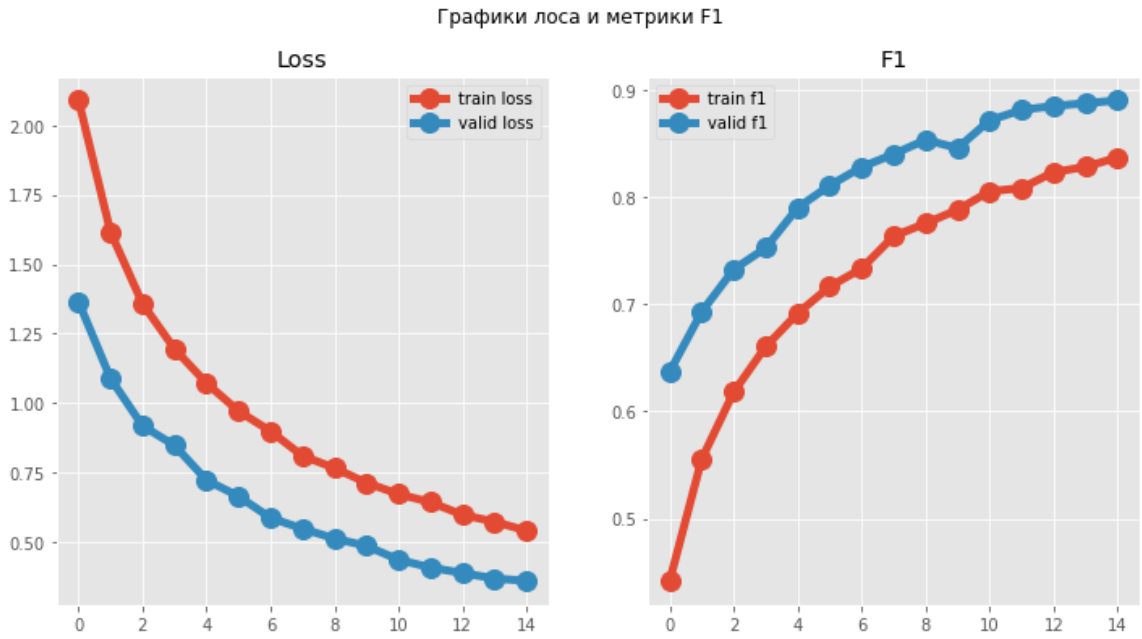
```
switch_grad_bert(model, grad_on = False)
set_seed()
optimizer = torch.optim.Adam(model.classifier.parameters(), lr=0.004)
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
            train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=1
5)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

```
15 epoch
    f1_train: 0.3311 | loss_train: 2.6227
    f1_valid: 0.4501 | loss_valid: 2.1495
```

In [ ]:

```
switch_grad_bert(model, grad_on = True)
set_seed()
optimizer = torch.optim.Adam([{'params':model.bert.parameters(), 'lr':3e-7},
                              {'params':model.classifier.parameters(), 'lr':0.002}])
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
           train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=1
5)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

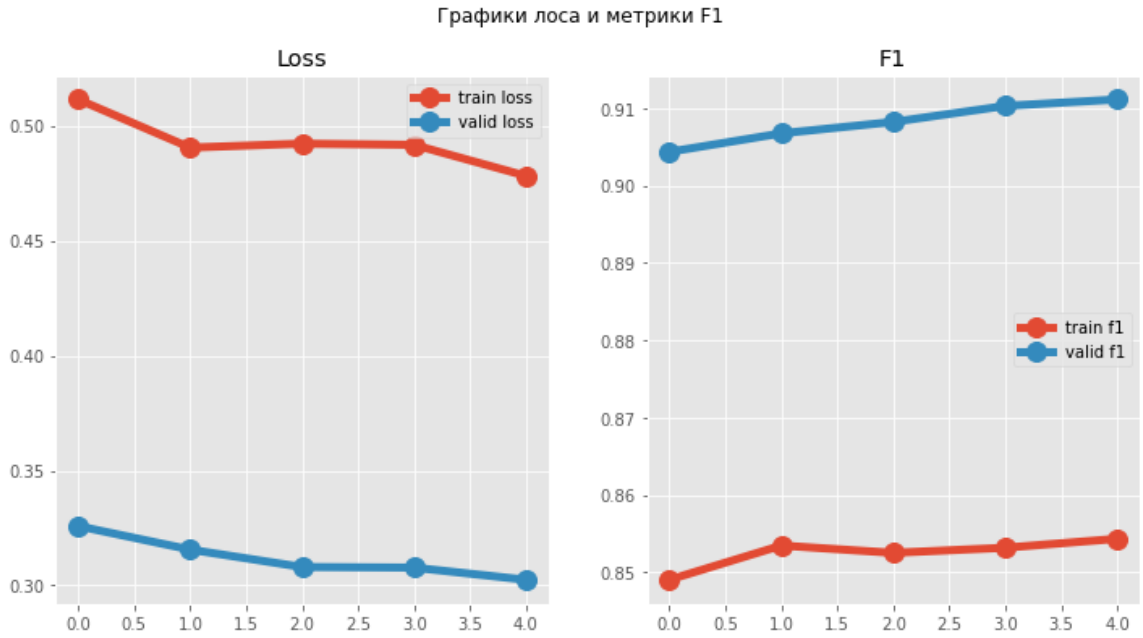
```
15 epoch
    f1_train: 0.8371 | loss_train: 0.5397
    f1_valid: 0.8903 | loss_valid: 0.3592
```



In [ ]:

```
switch_grad_bert(model, grad_on = False)
set_seed()
optimizer = torch.optim.Adam(model.classifier.parameters(), lr=0.001)
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
            train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=5
)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

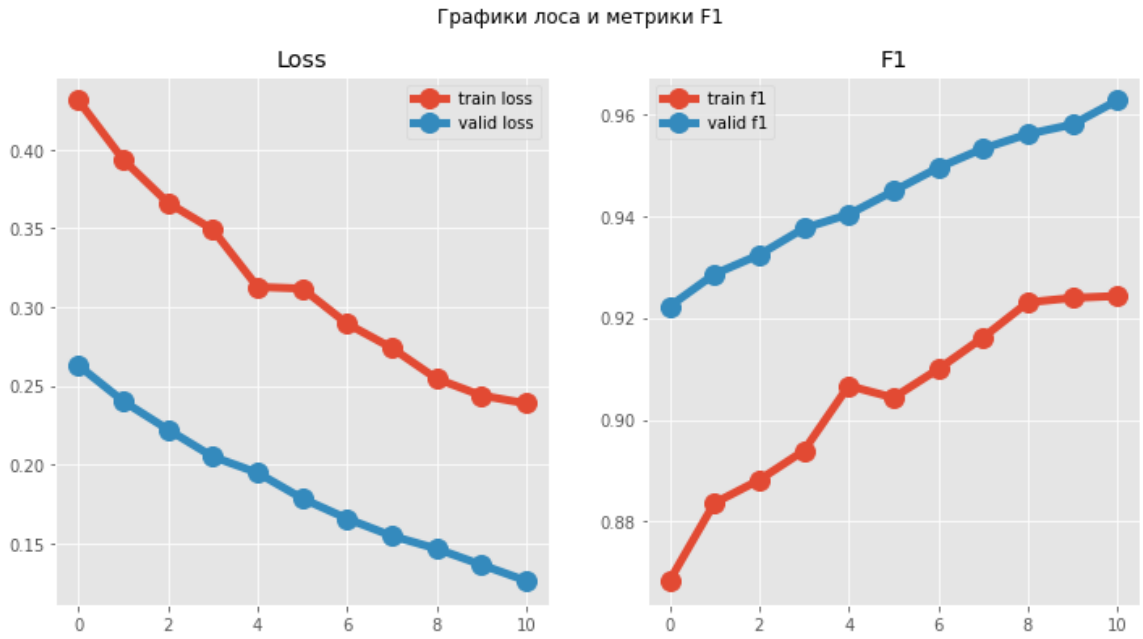
```
5 epoch
    f1_train: 0.8543 | loss_train: 0.4781
    f1_valid: 0.9111 | loss_valid: 0.3024
```



In [ ]:

