Ask Wiki Problem definition

AskWiki must perform 2 tasks sequentially, first is to construct a SPARQL query based on question, second task is to verbalize and generate an answer from the query results.

Author: shrinivasbjoshi@berkeley.edu

NL generation based on wikidata triples

To generate NL answer to a question, Askwiki must input wikidata triples into NLG model and generate and summarize an english language response.

We considered T5 and OpenAi model families as candidates for the NLG.

Intution behind choosing T5 & OpenAI models

1. T5 will offer larger training and fine tuning opportunities
2. OpenAI offers wider selection of models and easier few shot training approaches

This Notebook provides overview of our T5 small NLG training and generation.

Approach to NL generation using WEB NLG 2020 challenge data

1. Training on webnlg 2020 data set

   WebNLG dataset provided ready to use RDF triples [similar to how Askwiki will generate triples] and annotated human language responses for those triples

2. Askwiki intends to answer a specific question using NL and does not want to just summarize set of triples into a paragraph. For those purposes we have not done extensive tuning of T5 models in this notebook, onus of generating an answer is on the eariler pipeline of the code and not necessrily on the NLG model.
3. This model is just reacting to the input RDF tiples , AskWiki did not have access to any specific question answer database for finetuning and utilized the webnlg dataset as language generator and summarizer trainer [not as answering model]

Installing the required packages

```
!pip install transformers
!pip install sentencepiece
import pandas as pd
import os
import torch
from transformers import T5Tokenizer, T5ForConditionalGeneration
from transformers.optimization import  Adafactor
import time
import warnings
warnings.filterwarnings('ignore')
from huggingface_hub import notebook_login

notebook_login()
```

```
    Token is valid.
    Your token has been saved in your configured git credential helpers (store).
    Your token has been saved to /root/.cache/huggingface/token
    Login successful
```

```
!pip install textstat
```

```
    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
    Collecting textstat
      Downloading textstat-0.7.3-py3-none-any.whl (105 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 105.1/105.1 kB 6.2 MB/s eta 0:00:00
    Collecting pyphen
      Downloading pyphen-0.14.0-py3-none-any.whl (2.0 MB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 48.5 MB/s eta 0:00:00
    Installing collected packages: pyphen, textstat
    Successfully installed pyphen-0.14.0 textstat-0.7.3
```

If you just want to test the pre trained model jump to W210 testing section and import the model from hugging face

Preprocess the data

webnlg2020 data from gitlab directly

```
import urllib.request
import zipfile
url = 'https://gitlab.com/shimorina/webnlg-dataset/-/archive/master/webnlg-dataset-master.zip?path=release_v3.0/en/train'
urllib.request.urlretrieve(url, 'web.zip')
with zipfile.ZipFile('web.zip', 'r') as zip_ref:
    zip_ref.extractall('web')
import glob
import os
import re
import xml.etree.ElementTree as ET
import pandas as pd
files = glob.glob("/content/web/webnlg-dataset-master-release_v3.0-en-train/release_v3.0/en/train/**/*.xml", recursive=True)
triple_re=re.compile('(\d)triples')
data_dct={}
for file in files:
    tree = ET.parse(file)
    root = tree.getroot()
    triples_num=int(triple_re.findall(file)[0])
    for sub_root in root:
        for ss_root in sub_root:
            strutured_master=[]
            unstructured=[]
            for entry in ss_root:
                unstructured.append(entry.text)
                strutured=[triple.text for triple in entry]
                strutured_master.extend(strutured)
            unstructured=[i for i in unstructured if i.replace('\n','').strip()!='' ]
            strutured_master=strutured_master[-triples_num:]
            strutured_master_str=(' && ').join(strutured_master)
            data_dct[strutured_master_str]=unstructured
mdata_dct={"prefix":[], "input_text":[], "target_text":[]}
for st,unst in data_dct.items():
    for i in unst:
        mdata_dct['prefix'].append('AskWiki NLG: ')
        mdata_dct['input_text'].append(st)
        mdata_dct['target_text'].append(i)
```

```
df=pd.DataFrame(mdata_dct)
df.to_csv('webNLG2020_train.csv')
```

```
df[df['target_text']=='The Aarhus is the airport of Aarhus, Denmark.']
```

