CNN/Daily Mail dataset

Описание

CNN/Daily Mail - это набор данных для обобщения текста. Сгенерированные человеком абстрактные сводные маркеры были сгенерированы из новостных сюжетов на веб-сайтах CNN и Daily Mail в виде вопросов (с одним из скрытых объектов), а истории - в виде соответствующих отрывков, из которых система, как ожидается, ответит на вопрос "заполните пробел". Авторы выпустили скрипты, которые сканируют, извлекают и генерируют пары отрывков и вопросов с этих веб-сайтов.

В целом, датасет содержит 286 817 обучающих пар, 13 368 проверочных пар и 11 487 тестовых пар, как определено их скриптами. Исходные документы в обучающем наборе содержат 766 слов, охватывающих в среднем 29,74 предложения, в то время как резюме состоят из 53 слов и 3,72 предложений.

Кратко о датасете (из статьи Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond)

Существующие своды резюме абстрактного текста, включая Gigaword и DUC, состоят только из одного предложения в каждом резюме. CNN/Daily Mail - свод, который включает резюме из нескольких предложений. Чтобы создать этот свод, модифицировался существующий свод, который использовался для задания ответов на вопросы, основанные на отрывках (Hermann et al., 2015). В этой работе авторы использовали сгенерированные человеком абстрактные сводные маркеры из новых статей на веб-сайтах CNN и Daily Mail в качестве вопросов (со скрытой одной из сущностей), а истории - в качестве соответствующих отрывков, из которых система, как ожидается, ответит на вопрос "заполните пробел". С помощью простой модификации скрипта были восстановлены все сводные маркеры каждой истории в исходном порядке, чтобы получить резюме из нескольких предложений, где каждый маркер рассматривается как предложение.

Вычислительные затраты: авторы использовали один графический процессор Tesla K40 для обучения моделей на этом наборе данных. В то время как плоские модели (words-lvt2k и wordslvt2k-ptr) занимали менее 5 часов на эпоху, иерархический механизм внимания был очень дорогой, потребляя почти 12,5 часов на эпоху. Конвергенция всех моделей также происходит медленнее в этом наборе данных по сравнению с Gigaword, занимая почти 35 эпох для всех моделей. Таким образом, время обучения до конвергенции составляет около 7 дней для плоских моделей, но почти 18 дней для иерархического механизма внимания. Декодирование также происходит медленнее, с пропускной способностью 2 примера в секунду для плоских моделей и 1,5 примера в секунду для

иерархической модели внимания, при запуске на одном графическом процессоре с размером батча 1.

Оценка: авторы оценивали свои модели, используя полноразмерную метрику Rouge F1.

Результаты: Результаты базового кодера-декодера, а также иерархического механизма внимания показаны в таблице ниже. Хотя этот набор данных меньше и сложнее, чем корпус Gigaword, интересно отметить, что цифры Rouge находятся в том же диапазоне. Однако иерархический механизм внимания, описанный в разделе 2.4, лишь незначительно превосходит базовый декодер внимания.

Model	Rouge-1	Rouge-2	Rouge-L
words-lvt2k	32.49	11.84	29.47
words-lvt2k-hieratt	32.75	12.21	29.01
words-lvt2k-temp-att	***35.46**	***13.30**	***32.65**

После визуального осмотра выходных данных системы авторы статьи заметили, что в этом наборе данных обе эти модели создавали сводки, которые иногда содержат повторяющиеся фразы или даже предложения. Поскольку резюме в этом наборе данных содержат несколько предложений, вполне вероятно, что декодер "забывает", какая часть документа использовалась при создании более ранних основных моментов. Чтобы преодолеть эту проблему, они использовали модель временного внимания Шанкарана и др. (2016), который отслеживает прошлые веса внимания декодера и явно не рекомендует ему уделять внимание тем же частям документа на будущих временных шагах. Модель работает, как показано следующими простыми уравнениями:

$$\beta_t = \sum_{k=1}^{t-1} \alpha_k^{'};\ \alpha_t \propto \frac{\alpha_t^{'}}{\beta_t}$$

где $\alpha_k^{'}$ - ненормированный вектор весов внимания на t^{th} временном шаге декодера. Другими словами, временная модель внимания уменьшает веса внимания на текущем временном шаге, если прошлые веса внимания высоки в той же части документа.

Используя эту стратегию, модель временного внимания значительно повышает производительность как по сравнению с базовой моделью, так и по сравнению с иерархической моделью внимания.

Модели, применяющиеся к датасету (Abstractive Text Summarization)

Rank	Model	Rouge-1	Rouge-2	Rouge-L

Rank	Model	Rouge-1	Rouge-2	Rouge-L
1	*Pegasus 2B + SLiC*	47.97	24.18	44.88
2	*BRIO*	47.78	23.55	44.57
3	*PEGASUS + SummaReranker*	47.16	22.61	43.87
4	*BART + SimCLS*	46.67	22.15	43.54
5	*GLM-XXLarge*	44.7	21.4	41.4
6	*BART + R-Drop*	44.51	21.58	41.24
7	*CoCoNet + CoCoPretrain*	44.5	21.55	41.24
8	*MUPPET BART Large*	44.45	21.25	41.4
9	*CoCoNet*	44.39	21.41	41.05
10	*BART+R3F*	44.38	21.53	41.17
11	*ERNIE-GENLARGE (large-scale text corpora)*	44.31	21.35	41.6
12	*PALM*	44.3	21.12	41.41
13	*ProphetNet*	44.2	21.17	41.3
14	*PEGASUS*	44.17	21.47	41.11
15	*BART*	44.16	21.28	40.9
16	*ERNIE-GENLARGE*	44.02	21.17	41.26
17	*LongT5*	43.94	21.4	41.28
18	*T5*	43.52	21.55	40.69
19	*UniLMv2*	43.16	20.42	40.14
20	*UniLM*	43.08	20.43	40.34

Пример данных

Для каждого экземпляра существует строка для статьи, строка для основных моментов и строка для идентификатора.