```
switch_grad_bert(model, grad_on = True)
set_seed()
optimizer = torch.optim.Adam([{'params':model.bert.parameters(), 'lr':1e-6},
                              {'params':model.classifier.parameters(), 'lr':0.0001}])
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
           train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=1
1)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

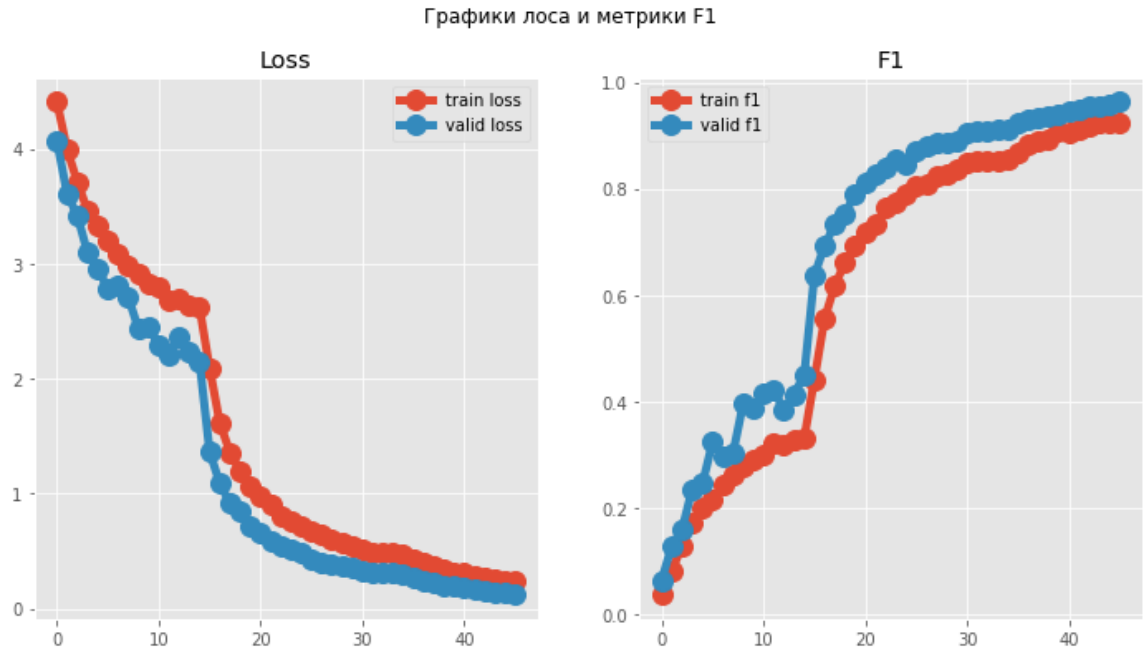
```
11 epoch
    f1_train: 0.9243 | loss_train: 0.2388
    f1_valid: 0.9627 | loss_valid: 0.1262
```



In [ ]:

```
print(f'Лучшее значение F1 на трейне: {max(f1_train):.4f} | на валидации: {max(f1_valid):.4f}')
show_metrics(loss_train, loss_valid, f1_train, f1_valid)
```

```
Лучшее значение F1 на трейне: 0.9243 | на валидации: 0.9627
```




Отличный результат, кажется немного странным, что всё время показатели на валидации были лучше чем на трейне, но вроде ошибок нигде нет и модель не обучается на валидационных данных, так что воспримем это как некоторая особенность модели.

Кроме в модели не полностью реализован потенциал и даже не наблюдается признаков переобучения, но у меня просто поджимали сроки, поэтому мне пришлось остановиться на этом результате.

In [ ]:

```
torch.save(model.state_dict(), '/content/drive/MyDrive/data_for_tinkoff/my_model_loss_with_weight.pt')
```

In [ ]:

```
model.load_state_dict(torch.load('/content/drive/MyDrive/data_for_tinkoff/my_model_loss_with_weight.pt'))
```

# Эксперимент 4.2

Обычный сэмплер и лосс без весов

In [ ]:

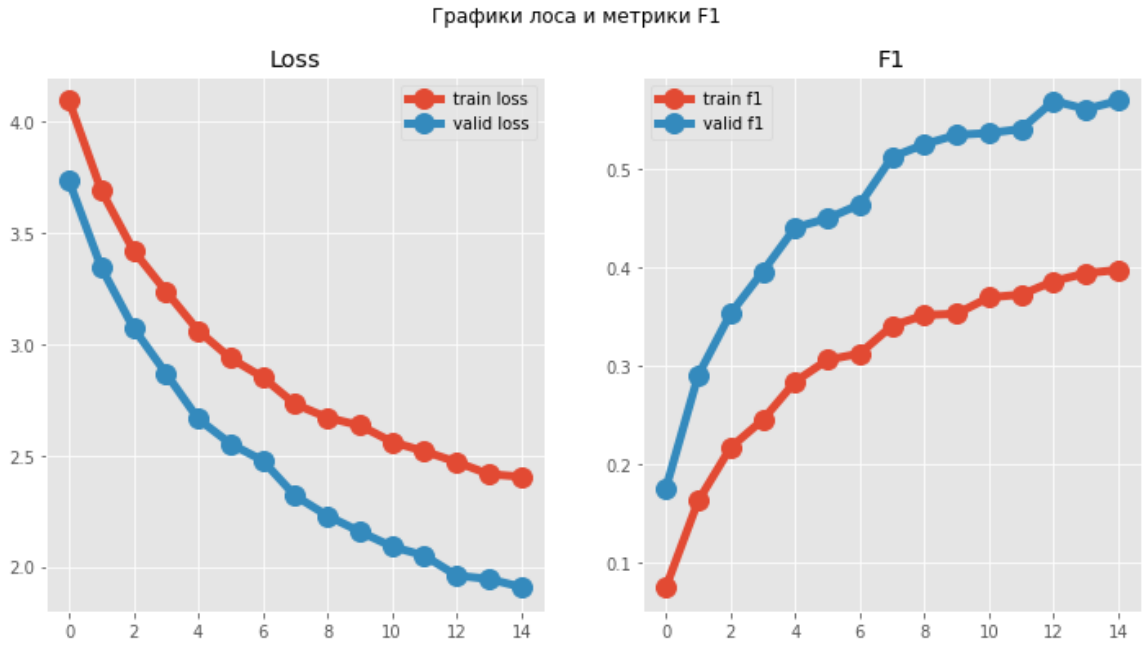
```
set_seed()
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=7
7)
model.to(device)

criterion = nn.CrossEntropyLoss() #
loss_train, loss_valid, f1_train, f1_valid = [], [], [], []
```

```
Some weights of the model checkpoint at bert-base-uncased were not used wh
en initializing BertForSequenceClassification: ['cls.predictions.decoder.w
eight', 'cls.predictions.transform.dense.bias', 'cls.predictions.transfor
m.LayerNorm.bias', 'cls.predictions.transform.LayerNorm.weight', 'cls.seq_
relationship.weight', 'cls.predictions.bias', 'cls.seq_relationship.bias',
'cls.predictions.transform.dense.weight']
- This IS expected if you are initializing BertForSequenceClassification f
rom the checkpoint of a model trained on another task or with another arch
itecture (e.g. initializing a BertForSequenceClassification model from a B
ertForPreTraining model).
- This IS NOT expected if you are initializing BertForSequenceClassificati
on from the checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a BertForSequence
Classification model).
Some weights of BertForSequenceClassification were not initialized from th
e model checkpoint at bert-base-uncased and are newly initialized: ['class
ifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to u
se it for predictions and inference.
```

