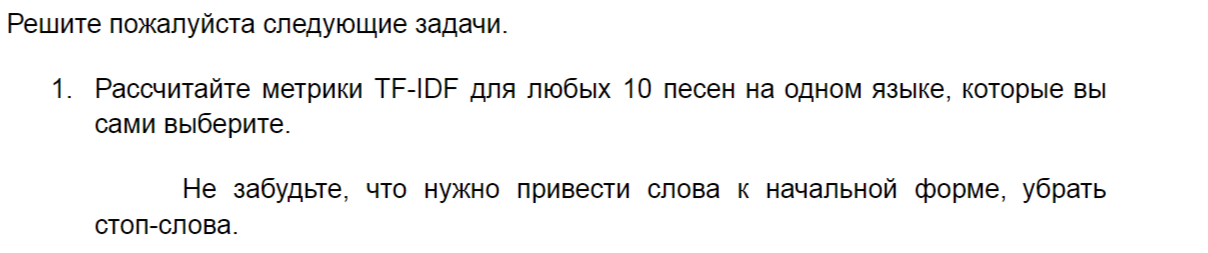
**Решение задач Обработки естественного языка**



# TODO вынести в def

fig = plt.figure()

fig.patch.set\_facecolor('white')

plt.subplots\_adjust(wspace=0.3, hspace=0.2)

i = 1

tokens = word\_tokenize(df['song\_text'][0])

text\_raw = " ".join(tokens)

wordcloud = WordCloud(colormap='Accent', background\_color='white', contour\_width=10).generate(text\_raw)

plt.tick\_params(labelsize=10)

plt.imshow(wordcloud)

plt.axis("off")

plt.title(df['name'][0],fontdict={'fontsize':7,'color':'grey'},y=0.93)

plt.tick\_params(labelsize=10)

i += 1



# Создание объекта TfidfVectorizer

tfidf\_vectorizer = TfidfVectorizer()

# Применение TF-IDF к текстовым данным

tfidf\_matrix = tfidf\_vectorizer.fit\_transform(songs\_list\_norm)

# Получение списка ключевых слов и их значения TF-IDF для первого документа

feature\_names = tfidf\_vectorizer.get\_feature\_names\_out()

tfidf\_scores = tfidf\_matrix.toarray()[0]

# Сортировка слов по значениям TF-IDF

sorted\_keywords = [word for \_, word in sorted(zip(tfidf\_scores, feature\_names), reverse=True)]

print("Ключевые слова:", sorted\_keywords)

count\_vectorizer = CountVectorizer()

bow = count\_vectorizer.fit\_transform(songs\_list\_norm)

bow.shape

(9, 461)

import operator

vocab\_sorted = sorted(count\_vectorizer.vocabulary\_.items(), key=operator.itemgetter(0))

