bart_storyline_2_story

December 1, 2021

1 Use Bart to generate story from storyline

References: article: https://towardsdatascience.com/teaching-bart-to-rap-fine-tuning-hugging-faces-bart-model-41749d38f3ef github repo: http://www.github.com/fpaupier/RapLyrics-Scraper

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
root_dir = "/content/drive/MyDrive/Colab Notebooks/544_bart/"
```

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

[3]:
```
# This run uses Pytorch Lightening to finetune the model
!pip install -q pytorch-lightning
!pip install -q transformers
```

```
     || 524 kB 4.1 MB/s
     || 329 kB 61.6 MB/s
     || 132 kB 81.0 MB/s
     || 829 kB 54.1 MB/s
     || 596 kB 45.9 MB/s
     || 1.1 MB 51.6 MB/s
     || 271 kB 67.9 MB/s
     || 192 kB 75.7 MB/s
     || 160 kB 62.1 MB/s
  Building wheel for future (setup.py) ... done
     || 3.1 MB 4.3 MB/s
     || 895 kB 58.7 MB/s
     || 3.3 MB 17.6 MB/s
     || 59 kB 6.9 MB/s
```

[4]:
```
# imports
import transformers
from torch.utils.data import DataLoader, TensorDataset, random_split,␣
 ↪RandomSampler, Dataset
import pandas as pd
```

```
import numpy as np

import torch.nn.functional as F
import pytorch_lightning as pl
import torch
from pytorch_lightning.callbacks import ModelCheckpoint

import math
import random
import re
import argparse
```

[5]:
```
df = pd.read_csv(root_dir + 'train_title_line_story.csv')
df
```

[5]:
```
                    storytitle  ...
titleLineConcat
0       David Drops the Weight  ...  David Drops the Weight <EOT> put try
realized ...
1                   Frustration  ...        Frustration <EOT> tom angry wall tom
regret
2           Marcus Buys Khakis  ...  Marcus Buys Khakis <EOT> business formal
pair ...
3           Different Opinions  ...  Different Opinions <EOT> trailer needed
much w...
4      Overcoming shortcomings  ...  Overcoming shortcomings <EOT> pastor tried
sin...
...                        ...  ...
...
52660                  Flavor  ...        Flavor <EOT> flavor tried get recipe
recipe
52661              After Death  ...      After Death <EOT> trouble day found
told week
52662  Janice breaks her wrist  ...  Janice breaks her wrist <EOT> janice legs
work...
52663   Jamie marries for love  ...  Jamie marries for love <EOT> jamie married
man...
52664                  Orange  ...                Orange <EOT> tree hit tree
tree nose

[52665 rows x 4 columns]
```

[6]:
```
train, valid, test = np.split(df, [int(.8*len(df)), int(.9*len(df))])
```

[7]:
```
train
```

[7]:
```
                    storytitle  ...
titleLineConcat
0       David Drops the Weight  ...  David Drops the Weight <EOT> put try
```

```
realized ...
1                  Frustration  ...        Frustration <EOT> tom angry wall tom
regret
2           Marcus Buys Khakis  ...  Marcus Buys Khakis <EOT> business formal
pair ...
3           Different Opinions  ...  Different Opinions <EOT> trailer needed
much w...
4      Overcoming shortcomings  ...  Overcoming shortcomings <EOT> pastor tried
sin...
...                        ...  ...
...
42127        Long Disney lines  ...  Long Disney lines <EOT> went waited longest
le...
42128                    Ropes  ...             Ropes <EOT> took went shore rope
great
42129                Detention  ...  Detention <EOT> punished waiting write
upset time
42130       Forgotten Homework  ...  Forgotten Homework <EOT> ryan mom school
schoo...
42131              The Dresser  ...    The Dresser <EOT> shopping found took
room well

[42132 rows x 4 columns]
```

[8]:
```
# Create a dataloading module as per the PyTorch Lightning Docs
SOURCE = 'titleLineConcat'
TARGET = 'story'

class SummaryDataModule(pl.LightningDataModule):
  def __init__(self, tokenizer, train, valid, batch_size):
    super().__init__()
    self.tokenizer = tokenizer
    self.train = train
    self.validate = valid
    self.batch_size = batch_size

  # Loads and splits the data into training, validation and test sets with a 80/
↪10/10 split
  # def prepare_data(self):
    # self.data = pd.read_csv(self.data_file)[:self.num_examples]
    # self.train, self.validate, self.test = np.split(self.data.sample(frac=1),␣
↪[int(.8*len(self.data))])

  # encode the sentences using the tokenizer
  def setup(self, stage):
    self.train = encode_sentences(self.tokenizer, self.train[SOURCE], self.
↪train[TARGET])
```

```
    self.validate = encode_sentences(self.tokenizer, self.validate[SOURCE],␣
↪self.validate[TARGET])
    # self.test = encode_sentences(self.tokenizer, self.test[SOURCE], self.
↪test[TARGET])

  # Load the training, validation and test sets in Pytorch Dataset objects
  def train_dataloader(self):
    dataset = TensorDataset(self.train['input_ids'], self.
↪train['attention_mask'], self.train['labels'])
    train_data = DataLoader(dataset, sampler = RandomSampler(dataset),␣
↪batch_size = self.batch_size)
    return train_data

  def val_dataloader(self):
    dataset = TensorDataset(self.validate['input_ids'], self.
↪validate['attention_mask'], self.validate['labels'])
    val_data = DataLoader(dataset, batch_size = self.batch_size)
    return val_data