In [ ]:

```
switch_grad_bert(model, grad_on = False)
set_seed()
optimizer = torch.optim.Adam(model.classifier.parameters(), lr=0.004)
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
            train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=1
5)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

```
15 epoch
    f1_train: 0.3970 | loss_train: 2.4053
    f1_valid: 0.5689 | loss_valid: 1.9098
```



In [ ]:

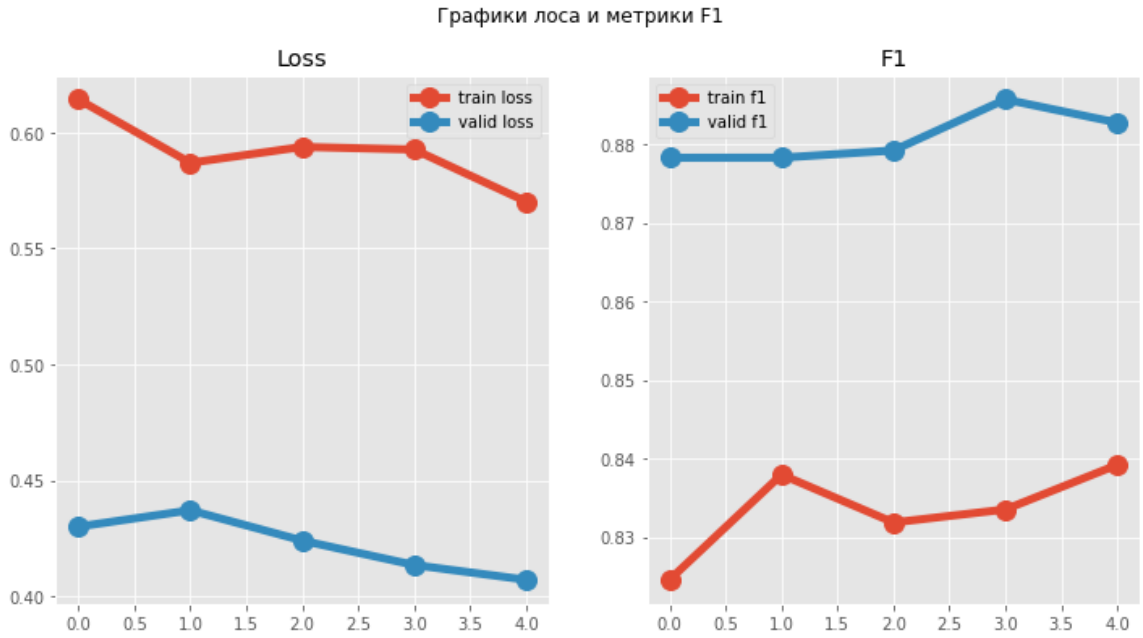
```
switch_grad_bert(model, grad_on = True)
set_seed()
optimizer = torch.optim.Adam([{'params':model.bert.parameters(), 'lr':3e-7},
                              {'params':model.classifier.parameters(), 'lr':0.002}])
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
           train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=1
5)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

```
15 epoch
    f1_train: 0.8175 | loss_train: 0.6470
    f1_valid: 0.8681 | loss_valid: 0.4673
```

In [ ]:

```
switch_grad_bert(model, grad_on = False)
set_seed()
optimizer = torch.optim.Adam(model.classifier.parameters(), lr=0.001)
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
            train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=5
)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

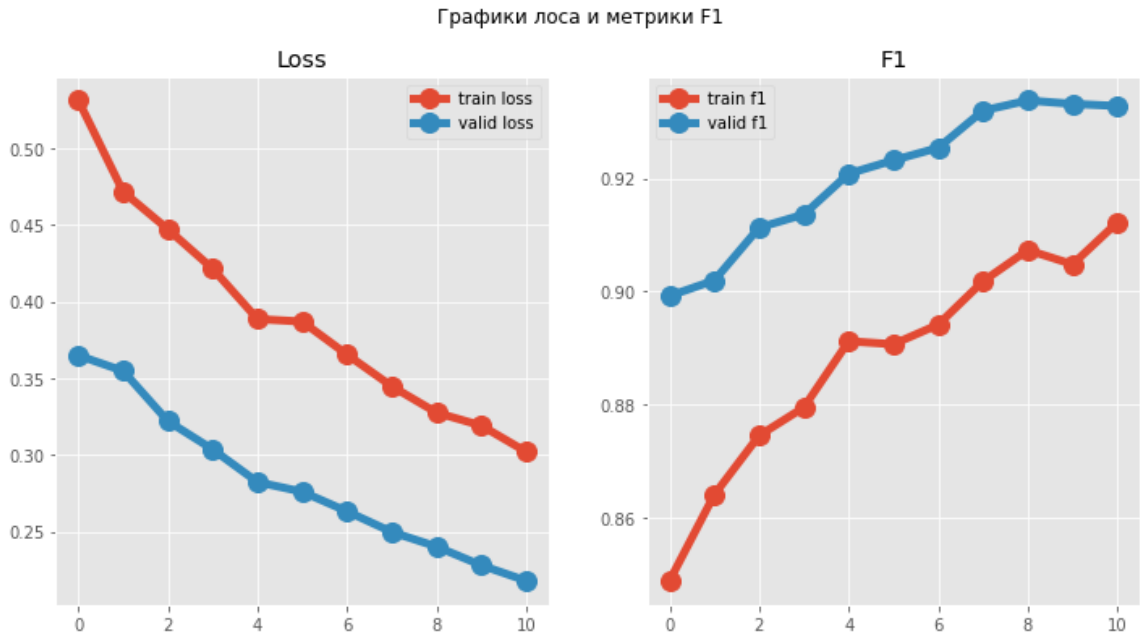
```
5 epoch
    f1_train: 0.8392 | loss_train: 0.5702
    f1_valid: 0.8828 | loss_valid: 0.4072
```



In [ ]:

```
switch_grad_bert(model, grad_on = True)
set_seed()
optimizer = torch.optim.Adam([{'params':model.bert.parameters(), 'lr':1e-6},
                              {'params':model.classifier.parameters(), 'lr':0.0001}])
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
           train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=1
1)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

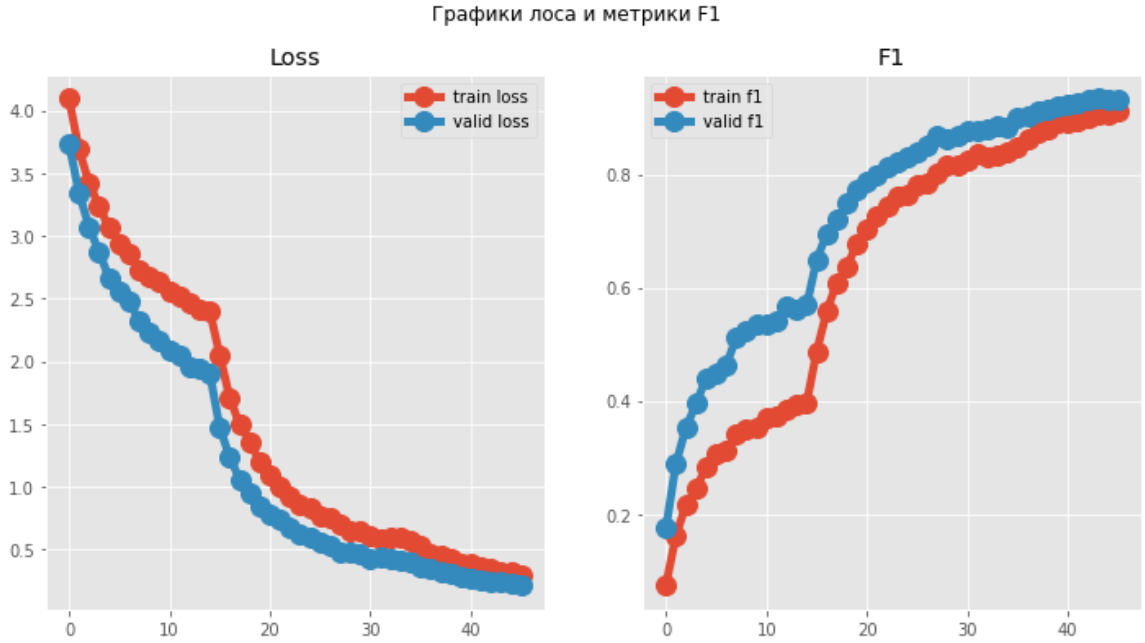
```
11 epoch
    f1_train: 0.9122 | loss_train: 0.3021
    f1_valid: 0.9328 | loss_valid: 0.2181
```



In [ ]:

```
print(f'Лучшее значение F1 на трейне: {max(f1_train):.4f} | на валидации: {max(f1_valid):.4f}')
show_metrics(loss_train, loss_valid, f1_train, f1_valid)
```

```
Лучшее значение F1 на трейне: 0.9122 | на валидации: 0.9338
```




Результаты неплохие, но до эксперимента 4.1 не дотягивают.

In [ ]:

```
torch.save(model.state_dict(), f'/content/drive/MyDrive/data_for_tinkoff/my_model_loss_without_weight.pt')
```

# Эксперимент 4.3

Сэмплер с балансировкой

In [ ]:

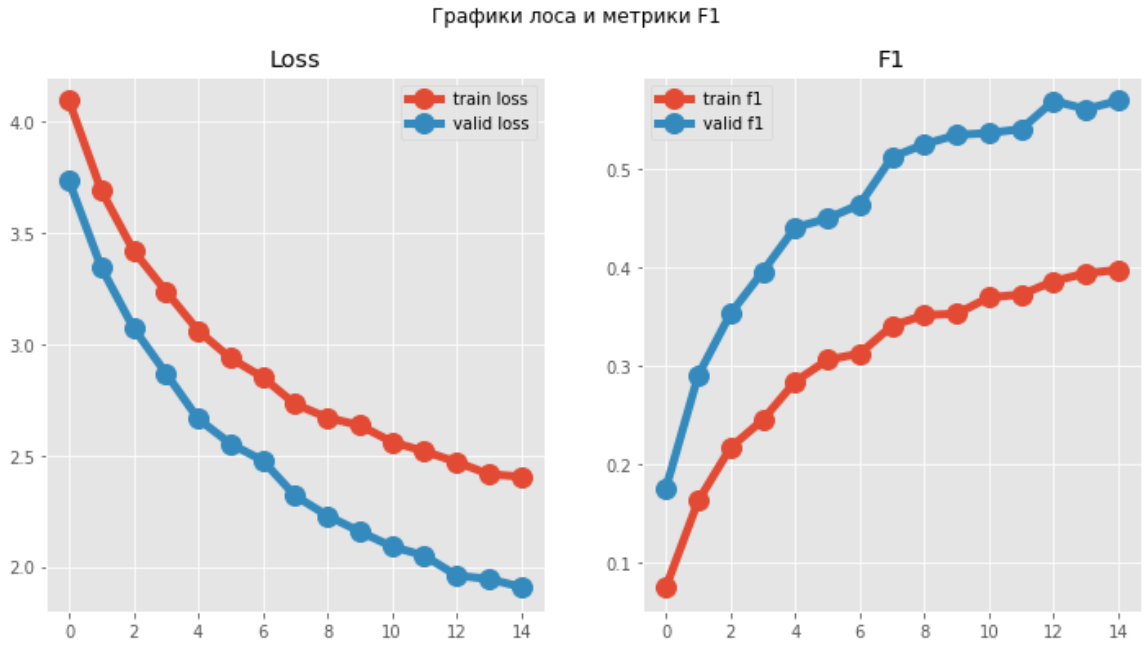
```
set_seed()
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=7
7)
model.to(device)
train_loader = DataIter(X_train1, y_train, tokenizer, device, batch_size=64, train = Tr
ue, make_balance = True)
valid_loader = DataIter(X_test1, y_test, tokenizer, device, batch_size=128, train = Fal
se, make_balance = False)
loss_train, loss_valid, f1_train, f1_valid = [], [], [], []
criterion = nn.CrossEntropyLoss()
```

```
Some weights of the model checkpoint at bert-base-uncased were not used wh
en initializing BertForSequenceClassification: ['cls.predictions.decoder.w
eight', 'cls.predictions.transform.dense.bias', 'cls.predictions.transfor
m.LayerNorm.bias', 'cls.predictions.transform.LayerNorm.weight', 'cls.seq_
relationship.weight', 'cls.predictions.bias', 'cls.seq_relationship.bias',
'cls.predictions.transform.dense.weight']
- This IS expected if you are initializing BertForSequenceClassification f
rom the checkpoint of a model trained on another task or with another arch
itecture (e.g. initializing a BertForSequenceClassification model from a B
ertForPreTraining model).
- This IS NOT expected if you are initializing BertForSequenceClassificati
on from the checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a BertForSequence
Classification model).
Some weights of BertForSequenceClassification were not initialized from th
e model checkpoint at bert-base-uncased and are newly initialized: ['class
ifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to u
se it for predictions and inference.
```