vocab\_sorted

('земной', 148), ('знамя', 149), ('знать', 150), ('значит', 151), ('игла', 152), ('играть', 153), ('идти', 154), ('избитый', 155), ('измениться', 156), ('иллюминатор', 157), ('имя', 158), ('источник', 159), ('кавказ', 160), ('казак', 161), ('казаться', 162), ('какой', 163), ('калач', 164), ('каменный', 165), ('камень', 166), ('камушек', 167), ('камчатка', 168), ('клён', 169), ('конец', 170), ('корабль', 171), ('король', 172), ('космический', 173), ('космодром', 174), ('косяк', 175), ('который', 176), ('красиво', 177), ('крепкий', 178), ('крикнуть', 179), ('кричать', 180), ('круг', 181), ('кто', 182), ('кубань', 183), ('кусок', 184), ('лyчахнуть', 185), ('ласкать', 186), ('ледяной', 187), ('лететь', 188), ('лето', 189), ('летопись', 190), ('лечить', 191), ('листва', 192), ('лицо', 193), ('лишить', 194), ('лишь', 195), ('лужа', 196), ('любовь', 197), ('мало', 198), ('манить', 199), ('маска', 200), ('матовый', 201), ('мать', 202), ('мгла', 203), ('междy', 204), ('менее', 205), ('место', 206), ('метаться', 207), ('метеорит', 208), ('мечтy', 209), ('мешок', 210), ('мир', 211), ('мой', 212), ('монолит', 213), ('море', 214), ('мороз', 215), ('мужество', 216), ('музыка', 217), ('мчаться', 218), ('мы', 219), ('нyжный', 220), ('награда', 221), ('налететь', 222), ('налить', 223), ('напpотить', 224), ('напомнить', 225), ('наравне', 226), ('народ', 227), ('начать', 228), ('наш', 229), ('небо', 230), ('нева', 231), ('недолго', 232), ('немало', 233), ('немой', 234), ('нервировать', 235), ('нести', 236), ('ниагар', 237), ('нидерланды', 238), ('никто', 239), ('новый', 240), ('нога', 241), ('носить', 242), ('ночной', 243), ('нс', 244), ('нужно', 245), ('облако', 246), ('обман', 247), ('обмануть', 248), ('обманчивый', 249), ('общий', 250), ('огонь', 251), ('огромный', 252), ('одесса', 253), ('один', 254), ('ожидать', 255), ('около', 256), ('он', 257), ('она', 258), ('опyстела', 259), ('опасность', 260), ('оправдать', 261), ('опьянять', 262), ('орбита', 263), ('осень', 264), ('остаться', 265), ('острый', 266), ('отвезти', 267), ('ответ', 268), ('отечество', 269), ('отшyмел', 270), ('оформить', 271), ('очень', 272), ('пpистать', 273), ('пpовожy', 274), ('пpогнать', 275), ('пpоидти', 276), ('память', 277), ('перерыв', 278), ('перу', 279), ('песня', 280), ('печаль', 281), ('пламя', 282), ('плачущий', 283), ('плечо', 284), ('плохой', 285), ('поpосло', 286), ('победа', 287), ('повеселиться', 288), ('повторять', 289), ('подарок', 290), ('поддатый', 291), ('подкова', 292), ('поднимать', 293), ('пожалеть', 294), ('позвать', 295), ('показаться', 296), ('покой', 297), ('пол', 298), ('поле', 299), ('польша', 300), ('понимать', 301), ('попёнок', 302), ('поток', 303), ('потренироваться', 304), ('поэт', 305), ('право', 306), ('праздник', 307), ('привет', 308), ('припев', 309), ('прислать', 310), ('притормозить', 311), ('приходить', 312), ('продолжиться', 313), ('простить', 314), ('просто', 315), ('простор', 316), ('прочь', 317), ('прошить', 318), ('проьесть', 319), ('птица', 320), ('пусть', 321), ('путь', 322), ('рада', 323), ('разбиться', 324), ('разговор', 325), ('разлетаться', 326), ('разлететься', 327), ('ранний', 328), ('рассвет', 329), ('рваный', 330), ('рвать', 331), ('рваться', 332), ('ребёнок', 333), ('ресторан', 334), ('риск', 335), ('родина', 336), ('рокот', 337), ('россия', 338), ('рука', 339), ('рукоплескать', 340), ('рюкзак', 341), ('рядом', 342), ('сpедь', 343), ('сам', 344), ('самар', 345), ('сбеpечь', 346), ('сбыться', 347), ('свет', 348), ('свободно', 349), ('святой', 350), ('сеpдцy', 351), ('себя', 352), ('сегодня', 353), ('седой', 354), ('сейчас', 355), ('сердце', 356), ('сидеть', 357), ('сила', 358), ('силач', 359), ('сильный', 360), ('синева', 361), ('сказать', 362), ('сколько', 363), ('скоро', 364), ('скороход', 365), ('слyчилось', 366), ('славный', 367), ('сладко', 368), ('слегка', 369), ('слеза', 370), ('слово', 371), ('случиться', 372), ('слышать', 373), ('смех', 374), ('смешить', 375), ('смешной', 376), ('смочь', 377), ('снегурочка', 378), ('сниться', 379), ('снова', 380), ('снять', 381), ('соpвалиться', 382), ('собственно', 383), ('сойти', 384), ('сон', 385), ('сосна', 386), ('спасти', 387), ('спросить', 388), ('среди', 389), ('сталь', 390), ('становиться', 391), ('старик', 392), ('стать', 393), ('стая', 394), ('стоpоной', 395), ('столик', 396), ('страсть', 397), ('стыд', 398), ('судьба', 399), ('счастие', 400), ('сын', 401), ('тpава', 402), ('тайвань', 403), ('такyть', 404), ('такой', 405), ('такси', 406), ('танцyть', 407), ('танцевать', 408), ('таять', 409), ('твой', 410), ('тело', 411), ('тепеpь', 412), ('тибет', 413), ('толк', 414), ('тонy', 415), ('тонуть', 416), ('тоска', 417), ('тост', 418), ('тот', 419), ('трава', 420), ('традиция', 421), ('трон', 422), ('трудный', 423), ('туда', 424), ('туша', 425), ('ты', 426), ('тысяча', 427), ('тёмный', 428), ('убить', 429), ('угодный', 430), ('узнавать', 431), ('уйти', 432), ('украсть', 433), ('ум', 434), ('умереть', 435), ('усталость', 436), ('форма', 437), ('хмельнyть', 438), ('ходить', 439), ('холодный', 440), ('цветок', 441), ('цепь', 442), ('чyя', 443), ('часы', 444), ('чемy', 445), ('чернеть', 446), ('четыpть', 447), ('четыре', 448), ('чили', 449), ('шyмить', 450), ('шальнyть', 451), ('шампанское', 452), ('шепчy', 453), ('шумный', 454), ('шут', 455), ('эй', 456)

