```
!pip install transformers
```

```
Collecting transformers
  Downloading https://files.pythonhosted.org/packages/fd/1a/41c644c963249fd7f
     |████████████████████████████████| 2.5MB 8.1MB/s
Requirement already satisfied: importlib-metadata; python_version < "3.8" in
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-pac
Collecting sacremoses
  Downloading https://files.pythonhosted.org/packages/75/ee/67241dc87f266093c
     |████████████████████████████████| 901kB 31.8MB/s
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/
Collecting huggingface-hub==0.0.12
  Downloading https://files.pythonhosted.org/packages/2f/ee/97e253668fda9b17e
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-pack
Collecting tokenizers<0.11,>=0.10.1
  Downloading https://files.pythonhosted.org/packages/d4/e2/df3543e8ffdab68f5
     |████████████████████████████████| 3.3MB 41.6MB/s
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: typing-extensions>=3.6.4; python_version < "3.
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7/d
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /us
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7
Installing collected packages: sacremoses, huggingface-hub, tokenizers, trans
Successfully installed huggingface-hub-0.0.12 sacremoses-0.0.45 tokenizers-0.
```

В современной компьютерной лингвистике вычисление вероятности текста производится в основном за счёт нейронных, а не энграммных моделей. Существует много разновидностей архитектур, мы рассмотрим 2: левостороннюю модель `gpt2` (её облегчённую версию `distilgpt2`) и языковую модель с пропусками `BERT`.

Односторонние языковые модели

Вначале создадим токенизатор и применим его к данным

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("gpt2")
```

```
Downloading: 100%                                   665/665 [00:00<00:00, 17.2kB/s]

Downloading: 100%                                   1.04M/1.04M [00:00<00:00, 4.10MB/s]

Downloading: 100%                                   456k/456k [00:00<00:00, 3.25MB/s]
```

```
text = "The Starship prototype descended under active aerodynamic control, accompl
tokenization = tokenizer(text)
print(tokenization)
```

```
{'input_ids': [464, 40172, 14879, 23667, 739, 4075, 9551, 34743, 1630, 11, 13
```

Что бы это могло значить? Наверное, это индексы токенов в словаре.

```
tokens = tokenizer.convert_ids_to_tokens(tokenization["input_ids"])
for index, token in zip(tokenization["input_ids"], tokens):
    print(index, token)
```

```
464 The
40172 ĠStarship
14879 Ġprototype
23667 Ġdescended
739 Ġunder
4075 Ġactive
9551 Ġaer
34743 odynamic
1630 Ġcontrol
11 ,
13013 Ġaccomplished
416 Ġby
1440 Ġfour
5672 Ġvehicles
13 .
```

Таким образом, индексы соответствуют не только словам, но и частотным символьным энграммам, это сделано для того, чтобы модель могла справляться с неизвестными словами. Токенизатор осуществляет преобразование из слов в индексы и обратно.

```
print(tokenizer.convert_tokens_to_ids(tokens))  ## результат совпадает с tokenizat
```

```
[464, 40172, 14879, 23667, 739, 4075, 9551, 34743, 1630, 11, 13013, 416, 1440
```

Токенизатор также умеет генерировать тензоры работать с батчами.

```
sents = [
    "Yesterday, all my troubles seemed so far away.",
    "I only want to say, if there is a way, take away this cup of poison, 'cause i
    "We do not need your education, we do not need your thought control.",
    "When the light begins to change, I sometimes feel a little strange, a little 
]
for elem in tokenizer(sents)["input_ids"]:
    print(elem)
```

```
[28065, 11, 477, 616, 14979, 3947, 523, 1290, 1497, 13]
[40, 691, 765, 284, 910, 11, 611, 612, 318, 257, 835, 11, 1011, 1497, 428, 65
[1135, 466, 407, 761, 534, 3707, 11, 356, 466, 407, 761, 534, 1807, 1630, 13]
[2215, 262, 1657, 6140, 284, 1487, 11, 314, 3360, 1254, 257, 1310, 6283, 11,
```