	prefix	input_text	target_text
7784	AskWiki NLG:	Aarhus_Airport | cityServed | "Aarhus, Denmark"	The Aarhus is the airport of Aarhus, Denmark.

```
train_df=pd.read_csv('webNLG2020_train.csv', index_col=[0])
```

```
#Perform Train and Test Split
from sklearn.model_selection import train_test_split
train_df, test_df = train_test_split(train_df, test_size=0.3)
```

```
train_df.count()
```

```
prefix         24639
input_text     24639
target_text    24639
dtype: int64
```

```
test_df.count()
```

```
prefix         10560
input_text     10560
target_text    10560
dtype: int64
```

```
batch_size=6
num_of_batches=int(len(train_df)/batch_size)
num_of_epochs=4
```

```
num_of_batches
```

```
4107
```

Checking for the GPU availability

```
if torch.cuda.is_available():
    dev = torch.device("cuda:0")
    print("Running on the GPU")
else:
    dev = torch.device("cpu")
    print("Running on the CPU")
```

```
Running on the GPU
```

Loading the pretrained model and tokenizer

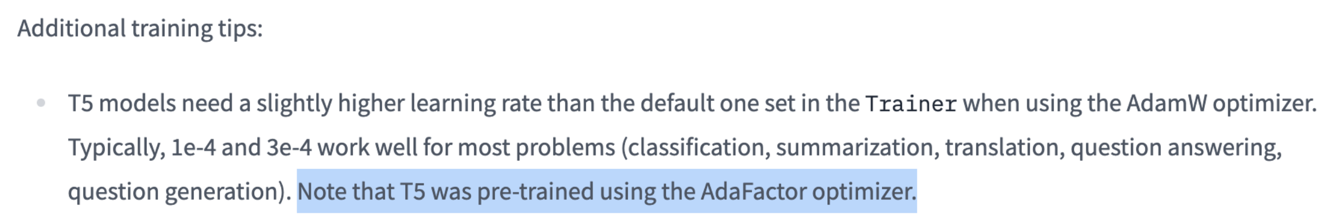
```
tokenizer = T5Tokenizer.from_pretrained('t5-large')
model = T5ForConditionalGeneration.from_pretrained('t5-large', return_dict=True)
#moving the model to device(GPU/CPU)
model.to(dev)
```

```
Downloading                                                    792k/792k [00:00<00:00,
(…)ve/main/spiece.model: 100%                                  20.7MB/s]
Downloading (…)lve/main/config.json:                          1.21k/1.21k [00:00<00:00,
100%                                                          58.4kB/s]
Downloading pytorch_model.bin:                              2.95G/2.95G [00:07<00:00,
100%                                                        396MB/s]
Downloading (…)neration_config.json:                            147/147 [00:00<00:00,
100%                                                            8.10kB/s]
```

```
T5ForConditionalGeneration(
  (shared): Embedding(32128, 1024)
  (encoder): T5Stack(
    (embed_tokens): Embedding(32128, 1024)
    (block): ModuleList(
      (0): T5Block(
        (layer): ModuleList(
          (0): T5LayerSelfAttention(
            (SelfAttention): T5Attention(
              (q): Linear(in_features=1024, out_features=1024, bias=False)
              (k): Linear(in_features=1024, out_features=1024, bias=False)
              (v): Linear(in_features=1024, out_features=1024, bias=False)
              (o): Linear(in_features=1024, out_features=1024, bias=False)
              (relative_attention_bias): Embedding(32, 16)
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (1): T5LayerFF(
            (DenseReluDense): T5DenseActDense(
              (wi): Linear(in_features=1024, out_features=4096, bias=False)
              (wo): Linear(in_features=4096, out_features=1024, bias=False)
              (dropout): Dropout(p=0.1, inplace=False)
              (act): ReLU()
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
      (1-23): 23 x T5Block(
        (layer): ModuleList(
          (0): T5LayerSelfAttention(
            (SelfAttention): T5Attention(
              (q): Linear(in_features=1024, out_features=1024, bias=False)
              (k): Linear(in_features=1024, out_features=1024, bias=False)
              (v): Linear(in_features=1024, out_features=1024, bias=False)
              (o): Linear(in_features=1024, out_features=1024, bias=False)
            )
            (layer_norm): T5LayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (1): T5LayerFF(
            (DenseReluDense): T5DenseActDense(
```