  # def test_dataloader(self):
  #   dataset = TensorDataset(self.test['input_ids'], self.
↪test['attention_mask'], self.test['labels'])
  #   test_data = DataLoader(dataset, batch_size = self.batch_size)
↪
  #   return test_data
```

[9]:
```
class LitModel(pl.LightningModule):
  # Instantiate the model
  def __init__(self, learning_rate, tokenizer, model, hparams):
    super().__init__()
    self.tokenizer = tokenizer
    self.model = model
    self.learning_rate = learning_rate
    # self.freeze_encoder = freeze_encoder
    # self.freeze_embeds_ = freeze_embeds
    # self.hparams = hparams
    self.save_hyperparameters(hparams)

    if self.hparams.freeze_encoder:
      freeze_params(self.model.get_encoder())

    if self.hparams.freeze_embeds:
      self.freeze_embeds()

  def freeze_embeds(self):
    ''' freeze the positional embedding parameters of the model; adapted from␣
↪finetune.py '''
```

```
    freeze_params(self.model.model.shared)
    for d in [self.model.model.encoder, self.model.model.decoder]:
      freeze_params(d.embed_positions)
      freeze_params(d.embed_tokens)

  # Do a forward pass through the model
  def forward(self, input_ids, **kwargs):
    return self.model(input_ids, **kwargs)

  def configure_optimizers(self):
    optimizer = torch.optim.Adam(self.parameters(), lr = self.learning_rate)
    return optimizer

  def training_step(self, batch, batch_idx):
    # Load the data into variables
    src_ids, src_mask = batch[0], batch[1]
    tgt_ids = batch[2]
    # Shift the decoder tokens right (but NOT the tgt_ids)
    decoder_input_ids = shift_tokens_right(tgt_ids, tokenizer.pad_token_id)

    # Run the model and get the logits
    outputs = self(src_ids, attention_mask=src_mask,␣
↪decoder_input_ids=decoder_input_ids, use_cache=False)
    lm_logits = outputs[0]
    # Create the loss function
    ce_loss_fct = torch.nn.CrossEntropyLoss(ignore_index=self.tokenizer.
↪pad_token_id)
    # Calculate the loss on the un-shifted tokens
    loss = ce_loss_fct(lm_logits.view(-1, lm_logits.shape[-1]), tgt_ids.
↪view(-1))

    return {'loss':loss}

  def validation_step(self, batch, batch_idx):

    src_ids, src_mask = batch[0], batch[1]
    tgt_ids = batch[2]

    decoder_input_ids = shift_tokens_right(tgt_ids, tokenizer.pad_token_id)

    # Run the model and get the logits
    outputs = self(src_ids, attention_mask=src_mask,␣
↪decoder_input_ids=decoder_input_ids, use_cache=False)
    lm_logits = outputs[0]

    ce_loss_fct = torch.nn.CrossEntropyLoss(ignore_index=self.tokenizer.
↪pad_token_id)
```

```
    val_loss = ce_loss_fct(lm_logits.view(-1, lm_logits.shape[-1]), tgt_ids.
↪view(-1))

    return {'loss': val_loss}

  # Method that generates text using the BartForConditionalGeneration's␣
↪generate() method
  def generate_text(self, text, eval_beams, early_stopping = True, max_len =␣
↪1000):
    ''' Function to generate text '''
    generated_ids = self.model.generate(
        text["input_ids"],
        attention_mask=text["attention_mask"],
        use_cache=True,
        decoder_start_token_id = self.tokenizer.pad_token_id,
        num_beams= eval_beams,
        max_length = max_len,
        early_stopping = early_stopping
    )
    return [self.tokenizer.decode(w, skip_special_tokens=True,␣
↪clean_up_tokenization_spaces=True) for w in generated_ids]

  def training_epoch_end(self,outputs):
        #  the function is called after every epoch is completed

        # calculating average loss
        avg_loss = torch.stack([x['loss'] for x in outputs]).mean()

        # calculating correect and total predictions
        # correct=sum([x["correct"] for  x in outputs])
        # total=sum([x["total"] for  x in outputs])

        # creating log dictionary
        # tensorboard_logs = {'loss': avg_loss,"Accuracy": correct/total}
        tensorboard_logs = {'loss': avg_loss}

        epoch_dictionary={
            # required
            'loss': avg_loss,

            # for logging purposes
            'log': tensorboard_logs}

        # return epoch_dictionary

def freeze_params(model):
```

```
  ''' Function that takes a model as input (or part of a model) and freezes the␣
 ↪layers for faster training
      adapted from finetune.py '''
  for layer in model.parameters():
    layer.requires_grade = False
```

[10]:
```
# Create the hparams dictionary to pass in the model
# I realise that this isn't really how this is meant to be used, but having␣
 ↪this here reminds me that I can edit it when I need
hparams = argparse.Namespace()

hparams.freeze_encoder = True
hparams.freeze_embeds = True
hparams.eval_beams = 4
```

[11]:
```
def shift_tokens_right(input_ids, pad_token_id):
  """ Shift input ids one token to the right, and wrap the last non pad token␣
 ↪(usually <eos>).
      This is taken directly from modeling_bart.py
  """
  prev_output_tokens = input_ids.clone()
  index_of_eos = (input_ids.ne(pad_token_id).sum(dim=1) - 1).unsqueeze(-1)
  prev_output_tokens[:, 0] = input_ids.gather(1, index_of_eos).squeeze()
  prev_output_tokens[:, 1:] = input_ids[:, :-1]
  return prev_output_tokens