Поля данных
статьи: строка, содержащая текст новостной статьи
основные моменты: строка, содержащая основные моменты статьи, написанные автором статьи
идентификатор: строка, содержащая максимально отформатированный хэш SHA1 URL-адреса, с которого была извлечена статья

{'article': 'Editor's note: In our Behind the Scenes series, CNN correspondents share their experiences in covering news and analyze the stories behind the events. Here, Soledad O'Brien takes users inside a jail where many of the inmates are mentally ill. An inmate housed on the "forgotten floor," where many mentally ill inmates are housed in Miami before trial. MIAMI, Florida (CNN) -- The ninth floor of the Miami-Dade pretrial detention facility is dubbed the "forgotten floor." Here, inmates with the most severe mental illnesses are incarcerated until they're ready to appear in court. Most often, they face

drug charges or charges of assaulting an officer --charges that Judge Steven Leifman says are usually "avoidable felonies." He says the arrests often result from confrontations with police. Mentally ill people often won't do what they're told when police arrive on the scene -- confrontation seems to exacerbate their illness and they become more paranoid, delusional, and less likely to follow directions, according to Leifman. So, they end up on the ninth floor severely mentally disturbed, but not getting any real help because they're in jail. We toured the jail with Leifman. He is well known in Miami as an advocate for justice and the mentally ill. Even though we were not exactly welcomed with open arms by the guards, we were given permission to shoot videotape and tour the floor. Go inside the 'forgotten floor' » . At first, it's hard to determine where the people are. The prisoners are wearing sleeveless robes. Imagine cutting holes for arms and feet in a heavy wool sleeping bag -- that's kind of what they look like. They're designed to keep the mentally ill patients from injuring themselves. That's also why they have no shoes, laces or mattresses. Leifman says about one-third of all people in Miami-Dade county jails are mentally ill. So, he says, the sheer volume is overwhelming the system, and the result is what we see on the ninth floor. Of course, it is a jail, so it's not supposed to be warm and comforting, but the lights glare, the cells are tiny and it's loud. We see two, sometimes three men -- sometimes in the robes, sometimes naked, lying or sitting in their cells. "I am the son of the president. You need to get me out of here!" one man shouts at me. He is absolutely serious, convinced that help is on the way -- if only he could reach the White House. Leifman tells me that these prisoner-patients will often circulate through the system, occasionally stabilizing in a mental hospital, only to return to jail to face their charges. It's brutally unjust, in his mind, and he has become a strong advocate for changing things in Miami. Over a meal later, we talk about how things got this way for mental patients. Leifman says 200 years ago people were considered "lunatics" and they were locked up in jails even if they had no charges against them. They were just considered unfit to be in society. Over the years, he says, there was some public outcry, and the mentally ill were moved out of jails and into hospitals. But Leifman says many of these mental hospitals were so horrible they were shut down. Where did the patients go? Nowhere. The streets. They became, in many cases, the homeless, he says. They never got treatment. Leifman says in 1955 there were more than half a million people in state mental hospitals, and today that number has been reduced 90 percent, and 40,000 to 50,000 people are in mental hospitals. The judge says he's working to change this. Starting in 2008, many inmates who would otherwise have been brought to the "forgotten floor" will instead be sent to a new mental health facility -- the first step on a journey toward long-term treatment, not just punishment. Leifman says it's not the complete answer, but it's a start. Leifman says the best part is that it's a win-win solution. The patients win, the families are relieved, and the state saves money by simply not cycling these prisoners through again and again. And, for Leifman, justice is served. E-mail to a friend .'

'highlights': 'Mentally ill inmates in Miami are housed on the "forgotten floor" Judge Steven Leifman says most are there as a result of "avoidable felonies" While CNN tours facility, patient shouts: "I am the son of the president" Leifman says the system is unjust and he's fighting for change .'

'id': 'ee8871b15c50d0db17b0179a6d2beab35065f1e9'}

Пример кода для загрузки тестового набора

In [1]:
```
from datasets import load_dataset
cnn_dm = load_dataset("cnn_dailymail", "3.0.0");
```

```
C:\Users\simpl\AppData\Local\Programs\Python\Python310\lib\site-packages\tqdm\auto.py:22: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
Found cached dataset cnn_dailymail (C:/Users/simpl/.cache/huggingface/datasets/cnn_dailymail/3.0.0/3.0.0/1b3c71476f6d152c31c1730e83ccb08bcf23e348233f4fcc11e182248e6bf7de)
100%|██████████| 3/3 [00:00<00:00, 22.63it/s]
```