Initializing the optimizer

(tips from hugging face, utilizing the same adapter on which t5 was trained)



Additional training tips:

- T5 models need a slightly higher learning rate than the default one set in the `Trainer` when using the AdamW optimizer. Typically, 1e-4 and 3e-4 work well for most problems (classification, summarization, translation, question answering, question generation). Note that T5 was pre-trained using the AdaFactor optimizer.

```
optimizer = Adafactor(
    model.parameters(),
    lr=1e-3,
    eps=(1e-30, 1e-3),
    clip_threshold=1.0,
    decay_rate=-0.8,
    beta1=None,
    weight_decay=0.0,
    relative_step=False,
    scale_parameter=False,
    warmup_init=False
)
```

```
from IPython.display import HTML, display

def progress(loss,value, max=100):
    return HTML(""" Batch loss :{loss}
        <progress
            value='{value}'
            max='{max}',
            style='width: 100%'
        >
            {value}
        </progress>
    """.format(loss=loss,value=value, max=max))
```

Training the model

```
train_df.count()
```

```
prefix         24647
input_text     24647
target_text    24647
dtype: int64
```

```
#modified training code
model.train()

loss_per_10_steps=[]
for epoch in range(1,num_of_epochs+1):
  print('Running epoch: {}'.format(epoch))

  running_loss=0

  out = display(progress(1, num_of_batches+1), display_id=True)
  for i in range(num_of_batches):
    inputbatch=[]
    labelbatch=[]
    new_df=train_df[i*batch_size:i*batch_size+batch_size]
    for indx,row in new_df.iterrows():
      input = 'AskWiki NLG: '+row['input_text']+'</s>'
      labels = row['target_text']+'</s>'
      inputbatch.append(input)
      labelbatch.append(labels)
    input=tokenizer.batch_encode_plus(inputbatch,padding=True,max_length=768,return_tensors='pt')
    label=tokenizer.batch_encode_plus(labelbatch,padding=True,max_length=768,return_tensors="pt")
    inputbatch=input["input_ids"]
    inputattention=input["attention_mask"]
    labelbatch=label["input_ids"]
    labelattention=label["attention_mask"]

    #send to GPU
    inputbatch=inputbatch.to(dev)
    inputattention=inputattention.to(dev)
    labelbatch=labelbatch.to(dev)
    labelattention=labelattention.to(dev)

    # clear out the gradients of all Variables
    optimizer.zero_grad()

    # Forward propogation
    outputs = model(input_ids=inputbatch
                    ,attention_mask=inputattention
                    ,labels=labelbatch
                    ,decoder_attention_mask=labelattention
                    )
    loss = outputs.loss
    loss_num=loss.item()
    logits = outputs.logits
    running_loss+=loss_num
    if i%10 ==0:
      loss_per_10_steps.append(loss_num)
    out.update(progress(loss_num,i, num_of_batches+1))

    # calculating the gradients
    loss.backward()

    #updating the params
    optimizer.step()

  running_loss=running_loss/int(num_of_batches)
  print('Epoch: {} , Running loss: {}'.format(epoch,running_loss))
```