def encode_sentences(tokenizer, source_sentences, target_sentences,␣
 ↪max_length=32, pad_to_max_length=True, return_tensors="pt"):
  ''' Function that tokenizes a sentence
      Args: tokenizer - the BART tokenizer; source and target sentences are the␣
 ↪source and target sentences
      Returns: Dictionary with keys: input_ids, attention_mask, target_ids
  '''

  input_ids = []
  attention_masks = []
  target_ids = []
  tokenized_sentences = {}

  for sentence in source_sentences:
    encoded_dict = tokenizer(
          sentence,
          max_length=max_length,
          padding="max_length" if pad_to_max_length else None,
          truncation=True,
          return_tensors=return_tensors,
          add_prefix_space = True
```

```
      )

    input_ids.append(encoded_dict['input_ids'])
    attention_masks.append(encoded_dict['attention_mask'])

  input_ids = torch.cat(input_ids, dim = 0)
  attention_masks = torch.cat(attention_masks, dim = 0)

  for sentence in target_sentences:
    encoded_dict = tokenizer(
          sentence,
          max_length=max_length,
          padding="max_length" if pad_to_max_length else None,
          truncation=True,
          return_tensors=return_tensors,
          add_prefix_space = True
      )
    # Shift the target ids to the right
    # shifted_target_ids = shift_tokens_right(encoded_dict['input_ids'],␣
↪tokenizer.pad_token_id)
    target_ids.append(encoded_dict['input_ids'])

  target_ids = torch.cat(target_ids, dim = 0)


  batch = {
      "input_ids": input_ids,
      "attention_mask": attention_masks,
      "labels": target_ids,
  }

  return batch


def noise_sentence(sentence_, percent_words, replacement_token = "<mask>"):
  '''
  Function that noises a sentence by adding <mask> tokens
  Args: sentence - the sentence to noise
        percent_words - the percent of words to replace with <mask> tokens; the␣
↪number is rounded up using math.ceil
  Returns a noised sentence
  '''
  # Create a list item and copy
  sentence_ = sentence_.split(' ')
  sentence = sentence_.copy()

  num_words = math.ceil(len(sentence) * percent_words)
```

```
    # Create an array of tokens to sample from; don't include the last word as an␣
    ↪option because in the case of lyrics
    # that word is often a rhyming word and plays an important role in song␣
    ↪construction
    sample_tokens = set(np.arange(0, np.maximum(1, len(sentence)-1)))

    words_to_noise = random.sample(sample_tokens, num_words)

    # Swap out words, but not full stops
    for pos in words_to_noise:
        if sentence[pos] != '.':
            sentence[pos] = replacement_token

    # Remove redundant spaces
    sentence = re.sub(r' {2,5}', ' ', ' '.join(sentence))

    # Combine concurrent <mask> tokens into a single token; this just does two␣
    ↪rounds of this; more could be done
    sentence = re.sub(r'<mask> <mask>', "<mask>", sentence)
    sentence = re.sub(r'<mask> <mask>', "<mask>", sentence)
    return sentence
```

2 Load BART

Here we load the model. I used "bart-base" because I had memory issues using "bart-large". "bart-base" appears to load without the use_cache argument, which by necessity must be turned to "False" for "bart-large".

[12]:
```
# Load the model
from transformers import BartTokenizer, BartForConditionalGeneration, AdamW,␣
 ↪BartConfig

tokenizer = BartTokenizer.from_pretrained('facebook/bart-base',␣
 ↪add_prefix_space=True)

bart_model = BartForConditionalGeneration.from_pretrained(
    "facebook/bart-base")
```

```
Downloading:   0%|          | 0.00/878k [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/446k [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/1.29M [00:00<?, ?B/s]
```

```
Downloading:   0%|          | 0.00/1.65k [00:00<?, ?B/s]
```

```
Downloading:   0%|          | 0.00/532M [00:00<?, ?B/s]
```

[13]:
```
# Load the data into the model for training
summary_data = SummaryDataModule(tokenizer, train, test, batch_size = 16)

# Load the model from a pre-saved checkpoint; alternatively use the code below␣
 ↪to start training from scratch
# model = LitModel.load_from_checkpoint(root_dir + "checkpoint_files_3/
 ↪5_epoch_11_29_ls.ckpt",
#                                       learning_rate = 2e-5, tokenizer =␣
 ↪tokenizer, model = bart_model, hparams = hparams)

model = LitModel(learning_rate = 2e-5, tokenizer = tokenizer, model =␣
 ↪bart_model, hparams = hparams)
```

3 Training the model with Pytorch Lightning

The below code utilises Pytorch Lightning's fantastic Trainer module that helps to control the training process. After creating a ModelCheckpoint object, the other options are fed into the Trainer module. I found that my colab crashed when I didn't explicitly set progress_bar_refresh_rate to something and I found that setting it to 500 seemed to work just fine.

[14]:
```
checkpoint = ModelCheckpoint(dirpath=root_dir + 'checkpoint_files_3/')
trainer = pl.Trainer(gpus = 1,
                     max_epochs = 3,
                     min_epochs = 3,
                     auto_lr_find = False,
                     checkpoint_callback = checkpoint,
                     progress_bar_refresh_rate = 500)
```