In [2]:
```
cnn_dm['train'][0]
```

Out[2]:
```
{'article': 'LONDON, England (Reuters) -- Harry Potter star Daniel Radcliffe gains access to a reported £20 million ($41.1 million) fortune as he turns 18 on Monday, but he insists the money won\'t cast a spell on him. Daniel Radcliffe as Harry Potter in "Harry Potter and the Order of the Phoenix" To the disappointment of gossip columnists around the world, the young actor says he has no plans to fritter his cash away on fast cars, drink and celebrity parties. "I don\'t plan to be one of those people who, as soon as they turn 18, suddenly buy themselves a massive sports car collection or something similar," he told an Australian interviewer earlier this month. "I don\'t think I\'ll be particularly extravagant. "The things I like buying are things that cost about 10 pounds -- books and CDs and DVDs." At 18, Radcliffe will be able to gamble in a casino, buy a drink in a pub or see the horror film "Hostel: Part II," currently six places below his number one movie on the UK box office chart. Details of how he\'ll mark his landmark birthday are under wraps. His agent and publicist had no comment on his plans. "I\'ll definitely have some sort of party," he said in an interview. "Hopefully none of you will be reading about it." Radcliffe\'s earnings from the first five Potter films have been held in a trust fund which he has not been able to touch. Despite his growing fame and riches, the actor says he is keeping his feet firmly on the ground. "People are always looking to say \'kid star goes off the rails,\'" he told reporters last month. "But I try very hard not to go that way because it would be too easy for them." His latest outing as the boy wizard in "Harry Potter and the Order of the Phoenix" is breaking records on both sides of the Atlantic and he will reprise the role in the last two films.  Watch I-Reporter give her review of Potter\'s latest » . There is life beyond Potter, however. The Londoner has filmed a TV movie called "My Boy Jack," about author Rudyard Kipling and his son, due for release later this year. He will also appear in "December Boys," an Australian film about four boys who escape an orphanage. Earlier this year, he made his stage debut playing a tortured teenager in Peter Shaffer\'s "Equus." Meanwhile, he is braced for even closer media scrutiny now that he\'s legally an adult: "I just think I\'m going to be more sort of fair game," he told Reuters. E-mail to a friend . Copyright 2007 Reuters. All rights reserved.This material may not be published, broadcast, rewritten, or redistributed.',
 'highlights': "Harry Potter star Daniel Radcliffe gets £20M fortune as he turns 18 Monday .\nYoung actor says he has no plans to fritter his cash away .\nRadcliffe's earnings from first five Potter films have been held in trust fund .",
 'id': '42c027e4ff9730fbb3de84c1af0d2c506e41c3e4'}
```

Тестируемые задачи, метрики

Задачи:

- Text Generation
- Question Answering
- Text Summarization
- Sequence-to-sequence Language Modeling
- Summarization
- Abstractive Text Summarization
- Document Summarization
- Extractive Text Summarization
- Extractive Document Summarization

Метрики:

- ROUGE-1
- ROUGE-2
- ROUGE-L

ROUGE - это набор метрик и программный пакет, используемый для оценки программного обеспечения для автоматического суммирования и машинного перевода при обработке естественного языка. Показатели сравнивают автоматически созданное резюме или перевод со ссылкой или набором ссылок (созданных человеком) резюме или перевода.
ROUGE-N измеряет количество совпадающих ‘n-граммов’ (N-grams) между текстом, сгенерированным нашей моделью, и ‘ссылкой’.

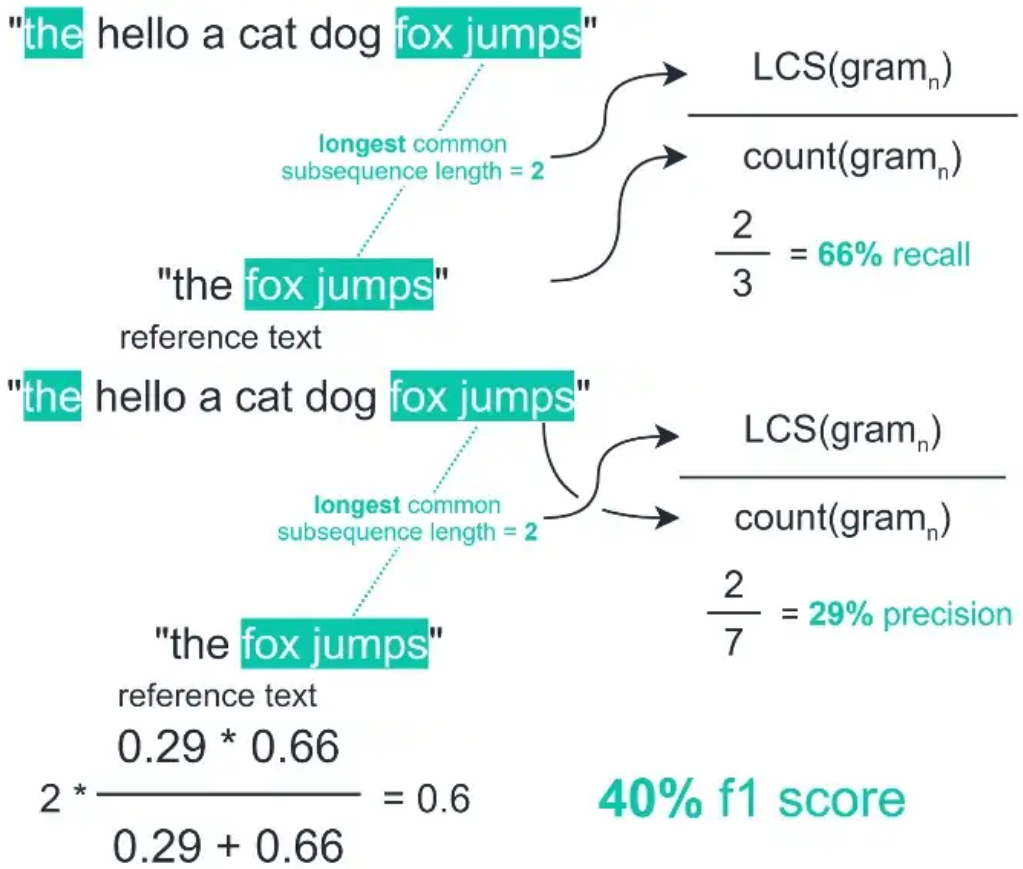
N-грамм - это просто группировка токенов / слов. Униграмма (1-gram) будет состоять из одного слова. Биграмма (2-gram) состоит из двух последовательных слов:
Оригинал: "the quick brown fox jumps over"
Униграммы: ['the', 'quick', 'brown', 'fox', 'jumps', 'over']
Биграммы: ['the quick', 'quick brown', 'brown fox', 'fox jumps', 'jumps over']

Ссылка представляет собой результат, созданный человеком в лучшем случае, поэтому для автоматической саммаризации это будет созданное человеком резюме входного текста.