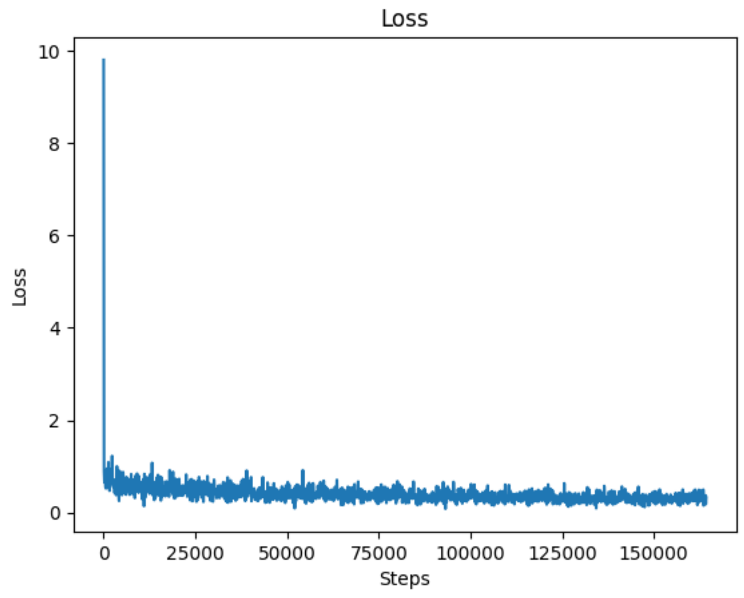
```
Running epoch: 1
Batch loss :0.5199187994003296
Epoch: 1 , Running loss: 0.5496556688492856
Running epoch: 2
Batch loss :0.4018687605857849
Epoch: 2 , Running loss: 0.40919128663534954
Running epoch: 3
Batch loss :0.3515186309814453
Epoch: 3 , Running loss: 0.3523144597373132
Running epoch: 4
Batch loss :0.24046590924263
Batch loss :0.29740798473358154
Epoch: 4 , Running loss: 0.3130683382392104
```

Plotting the loss over time

```
import matplotlib.pyplot as plt
```

```
steps = [i*100 for i in range(len(loss_per_10_steps))]

plt.plot(steps, loss_per_10_steps)
plt.title('Loss')
plt.xlabel('Steps')
plt.ylabel('Loss')
plt.show()
```



Push to huggingface

```
model.push_to_hub("V4T5LARGE")
```

```
Upload 1 LFS files: 100%                                   1/1 [04:57<00:00, 297.80s/it]

pytorch_model.bin: 100%                                    2.95G/2.95G [04:57<00:00, 10.3MB/s]

CommitInfo(commit_url='https://huggingface.co/shrinivasbjoshi/V4T5LARGE/commit/5d1cc2305e717081e73b4733e3a0e0e0fcdca6be', cor
T5ForConditionalGeneration', commit_description='', oid='5d1cc2305e717081e73b4733e3a0e0e0fcdca6be', pr_url=None, pr_revision=
```