```
/usr/local/lib/python3.7/dist-
packages/pytorch_lightning/trainer/connectors/callback_connector.py:148:
LightningDeprecationWarning: Setting `Trainer(checkpoint_callback=<pytorch_light
ning.callbacks.model_checkpoint.ModelCheckpoint object at 0x7f368f304c50>)` is
deprecated in v1.5 and will be removed in v1.7. Please consider using `Trainer(e
nable_checkpointing=<pytorch_lightning.callbacks.model_checkpoint.ModelCheckpoin
t object at 0x7f368f304c50>)`.
  f"Setting `Trainer(checkpoint_callback={checkpoint_callback})` is deprecated
in v1.5 and will "
/usr/local/lib/python3.7/dist-
packages/pytorch_lightning/trainer/connectors/callback_connector.py:91:
LightningDeprecationWarning: Setting `Trainer(progress_bar_refresh_rate=500)` is
deprecated in v1.5 and will be removed in v1.7. Please pass
```

```
`pytorch_lightning.callbacks.progress.TQDMProgressBar` with `refresh_rate`
directly to the Trainer's `callbacks` argument instead. Or, to disable the
progress bar pass `enable_progress_bar = False` to the Trainer.
  f"Setting `Trainer(progress_bar_refresh_rate={progress_bar_refresh_rate})` is
deprecated in v1.5 and"
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
```

[15]:
```
# Fit the instantiated model to the data
trainer.fit(model, summary_data)
```

```
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name  | Type                          | Params
-------------------------------------------------------
0 | model | BartForConditionalGeneration | 139 M
-------------------------------------------------------
139 M     Trainable params
0         Non-trainable params
139 M     Total params
557.682   Total estimated model params size (MB)

Validation sanity check: 0it [00:00, ?it/s]


Training: 0it [00:00, ?it/s]


Validating: 0it [00:00, ?it/s]


Validating: 0it [00:00, ?it/s]


Validating: 0it [00:00, ?it/s]
```

[16]:
```
# %load_ext tensorboard
# %tensorboard --logdir /content/lightning_logs/
```

[17]:
```
# If you want to manually save a checkpoint, this works, although the model␣
 ↪should automatically save (progressively better)
# checkpoints as it moves through the epochs
trainer.save_checkpoint(root_dir + "checkpoint_files_3/3_epoch_11_30_ls.ckpt")
```

4 Generate Story from StoryLine

[18]:
```
# def generate_lyrics(seed_line, num_lines, model_, noise_percent = 0.25,␣
 ↪multiple_lines = False, max_line_history = 3):
#   ''' Function that generates lyrics based on previously generated lyrics
#       Args: seed_line - a line to start off the machine
#             num_lines - the number of lines to generate
#             model_ - the model used to generate the text
#             multiple_lines - whether the model generates based on multiple␣
 ↪previous lines or just the past line
#             max_line_history - the maximum number of previous lines used in␣
 ↪the current input
#       Returns a list with num_lines of rap lines
#   '''
#   # Put the model on eval mode
#   model_.to(torch.device('cpu'))
#   model_.eval()
#   lyrics = []
#   lyrics.append(seed_line)
#   #skip noising for now
#   # prompt_line_tokens = tokenizer(noise_sentence(seed_line, 0.2), max_length␣
 ↪= 600, return_tensors = "pt", truncation = True)
#   prompt_line_tokens = tokenizer(seed_line, max_length = 100, return_tensors␣
 ↪= "pt", truncation = True)
#   # Loop through the number of lines generating a new line based on the old

#   line = [seed_line]
#   for i in range(num_lines):
#     # Print out the new line
#     print(line[0].strip())
#     lyrics.append(line[0])
#     line = model.generate_text(prompt_line_tokens, eval_beams = 4, max_len =␣
 ↪60000)
#     # This deals with an artefact in the training data that I had an issue␣
 ↪cleaning
#     if line[0].find(":") != -1:
#       line[0] = re.sub(r'[A-Z]+: ', '', line[0])
#     # This allows the model to generate a new line conditioned on more than␣
 ↪one line
#     if multiple_lines:
#       start_line = np.maximum(0, i - max_line_history)
#       end_line = i
#       prompt_line = ' '.join(lyrics[start_line:end_line]) # Going to end_line␣
 ↪is fine because it is non-inclusive
#     else:
#       prompt_line = lyrics[i]
#     # prompt_line_tokens = tokenizer(noise_sentence(prompt_line,␣
 ↪noise_percent), max_length = 32, return_tensors = "pt", truncation = True)
```

```
#     prompt_line_tokens = tokenizer(prompt_line, max_length = 32,␣
 ↪return_tensors = "pt", truncation = True)