2.Цель этого задания - использовать предварительно обученную модель BERT для классификации тональности отзывов на фильмы

import torch

import transformers as bert\_trained

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from torch.utils.data import Dataset, DataLoader

from sklearn.model\_selection import train\_test\_split

df = pd.read\_csv(f"/content/Film/IMDB Dataset.csv")

| **eview** | **sentiment** |
| --- | --- |
| **0** | One of the other reviewers has mentioned that ... | positive |
| **1** | A wonderful little production. <br /><br />The... | positive |
| **2** | I thought this was a wonderful way to spend ti... | positive |
| **3** | Basically there's a family where a little boy ... | negative |
| **4** | Petter Mattei's "Love in the Time of Money" is... | positive |

df['sentiment'] = df['sentiment'].apply(lambda x: 1 if x=='positive' else 0)

df.head()

| **review** | **sentiment** |
| --- | --- |
| **0** | One of the other reviewers has mentioned that ... | 1 |
| **1** | A wonderful little production. <br /><br />The... | 1 |
| **2** | I thought this was a wonderful way to spend ti... | 1 |
| **3** | Basically there's a family where a little boy ... | 0 |
| **4** | Petter Mattei's "Love in the Time of Money" is... | 1 |

df.describe(include="all")

| **review** | **sentiment** |
| --- | --- |
| **count** | 50000 | 50000.000000 |
| **unique** | 49582 | NaN |
| **top** | Loved today's show!!! It was a variety and not... | NaN |
| **freq** | 5 | NaN |
| **mean** | NaN | 0.500000 |
| **std** | NaN | 0.500005 |
| **min** | NaN | 0.000000 |
| **25%** | NaN | 0.000000 |
| **50%** | NaN | 0.500000 |
| **75%** | NaN | 1.000000 |
| **max** | NaN | 1.000000 |

# удалим знаки препинания и приведем к нижнему регистру

df['review'] = df['review'].apply(lambda x: re.sub(r'[^\w\s]', '', x.lower()))

df

| **review** | **sentiment** |
| --- | --- |
| **0** | one of the other reviewers has mentioned that ... | 1 |
| **1** | a wonderful little production br br the filmin... | 1 |
| **2** | i thought this was a wonderful way to spend ti... | 1 |
| **3** | basically theres a family where a little boy j... | 0 |
| **4** | petter matteis love in the time of money is a ... | 1 |
| **...** | ... | ... |
| **49995** | i thought this movie did a down right good job... | 1 |
| **49996** | bad plot bad dialogue bad acting idiotic direc... | 0 |
| **49997** | i am a catholic taught in parochial elementary... | 0 |
| **49998** | im going to have to disagree with the previous... | 0 |
| **49999** | no one expects the star trek movies to be high... | 0 |