AskWiki Testing

```
#if you have come here without training model then start from here
from transformers import AutoModel
tokenizer = T5Tokenizer.from_pretrained('t5-large')
```

```
Downloading (…)ve/main/spiece.model: 100%                  792k/792k [00:00<00:00, 2.88MB/s]

Downloading (…)lve/main/config.json: 100%                  1.21k/1.21k [00:00<00:00, 82.9kB/s]
```

```
AskWiki_NLG = T5ForConditionalGeneration.from_pretrained('shrinivasbjoshi/V4T5LARGE', return_dict=True)
```

```
Downloading (…)lve/main/config.json: 100%                  1.48k/1.48k [00:00<00:00, 74.2kB/s]

Downloading pytorch_model.bin: 100%                        2.95G/2.95G [01:01<00:00, 50.7MB/s]

Downloading (…)neration_config.json: 100%                  142/142 [00:00<00:00, 8.02kB/s]
```

```
AskWiki_NLG.to(dev)
```

```
            (wi): Linear(in_features=1024, out_features=4096, bias=False)
            (wo): Linear(in_features=4096, out_features=1024, bias=False)
            (dropout): Dropout(p=0.1, inplace=False)
            (act): ReLU()
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
    (1-23): 23 x T5Block(
      (layer): ModuleList(
        (0): T5LayerSelfAttention(
          (SelfAttention): T5Attention(
            (q): Linear(in_features=1024, out_features=1024, bias=False)
            (k): Linear(in_features=1024, out_features=1024, bias=False)
            (v): Linear(in_features=1024, out_features=1024, bias=False)
            (o): Linear(in_features=1024, out_features=1024, bias=False)
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (1): T5LayerCrossAttention(
          (EncDecAttention): T5Attention(
            (q): Linear(in_features=1024, out_features=1024, bias=False)
            (k): Linear(in_features=1024, out_features=1024, bias=False)
            (v): Linear(in_features=1024, out_features=1024, bias=False)
            (o): Linear(in_features=1024, out_features=1024, bias=False)
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (2): T5LayerFF(
          (DenseReluDense): T5DenseActDense(
            (wi): Linear(in_features=1024, out_features=4096, bias=False)
            (wo): Linear(in_features=4096, out_features=1024, bias=False)
            (dropout): Dropout(p=0.1, inplace=False)
            (act): ReLU()
          )
          (layer_norm): T5LayerNorm()
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
  )
  (final_layer_norm): T5LayerNorm()
  (dropout): Dropout(p=0.1, inplace=False)
)
(lm_head): Linear(in_features=1024, out_features=32128, bias=False)
)
```

```
input_ids = tokenizer.encode("AskWiki NLG: shrinivas | description | student && shrinivas | surname | joshi && shrinivas | student
input_ids=input_ids.to(dev)
outputs = AskWiki_NLG.generate(input_ids)
tokenizer.decode(outputs[0])
```

```
'<pad>shrinivas is a student at UC Berkeley and has the full name of'
```

```
t_input_ids = tokenizer.encode("AskWiki NLG: shrinivas | description | student && shrinivas | surname | joshi && shrinivas | stude
t_input_ids=t_input_ids.to(dev)
#outputs = AskWiki_NLG.generate(input_ids)
```

```
input=tokenizer.batch_encode_plus(inputbatch,padding=True,max_length=768,return_tensors='pt')
```

```
tensor([[ 8366,   518,  9069,   445, 24214,    10,     3, 31763,    29,  6823,
             7,  1820,  4210,  1820,  1236,     3,   184,   184,     3, 31763,
            29,  6823,     7,  1820,   244,  4350,  1820,     3,  1927,  5605,
             3,   184,   184,     3, 31763,    29,  6823,     7,  1820,  1236,
          1820,     3,  6463, 20776,     3,   184,   184,     3, 31763,    29,
          6823,     7,  1820,  1268,   945,  1820,  3136,   184,   855,  8153,
             3,   184,   184,     3, 31763,    29,  6823,     7,  1820,  1246,
          1820,  6426,     1]], device='cuda:0')
```

```
d_outputs_1 = model.generate(t_input_ids,do_sample=False,num_beams=6)
```

```
tokenizer.decode(d_outputs_1[0])
```

```
'<pad>Srinivas is a student at UC Berkeley and has the full name of'
```

T5 large does generate random outputs based on num of beams as evidenced above, the model would definitely need more rigorous training for AskWiki purposes, ideally on multiple GPUs and larger dataset and additional number of epochs

Compute Metrics on Generation

```
test_df.count()
```

```
prefix         10560
input_text     10560
target_text    10560
dtype: int64
```

```
batch_size=6
num_of_batches=int(len(test_df)/batch_size)
num_of_epochs=4
```

```
test_df
```