#   return lyrics
```

[19]:
```
def generate_story(title_line, model_):
  # Put the model on eval mode
  model_.to(torch.device('cpu'))
  model_.eval()
  prompt_line_tokens = tokenizer(title_line, max_length = 100, return_tensors =␣
 ↪"pt", truncation = True)
  out = model.generate_text(prompt_line_tokens, eval_beams = 4, early_stopping␣
 ↪= False, max_len=100000)
  return out[0].strip()
```

[20]:
```
df
```

[20]:
```
                    storytitle  ...
titleLineConcat
0       David Drops the Weight  ...  David Drops the Weight <EOT> put try
realized ...
1                   Frustration  ...        Frustration <EOT> tom angry wall tom
regret
2           Marcus Buys Khakis  ...  Marcus Buys Khakis <EOT> business formal
pair ...
3           Different Opinions  ...  Different Opinions <EOT> trailer needed
much w...
4      Overcoming shortcomings  ...  Overcoming shortcomings <EOT> pastor tried
sin...
...                        ...  ...
...
52660                  Flavor  ...        Flavor <EOT> flavor tried get recipe
recipe
52661              After Death  ...      After Death <EOT> trouble day found
told week
52662  Janice breaks her wrist  ...  Janice breaks her wrist <EOT> janice legs
work...
52663   Jamie marries for love  ...  Jamie marries for love <EOT> jamie married
man...
52664                  Orange  ...               Orange <EOT> tree hit tree
tree nose

[52665 rows x 4 columns]
```

[21]:
```
titleLineConcat = df['titleLineConcat'][0]
titleLineConcat
```

[21]:
```
'David Drops the Weight <EOT> put try realized started weeks'
```

[22]:
```
story = df['story'][0]
story
```

[22]:
```
"David noticed he had put on a lot of weight recently.He examined his habits to
try and figure out the reason.He realized he'd been eating too much fast food
lately.He stopped going to burger places and started a vegetarian diet.After a
few weeks, he started to feel much better."
```

[23]:
```
story = generate_story(titleLineConcat, model)
story
#3 epoch
#Frustration
#jimmy told told told said
#David Drops the Weight
#dave overweight weight weight weight
#8 epoches
#David Drops the Weight
#lose wanted diet doctor doctor
#Frustration
#jim jim told told jim
#13 epoches
# Frustration
# tom angry wall tom tom
# David Drops the Weight
# pounds decided went month pounds
```

[23]:
```
'David decided to put on a few pounds.He decided to try to lose weight.He
realized that he was gaining a lot.He started to lose'
```

[24]:
```
reduced_test = pd.read_csv(root_dir + 'test_predicted_storyline_15_ep.csv')
reduced_test
```

[24]:
```
               storytitle  ...
predicted_titleLineConcat
0                  Crazed  ...              Crazed <EOT> kate door kate room
kate
1    Thanksgiving Dinner.  ...  Thanksgiving Dinner. <EOT> thanksgiving
turkey...
2                 On Sale  ...             On Sale <EOT> went saw tried tried
one
3      First Scraped Knee  ...  First Scraped Knee <EOT> niece scraped knee
sc...
4                      tv  ...                         tv <EOT> tv sudden tv tv
tv
..                    ...  ...
...
256            Rear Ended  ...         Rear Ended <EOT> tom tried right ended
tom
257                   Bat  ...                      Bat <EOT> bat hit leg leg
```

```
leg
258                Her Mom  ...       Her Mom <EOT> mother year year mother
mother
259  Pat the Photographer  ...  Pat the Photographer <EOT> pat photographer
st...
260               Pressure  ...  Pressure <EOT> undercooked undercooked finger
...

[261 rows x 6 columns]
```

[25]:
```
titleLineConcat = reduced_test['predicted_titleLineConcat'][0]
titleLineConcat
```

[25]:
```
'Crazed <EOT> kate door kate room kate'
```

[26]:
```
story = reduced_test['story'][0]
story
```

[26]:
```
"I was cleaning nonstop.I couldn't seem to get anything clean enough.I went out
and bought some cleaning supplies when I ran out.I went back home and started
power washing the exterior.When I was done, the sun was going down."
```

[27]:
```
story = generate_story(titleLineConcat, model)
story
```

[27]:
```
'Kate was playing outside with her friends.A man knocked on the door.Kate was
startled.The man ran out of the room.Kate had to'
```

[28]:
```
#TODO: Figure out how to generate complete sentences
#maybe sep_token for sentences?
keep_index = []
generated_stories = []
for index, row in reduced_test.iterrows():
    out = generate_story(row['predicted_titleLineConcat'], model)
    if len(out.split('.')) >= 6:
      story = out.split('.')[:5]
      story = '.'.join(story) + '.'
      generated_stories.append(story)
      keep_index.append(index)
      print('title: ', row['storytitle'])
      print('origin_storyline: ', row['storyline'])
      print('origin_story: ', row['story'])
      print('generated_storylines: ', row['predicted_storylines'])
      print('generated_story: ', story)
      print()
```