```
tokenizer.pad_token = tokenizer.eos_token
for elem in tokenizer(sents, return_tensors="pt", padding=True)["input_ids"]:
    print(elem)
```

```
tensor([28065,    11,   477,   616, 14979,  3947,   523,  1290,  1497,    13,
        50256, 50256, 50256, 50256, 50256, 50256, 50256, 50256, 50256, 50256,
        50256, 50256, 50256, 50256, 50256])
tensor([   40,   691,   765,   284,   910,    11,   611,   612,   318,   257,
          835,    11,  1011,  1497,   428,  6508,   286,  8764,    11,   705,
        25587,   340, 20246,   502,    13])
tensor([ 1135,   466,   407,   761,   534,  3707,    11,   356,   466,   407,
          761,   534,  1807,  1630,    13, 50256, 50256, 50256, 50256, 50256,
        50256, 50256, 50256, 50256, 50256])
tensor([ 2215,   262,  1657,  6140,   284,  1487,    11,   314,  3360,  1254,
          257,  1310,  6283,    11,   257,  1310, 18116,   618,   340,   338,
         3223,    13, 50256, 50256, 50256])
```

Загрузим модель на видеокарту.

```
from transformers import AutoModelWithLMHead, AutoModel

model = AutoModelWithLMHead.from_pretrained("gpt2").to("cuda")
type(model)
```

```
/usr/local/lib/python3.7/dist-packages/transformers/models/auto/modeling_auto
  FutureWarning,
Downloading: 100%                      548M/548M [00:11<00:00, 48.1MB/s]

transformers.models.gpt2.modeling_gpt2.GPT2LMHeadModel
```

Вычислим вероятность каждого сабтокена в этом тексте, включая начальный. Модель `gpt2` получает на вход список токенов и в каждой позиции предсказывает распределение вероятностей для следующего "слова".

Поэтому добавим в текст символ начала строки, чтобы иметь возможность посмотреть распределение вероятностей для первого слова. По умолчанию `gpt2` этого не делает.

```
import torch
from torch import LongTensor

text = "The Starship prototype descended under active aerodynamic control, accompl
tokenization = tokenizer(text)
tokenization = tokenizer(text)
tokens = tokenizer.convert_ids_to_tokens(tokenization["input_ids"])
input_ids = [tokenizer.bos_token_id] + tokenization["input_ids"]
batch = LongTensor([input_ids]).to("cuda")
with torch.no_grad():
    output = model(batch)
logits = output["logits"]
logits.shape
```

```
torch.Size([1, 16, 50257])
```

Пока сеть посчитала так называемые логиты $z_1, \ldots, z_n$. Чтобы по ним получить вероятности, нужно применить операцию `softmax`:

$$\mathrm{softmax}(z_1, \ldots, z_n) = [\frac{e^{z_1}}{\sum\limits_j e^{z_j}}, \ldots, \frac{e^{z_n}}{\sum\limits_j e^{z_j}}]$$

Найдём вероятность каждого токена

```
import numpy as np

probs = torch.softmax(logits[0], dim=-1)
probs =  probs.cpu().numpy()
# for i, (index, token) in enumerate(zip(input_ids[1:], tokens)):
#     print(i, token, probs[i,index])
token_probs = probs[np.arange(len(probs)-1), input_ids[1:]]
for i, (token, prob) in enumerate(zip(tokens, token_probs)):
    print(i, token, prob)
```

```
0 The 0.037699427
1 ĠStarship 1.164296e-05
2 Ġprototype 2.3631192e-05
3 Ġdescended 4.776689e-06
4 Ġunder 0.0038189965
5 Ġactive 0.00015830748
6 Ġaer 0.0013223129
7 odynamic 0.789692
8 Ġcontrol 0.1038841
9 , 0.12653658
10 Ġaccomplished 2.7169448e-05
11 Ġby 0.118614025
12 Ġfour 0.0030419987
13 Ġvehicles 0.00068965525
14 . 0.14785583
```

Посчитаем в каждой позиции 5 самых вероятных токенов. Здесь удобнее использовать исходные логарифмические вероятности:

```
top_log_probs, top_indexes = torch.topk(logits[0], k=5, dim=-1)
top_indexes = top_indexes.cpu().numpy()
for i in range(len(logits[0])-1):
    print("<BEGIN> " + " ".join([x.strip("ĠĊ") for x in tokens[:i]]))
    curr_top_tokens = tokenizer.convert_ids_to_tokens(top_indexes[i])
    for index, token in zip(top_indexes[i], curr_top_tokens):
        token = token.strip("ĠĊ")
        print(f"{token}:{probs[i,index]:.3f}", end=" ")
    print("")
```

```
<BEGIN>
:0.062 The:0.038 ":0.024 A:0.019 I:0.018
<BEGIN> The
first:0.010 U:0.009 following:0.008 United:0.006 US:0.005
<BEGIN> The Starship
Enterprise:0.213 Tro:0.212 Trooper:0.021 Ant:0.008 Crew:0.008
<BEGIN> The Starship prototype
is:0.146 was:0.063 of:0.052 has:0.044 ,:0.042
<BEGIN> The Starship prototype descended
into:0.391 to:0.135 from:0.115 on:0.041 in:0.019
<BEGIN> The Starship prototype descended under
the:0.259 a:0.132 heavy:0.109 its:0.050 fire:0.041
<BEGIN> The Starship prototype descended under active
investigation:0.103 fire:0.084 development:0.066 radar:0.043 construction:0.0
<BEGIN> The Starship prototype descended under active aer
odynamic:0.790 odynamics:0.127 ob:0.036 on:0.022 opl:0.007
<BEGIN> The Starship prototype descended under active aer odynamic
drag:0.327 control:0.104 braking:0.033 and:0.027 stress:0.025
<BEGIN> The Starship prototype descended under active aer odynamic control
,:0.127 and:0.073 during:0.066 on:0.064 at:0.062
<BEGIN> The Starship prototype descended under active aer odynamic control ,
and:0.114 but:0.085 which:0.041 with:0.041 the:0.032
<BEGIN> The Starship prototype descended under active aer odynamic control ,
in:0.143 by:0.119 its:0.087 with:0.069 a:0.060
<BEGIN> The Starship prototype descended under active aer odynamic control ,
the:0.175 a:0.169 an:0.044 its:0.017 using:0.010
<BEGIN> The Starship prototype descended under active aer odynamic control ,
engines:0.038 -:0.027 separate:0.018 different:0.017 small:0.015
<BEGIN> The Starship prototype descended under active aer odynamic control ,
.:0.148 ,:0.134 ::0.098 of:0.046 in:0.043
```

Теперь посмотрим, насколько модель знает грамматику.

```
texts = [
    "Alexandra is very proud of herself.", "Alexandra is very proud of himself.",
    "Alexander is very proud of herself.", "Alexander is very proud of himself.",
    "Alexandra is very proud of she.", "Alexandra is very proud of her.",
    "Alexandra is very proud of her son."
]
tokenizer.pad_token = tokenizer.eos_token
batch = tokenizer(texts, return_tensors="pt", padding=True).to("cuda")
# добавляем индекс начала строки (склейка массивов по первой координате)
batch["input_ids"] = torch.cat([
   torch.ones_like(batch["input_ids"][:,:1])*tokenizer.bos_token_id,
   batch["input_ids"]
], dim=1)
#  batch["attention_mask"] = torch.cat([
#     torch.ones_like(batch["attention_mask"][:,:1]),
#     batch["attention_mask"]
#  ], dim=-1)
```

```
with torch.no_grad():
    logits = model(batch["input_ids"])["logits"]
probs = torch.softmax(logits, dim=-1).cpu().numpy()
print(probs.shape)
```