	prefix	input_text	target_text
1133	AskWiki NLG:	ALCO_RS-3 | builder | American_Locomotive_Comp...	The ALCO RS-3 has a V12 engine and is built by...
8795	AskWiki NLG:	A.F.C._Fylde | manager | Dave_Challinor && Dav...	Affiliated with Tranmere Rovers F.C., Dave Cha...
10657	AskWiki NLG:	Bakewell_pudding | region | Derbyshire_Dales &...	The dessert Bakewell pudding is from the Derby...
4287	AskWiki NLG:	Aleksandr_Prudnikov | club | FC_Terek_Grozny &...	Aleksandr Prudnikov played for FC Terek Grozny...
20545	AskWiki NLG:	Richland_Township,_Madison_County,_Indiana | c...	Richland Township, in Madison County, Indiana ...
...	...	...	...
29259	AskWiki NLG:	Above_the_Veil | numberOfPages | "248" && Abov...	Above the Veil by Garth Nix was produced in Pr...
6312	AskWiki NLG:	109_Felicitas | mass | 7.5 (kilograms) && 109_...	109 Felicitas has a mass of 7.5 kg and an apoa...
34528	AskWiki NLG:	Alcatraz_Versus_the_Evil_Librarians | language...	Alcatraz Versus the Evil Librarians is an Engl...
22373	AskWiki NLG:	Adam_Holloway | battle | Gulf_War && Adam_Holl...	Adam Holloway was in the Grenadier Guards in t...
18433	AskWiki NLG:	Superleague_Greece | champions | Olympiacos_F.C.	The Superleague Greece champions are Olympiaco...