```
title:  Drunk Dialing
origin_storyline:  tom decided started exes problems
origin_story:  Tom had problems with depression.He decided to drink one night.He
went overboard and started calling people.He called some exes.It create a lot of
problems in his life.
```

generated_storylines: phone conversation man man apologized
generated_story: The man hung up the phone.The woman continued to have a conversation.The man was drunk.The women confronted the man.They apologized.

title: Scrambled Eggs
origin_storyline: surprise usually morning woke missed
origin_story: Aya woke early one morning to surprise her husband with a hot meal.He began work at 6AM, so usually she was only awake to say goodbye.But this morning, she made him a big hot dish of scrambled eggs.Then she woke him and smiled as he dove in - until he grimaced.Aya had missed a large chunk of eggshell, and it had cut her husband!
generated_storylines: eggs eggs eggs scrambled eggs
generated_story: The man made some eggs.He scrambled the eggs.The eggs were scrambled.The man scrambled them again.He made more eggs.

title: Trick or Treating
origin_storyline: loved moved treating try candy
origin_story: Tim loved Halloween.He moved into a nice neighborhood.He had never gone trick or treating before.Tim decided to try it out.He had great fun and got lots of candy.
generated_storylines: halloween dressed year dressed year
generated_story: It was Halloween.Everyone was dressed up.This year was no different.All of the kids dressed up for Halloween.They dressed up as ghosts.

title: Breakfast
origin_storyline: suzanne family time served time
origin_story: Suzanne was in the kitchen making breakfast.Her family was not awake yet.She spent a lot of time doing everything just right.Finally, breakfast was ready to be served.Her family woke up just in time and everyone enjoyed breakfast.
generated_storylines: hungry decided ingredients followed hunger
generated_story: Tom woke up hungry.He decided to make breakfast.He bought all the ingredients.He followed the instructions.Tom ate his breakfast with no hunger.

title: The Internet
origin_storyline: trouble weeks seem turned uses
origin_story: Lynn was having a lot of trouble with her internet service.For weeks she kept having to call technical support off and on.After many technician visits they seem to of found the problem.It turned out that her roku was soaking up all the bandwidth.For now Lynn doesn't use the roku and just uses her other devices.
generated_storylines: internet went called fixed internet
generated_story: The internet was broken.I went to the internet provider.They called me.They fixed it.The internet was back on.

title: Wrong Party
origin_storyline: tom sure went party one

origin_story: Tom was invited to a party.He wasn't sure about the address.He went to the neighborhood and looked for it.Tom eventually wound up at a party he found.It wasn't the one he was looking for but he had fun.
generated_storylines: tim birthday tim wrong tim
generated_story: Tim was invited to a birthday party.It was his birthday.Tim was excited.But the party was wrong.Tim had to cancel.

title: Pets
origin_storyline: son needed home every happy
origin_story: My son had always wanted a pet.I found two cats that needed a new home.We decided to bring them home.My son feeds them every morning.Being a pet owner makes him happy.
generated_storylines: pet went found took named
generated_story: The man wanted a pet.He went to the pet store.He found a dog.He took the dog home.He named the dog.

title: Too sweet
origin_storyline: get would took sweet could
origin_story: Allie wanted to get some candy.But she was not sure if it would be too sweet.So she took a bite.It was far too sweet.She could no longer eat it anymore.
generated_storylines: sweet drink sweet drink stomach
generated_story: The man had a sweet tooth.He decided to drink it.It was too sweet.He couldn't drink it anymore.His stomach hurt.

title: Boring Movie
origin_storyline: romance really tim whining movie
origin_story: Tim hated romance movies.His girlfriend really liked them.Tim was forced to watch one with her.He kept complaining and whining throughout.Eventually his girlfriend stopped the movie and kicked him out.
generated_storylines: tom watching tom wound tom
generated_story: Tom was at the movies.He was watching a horror movie.Tom was really bored.He wound up watching it.Tom couldn't finish it.

title: Stolen Car
origin_storyline: tim parked missing report unfortunately
origin_story: Tim was visiting friends.He parked in a bad neighborhood.When he went back to his car it was missing.Tom called the cops and filed a report.Unfortunately the car was never found.
generated_storylines: car stolen tom police towed
generated_story: Tom bought a car.The car was stolen.Tom was angry.He called the police.They towed the car.

title: Clock Batteries
origin_storyline: manny realize manny time manny
origin_story: The batteries in Manny's clock had gone out.He didn't realize the batteries were dead.Manny had an appointment.He was going by the time on the clock.Manny was late for his appointment.

generated_storylines: batteries turn burned replacing batteries
generated_story: The batteries in my clock were old.I had to turn them off.They burned out.I ended up replacing them.I replaced the batteries.

title: Horror
origin_storyline: mail horror hand also open
origin_story: We got a box in the mail.It contained collectibles from horror movies.We found chop sticks that looked like Krueger's hand.We also had a chainsaw massacre shirt.We were very excited to open the box.
generated_storylines: horror movie scared movie movie
generated_story: The man watched a horror movie.The movie was scary.The man was scared.He got out of the movie.He watched the movie again.

title: Game at the Bar
origin_storyline: watch checked watch local watched
origin_story: Terry wanted to watch the football game.He checked his channels.Terry wasn't able to watch the game on his television.Terry went to the local bar.He watched the game there.
generated_storylines: tom bar game tom leave
generated_story: Tom was at a bar.There was a game at the bar.The game was going well.Tom was very nervous.He decided to leave.

title: The Roulette Wheel
origin_storyline: mac go night put black
origin_story: Mac and his friends were off on a boys weekend.They had decided to go to the casino.They were playing different games throughout the night.Mac approached the roulette wheel and put it all on black.The ball bounced around forever before finally settling on black.
generated_storylines: jim twisted tried hit leg
generated_story: Jim was playing roulette.He twisted his ankle.He tried to get back up.He hit the roulette wheel.He broke his leg.

title: Accidents Happen
origin_storyline: job work desk would turns
origin_story: I was at my job.It was my first day of work at this job.I accidentally spilled coffee on my boss's desk.I was sure I would be fired.It turns out he was very forgiving because I was new.
generated_storylines: ken ken stepped fell scraped
generated_story: Ken was playing outside.Ken was playing basketball.He stepped on the ball.He fell.Ken scraped his knee.

title: first place
origin_storyline: john change john got even
origin_story: John's baseball team came in last place last year.That is about to change.John and the other players worked harder to get better.His team got first place.They even won championship.
generated_storylines: first nervous ran ended happy
generated_story: It was my first race.I was very nervous.I ran for a long

time.I ended up winning.I am happy about that.

title: Chicken Soup
origin_storyline: james stomach realized went chicken
origin_story: James was eating a snickers bar.He got sick to his stomach.He realized he had been eating too many sweets.He went to lie down.His wife brought him some chicken soup.
generated_storylines: make make ingredients followed great
generated_story: Tom wanted to make chicken soup.He decided to make it himself.He bought all the ingredients.He followed the recipe.It was great.

title: Old Coffee
origin_storyline: work poured realize tried threw
origin_story: Tom was running late for work.He poured out some old coffee.He didn't realize how long it had been out.He tried to drink some.Tom spat it back out and threw it out.