```
(7, 10, 50257)
```

```
for i, text in enumerate(texts):
    print(text)
    text_token_ids = batch["input_ids"][i,1:]
    text_tokens = [x.strip("ĠĊ") for x in tokenizer.convert_ids_to_tokens(text_tok
    for j, (index, token) in enumerate(zip(text_token_ids, text_tokens)):
        print(f"{token}:{probs[i,j,index]:.3f}", end=" ")
    print("")
```

```
Alexandra is very proud of herself.
Alex:0.000 andra:0.050 is:0.014 very:0.004 proud:0.028 of:0.579 herself:0.040
Alexandra is very proud of himself.
Alex:0.000 andra:0.050 is:0.014 very:0.004 proud:0.028 of:0.579 himself:0.002
Alexander is very proud of herself.
Alexander:0.000 is:0.011 very:0.004 proud:0.030 of:0.679 herself:0.001 .:0.20
Alexander is very proud of himself.
Alexander:0.000 is:0.011 very:0.004 proud:0.030 of:0.679 himself:0.020 .:0.19
Alexandra is very proud of she.
Alex:0.000 andra:0.050 is:0.014 very:0.004 proud:0.028 of:0.579 she:0.001 .:0
Alexandra is very proud of her.
Alex:0.000 andra:0.050 is:0.014 very:0.004 proud:0.028 of:0.579 her:0.356 .:0
Alexandra is very proud of her son.
Alex:0.000 andra:0.050 is:0.014 very:0.004 proud:0.028 of:0.579 her:0.356 son
```

```
!apt install -qq enchant
!pip install pyenchant
```

```
The following additional packages will be installed:
  aspell aspell-en dictionaries-common emacsen-common hunspell-en-us
  libaspell15 libenchant1c2a libhunspell-1.6-0 libtext-iconv-perl
Suggested packages:
  aspell-doc spellutils wordlist hunspell openoffice.org-hunspell
  | openoffice.org-core libenchant-voikko
The following NEW packages will be installed:
  aspell aspell-en dictionaries-common emacsen-common enchant hunspell-en-u
  libaspell15 libenchant1c2a libhunspell-1.6-0 libtext-iconv-perl
0 upgraded, 10 newly installed, 0 to remove and 39 not upgraded.
Need to get 1,310 kB of archives.
After this operation, 5,353 kB of additional disk space will be used.
Preconfiguring packages ...
Selecting previously unselected package libtext-iconv-perl.
(Reading database ... 160837 files and directories currently installed.)
Preparing to unpack .../0-libtext-iconv-perl_1.7-5build6_amd64.deb ...
Unpacking libtext-iconv-perl (1.7-5build6) ...
Selecting previously unselected package libaspell15:amd64.
Preparing to unpack .../1-libaspell15_0.60.7~20110707-4ubuntu0.1_amd64.deb
Unpacking libaspell15:amd64 (0.60.7~20110707-4ubuntu0.1) ...
Selecting previously unselected package emacsen-common.
Preparing to unpack .../2-emacsen-common_2.0.8_all.deb ...
Unpacking emacsen-common (2.0.8) ...
Selecting previously unselected package dictionaries-common.
```