ROUGE-L измеряет самую длинную общую подпоследовательность (LCS) между выводом модели и ссылкой. Т.е. подсчитывается самая длинная последовательность

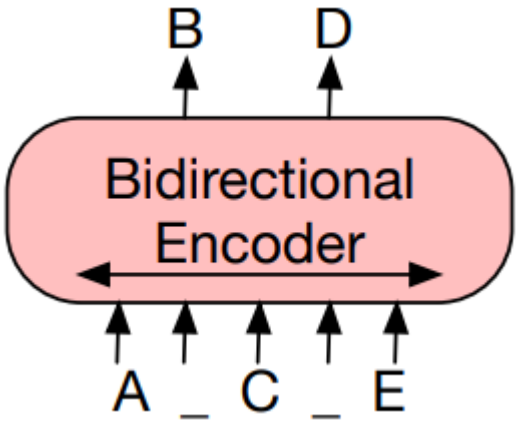
токенов, которая является общей для обоих:

Идея здесь заключается в том, что более длинная общая последовательность указывала бы на большее сходство между двумя последовательностями. Мы можем применить наши вычисления отзыва и точности, как и раньше, но на этот раз мы заменяем совпадение на LCS:

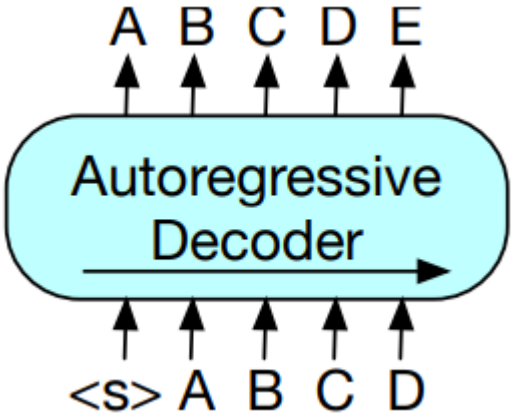


Модель BART

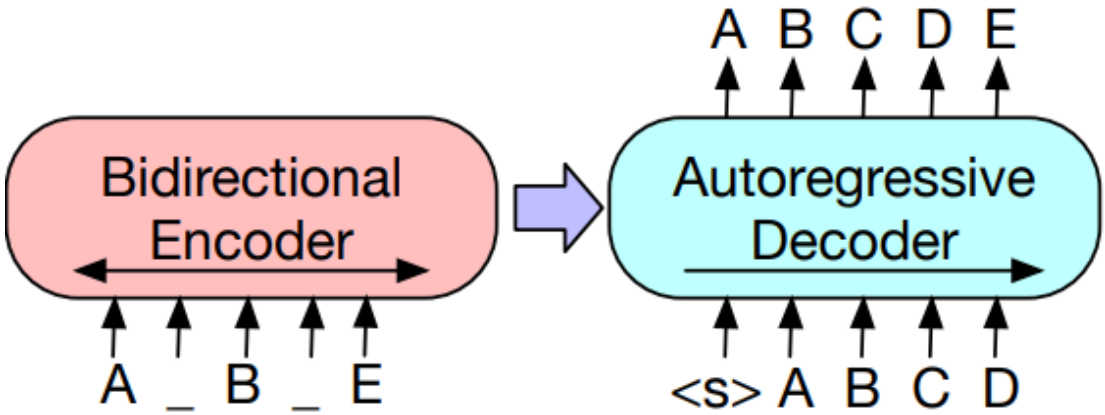
Схематическое сравнение BART с BERT (Devlin et al., 2019) и GPT (Radford et al., 2018):



(a) BERT: случайные токены заменяются масками, и документ кодируется двунаправленно. Отсутствующие токены предсказываются независимо, поэтому BERT не может быть легко использован для генерации.



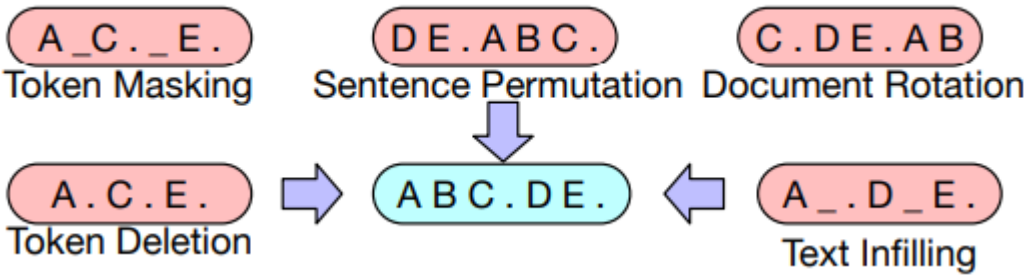
(b) GPT: токены прогнозируются авторегрессионно, что означает, что GPT можно использовать для генерации. Однако слова могут зависеть только от левого контекста, поэтому он не может изучать двунаправленные взаимодействия.



(c) BART: входы кодера не обязательно должны быть выровнены с выходами декодера, что позволяет произвольно преобразовывать шум. Здесь документ был поврежден путем замены фрагментов текста символами маски. Поврежденный документ (слева) кодируется с помощью двунаправленной модели, а затем вероятность исходного документа (справа) вычисляется с помощью авторегрессионного декодера. Для точной настройки неповрежденный документ вводится как в кодер, так и в декодер, и мы используем представления из конечного скрытого состояния декодера.