generated_storylines: tom noticed old use tom
generated_story: Tom was at the coffee shop.He noticed a coffee machine.It was old.Tom decided to use it.Tom had a great time.

title: Birthday
origin_storyline: father take leaving us steak
origin_story: It was my father's birthday.We made plans to take him out for a steak.He doesn't like leaving the house much.We finally talked him into going with us.He greatly enjoyed his steak that day.
generated_storylines: jessica turning party cake great
generated_story: Jessica's birthday was coming up.She was turning sixteen.She decided to have a party.She baked a cake.Everyone had a great time.

title: Storm
origin_storyline: snowstorm including stove tank strenuous
origin_story: Last October, there was a bad snowstorm and the power went out.I lost power for 6 days, including my electric-start heat.I had 4 tropical fish tanks, so I kept water boiling on the stove.Each 15 minutes, I'd add warm water and aerate the tank with a cup.It was 6 days of a strenuous nightmare-but all my fish lived!
generated_storylines: tom storm tom wind storm
generated_story: Tom was at the beach.There was a storm.Tom was not paying attention.The wind was very strong.Tom got scared of the storm.

title: Rainy Halloween
origin_storyline: emma woke parade parade still
origin_story: Emma was super excited for the Halloween pet parade.When she woke up on Halloween it was pouring!Her mom said she couldn't go to the parade.She made her own parade with her stuffed animals!It was still the best Halloween ever.
generated_storylines: halloween scariest looked sadly soaked
generated_story: It was Halloween.It was the scariest day ever.I looked out the

window.Sadly it was raining.I was soaked.

title: Clipping Coupons
origin_storyline: loved shopped went time believe
origin_story: Alicia loved coupons.She always printed them before she shopped.But one day she brought her book with her when she went to the grocer.By the time she got rung up, she saved one thousand.Alicia could not believe it.
generated_storylines: went went line line worth
generated_story: I went to the grocery store yesterday.I went to get some coupons.There was a line.I got in line.It was worth it.

title: Wrong Glasses
origin_storyline: tom left blurry realized exchange
origin_story: Tom was doing a group project.At the end of it he grabbed his glasses and left.Tom noticed everything was blurry.Eventually he realized he got his friend's glasses by mistake.Tom called him up to do an exchange.
generated_storylines: glasses went wrong pick pick
generated_story: The man needed glasses.He went to the store.They were wrong.He had to pick them up.He couldn't pick them out.

title: Racing
origin_storyline: racing thunderbird wary let showed
origin_story: Kurt loved racing cars.One day, he came over and asked to borrow my thunderbird.I was wary, as he promised he wasn't going to race it.I let him borrow the car.After he died, I found a paper that showed he had indeed raced my car.
generated_storylines: car race took took race
generated_story: The man bought a car.He decided to race it.He took it to the track.It took him a while.He won the race.

title: Climate
origin_storyline: watched climate talked subject together
origin_story: I watched a documentary.It was about climate change.I talked to my husband about the implications of such changes.We agreed completely on the subject.Together, we decided to start recycling.
generated_storylines: world weather temperature man winter
generated_story: The world was changing.The weather was getting colder.The temperature was dropping.The man had to change his climate.It was winter.

title: Social Media Problems
origin_storyline: twitter politicians send started account
origin_story: Tim was all over Twitter.He followed a lot of politicians.He would get drunk and send insults to them.After a while many people started getting upset with him.Tim eventually had his account banned.
generated_storylines: sam number insulted insulted insulted
generated_story: Sam was on social media.He had a number of friends.One of them insulted him.Sam insulted them.He insulted them again.

title: Roaches.
origin_storyline: roaches walls house cleaned longer
origin_story: In a house on a hill there was a house littered with roaches.Along the walls the floors the ceiling everywhere.No one lived there, until a new family bought the house.They had it cleaned and fumigated getting rid of every bug.Now it is no longer the roach house.
generated_storylines: roaches roaches exterminator sprayed free
generated_story: The man sprayed the roaches.The roaches got infected.The man called an exterminator.The exterminator sprayed them.They were free.

title: Garage Door
origin_storyline: pick open concierge pressed returned
origin_story: My wife went out to pick up a pizza tonight.When she got home, the garage door to our building would not open.She called the concierge but he was away.She pressed the buzzer for the concierge to open the door.After five minutes he returned and opened the door.
generated_storylines: greg window saw dog dog
generated_story: Greg's garage door was open.He looked out the window.He saw a dog running around.Greg called the dog.The dog ran away.

title: Fried
origin_storyline: husband however every talked decided
origin_story: My husband was determined to stop eating fried foods.His favorite food, however, is fried chicken.He was tempted every day by it.We talked about the situation.We decided he could eat it once per week instead.
generated_storylines: fried decided ingredients followed turned
generated_story: Tom wanted fried chicken.He decided to make it himself.He bought all the ingredients.He followed the instructions.It turned out great.

title: Earthquake
origin_storyline: suddenly shaking ran soon ever
origin_story: Laura woke up suddenly.The ground was shaking!Laura hurried and ran for the doorway.Soon after the ground stopped shaking.That was the first earthquake Laura has ever been in.
generated_storylines: california woke scared scared scared
generated_story: Tom lived in California.One day he woke up to an earthquake.Tom was scared.He was scared for his life.Tom got scared.

title: Politician at Heart
origin_storyline: politics vote popular sad mp
origin_story: Clair was extremely interested in politics from a young age.At 13 years old, she campaigned for the vote for 16 year olds.This didn't make her popular, as everyone thought she was a nerd.This made her sad, but she knew she was fighting for a good cause.Years later, she is now an MP and looks down on her previous peers.
generated_storylines: politician wanted switched switched switched
generated_story: John was a politician at heart.He wanted to change that.He switched his political views.John switched his politics.John was happy he

switched.

[29]:
```
reduced_test = reduced_test.iloc[keep_index]
```

[30]:
```
reduced_test['predicted_story'] = generated_stories
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py: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
  """Entry point for launching an IPython kernel.
```

[31]:
```
for index, row in reduced_test.iterrows():
    out = row['predicted_story']
    if len(out.split('.')) != 6:
      print(out)
```

[32]:
```
reduced_test.reset_index(drop=True, inplace=True)
```

[33]:
```
reduced_test
```

[33]:
```
               storytitle  ...
predicted_story
0           Drunk Dialing  ...  The man hung up the phone.The woman continued
...
1          Scrambled Eggs  ...  The man made some eggs.He scrambled the
eggs.T...
2       Trick or Treating  ...  It was Halloween.Everyone was dressed up.This
...
3               Breakfast  ...  Tom woke up hungry.He decided to make
breakfas...
4            The Internet  ...  The internet was broken.I went to the
internet...
5             Wrong Party  ...  Tim was invited to a birthday party.It was
his...
6                    Pets  ...  The man wanted a pet.He went to the pet
store...
7              Too sweet  ...  The man had a sweet tooth.He decided to drink
...
8            Boring Movie  ...  Tom was at the movies.He was watching a
horror...
9              Stolen Car  ...  Tom bought a car.The car was stolen.Tom was
an...
10        Clock Batteries  ...  The batteries in my clock were old.I had to
tu...
```