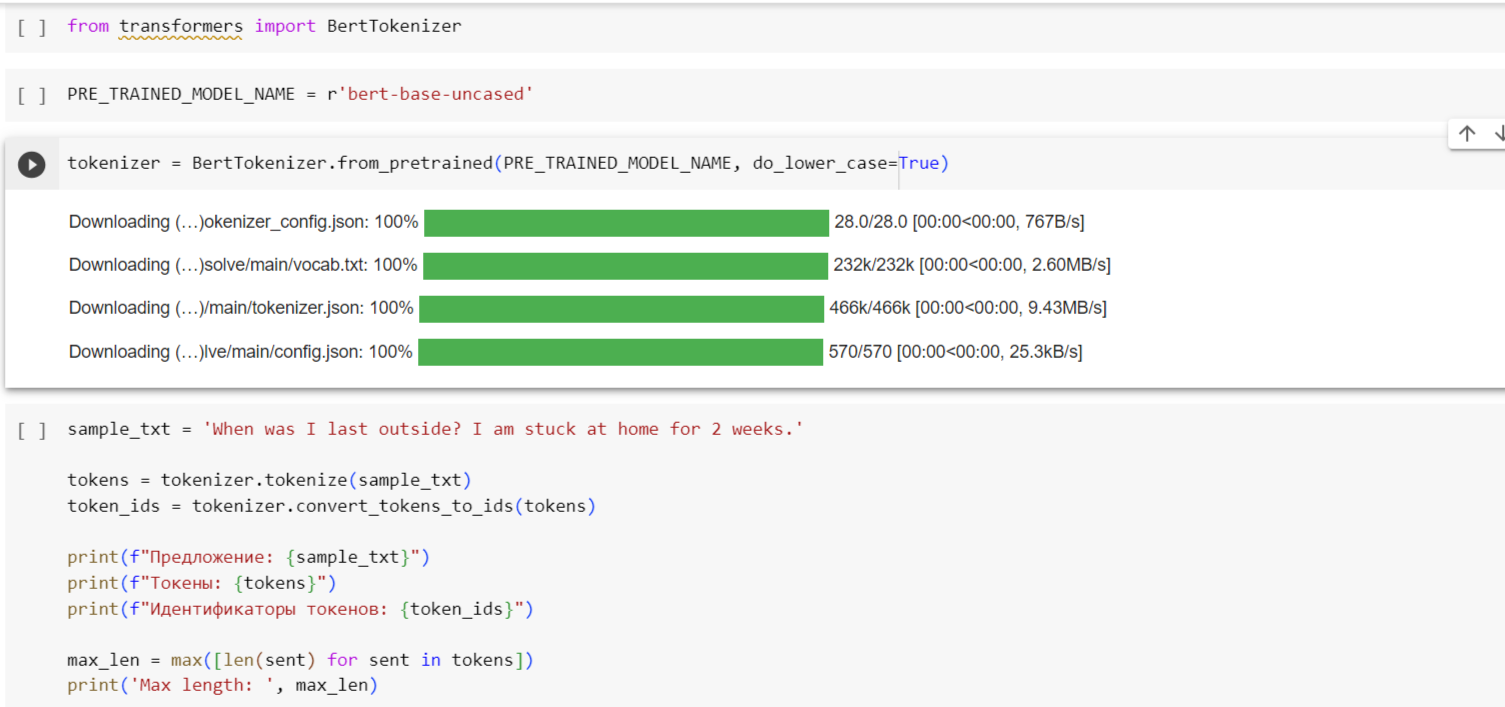
50000 rows × 2 columns

# загрузка и удаление стоп-слов

nltk.download('stopwords')

stop\_words = stopwords.words('english')

Построение bert модели











model\_class, tokenizer\_class, pretrained\_weights = (ppb.BertModel, ppb.BertTokenizer, 'bert-base-uncased')

# Загрузка предобученной модели/токенизатора

tokenizer = tokenizer\_class.from\_pretrained(pretrained\_weights)

model = model\_class.from\_pretrained(pretrained\_weights)

short\_df['review'].value\_counts()