```
Preparing to unpack .../3-dictionaries-common_1.27.2_all.deb ...
Adding 'diversion of /usr/share/dict/words to /usr/share/dict/words.pre-dict
Unpacking dictionaries-common (1.27.2) ...
Selecting previously unselected package aspell.
Preparing to unpack .../4-aspell_0.60.7~20110707-4ubuntu0.1_amd64.deb ...
Unpacking aspell (0.60.7~20110707-4ubuntu0.1) ...
Selecting previously unselected package aspell-en.
Preparing to unpack .../5-aspell-en_2017.08.24-0-0.1_all.deb ...
Unpacking aspell-en (2017.08.24-0-0.1) ...
Selecting previously unselected package hunspell-en-us.
Preparing to unpack .../6-hunspell-en-us_1%3a2017.08.24_all.deb ...
Unpacking hunspell-en-us (1:2017.08.24) ...
Selecting previously unselected package libhunspell-1.6-0:amd64.
Preparing to unpack .../7-libhunspell-1.6-0_1.6.2-1_amd64.deb ...
Unpacking libhunspell-1.6-0:amd64 (1.6.2-1) ...
Selecting previously unselected package libenchant1c2a:amd64.
Preparing to unpack .../8-libenchant1c2a_1.6.0-11.1_amd64.deb ...
Unpacking libenchant1c2a:amd64 (1.6.0-11.1) ...
Selecting previously unselected package enchant.
Preparing to unpack .../9-enchant_1.6.0-11.1_amd64.deb ...
Unpacking enchant (1.6.0-11.1) ...
Setting up libhunspell-1.6-0:amd64 (1.6.2-1) ...
Setting up libaspell15:amd64 (0.60.7~20110707-4ubuntu0.1) ...
Setting up emacsen-common (2.0.8) ...
Setting up libtext-iconv-perl (1.7-5build6) ...
Setting up dictionaries-common (1.27.2) ...
Setting up aspell (0.60.7~20110707-4ubuntu0.1) ...
Setting up hunspell-en-us (1:2017.08.24) ...
Setting up libenchant1c2a:amd64 (1.6.0-11.1) ...
Setting up aspell-en (2017.08.24-0-0.1) ...
Setting up enchant (1.6.0-11.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.2) ...
/sbin/ldconfig.real: /usr/local/lib/python3.7/dist-packages/ideep4py/lib/li
```

```python
import enchant
#function for finding misspeled word in sentence
def find_misspeled(sentance):
  word_list = sentance.split(' ')
  misspelled_words = dict()
  #print(len(word_list))
  for i in range(len(word_list)):
    if not d.check(word_list[i]):
      misspelled_words[word_list[i]] = i
  return misspelled_words


#check fuction
### check the useful function in package enchant
#load english dictionary
d = enchant.Dict("en_US")
find_misspeled('He is intelligen')
```

```
{'intelligen': 2}
```

```python
def generate_correction(sentance):
  sentance_list = [sentance]
```

```
    possible_correction_list = []
    word_dict = find_misspeled(sentance)

    for word in word_dict.keys():
      for sent in sentance_list:
        word_list = sent.split(' ')
        for correction_word in d.suggest(word):
          new_sentance = ' '.join(word_list[:word_dict[word]] + [correction_word] + w
          possible_correction_list  += [new_sentance]
      #print(sentance_list, word)
      sentance_list = possible_correction_list
      possible_correction_list = []
      #print(sentance_list)
    return sentance_list
```

```
#check function
generate_correction('He is intelligen')
```

```
['He is intelligent',
 'He is intelligence',
 'He is intelligible',
 'He is intelligibly',
 'He is belligerent']
```

```
#check function
generate_correction('He liks intelligen peaple')
```

```
 'He ilks intelligible people',
 'He ilks intelligible Peale',
 'He ilks intelligible leaper',
 'He ilks intelligible apple',
 'He ilks intelligible appeal',
 'He ilks intelligibly people',
 'He ilks intelligibly Peale',
 'He ilks intelligibly leaper',
 'He ilks intelligibly apple',
 'He ilks intelligibly appeal',
 'He ilks belligerent people',
 'He ilks belligerent Peale',
 'He ilks belligerent leaper',
 'He ilks belligerent apple',
 'He ilks belligerent appeal',
 'He likes intelligent people',
 'He likes intelligent Peale',
 'He likes intelligent leaper',
 'He likes intelligent apple',
 'He likes intelligent appeal',
 'He likes intelligence people',
 'He likes intelligence Peale',
 'He likes intelligence leaper',
 'He likes intelligence apple',
 'He likes intelligence appeal',
 'He likes intelligible people',
 'He likes intelligible Peale',
 'He likes intelligible leaper',
 'He likes intelligible apple',
 'He likes intelligible appeal',
 'He likes intelligibly people',
 'He likes intelligibly Peale',
 'He likes intelligibly leaper',
 'He likes intelligibly apple',
 'He likes intelligibly appeal',
 'He likes belligerent people',
 'He likes belligerent Peale',
 'He likes belligerent leaper',
 'He likes belligerent apple',
 'He likes belligerent appeal',
 'He links intelligent people',
 'He links intelligent Peale',
 'He links intelligent leaper',
 'He links intelligent apple',
 'He links intelligent appeal',
 'He links intelligence people',
 'He links intelligence Peale',
 'He links intelligence leaper',
 'He links intelligence apple',
 'He links intelligence appeal',
 'He links intelligible people',
 'He links intelligible Peale',
 'He links intelligible leaper',
 'He links intelligible apple',
 'He links intelligible appeal',
 'He links intelligibly people',
 'He links intelligibly Peale',
 'He links intelligibly leaper',
 'He links intelligibly apple',
```

