This document explains in more details about how functions work, how some of the parameters are configured, etc.

PREPROCESSING:

1 The code commented out(df_.text = 'summarize: ' + df_.text) is required only for t5, because t5 has a unified model structure for various downstream tasks and the user must specify the need by adding a keyword.

2 train_test_split(df_,test_size=0.1,shuffle=True, random_state=42)

""" Split dataset into training set and test set."""

Args:

*arrays:

dataset, vaild for lists, numpy arrays, spcipy-sparse matrices or pandas dataframes.

test_size:

a ratio that indicates the proportion of test set.

shuffle:

a Boolean value, True by default.

random_state:

an option whether preserves the same shuffle at each run. When a number is given, then the results become reproducible.

Returns:

List containing train-test split of inputs.

MODELS:

1 tokenizer.batch_encode_plus(train['text'], return_tensors='pt', max_length=max_length, pad_to_max_length=True,truncation=True):

""" Encode the input sequence"""

Args:

text:

a list of sequence to be encoded.

return_tensors:

If set, will return tensors. 'pt' represents torch.Tensor objects.

max_length:

```
    If not set, it will use the predefined max_length of model. Here it is
    still set in order to make it clear - for bart max_length=1024,for t5
    max_length=512 and for gpt-2 it is set to be 1024 although there's no
    restriction.
  pad_to_max_length:
    A Boolean value, if set True, then the input sequence will be padded to
    max_length.
  truncation:
    A Boolean value, if set True, the input sequence will be truncated to
    max_length.


Returns:
  A dictionary of input_ids, attention_masks, token_type_ids.


2def train(dataset, lr, eps, batch_size, epochs, model, tokenizer, d
evice):
  """ Train GPT2 model and print out training details."""

  Args:
    dataset:
    training data, it's a 2-dim tensor, i.e.[4500,1024]represents
    4500 samples in total, and each of them has 1024 dims.
    lr:
    learning rate, set as a rather small number 1e-05 to avoid large change on
    the weights, default value is 1e-03.
    eps:
    term added to the denominator to improve numerical stability, set by
    default 1e-08.
    batch_size:
    number of samples in a batch for one backpropagation, set to be 2 but it
    can be larger if GPU allows.
    epochs:
    number of iterations over the whole input data, set by 5 due to GPU limits,
    but a larger number would be better.
    model:
    the model for training, it's loaded as a pre-trained model from the
    huggingface's libraries.
    tokenizer:
    the tokenizer for generating tokens, it's also directly loaded from the
    huggingface's libraries.
    device:
    an option for GPU.
```

```
 Returns:
      a list containing average training loss for each epoch.
```

[*Note: those hyperparameters on training() are set the same way among different language models for simplicity, but it's worth exploring different optimal settings for each of the models.]

```
3model.generate(input_ids, attention_mask, decoder_start_token_id,
max_length, min_length, num_beams, no_repeat_ngram_size):
  """ Generates sequence for models with a LM head."""

  Args:
     input_ids, attention_mask, decoder_start_token_id:
     input_ids, attention_mask and decoder_start_token_id of the input tokens,
     directly obtained from the tokenizing step. for gpt-2, it only has
     input_ids, but for encoder-decoder structure like t5 and bart, all three
     are required.
      max_length, min_length:
     max and min length of the generated sequence, but from my experiments,
     these constraints are not accurate, thus for a generated summary of length
     around 100, different max_length and min_length are set for different
     models and they are all chosen by the observations of limited experiments.
      num_beams:
     Number of beams for beam search. 1 means no beam search. Default to
     1. It's set to be 3 for all models when apply beam search.
      no_repeat_ngram_size:
     the number defines that the ngram can only appear once. For gpt-2 and bart,
     the value is set to be 2 but for t5 it's 3. The reason is that when it's 2,
     t5 outputs bad summaries and only with 3 it performs in a normal way.

  Returns:
     `torch.LongTensor` of shape `(batch_size * num_return_sequences,
     sequence_length)`
```

[*Note: Only greedy search and beam search are discussed and compared in the project, thus many other settings are absent, but it would be interesting to also include method like top-k sampling. Besides, there are also parameters like repetition_penalty, length_penalty to penalize on word repetitions or sequence length which can make an difference on the results.]

[source code from https://github.com/huggingface/transformers/blob/c4d4e8bdbd25d9463d41de6398940329c89b7fb6/src/transformers/generation_utils.py#L101

more explanations please refer to https://huggingface.co/blog/how-to-generate]

EVALUATIONS:

```
datasets.load_metric('rouge'):

""" Provide evaluation metrics for NLP tasks."""

Args:
     metric_name:
     name of the metric chosen from bertscore, bleu, bleurt, coval, gleu, glue,
     meteor, rouge, sacrebleu, seqeval, squad, squad_v2, xnli.


Returns:
     the metric object.

metric.compute(rouge_types=['rouge1',"rouge2","rouge3",'rougeL']):


""" Compute the metric value."""

Args:
     rouge_types:
     a list of the rouge types that to be returned.


Returns:
     A dictionary containing all specified touge_types. To access the rouge-1,
     use results["rouge1"], to access the recall of rouge-1, use
     results["rouge1"].mid.recall
```