one reviewers mentioned watching 1 oz episode youll hooked right exactly happened mebr first thing struck oz brutality unflinching scenes violence set right word go trust show faint hearted timid show pulls punches regards drugs sex violence hardcore classic use wordbr called oz nickname given oswald maximum security state penitentary focuses mainly emerald city experimental section prison cells glass fronts face inwards privacy high agenda em city home manyaryans muslims gangstas latinos christians italians irish moreso scuffles death stares dodgy dealings shady agreements never far awaybr would say main appeal show due fact goes shows wouldnt dare forget pretty pictures painted mainstream audiences forget charm forget romanceoz doesnt mess around first episode ever saw struck nasty surreal couldnt say ready watched developed taste oz got accustomed high levels graphic violence violence injustice crooked guards wholl sold nickel inmates wholl kill order get away well mannered middle class inmates turned prison bitches due lack street skills prison experience watching oz may become comfortable uncomfortable viewingthats get touch darker side 1 liked william hickey prizzis honor resurrects character anthony mob boss weak godfather satire laughs stuart whitman looks perplexed hes schlockfest morgan fairchilds performance one better efforts movie alone good sign sure eddie deezen vacillates three stooges slapstick bad woody allen imitation fatally flawed mob boss derivative boredom quickly overcomes comedy film drags car chases hidden weapons restaurant bathroom numerous nonsense merk 1 one say wasnt warned read reviews user external like us attracted horror movies curiosity got cat come scream people movie go dark room know thats horror aficionados always dying know whats even know itll badbr bottom line movie left angry pretends real caresgimmicks allowed actors dialogue lame unusual event horror movies even movie bad polite really got mad film rip bwp also halfhearted lazy rip thatbr dont believe sacred cows thought could outdo bwp kudos didnt even try movie made little effort care unforgivable sin horror movie 1 absolutely love film everything almost felt like watching friends screen way movie filmed pure masterpiece original creative related characters even thoughts im really glad ran across movie genius like justin 1 brilliant horror film utterly gruesome scary thing remake john carpenter please let put film simply brilliant start film aliens spacecraft hurtling towards earth centuries mankind walked planet explosion unleashes films title amazing shining white blue stating thing one best opening credits horror film everbr cast actors play twelve man science team joy behold locations setting station antartica visually impressive dvd widescreen must great cinema regret seeing big screenbr kurt russell excellent macready helicopter pilot reluctantly becomes leader men trying combat lethal shape changing monstrosity infiltrated base actors really good create terrific scenes paranoia tension thing infected favourite scene whole film macready tests everyone thats still alive infection tense scary finally spectacular love funny wellbr special mention must go rob bottin truly amazing make effects shape changing designs alien didnt get oscar best visual effects time damn well also debatable whether john carpenters greatest filmits certainly gruesome masterpiecebr wait cold winter night get budweiser fridge sit watch thing horror masterpiece flame throwing heroes fighting shape changing towers gore slimebr utterly brilliantbr ten ten 1 .. mario fan long remember fond memories playing super mario world kid game brought back many memories adding something new super mario galaxy latest installment amazing mario franchise much different game mario still keeping intact greatest elements mario first noticeable difference story takes place spacebr story begins much like mario game mario receives letter princess peach inviting celebration castle mushroom kingdom upon arriving peachs castle mario finds bowser son bowser jr attacking castle airships bowser kidnaps princess peach lifts castle space midst castle lifted space mario falls lands unknown planet mario found talking star named luma taken back lumas home floating space station mario meets many lumas also meets leader woman named rosalina rosalina tells mario bowser taken away space stations power stars scattered across universe mario help lumas find save peach thus adventure beginsbr way play game flying space station galaxies galaxy consists multiple planets mario travels amongst levels via shooting stars retrieve power stars mario many times walk way around planets without losing gravity planets small others big many planets similar classic mario environments best thing game controls stuff like jumping still wiimote used many unique ways game shake remote mario perform spin used primary attack game well activate shooting stars also point remote screen use pointer fire star bits enemies objects environment graphics far best graphics wii hard describe great game looks could probably almost say looks good 360 gamesbr minor gripes going upside effect takes getting used also story pretty weak worst part lose lives turn game matter many last quit restart 4 lives still minor problems aside superb game highly entertaining challenging type game weve waiting wiibr perfect 10 10 1 worst movie ever seen well worst probably ever see see need rehash others said previously forewarnedbr one bad movies think want watch want able make fun plain bad bad bad bad badbr movie equivalent pet rock friend wait wait wait wait wait wait wait wait something happen unfortunately never least pet rock knew getting lions gate completely deceives bombshell nothis disaster watching film would swear george w bush hands making film yes idioticbr stay away unless course want watch worst movie time probably lions gate figured would make money piece tripe 1 well like watch bad horror bmovies cause think interesting see stupidity unability creators shoot seriously good movie always compare movies example spielbergs works againandagain dont understand huge difference see like ed woods movies cause inept funny people chilling funny even interesting extremely boring horror movie without anything makes even bad movies watchable theres acting screenplay direction thrills even blood extremely inept amateurish film definitely worst movie ever seen seen lot worst movies believe warned 110 1 guess would originally going least two parts thus least quarter longer otherwise one explain confused abbreviated storyline never completely lost often partially lost usually unclear character motivation movie feels though joining plot points dropped squeeze time slotbr longer might make sense still wouldnt much good movies interesting idea war zeus hera war male female movie drops ball making heras followers fairly horrible clear zeus followers believe movie also interesting dont see gods theres real certainty exist got couple intriguing ideas doesnt anything useful thembr bad dialog cardboard characters one interesting scene involving hercules three antagonistic sons unwatchable also worth watching 1 saw brothers shadow tribeca film festival loved judd hirsch scott cohen great father son film follows scott cohen parole alaska back family brooklyn shows brother died embarks journey slowly repair estranged relationships brothers wife child father never forgiven black sheep family story takes us deep hearts minds family allows deeply understand complexity lives also imagery woodworking business brooklyn backdrop sets tone rich revealing family portrait 1 Name: review, Length: 300, dtype: int64