In [ ]:

```
switch_grad_bert(model, grad_on = False)
set_seed()
optimizer = torch.optim.Adam(model.classifier.parameters(), lr=0.004)
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
            train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=1
5)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

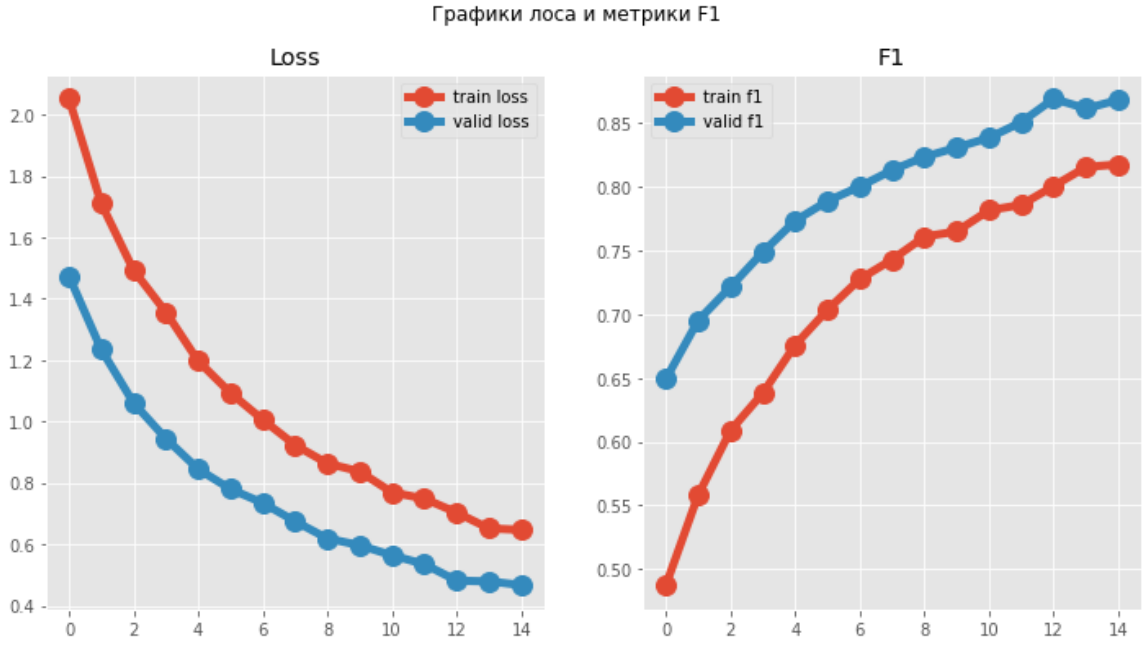
```
15 epoch
    f1_train: 0.3970 | loss_train: 2.4053
    f1_valid: 0.5689 | loss_valid: 1.9098
```



In [ ]:

```
switch_grad_bert(model, grad_on = True)
set_seed()
optimizer = torch.optim.Adam([{'params':model.bert.parameters(), 'lr':3e-7},
                              {'params':model.classifier.parameters(), 'lr':0.002}])
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
           train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=1
5)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

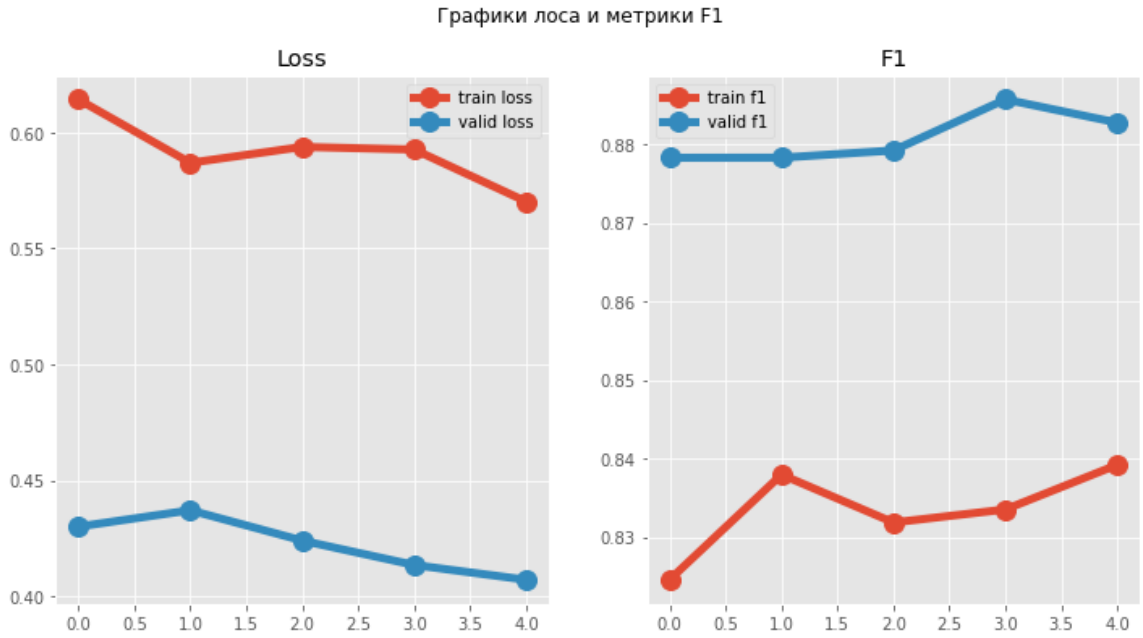
```
15 epoch
    f1_train: 0.8175 | loss_train: 0.6470
    f1_valid: 0.8681 | loss_valid: 0.4673
```



In [ ]:

```
switch_grad_bert(model, grad_on = False)
set_seed()
optimizer = torch.optim.Adam(model.classifier.parameters(), lr=0.001)
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
            train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=5
)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

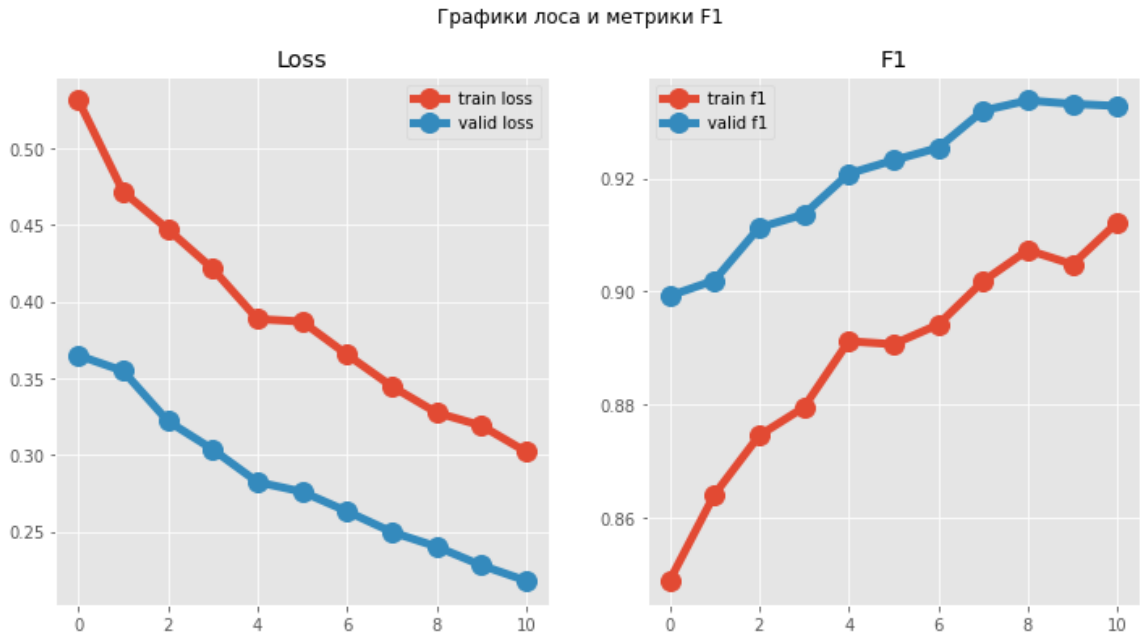
```
5 epoch
    f1_train: 0.8392 | loss_train: 0.5702
    f1_valid: 0.8828 | loss_valid: 0.4072
```



In [ ]:

```
switch_grad_bert(model, grad_on = True)
set_seed()
optimizer = torch.optim.Adam([{'params':model.bert.parameters(), 'lr':1e-6},
                              {'params':model.classifier.parameters(), 'lr':0.0001}])
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
           train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=1
1)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

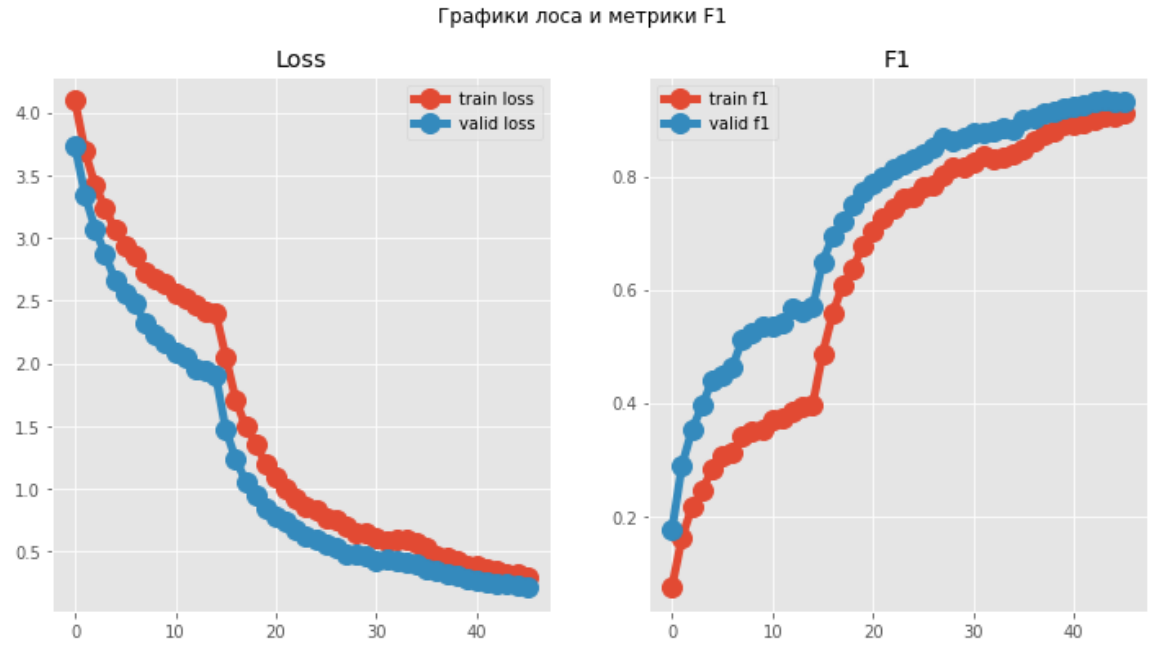
```
11 epoch
    f1_train: 0.9122 | loss_train: 0.3021
    f1_valid: 0.9328 | loss_valid: 0.2181
```



In [ ]:

```
print(f'Лучшее значение F1 на трейне: {max(f1_train):.4f} | на валидации: {max(f1_valid):.4f}')
show_metrics(loss_train, loss_valid, f1_train, f1_valid)
```

```
Лучшее значение F1 на трейне: 0.9122 | на валидации: 0.9338
```




С помощью сэмплера модель на ранних этапах быстрее обучалась, но до результатов эксперимента 4.1 не доросла.

In [ ]:

```
torch.save(model.state_dict(), f'/content/drive/MyDrive/data_for_tinkoff/my_model_loss_with_sampl.pt')
```

# Эксперимент 4.4

Лосс из статьи

Пытался сделать свой лосс, но работает не очень хорошо.

In [31]:

```
class SADL(torch.nn.Module):

    def __init__(self, alpha = 1.0, gamma = 1.0):
        super(SADL, self).__init__()
        self.alpha = alpha
        self.gamma = gamma


    def forward(self, logits, targets):
        p = torch.softmax(logits, dim=1)
        y = torch.zeros_like(logits)
        y[range(logits.shape[0]),targets] = 1
        a = ((1 - p) ** self.alpha) * p
        loss =  1 - (2 * a * y + self.gamma) / (a + y + self.gamma)
        return loss.mean()
```

In [33]:

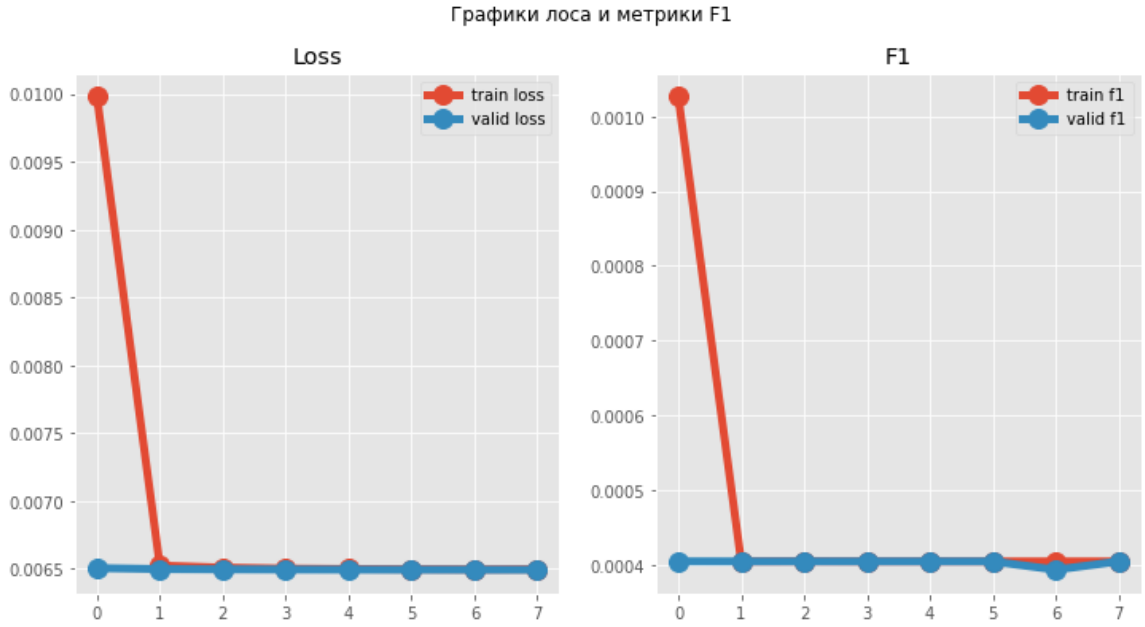
```
criterion = SADL()
set_seed()
train_loader = DataIter(X_train1, y_train, tokenizer, device, batch_size=64, train = Tr
ue, make_balance = False)
valid_loader = DataIter(X_test1, y_test, tokenizer, device, batch_size=128, train = Fal
se, make_balance = False)
loss_train, loss_valid, f1_train, f1_valid = [], [], [], []

model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=7
7)
model.to(device)
print(1)
```

```
Some weights of the model checkpoint at bert-base-uncased were not used wh
en initializing BertForSequenceClassification: ['cls.seq_relationship.bia
s', 'cls.predictions.transform.LayerNorm.weight', 'cls.seq_relationship.we
ight', 'cls.predictions.transform.LayerNorm.bias', 'cls.predictions.transf
orm.dense.bias', 'cls.predictions.bias', 'cls.predictions.transform.dense.
weight', 'cls.predictions.decoder.weight']
- This IS expected if you are initializing BertForSequenceClassification f
rom the checkpoint of a model trained on another task or with another arch
itecture (e.g. initializing a BertForSequenceClassification model from a B
ertForPreTraining model).
- This IS NOT expected if you are initializing BertForSequenceClassificati
on from the checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a BertForSequence
Classification model).
Some weights of BertForSequenceClassification were not initialized from th
e model checkpoint at bert-base-uncased and are newly initialized: ['class
ifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to u
se it for predictions and inference.

1
```