```
texts = generate_correction('He liks intelligen peaple')
tokenizer.pad_token = tokenizer.eos_token
batch = tokenizer(texts, return_tensors="pt", padding=True).to("cuda")
# добавляем индекс начала строки (склейка массивов по первой координате)
batch["input_ids"] = torch.cat([
   torch.ones_like(batch["input_ids"][:,:1])*tokenizer.bos_token_id,
   batch["input_ids"]
], dim=1)
#  batch["attention_mask"] = torch.cat([
#     torch.ones_like(batch["attention_mask"][:,:1]),
#     batch["attention_mask"]
#  ], dim=-1)
with torch.no_grad():
   logits = model(batch["input_ids"])["logits"]
probs = torch.softmax(logits, dim=-1).cpu().numpy()
print(probs.shape)
```

```
    (250, 9, 50257)
```

```
for i, text in enumerate(texts):
    print(text)
    text_token_ids = batch["input_ids"][i,1:]
    text_tokens = [x.strip("ĠĊ") for x in tokenizer.convert_ids_to_tokens(text_tok
    for j, (index, token) in enumerate(zip(text_token_ids, text_tokens)):
        print(f"{token}:{probs[i,j,index]:.3f}", end=" ")
    print("")
```

```
He leeks intelligence Peale
He:0.003 le:0.000 eks:0.000 intelligence:0.000 Pe:0.000 ale:0.001 <|endofte
He leeks intelligence leaper
He:0.003 le:0.000 eks:0.000 intelligence:0.000 le:0.000 aper:0.000 <|endoft
He leeks intelligence apple
He:0.003 le:0.000 eks:0.000 intelligence:0.000 apple:0.000 <|endoftext|>:0.
He leeks intelligence appeal
He:0.003 le:0.000 eks:0.000 intelligence:0.000 appeal:0.000 <|endoftext|>:0
He leeks intelligible people
He:0.003 le:0.000 eks:0.000 intellig:0.000 ible:0.002 people:0.000 <|endoft
He leeks intelligible Peale
He:0.003 le:0.000 eks:0.000 intellig:0.000 ible:0.002 Pe:0.000 ale:0.000 <|
He leeks intelligible leaper
He:0.003 le:0.000 eks:0.000 intellig:0.000 ible:0.002 le:0.001 aper:0.000 <
He leeks intelligible apple
He:0.003 le:0.000 eks:0.000 intellig:0.000 ible:0.002 apple:0.000 <|endofte
He leeks intelligible appeal
He:0.003 le:0.000 eks:0.000 intellig:0.000 ible:0.002 appeal:0.000 <|endoft
He leeks intelligibly people
He:0.003 le:0.000 eks:0.000 intellig:0.000 ibly:0.015 people:0.000 <|endoft
He leeks intelligibly Peale
He:0.003 le:0.000 eks:0.000 intellig:0.000 ibly:0.015 Pe:0.000 ale:0.001 <|
He leeks intelligibly leaper
He:0.003 le:0.000 eks:0.000 intellig:0.000 ibly:0.015 le:0.000 aper:0.000 <
He leeks intelligibly apple
He:0.003 le:0.000 eks:0.000 intellig:0.000 ibly:0.015 apple:0.000 <|endofte
He leeks intelligibly appeal
He:0.003 le:0.000 eks:0.000 intellig:0.000 ibly:0.015 appeal:0.000 <|endoft
He leeks belligerent people
He:0.003 le:0.000 eks:0.000 bellig:0.000 erent:0.958 people:0.000 <|endofte
He leeks belligerent Peale
He:0.003 le:0.000 eks:0.000 bellig:0.000 erent:0.958 Pe:0.000 ale:0.001 <|e
He leeks belligerent leaper
He:0.003 le:0.000 eks:0.000 bellig:0.000 erent:0.958 le:0.000 aper:0.000 <|
He leeks belligerent apple
He:0.003 le:0.000 eks:0.000 bellig:0.000 erent:0.958 apple:0.000 <|endoftex
He leeks belligerent appeal
He:0.003 le:0.000 eks:0.000 bellig:0.000 erent:0.958 appeal:0.000 <|endofte
He ilks intelligent people
He:0.003 il:0.000 ks:0.014 intelligent:0.000 people:0.059 <|endoftext|>:0.0
He ilks intelligent Peale
He:0.003 il:0.000 ks:0.014 intelligent:0.000 Pe:0.000 ale:0.001 <|endoftext
He ilks intelligent leaper
He:0.003 il:0.000 ks:0.014 intelligent:0.000 le:0.000 aper:0.001 <|endoftex
He ilks intelligent apple
He:0.003 il:0.000 ks:0.014 intelligent:0.000 apple:0.000 <|endoftext|>:0.00
He ilks intelligent appeal