tokenized\_df = short\_df['review'].apply((lambda x: tokenizer.encode(x, add\_special\_tokens=True,truncation=True)))

0 [101, 2028, 15814, 3855, 3666, 1015, 11472, 27... 1 [101, 6919, 2210, 2537, 7467, 6028, 14477, 475... 2 [101, 2245, 6919, 2126, 5247, 2051, 2980, 2621... 3 [101, 10468, 2045, 2015, 2155, 2210, 2879, 518... 4 [101, 9004, 3334, 4717, 17580, 2293, 2051, 276... ... 295 [101, 2750, 6135, 22369, 6475, 3049, 17312, 43... 296 [101, 3185, 19237, 4632, 2242, 3684, 16535, 26... 297 [101, 2559, 2830, 6697, 2939, 4258, 2347, 2102... 298 [101, 2183, 2156, 2698, 7038, 2347, 2102, 4415... 299 [101, 2387, 3428, 5192, 5917, 3540, 2143, 2782... Name: review, Length: 300, dtype: object

# Find the maximum length

max\_len = max([len(sent) for sent in tokenized\_df])

print('Max length: ', max\_len)

Max length: 512

short\_df.dtypes

review object sentiment int64 dtype: object

short\_df['tokenized\_review'] = tokenized\_df

<ipython-input-104-322268641750>:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

short\_df['tokenized\_review'] = tokenized\_df

from sklearn.model\_selection import train\_test\_split

X\_train, X\_val, y\_train, y\_val =\

    train\_test\_split(short\_df['tokenized\_review'], short\_df['sentiment'], test\_size=0.1, random\_state=2020)

# Create a function to tokenize a set of texts

def preprocessing\_for\_bert(data):

    """Perform required preprocessing steps for pretrained BERT.

    @param    data (np.array): Array of texts to be processed.

    @return   input\_ids (torch.Tensor): Tensor of token ids to be fed to a model.

    @return   attention\_masks (torch.Tensor): Tensor of indices specifying which

                  tokens should be attended to by the model.

    """

    # Create empty lists to store outputs

    input\_ids = []

    attention\_masks = []

    # For every sentence...

    for sent in data:

        # `encode\_plus` will:

        #    (1) Tokenize the sentence

        #    (2) Add the `[CLS]` and `[SEP]` token to the start and end

        #    (3) Truncate/Pad sentence to max length

        #    (4) Map tokens to their IDs

        #    (5) Create attention mask

        #    (6) Return a dictionary of outputs

        encoded\_sent = tokenizer.encode\_plus(

            text=text\_preprocessing(sent),  # Preprocess sentence

            add\_special\_tokens=True,        # Add `[CLS]` and `[SEP]`

            max\_length=MAX\_LEN,                  # Max length to truncate/pad

            pad\_to\_max\_length=True,         # Pad sentence to max length

            #return\_tensors='pt',           # Return PyTorch tensor

            return\_attention\_mask=True      # Return attention mask

            )

        # Add the outputs to the lists

        input\_ids.append(encoded\_sent.get('input\_ids'))

        attention\_masks.append(encoded\_sent.get('attention\_mask'))

    # Convert lists to tensors

    input\_ids = torch.tensor(input\_ids)

    attention\_masks = torch.tensor(attention\_masks)

    return input\_ids, attention\_masks

# Specify `MAX\_LEN`

MAX\_LEN = 64

# Print sentence 0 and its encoded token ids

token\_ids = list(preprocessing\_for\_bert([X[0]])[0].squeeze().numpy())

print('Original: ', X[0])

print('Token IDs: ', token\_ids)

# Run function `preprocessing\_for\_bert` on the train set and the validation set

print('Tokenizing data...')

train\_inputs, train\_masks = preprocessing\_for\_bert(X\_train)

val\_inputs, val\_masks = preprocessing\_for\_bert(X\_val)