In [34]:

```
switch_grad_bert(model, grad_on = False)
set_seed()
optimizer = torch.optim.Adam(model.classifier.parameters(), lr=0.0003)
loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
            train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=8
)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

```
8 epoch
    f1_train: 0.0004 | loss_train: 0.0065
    f1_valid: 0.0004 | loss_valid: 0.0065
```

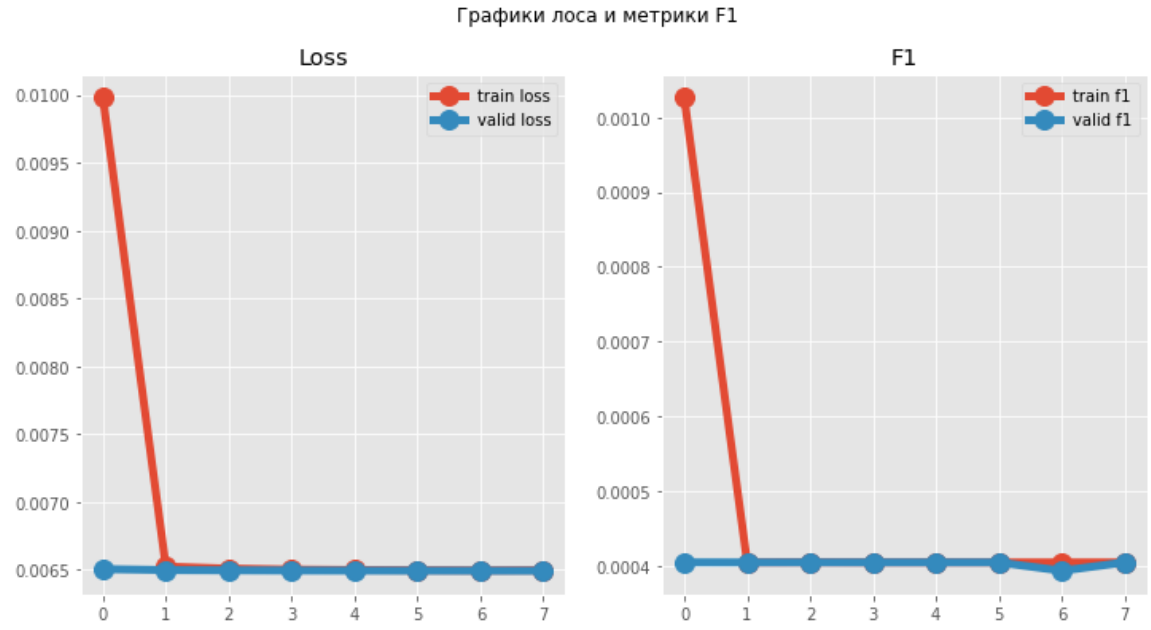



Модель кажется совсем не учится, попробуем взять модель из эксперимента 4.1 и дообучить.

In [35]:

```
print(f'Лучшее значение F1 на трейне: {max(f1_train):.4f} | на валидации: {max(f1_valid):.4f}')
show_metrics(loss_train, loss_valid, f1_train, f1_valid)
```

```
Лучшее значение F1 на трейне: 0.0010 | на валидации: 0.0004
```




Я нашёл реализацию данного лосса на гитхабе ссылка: [https://github.com/fursovia/self-adj-dice/tree/4d70d87a05afa154d1002acebd95848f833db342 (https://github.com/fursovia/self-adj-dice/tree/4d70d87a05afa154d1002acebd95848f833db342)](https://github.com/fursovia/self-adj-dice/tree/4d70d87a05afa154d1002acebd95848f833db342)

Кажется это Ваш сотрудник Иван Фурсов (Чайников Константин с тинькофф поколения передаёт тебе привет), кажется у него правильно реализован лосс

In [36]:

```
!pip install sadice
from sadice import SelfAdjDiceLoss #
```

```
Collecting sadice
  Downloading https://files.pythonhosted.org/packages/9f/ed/f566e4791f0c1e
b1a2edc459d571ee93a4579fada122f4028c69d4d8f131/sadice-0.1.3-py3-none-any.w
hl
Requirement already satisfied: torch<2.0.0,>=1.0.0 in /usr/local/lib/pytho
n3.7/dist-packages (from sadice) (1.9.0+cu102)
Requirement already satisfied: typing-extensions in /usr/local/lib/python
3.7/dist-packages (from torch<2.0.0,>=1.0.0->sadice) (3.7.4.3)
Installing collected packages: sadice
Successfully installed sadice-0.1.3
```

In [37]:

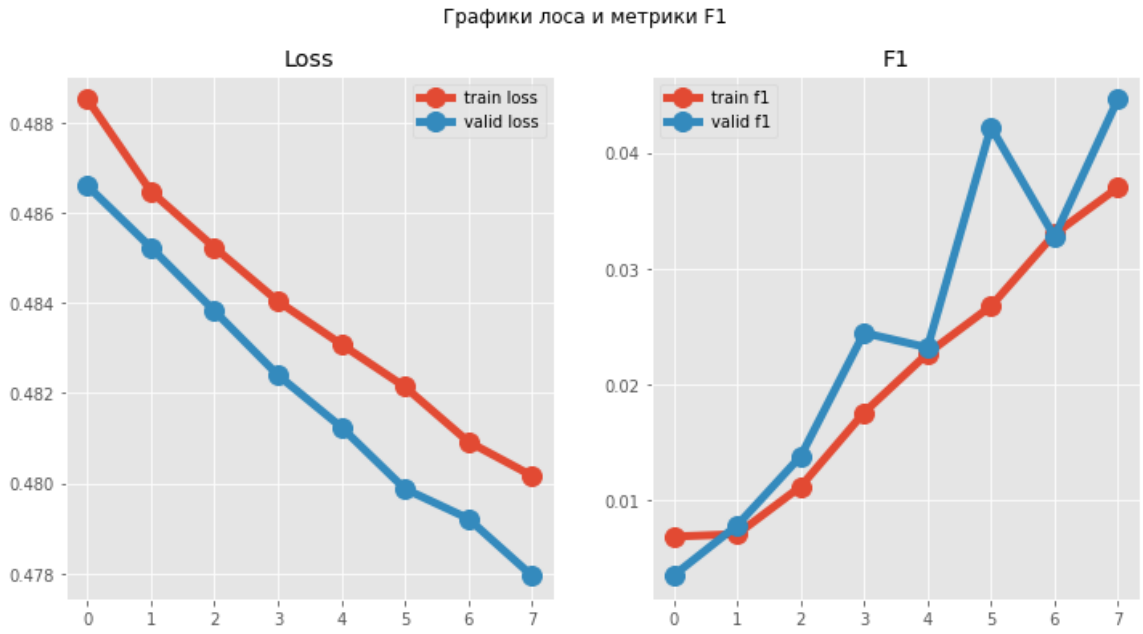
```
criterion = SelfAdjDiceLoss()
set_seed()
train_loader = DataIter(X_train1, y_train, tokenizer, device, batch_size=64, train = Tr
ue, make_balance = False)
valid_loader = DataIter(X_test1, y_test, tokenizer, device, batch_size=128, train = Fal
se, make_balance = False)
loss_train, loss_valid, f1_train, f1_valid = [], [], [], []

model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=7
7)
model.to(device)
print(1)
```

```
Some weights of the model checkpoint at bert-base-uncased were not used wh
en initializing BertForSequenceClassification: ['cls.seq_relationship.bia
s', 'cls.predictions.transform.LayerNorm.weight', 'cls.seq_relationship.we
ight', 'cls.predictions.transform.LayerNorm.bias', 'cls.predictions.transf
orm.dense.bias', 'cls.predictions.bias', 'cls.predictions.transform.dense.
weight', 'cls.predictions.decoder.weight']
- This IS expected if you are initializing BertForSequenceClassification f
rom the checkpoint of a model trained on another task or with another arch
itecture (e.g. initializing a BertForSequenceClassification model from a B
ertForPreTraining model).
- This IS NOT expected if you are initializing BertForSequenceClassificati
on from the checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a BertForSequence
Classification model).
Some weights of BertForSequenceClassification were not initialized from th
e model checkpoint at bert-base-uncased and are newly initialized: ['class
ifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to u
se it for predictions and inference.

1
```

In [38]:

```
switch_grad_bert(model, grad_on = False)

set_seed()
optimizer = torch.optim.Adam(model.classifier.parameters(), lr=0.0003)

loss_train_e, loss_valid_e, f1_train_e, f1_valid_e = train(model,
            train_loader, valid_loader, optimizer, criterion, device=device, n_epochs=8
)
loss_train.extend(loss_train_e)
loss_valid.extend(loss_valid_e)
f1_train.extend(f1_train_e)
f1_valid.extend(f1_valid_e)
```

```
8 epoch
    f1_train: 0.0370 | loss_train: 0.4802
    f1_valid: 0.0446 | loss_valid: 0.4780
```

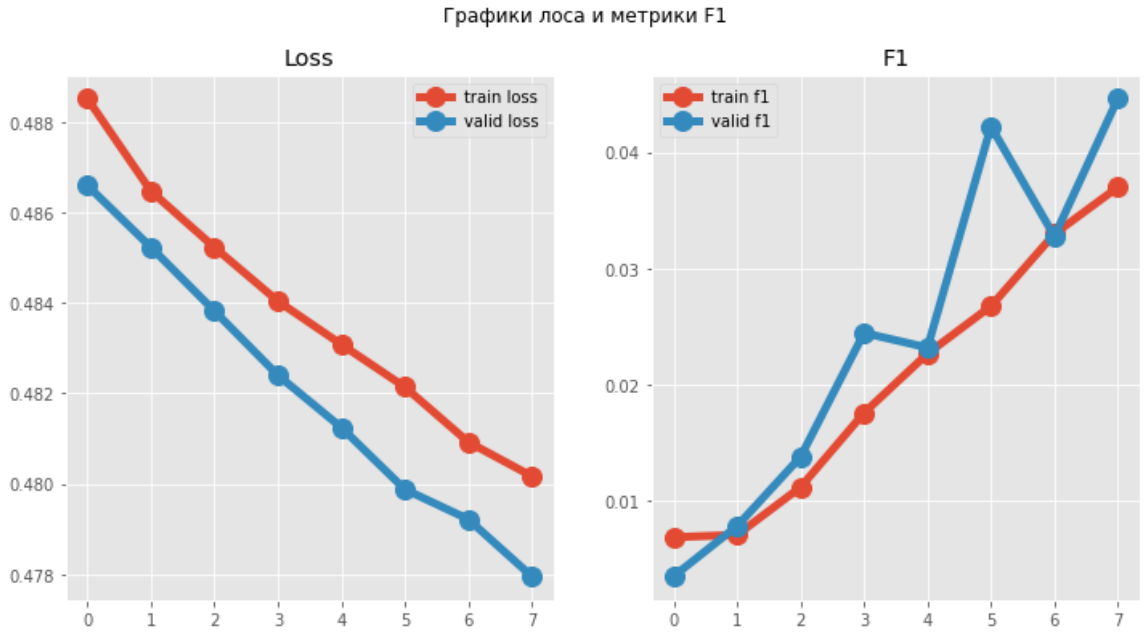



Явного прогресса не наблюдается

In [39]:

```
print(f'Лучшее значение F1 на трейне: {max(f1_train):.4f} | на валидации: {max(f1_valid):.4f}')
show_metrics(loss_train, loss_valid, f1_train, f1_valid)
```

```
Лучшее значение F1 на трейне: 0.0370 | на валидации: 0.0446
```




# Выводы ¶

Итого берт с лосом кросс энтропии с весами показал наилучшие результаты. Моя реализация Лосса из статьи не работает здесь, лосс из статьи Ивана Фурсова тоже не показал хорошие результаты.

**Возможные улучшения:**

- Можно попробовать дальше поэкспериментировать с комбинированием подходов, например взять в самом начале балансированное сэмплирование и обычным лоссом, потом вернуться к обычному сэплмированию, но взять лосс с весами.
- Взять берт побольше или дольше обучить этот берт
- Возможно стоит попробовать обучить сеть на tfidf/count кодировке предложений.
- Реализовать Labelsmoothing

In [ ]:

```
!jupyter nbconvert --to html Testovoe_tinkoff_chainikov_konstantin.ipynb
```

```
[NbConvertApp] Converting notebook Testovoe_tinkoff_chainikov_konstantin.i
pynb to html
[NbConvertApp] Writing 1695768 bytes to Testovoe_tinkoff_chainikov_konstan
tin.html
```

In [ ]:

```

```