American History GPT

- Goal: Learn more about American history
- Subgoal: Replace history teachers

Libraries

In [47]:
```
import torch
import numpy as np
import torch.nn as nn

import time
import os

import shutil
import glob

from torch.nn import functional as F
```

Check that cuda is being used

In [78]:
```
torch.cuda.is_available()
```

Out[78]:
```
True
```

Params

In [80]:
```
torch.manual_seed(256)
device = 'cuda' if torch.cuda.is_available() else 'cpu'

block_size        = 40      ## N tokens in sequence
batch_size        = 64
max_iters         = 300000
eval_interval     = 10000
learning_rate     = 0.0003
eval_iters        = 300
vocab_size        = 168  ## 65

## every id for a given token is embedded to vector of this size
n_embd            = 512
n_head            = 8         ## 8 attention heads
n_layer           = 6         ## 6 eoncoder layers
dropout           = 0.2
```

Data

- 102 United States History Books from Project Gutenburg

Merge the Files

In [49]:
```
filenames = glob.glob("E:/ITS 530/datasets/Books/Seperate/American_History/*.txt")
with open("output_file.txt", "wb") as outfile:
    for filename in filenames:
        with open(filename, "rb") as infile:
            shutil.copyfileobj(infile, outfile)
```

In [50]:
```
text = ''

input_file2 = 'output_file.txt'

with open(input_file2, 'r', encoding='utf-8') as f:
    text = f.read()
```

In [51]:
```
print("length of data in letter or characters")
len(text)
```

```
length of data in letter or characters
```

Out[51]: 53531405

In [52]:
```
list(set(text))
```

Out[52]: ['Ç',
'?',
'H',
'#',
'\xad',
'”',
'E',
'{',
'%',
'ë',
'W',
'3',
'\t',
'í',
'Ñ',
'0',
'î',
']',
'Y',
'(',
'r',
'5',
'9',
'´',
'P',
'è',
'4',
'}',
'\n',
'h',
'|',
'2',
'ú',
'\ufeff',
'f',
'6',
'³',
'F',
'Æ',
'Ê',
'ß',
'T',
'D',
'Ü',
'Z',
'i',
'J',
'ñ',
'ê',
'[',
'Ë',
'È',
'`',
'u',
'.',
'z',

```
'A',
'^',
'!',
'ä',
'B',
'à',
'X',
'ö',
'×',
'É',
'M',
'§',
'á',
'x',
'*',
'I',
'•',
's',
'+',
'=',
'y',
'k',
'·',
'Î',
'n',
'ô',
'$',
'p',
'N',
'Ò',
'—',
''',
'\xa0',
'm',
'b',
'v',
'1',
'Â',
'>',
'\x8a',
'Ô',
'¹',
'œ',
'K',
'À',
'ù',
'"',
'e',
'∴',
'U',
'd',
'8',
'ï',
'£',
'…',
'c',
```

```
'V',
'¢',
'O',
',',
'&',
'<',
'Å',
'R',
'‘',
'é',
':',
')',
'¨',
'º',
'L',
'w',
'\\',
'Ú',
'l',
'°',
'»',
'^',
'²',
'ü',
'@',
"'",
'“',
'ç',
'Ö',
'™',
'-',
'q',
'a',
'_',
';',
'æ',
'½',
'G',
'û',
'S',
'o',
'Í',
'/',
'\x9c',
'Œ',
'ó',
't',
'7',
'Ä',
'â',
'g',
'å',
'j',
' ',
'Q',
'C']
```

In [55]:
```
the_chars  = sorted(     list(set(text))     )

vocab_size = len( the_chars )      ## 65

print(  len(the_chars)  )

print(  ''.join(the_chars)  )

## The printed oputput
## !$&',-.3:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
```

```
168

 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrs
tuvwxyz{|}¢£§¨º²³´.¹º»¼ÀÂÄÅÆÇÈÉÊËÍÎÑÒÔÖ×ÚÜßàáâäåæçèéêëíîïñóôöùúûüŒœˆ–‘’“”•…™
```

In [56]:
```
stoi = { ch:i for i, ch in enumerate(the_chars) }
itos = { i:ch for i, ch in enumerate(the_chars) }
```