He:0.003 il:0.000 ks:0.014 intelligent:0.000 appeal:0.000 <|endoftext|>:0.0
He ilks intelligence people
He:0.003 il:0.000 ks:0.014 intelligence:0.000 people:0.006 <|endoftext|>:0.
He ilks intelligence Peale
He:0.003 il:0.000 ks:0.014 intelligence:0.000 Pe:0.000 ale:0.002 <|endoftex
He ilks intelligence leaper
He:0.003 il:0.000 ks:0.014 intelligence:0.000 le:0.004 aper:0.001 <|endofte
He ilks intelligence apple
He:0.003 il:0.000 ks:0.014 intelligence:0.000 apple:0.000 <|endoftext|>:0.0
He ilks intelligence appeal
He:0.003 il:0.000 ks:0.014 intelligence:0.000 appeal:0.000 <|endoftext|>:0.
He ilks intelligible people
```

```
sentence_probability = dict()
for i, text in enumerate(texts):
    print(text)
    s = 1

    text_token_ids = batch["input_ids"][i,1:]
    text_tokens = [x.strip("ĠĊ") for x in tokenizer.convert_ids_to_tokens(text_toke
    for j, (index, token) in enumerate(zip(text_token_ids, text_tokens)):
        if j < len(tokenizer(text)["input_ids"]) - 2:
          print(probs[i,j,index], j)
          s *= probs[i,j,index]
    sentence_probability[text] = s
    print(s)
    print("")
```

```
1.8046772e-07 1
2.6963878e-06 2
0.0024347224 3
3.0900791703450004e-18

He lips intelligible leaper
0.0026081842 0
1.804677e-07 1
2.6963876e-06 2
0.0024347224 3
3.0900784231192662e-18

He lips intelligible apple
0.0026081845 0
1.804677e-07 1
2.696387e-06 2
1.2691705394141942e-15

He lips intelligible appeal
0.0026081847 0
1.8046772e-07 1
2.696388e-06 2
1.2691713877088914e-15

He lips intelligibly people
0.0026081842 0
1.8046772e-07 1
2.6963878e-06 2
1.2691710540897698e-15

He lips intelligibly Peale
0.0026081842 0
1.804677e-07 1
2.6963876e-06 2
0.017325673 3
2.1989237667176268e-17

He lips intelligibly leaper
0.0026081845 0
1.804677e-07 1
2.696387e-06 2
0.017325671 3
2.198923170337733e-17
```

```
He lips intelligibly apple
0.0026081847 0
1.8046772e-07 1
2.696388e-06 2
1.2691713877088914e-15