10564 rows × 3 columns

```
#take a sample of  rows from test data to evaluate
sample_test_df=test_df.sample(frac=0.01)
```

```
sample_test_df.count()
```

```
prefix         106
input_text     106
target_text    106
dtype: int64
```

```
import inspect
inspect.signature(AskWiki_NLG.generate)
```

```
<Signature (inputs: Optional[torch.Tensor] = None, generation_config:
Optional[transformers.generation.configuration_utils.GenerationConfig] = None, logits_processor:
Optional[transformers.generation.logits_process.LogitsProcessorList] = None, stopping_criteria:
Optional[transformers.generation.stopping_criteria.StoppingCriteriaList] = None, prefix_allowed_tokens_fn:
Optional[Callable[[int, torch.Tensor], List[int]]] = None, synced_gpus: Optional[bool] = None, streamer:
Optional[ForwardRef('BaseStreamer')] = None, **kwargs) ->
Union[transformers.generation.utils.GreedySearchEncoderDecoderOutput,
transformers.generation.utils.GreedySearchDecoderOnlyOutput, transformers.generation.utils.SampleEncoderDecoderOutput,
transformers.generation.utils.SampleDecoderOnlyOutput, transformers.generation.utils.BeamSearchEncoderDecoderOutput,
transformers.generation.utils.BeamSearchDecoderOnlyOutput, transformers.generation.utils.BeamSampleEncoderDecoderOutput,
transformers.generation.utils.BeamSampleDecoderOnlyOutput,
transformers.generation.utils.ContrastiveSearchEncoderDecoderOutput,
transformers.generation.utils.ContrastiveSearchDecoderOnlyOutput, torch.LongTensor]>
```

```
#test cycle
AskWiki_outputs_top=[]
AskWiki_outputs_beams =[]
```

```
for indx,row in sample_test_df.iterrows():
  input = 'AskWiki NLG: '+row['input_text']+'</s>'
  input=tokenizer(input,padding=True,max_length=768,return_tensors='pt')
  input=input.to(dev)
  #Generate inferences
  #output_sequences = AskWiki_NLG.generate(input_ids=input["input_ids"],attention_mask=input["attention_mask"],do_sample=False,)
  #AskWiki_outputs.append(tokenizer.batch_decode(output_sequences, skip_special_tokens=True))
  outputs_top = AskWiki_NLG.generate(input_ids=input["input_ids"],attention_mask=input["attention_mask"],do_sample=False,num_beams
  AskWiki_outputs_top.append(tokenizer.batch_decode(outputs_top, skip_special_tokens=True))
  outputs_beams = AskWiki_NLG.generate(input_ids=input["input_ids"],attention_mask=input["attention_mask"],do_sample=True,top_k=50
  AskWiki_outputs_beams.append(tokenizer.batch_decode(outputs_beams, skip_special_tokens=True))
AskWiki_outputs_beams= list(np.concatenate(AskWiki_outputs_beams))
AskWiki_outputs_top= list(np.concatenate(AskWiki_outputs_top))
```

```
len(AskWiki_outputs_beams)
```

```
106
```

```
len(AskWiki_outputs_top)
```

```
106
```

```
AskWiki_outputs_top
```

```
['Alfred Moore Scales was a member of the Democratic Party in the U.S. He was Governor of North Carolina and was preceded by
James Madison Leach. Mr. Watson was also the successor to Alfred Mollinle.',
 'Len Wein was awarded an award from the Academy of Comic Book Arts (.) of the same year. The book is also home to the award
"Assocessor to another one. "In a \'Dean of comic book arts.',
 'AmeriGas is a company in the energy industry that serves the United States and has an operating income of $380,700,000
from operating revenue of 380700000. The U.S. is home to an impressive income.',
 'Associazione Calcio Chievo Verona is managed by Rolando Maran, who is a member of the Unione Triestina 2012 S.S.D. club.
The manager of A.C. Chiavo veronara.',
 "The ground of A.E Dimitra Efxeinoupolis is located in the town of Etxinoupoli. The town is the location of the home to the
A 'Alea di Mendrisio.",
 "Alison O'Donnell started performing in 1963 and was a member of the Mellow Candle and Bajik bands. Her musical genre is
folk music and she is associated with the musical artist, Alision O'Donnell. She is also in the same year.",
 "Batchoy includes the ingredient chicken which is a bird of the order 'birds. The chicken is an ingredient in the
batchoon.'. is from the chinese gen. are many different names including chicken.",
 'Alfred Moore Scales was born in Reidsville, North. Carolina and was a member of the Democratic Party in the U.S. He was
the Governor of North Carolina where he was succeeded by the politician James W. Reid.',
 'Eric Flint, born in Burbank, California, wrote 1634: The Ram Rebellion and was the author of the book "1634 A. The book
was also written by Eric Flint. "Rould" was his prequel.',
 'Aenir, written in English, was followed up by Above the Veil and is a prequel to the book "Above the Wilder". English is
the language spoken in the U.S.A. book.',
 'Spanish is the official language of Argentina where one of the ethnic groups are the Spanish language. It is also where
this country is where you will find the town known as "Alma mater". The language used in Argentina is Spanish.',
 'The Appleton International Airport\'s location id ATW is also the location ID for this airport. It has an area code of
"ATW". It is located in that same city. The name of the nearest airport to it is "Ahm".',
 'Susana Diaz is the leader of Andalusia, where the dish ajoblanco can be found in the Andalusian region of the country. The
food is also where you will find the food Ajo blanco.',
 'The chairman of A.S. Roma is James Pallotta.s. Claudiona is also the chair of the A S Roma oma. is a company in the same
name..A.C. Rome.',
 'Brussels is the location of the European University Association headquartered in Brussels. The university is known for
being home to the ethnic group of European Universities Association. Located in the city is also where the university itself
is a headquarters of other important people.',
 "Fuad Masum is the leader of Iraq, where you will find the town of Baku. is also where Iraq is where the dish originates
from. a similar dish is called 'Alsol dish's origins.",
 'Roy Thomas received an award from the Academy of Comic Book Arts. Mr. Thomas is also an exponent of the same religion in
that same city. The book is available in an impressive background...A. is where Roy Tomomas was born.',
 'The 11th Mississippi Infantry Monument is located in Adams County, Pennsylvania, which has Cumberland County to its north
and to the west of Franklin County in Maryland. To the north of Adams county lies Frederick County of Maryland, both of
which are also located.',
 'Bionico is a dessert from the Guadalajara region of Mexico. The leader of the country is Silvano Aureoles Conejo. Cottage
cheese can also be added there. It is also served as an alternative.',
 'Ajoblanco includes the ingredient almonds, which are from the Rosales order, and are classed as a flowering plant. The
order of Almonds is also home to the division of the order Arosae.',
 'Anders Osborne is signed to Rabadash Records and Alligator Records, the latter of which is a performer of the Blues.
Allogan Records main genre of music is the music genre, partly coming from blues music.',
 'The ground of A.S. Gubbio 1910 is located in Italy, where the Italian language is spoken and the capital is Rome. Sergio
Mattarella is the leader of the country. The inhabitants of Italy are called Italians.',
 '1097 Vicia, formerly known as 1928 PC, has an orbital period of 135589000.0 and a periapsis of 279142000000. It has the
epoch date of 31 December 2006 and was once called "1928 PC".',
 'Native Americans are an ethnic group in the United States, the birth place of Abraham A. Ribicoff, who was an American,
and died in New York City, where he was born. He was married to Ruth in Washington, U.S.',
 'Abdul Taib Mahmud was born in Miri, Malaysia and resides in Sarawak. He represented Kota Samarahan and was a member of the
Parti Pesaka Bumiputera Bersatu.',
 'Jorge Humberto Rodriguez plays for the Alianza F.C. club. Mr. Rodriguez is also a former club of his. The name of the club
he used to play for is Alianga FC in the U.S.',
 "Alex Tyus had been drafted in 2011 to make his a draft year of the same year in turn was number '10's draft team. Mr.
```

```
     Tylus was born in that year. Hma is where you will find the asteroid.",
      'The Ariane 5 was manufactured by the European Space Agency and launched from ELA-3 launchpad. Its maiden flight was on the
     2nd of March 2004 and its final flight took place on 18th of December 2009. It has a diameter of 5.4 metres.',
```

```
from nltk.translate.bleu_score import sentence_bleu
def bleu_calc(reference, candidate):
    return sentence_bleu(reference.split(), candidate.split())
```

```
sample_test_df["candidate_text_beams"] = AskWiki_outputs_beams
sample_test_df["candidate_text_top"] = AskWiki_outputs_top
```

```
sample_test_df
```