BART использует стандартную архитектуру преобразования последовательности в

последовательность из (Vaswani et al., 2017), за исключением того, что, следуя GPT, функции активации модифицированы ReLU для GeLUs (Hendrycks & Gimpel, 2016) и инициализированы параметры из N (0, 0.02). Для базовой модели авторы использовали 6 слоев в кодере и декодере, а для большой модели - по 12 слоев в каждом. Архитектура тесно связана с архитектурой, используемой в BERT, со следующими отличиями: (1) каждый уровень декодера дополнительно выполняет перекрестное внимание к конечному скрытому слою кодера (как в модели преобразования последовательности в последовательность); и (2) BERT использует дополнительную подачу-прямая сеть перед предсказанием слов, чего не делает BART. В общей сложности BART содержит примерно на 10% больше параметров, чем модель BERT аналогичного размера.



Преобразования для шумоподавления входных данных, с которыми экспериментируют авторы.

Маскировка токенов.

Следуя за BERT (Devlin et al., 2019), отбираются случайные токены и заменяются элементами [MASK].

Удаление токена.

Случайные токены удаляются из входных данных. В отличие от маскировки токенов, модель должна решить, в каких позициях отсутствуют входные данные.

Заполнение текста.

Отбирается несколько текстовых промежутков, причем длины промежутков берутся из распределения Пуассона (λ = 3). Каждый интервал заменяется одним токеном [MASK]. Промежутки длиной 0 соответствуют вставке токенов [MASK]. Заполнение текста вдохновлено SpanBERT (Joshi et al., 2019), но SpanBERT выбирает длины интервалов из другого (ограниченного геометрического) распределения и заменяет каждый интервал последовательностью токенов [MASK] точно такой же длины. Заполнение текстом учит модель предсказывать, сколько токенов отсутствует в промежутке.

Перестановка предложений.

Документ делится на предложения на основе полных остановок, и эти предложения перемешиваются в случайном порядке.

Поворот документа.

Маркер выбирается равномерно случайным образом, и документ поворачивается так, чтобы он начинался с этого маркера. Эта задача обучает модель определять начало документа.

Ниже приведены два варианта с применением модели. Первый через AutoTokenizer

и AutoModel, второй через pipeline.

Часто мы хотим автоматически получить соответствующую модель, присвоив имя предварительно подготовленной конфигурации. Это возможно благодаря Huggingface AutoClasses. AutoClasses разделяются в AutoConfig, AutoModel и AutoTokenizer. Создание экземпляра одного из них в отношении имени модели или пути создаст соответствующую архитектуру для модели, имя которой указано.

In [3]:
```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

tokenizer = AutoTokenizer.from_pretrained("facebook/bart-large-cnn")

model = AutoModelForSeq2SeqLM.from_pretrained("facebook/bart-large-cnn")

tokens_input = tokenizer.encode("summarize: "+cnn_dm['train'][0]['article'], ret
ids = model.generate(tokens_input, min_length=80, max_length=120)
summary = tokenizer.decode(ids[0], skip_special_tokens=True)

print(summary)
```

```
Harry Potter star Daniel Radcliffe turns 18 on Monday. He gains access to a rep
orted £20 million ($41.1 million) fortune as he turns 18. Radcliffe's earnings
from the first five Potter films have been held in a trust fund. The Londoner h
as filmed a TV movie called "My Boy Jack" due for release later this year.. He
will also appear in "December Boys," an Australian film about four boys.
```

У трансформеров HuggingFace есть возможность загрузить модель с помощью pipeline, и это самый простой способ попробовать и посмотреть, как работает модель.

Pipeline содержит в фоновом режиме сложный код из библиотеки transformers и представляет собой API для множества задач, таких как суммирование, анализ настроений, распознавание именованных объектов и многих других.

In [4]:
```
from transformers import pipeline

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

for i in range(3):
    ARTICLE = cnn_dm['train'][i]['article']
    print(summarizer(ARTICLE, max_length=130, min_length=30, do_sample=False))
```

```
[{'summary_text': "Harry Potter star Daniel Radcliffe turns 18 on Monday. He ga
ins access to a reported £20 million ($41.1 million) fortune. Radcliffe's earni
ngs from the first five Potter films have been held in a trust fund."}]
[{'summary_text': 'Judge Steven Leifman is an advocate for justice and the ment
ally ill. About one-third of all people in Miami-Dade county jails are mentally
ill, he says. He says the sheer volume is overwhelming the system. Starting in
2008, many inmates will be sent to a new mental health facility.'}]
[{'summary_text': 'NEW: "I probably had a 30-, 35-foot free fall," survivor Gar
y Babineau says. NEW: "My truck was completely face down, pointed toward the gr
ound, and my truck got ripped in half," he says. Emergency room physician John
Hink rushed to the scene in 15 minutes. Rescue effort was controlled and organi
zed, he says; opposite of lightning-quick collapse.'}]
```

In [5]:
```
from transformers import pipeline
```

```
summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

summarizers = []

for i in range(500):
    ARTICLE = cnn_dm['train'][i]['article']
    summarizers.append(summarizer(ARTICLE, max_length=130, min_length=30, do_sam
```

```
Your max_length is set to 130, but you input_length is only 89. You might consi
der decreasing max_length manually, e.g. summarizer('...', max_length=44)
```

Подготовка данных

In [6]:
```
from nltk.tokenize import WhitespaceTokenizer
import re

tokenizer_w = WhitespaceTokenizer()
def preprocess_text(text, category):
    res = []

    for k in range(500):
        if (category == 'summary_text'):
            tokenized_list = tokenizer_w.tokenize(text[k][0][category])
        else:
            tokenized_list = tokenizer_w.tokenize(text[k][category])
        temp_list = []
        for i in range(len(tokenized_list)):
            if (bool(re.search(r"[A-z]{1,25}", tokenized_list[i]))):
                temp_list.append(tokenized_list[i])
        for i in range(len(temp_list)):
            if (re.match(r"[^A-z]", temp_list[i][0])):
                temp_list[i] = temp_list[i][1:]
            if (re.match(r"[^A-z]", temp_list[i][-1]) and temp_list[i][-1] != "'
                temp_list[i] = temp_list[i][:-1]

        for i in temp_list:
            if i not in res:
                res.append(i.lower())
    return res

articles_prepocessed = preprocess_text(cnn_dm['train'], 'article')
summaries_preprocessed = preprocess_text(summarizers, 'summary_text')
```