```
11                 Horror  ...  The man watched a horror movie.The movie was
s...
12        Game at the Bar  ...  Tom was at a bar.There was a game at the
bar.T...
13     The Roulette Wheel  ...  Jim was playing roulette.He twisted his
ankle...
14       Accidents Happen  ...  Ken was playing outside.Ken was playing
basket...
15            first place  ...  It was my first race.I was very nervous.I ran
...
16           Chicken Soup  ...  Tom wanted to make chicken soup.He decided to
...
17             Old Coffee  ...  Tom was at the coffee shop.He noticed a
coffee...
18               Birthday  ...  Jessica's birthday was coming up.She was
turni...
19                  Storm  ...  Tom was at the beach.There was a storm.Tom
was...
20        Rainy Halloween  ...  It was Halloween.It was the scariest day
ever...
21       Clipping Coupons  ...  I went to the grocery store yesterday.I went
t...
22          Wrong Glasses  ...  The man needed glasses.He went to the
store.Th...
23                Racing  ...  The man bought a car.He decided to race it.He
...
24               Climate  ...  The world was changing.The weather was
getting...
25  Social Media Problems  ...  Sam was on social media.He had a number of
fri...
26               Roaches.  ...  The man sprayed the roaches.The roaches got
in...
27            Garage Door  ...  Greg's garage door was open.He looked out the
...
28                 Fried  ...  Tom wanted fried chicken.He decided to make
it...
29             Earthquake  ...  Tom lived in California.One day he woke up to
...
30    Politician at Heart  ...  John was a politician at heart.He wanted to
ch...

[31 rows x 7 columns]
```

[34]:
```
reduced_test.to_csv(root_dir+'test_predicted_story_3_ep.csv', encoding='utf-8',␣
↪index=False)
```

5 Evaluate BLEU Score

[38]:
```
from nltk.translate.bleu_score import sentence_bleu
from nltk.translate.bleu_score import SmoothingFunction
score_list = []
for index, row in reduced_test.iterrows():
    gt = row['story']
    pred = row['predicted_story']
    refs = []
    for s in gt.split('.')[:5]:
      refs.append(s.split())
    # print(refs)
    cans = []
    for s in pred.split('.')[:5]:
      can = s.split()
      cans.extend(can)
    score = sentence_bleu(refs, cans)
    score_list.append(score)

print('sentence bleu score is: ', np.mean(score_list))
```

```
sentence bleu score is:  0.4354171330439157

/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:490:
UserWarning:
Corpus/Sentence contains 0 counts of 2-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:490:
UserWarning:
Corpus/Sentence contains 0 counts of 3-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:490:
UserWarning:
Corpus/Sentence contains 0 counts of 4-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
```

[ ]:
```
#output notebook as pdf
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('bart_storyline_2_story.ipynb')
```

```
File colab_pdf.py already there; not retrieving.
```

```
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```