He lips intelligibly appeal
0.0026081842 0
1.8046772e-07 1
2.6963878e-06 2
1.2691710540897698e-15

He lips belligerent people
0.0026081842 0
1.804677e-07 1
4.4202068e-07 2
```

```
markdict = sentence_probability
marklist = sorted(markdict.items(), key=lambda x:x[1], reverse=True)
sortdict = dict(marklist)
a = list(sortdict.keys())
print(a[0], sortdict[a[0]])
print(a[1], sortdict[a[1]])
print(a[2], sortdict[a[2]])
```

```
He likes intelligence people 2.770870599368776e-06
He likes intelligence appeal 2.770870599368776e-06
He likes intelligent people 2.770870048382653e-06
```

```
def generate_possible_sentence(sentance):
  texts = generate_correction(sentance)
  tokenizer.pad_token = tokenizer.eos_token
  batch = tokenizer(texts, return_tensors="pt", padding=True).to("cuda")
  # добавляем индекс начала строки (склейка массивов по первой координате)
  batch["input_ids"] = torch.cat([
      torch.ones_like(batch["input_ids"][:,:1])*tokenizer.bos_token_id,
      batch["input_ids"]
  ], dim=1)
  #  batch["attention_mask"] = torch.cat([
  #     torch.ones_like(batch["attention_mask"][:,:1]),
  #     batch["attention_mask"]
  #  ], dim=-1)
  with torch.no_grad():
      logits = model(batch["input_ids"])["logits"]
  probs = torch.softmax(logits, dim=-1).cpu().numpy()
  #print(probs.shape)
  sentence_probability = dict()
  for i, text in enumerate(texts):
      #print(text)
      s = 1

      text_token_ids = batch["input_ids"][i,1:]
      text_tokens = [x.strip("ĠĊ") for x in tokenizer.convert_ids_to_tokens(text_t
      for j, (index, token) in enumerate(zip(text_token_ids, text_tokens)):
          if j < len(tokenizer(text)["input_ids"]) - 2:
            #print(probs[i,j,index], j)
            s *= probs[i,j,index]
          s *= probs[i,j,index]
      sentence_probability[text] = s
      #print(s)
      #print("")
  markdict = sentence_probability
  marklist = sorted(markdict.items(), key=lambda x:x[1], reverse=True)
  sortdict = dict(marklist)
  a = list(sortdict.keys())
  print(a[0], sortdict[a[0]])
  if len(a) > 1:
    print(a[1], sortdict[a[1]])
  if len(a) > 2:
    print(a[2], sortdict[a[2]])
```

```
generate_possible_sentence('He liks intelligen peaple')
```

```
(250, 9, 50257)
He likes intelligence people 2.770870599368776e-06
He likes intelligence appeal 2.770870599368776e-06
He likes intelligent people 2.770870048382653e-06
```

```
generate_possible_sentence('Nice to meet you')
```

```
Nice to meet you 5.292847503609666e-06
```

```
generate_possible_sentence('Nice to ment you')
```

```
Nice to meant you 5.292997817010932e-06
Nice to meat you 5.292997817010932e-06
Nice to lent you 5.292997817010932e-06
```

```
generate_possible_sentence('Where are you from')
```

```
Where are you from 3.5786616254334346e-05
```

```
generate_possible_sentence('Where are you frm')
```

```
Where are you fem 3.5786530514742704e-05
Where are you rm 3.5786530514742704e-05
Where are you farm 3.5786530514742704e-05
```

```
generate_possible_sentence('Wher are you from')
```

```
Where are you from 3.578611335927997e-05
When are you from 2.152745625026188e-06
Her are you from 4.045405216579659e-08
```

```
generate_possible_sentence('Where are yu from')
```

```
Where are y from 3.578633859899552e-05
Where are you from 3.578633859899552e-05
Where are ye from 3.578633859899552e-05
```

```
generate_possible_sentence('Thanks so much for the birthday mony.')
```

✓ 0s completed at 4:25 PM ● ✕