Применение Word2Vec

In [7]:
```
import gensim
from gensim.models import Word2Vec

model_articles = gensim.models.Word2Vec([articles_prepocessed], min_count = 1, v
model_summaries = gensim.models.Word2Vec([summaries_preprocessed], min_count = 1
```

In [8]:
```
#get the model's vocabulary size
max_size = len(model_articles.wv.index_to_key)-1
```

In [9]:
```
import tensorflow as tf
```

```
#initializing tf session
sess = tf.compat.v1.InteractiveSession()
```

In [10]:
```
from torch.utils.tensorboard import SummaryWriter

#with SummaryWriter,we save summary and events to the event file
writer = SummaryWriter('logs', sess.graph)
```

In [11]:
```
from tensorboard.plugins import projector

# Initialize the projectors and add the embeddings
config = projector.ProjectorConfig()
embed= config.embeddings.add()
```

In [12]:
```
#specify our tensor_name as embedding and metadata_path to the metadata.tsv file
embed.tensor_name = 'articles'
embed.metadata_path = 'metadata.tsv'
```

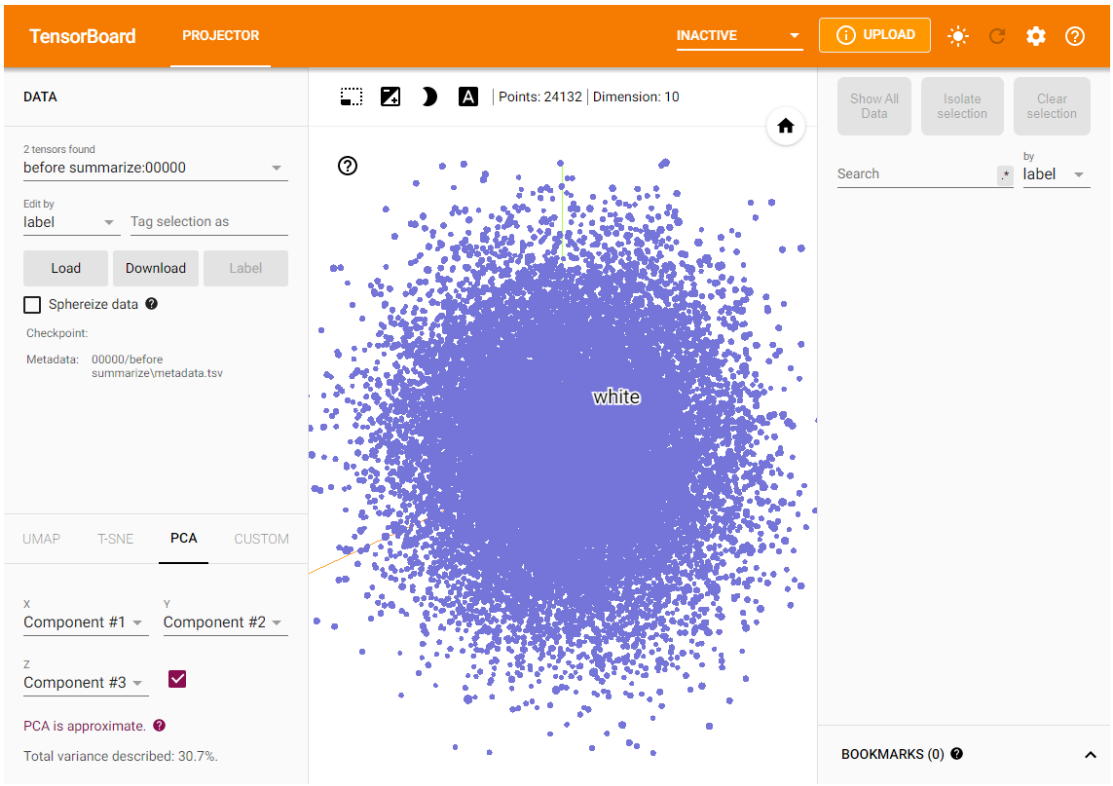
In [13]:
```
import tensorboard as tb
tf.io.gfile = tb.compat.tensorflow_stub.io.gfile

v1 = tf.Variable(1, name='v1')
v2 = tf.Variable(2, name='v2')
saver = tf.compat.v1.train.Saver([v1,v2])

writer.add_embedding(model_articles.wv.vectors, metadata=model_articles.wv.index
writer.add_embedding(model_summaries.wv.vectors, metadata=model_summaries.wv.inc
writer.close()
```

```
WARNING:tensorflow:Saver is deprecated, please switch to tf.train.Checkpoint or
tf.keras.Model.save_weights for training checkpoints. When executing eagerly va
riables do not necessarily have unique names, and so the variable.name-based lo
okups Saver performs are error-prone.
```