	prefix	input_text	target_text	candidate_text	blue_score	candidate_text_beams	candidate_text_top	blu
2550	AskWiki NLG:	Alfred_Moore_Scales | office | "Governor of No...	Alfred Moore Scales, a member of the Democrati...	Alfred Moore Scales, who was Governor of North...	0.0	Democratic Governor Alfred Moore Scales of Nor...	Alfred Moore Scales was a member of the Democr...	
9126	AskWiki NLG:	Len_Wein | award | Academy_of_Comic_Book_Arts	Len Wein has won awards from the Academy of Co...	Len Wein was awarded an award from the Academy...	0.0	Len Wein was awarded the Academy of Comic Book...	Len Wein was awarded an award from the Academy...	
6732	AskWiki NLG:	AmeriGas | regionServed | United_States && Ame...	AmeriGas an energy industry provides services ...	AmeriGas is a company in the energy industry t...	0.0	AmeriGas, an energy industry produces that ear...	AmeriGas is a company in the energy industry t...	
28333	AskWiki NLG:	A.C._Chievo_Verona | manager | Rolando_Maran &...	Associazione Calcio Chievo Verona is managed b...	Associazione Calcio Chievo Verona is managed b...	0.0	The manager of A.C. Chievo Verona is Rolando M...	Associazione Calcio Chievo Verona is managed b...	
			The ground of	The ground of A.F				

```
# Apply the function to the DataFrame
sample_test_df['blue_score_beam'] = sample_test_df.apply(lambda row: bleu_calc(row['target_text'], row['candidate_text_beams']), a
sample_test_df['blue_score_top'] = sample_test_df.apply(lambda row: bleu_calc(row['target_text'], row['candidate_text_top']), axis
```

```
import textstat
```

```
#reading score
sample_test_df['top_reading_score'] = sample_test_df.apply(lambda row: textstat.flesch_reading_ease(row['candidate_text_top']), ax
sample_test_df['beam_reading_score'] = sample_test_df.apply(lambda row: textstat.flesch_reading_ease(row['candidate_text_beams']),
```

```
sample_test_df['top_words']=sample_test_df['candidate_text_top'].apply(lambda x: len(x.split()))
sample_test_df['beam_words']=sample_test_df['candidate_text_beams'].apply(lambda x: len(x.split()))
```

```
np.mean(sample_test_df['top_words'])
```

```
     35.632075471698116
```

```
np.mean(sample_test_df['beam_words'])
```

```
     33.91509433962264
```

```
sample_test_df
```