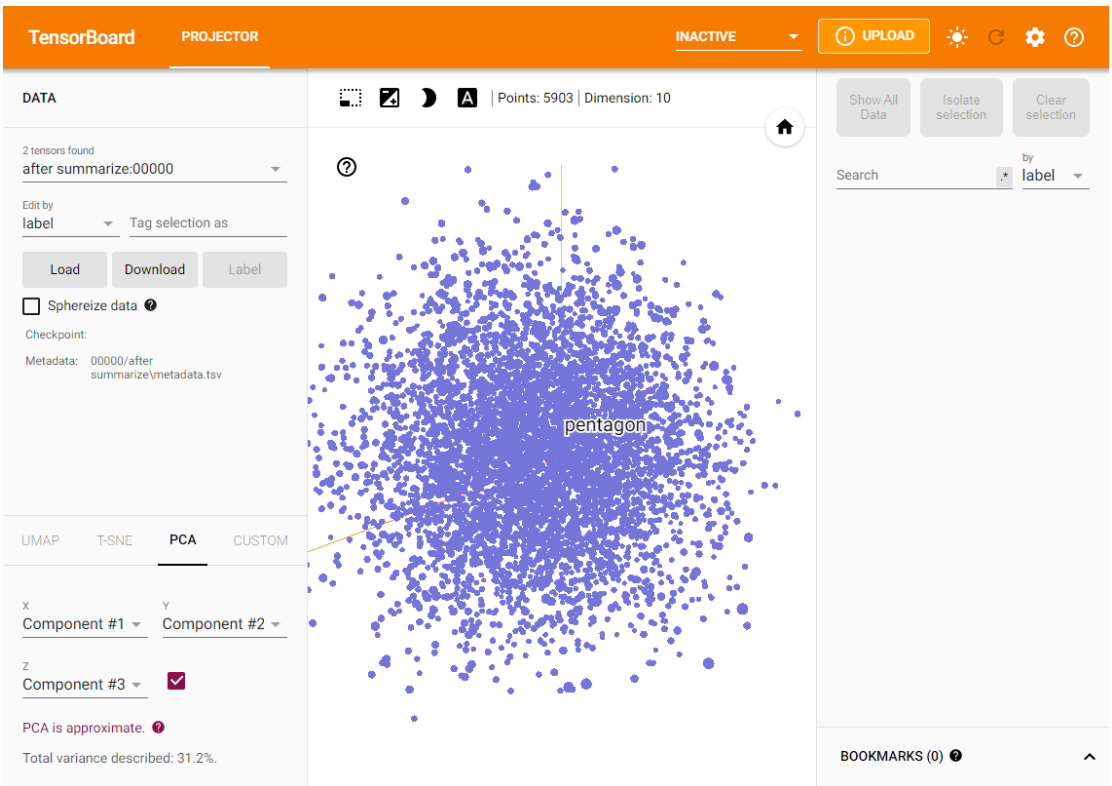
In [14]:
```
%load_ext tensorboard
```

In [15]:
```
%tensorboard --logdir=logs --host=localhost
```

Tensorboard before summarize



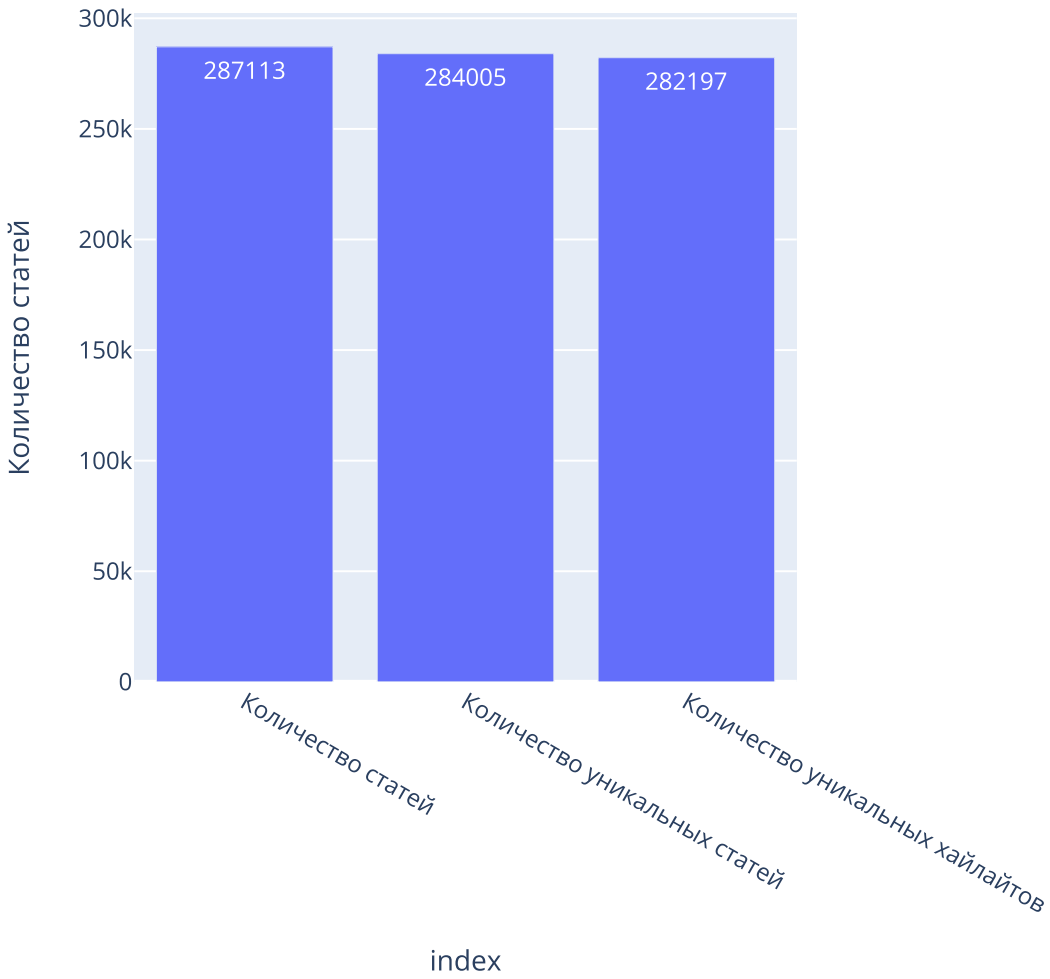
Tensorboard after summarize



Статистические характеристики датасета в plotly

In [16]:
```
import pandas as pd
import plotly.express as px

data = cnn_dm['train'].to_pandas()
```

```
count_rows = data['article'].count()
count_unique_articles = data['article'].nunique()
count_unique_highlights = data['highlights'].nunique()
temp_dict = {"Количество статей": [count_rows, count_unique_articles,count_uniqu
ind = ["Количество статей","Количество уникальных статей","Количество уникальных
df = pd.DataFrame(temp_dict, index = ind)

fig = px.bar(df, y='Количество статей', text='Количество статей')

fig.show()
```



Выборка городов и авторов статей

In [17]:
```
res_lst_cnn=[]
for i in range(90151):
    temp = cnn_dm['train'][i]['article']
    res = ''
    lst_temp = re.findall(r'[A-Z]{2,25}, [A-Z][a-z]{1,30}', temp)
    if (lst_temp!=[] and re.match(r'[I]{1,}', lst_temp[0])):
        lst_temp=[]
    if lst_temp==[]:
        lst_temp = re.findall(r'([A-Z]{1,30})(,|, | |\. |\.)((\(CNN\))|([A-Z][a-
    if (lst_temp != []):
        if (type(lst_temp[0]) == tuple):
```

```
            res += lst_temp[0][0]
        elif (type(lst_temp[0] == str)):
            res += lst_temp[0].split(',')[0]
    if (len(res) == 1):
        res = ''
    if res!='':
        res_lst_cnn.append(res)


res_lst_dm = []
for i in range(90151, 287113):

    temp_dm = cnn_dm['train'][i]['article']
    lst_temp_dm = re.findall(r'(By)( \. )([a-zA-z ]{1,})( \. )', temp_dm)
    if (lst_temp_dm!=[]):
        res_lst_dm.append(lst_temp_dm[0][2])
```

Частотное распределение

In [18]:
```
from nltk import FreqDist

freq_dist_cnn = dict(FreqDist(res_lst_cnn).most_common())
freq_dist_dm = dict(FreqDist(res_lst_dm).most_common())
```

In [19]:
```
import plotly.graph_objects as go

fig_cnn = go.Figure(data=[go.Table(header=dict(values=['Город', 'Как часто встре
                 cells=dict(values=[list(freq_dist_cnn.keys()), list(freq_dist_c
                    ])
fig_cnn.update_layout(
    autosize=True,
    height=610,
    paper_bgcolor="LightSteelBlue",

)
fig_cnn.show()
```

Город	Как часто встречалась новость
WASHINGTON	1477
LONDON	1209
YORK	594
CNN	542
ANGELES	495
ATLANTA	307
BAGHDAD	230
ISLAMABAD	152
CEO	135
CITY	130
DELHI	116
BEIJING	109
UK	107
TEHRAN	100
KABUL	97
MIAMI	96
TMV	89
TV	81
MADRID	75
JERUSALEM	74
PARIS	72
MOSCOW	71

In [24]:
```
fig_dm = go.Figure(data=[go.Table(header=dict(values=['Репортер', 'Как часто вст
                 cells=dict(values=[list(freq_dist_dm.keys()), list(freq_dist_dm
                    ])
fig_dm.update_layout(
    autosize=True,
    height=610,
    paper_bgcolor="LightSteelBlue",

)
fig_dm.show()
```

Репортер	Как часто встречалась новость от него
Daily Mail Reporter	13321
Associated Press	2162
Mark Duell	1442
Bianca London	1203
Jill Reilly	1052
Martin Robinson	957
Leon Watson	917
Sam Webb	897
Sarah Griffiths	766
Anna Edwards	736
Victoria Woollaston	734
Damien Gayle	729
Mark Prigg	721
Tara Brady	716
Emma Innes	703
James Rush	701
Sara Malm	667
Ryan Gorman	657
Amanda Williams	647

Облако слов (города)

In [21]:
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
%matplotlib inline

text_cnn = ''
freq_dist_cnn.items()
for k,v in freq_dist_cnn.items():
    for i in range(v):
        text_cnn += k + " "

wordcloud = WordCloud(collocations=False,
                      background_color="white",
                      colormap='Dark2',
                      width=1920,
                      height=1080,
                      min_font_size=14).generate(text_cnn)

plt.imshow(wordcloud, interpolation='bilinear', aspect='auto')
plt.axis("off")
plt.show()
```

Облако слов (репортеры)

In [23]:

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
%matplotlib inline

text_dm = ''
freq_dist_dm.items()
for k,v in freq_dist_dm.items():
    for i in range(v):
        text_dm += k + " "

wordcloud = WordCloud(collocations=False,
                      background_color="white",
                      colormap='Dark2',
                      width=1920,
                      height=1080,
                      min_font_size=14).generate(text_dm)

plt.imshow(wordcloud, interpolation='bilinear', aspect='auto')
plt.axis("off")
plt.show()
```


Reporter
Mail
Daily
Associated
MailOnline
Press
Robinson
James
Mark
Martin
Steve
Chris
David
Simon
Sarah
Emma
Sam
Duell
Anna
London
Richard
Edward
Reilly
Amanda
Jill
Hugo
Kerry
Malm
Allen
Rush
Smith
Ashley
Alex
Webb
Tomlinson
Paul
Arkell
Louise
Harriet
Damien
Gye
Sophie
Warren
Woollaston
Gardner
Victoria
Pleasance
Tara
Kent
Rob
Lizzie
Snejana
Katy
Styles
Nye
Emily
Will
Ellie
Winter
Matt
Suzannah
Bond
Leon
Sara
Nolan
Collman
Helen
Gorman
Watson
Blake
Becky
Farberov
Ryan
Daniel
Anthony
Mcdermott
Peter
Michael
Hill
Miller
Boyle
Cooper
Gayle
Brady
Prigg
Lucy
Australia
De
Ruth
Nick
Lydia
Griffiths
Innes
William
Evans
Bianca
John