In [57]:
```
print( stoi )
print( itos )
```

```
{'\t': 0, '\n': 1, ' ': 2, '!': 3, '"': 4, '#': 5, '$': 6, '%': 7, '&': 8, "'": 9,
'(': 10, ')': 11, '*': 12, '+': 13, ',': 14, '-': 15, '.': 16, '/': 17, '0': 18,
'1': 19, '2': 20, '3': 21, '4': 22, '5': 23, '6': 24, '7': 25, '8': 26, '9': 27,
':': 28, ';': 29, '<': 30, '=': 31, '>': 32, '?': 33, '@': 34, 'A': 35, 'B': 36,
'C': 37, 'D': 38, 'E': 39, 'F': 40, 'G': 41, 'H': 42, 'I': 43, 'J': 44, 'K': 45,
'L': 46, 'M': 47, 'N': 48, 'O': 49, 'P': 50, 'Q': 51, 'R': 52, 'S': 53, 'T': 54,
'U': 55, 'V': 56, 'W': 57, 'X': 58, 'Y': 59, 'Z': 60, '[': 61, '\\': 62, ']': 63,
'^': 64, '_': 65, '`': 66, 'a': 67, 'b': 68, 'c': 69, 'd': 70, 'e': 71, 'f': 72,
'g': 73, 'h': 74, 'i': 75, 'j': 76, 'k': 77, 'l': 78, 'm': 79, 'n': 80, 'o': 81,
'p': 82, 'q': 83, 'r': 84, 's': 85, 't': 86, 'u': 87, 'v': 88, 'w': 89, 'x': 90,
'y': 91, 'z': 92, '{': 93, '|': 94, '}': 95, '\x8a': 96, '\x9c': 97, '\xa0': 98,
'¢': 99, '£': 100, '§': 101, '¨': 102, '\xad': 103, '°': 104, '²': 105, '³': 106,
'´': 107, '·': 108, '¹': 109, 'º': 110, '»': 111, '½': 112, 'À': 113, 'Â': 114, 'Ä':
115, 'Å': 116, 'Æ': 117, 'Ç': 118, 'È': 119, 'É': 120, 'Ê': 121, 'Ë': 122, 'Í': 123,
'Î': 124, 'Ñ': 125, 'Ò': 126, 'Ô': 127, 'Ö': 128, '×': 129, 'Ú': 130, 'Ü': 131, 'ß':
132, 'à': 133, 'á': 134, 'â': 135, 'ä': 136, 'å': 137, 'æ': 138, 'ç': 139, 'è': 140,
'é': 141, 'ê': 142, 'ë': 143, 'í': 144, 'î': 145, 'ï': 146, 'ñ': 147, 'ó': 148, 'ô':
149, 'ö': 150, 'ù': 151, 'ú': 152, 'û': 153, 'ü': 154, 'Œ': 155, 'œ': 156, 'ˆ': 157,
'–': 158, '‘': 159, '’': 160, '“': 161, '”': 162, '•': 163, '…': 164, '™': 165, '∴':
166, '\ufeff': 167}
{0: '\t', 1: '\n', 2: ' ', 3: '!', 4: '"', 5: '#', 6: '$', 7: '%', 8: '&', 9: "'", 1
0: '(', 11: ')', 12: '*', 13: '+', 14: ',', 15: '-', 16: '.', 17: '/', 18: '0', 19:
'1', 20: '2', 21: '3', 22: '4', 23: '5', 24: '6', 25: '7', 26: '8', 27: '9', 28:
':', 29: ';', 30: '<', 31: '=', 32: '>', 33: '?', 34: '@', 35: 'A', 36: 'B', 37:
'C', 38: 'D', 39: 'E', 40: 'F', 41: 'G', 42: 'H', 43: 'I', 44: 'J', 45: 'K', 46:
'L', 47: 'M', 48: 'N', 49: 'O', 50: 'P', 51: 'Q', 52: 'R', 53: 'S', 54: 'T', 55:
'U', 56: 'V', 57: 'W', 58: 'X', 59: 'Y', 60: 'Z', 61: '[', 62: '\\', 63: ']', 64:
'^', 65: '_', 66: '`', 67: 'a', 68: 'b', 69: 'c', 70: 'd', 71: 'e', 72: 'f', 73:
'g', 74: 'h', 75: 'i', 76: 'j', 77: 'k', 78: 'l', 79: 'm', 80: 'n', 81: 'o', 82:
'p', 83: 'q', 84: 'r', 85: 's', 86: 't', 87: 'u', 88: 'v', 89: 'w', 90: 'x', 91:
'y', 92: 'z', 93: '{', 94: '|', 95: '}', 96: '\x8a', 97: '\x9c', 98: '\xa0', 99:
'¢', 100: '£', 101: '§', 102: '¨', 103: '\xad', 104: '°', 105: '²', 106: '³', 107:
'´', 108: '·', 109: '¹', 110: 'º', 111: '»', 112: '½', 113: 'À', 114: 'Â', 115: 'Ä',
116: 'Å', 117: 'Æ', 118: 'Ç', 119: 'È', 120: 'É', 121: 'Ê', 122: 'Ë', 123: 'Í', 124:
'Î', 125: 'Ñ', 126: 'Ò', 127: 'Ô', 128: 'Ö', 129: '×', 130: 'Ú', 131: 'Ü', 132: 'ß',
133: 'à', 134: 'á', 135: 'â', 136: 'ä', 137: 'å', 138: 'æ', 139: 'ç', 140: 'è', 141:
'é', 142: 'ê', 143: 'ë', 144: 'í', 145: 'î', 146: 'ï', 147: 'ñ', 148: 'ó', 149: 'ô',
150: 'ö', 151: 'ù', 152: 'ú', 153: 'û', 154: 'ü', 155: 'Œ', 156: 'œ', 157: 'ˆ', 158:
'–', 159: '‘', 160: '’', 161: '“', 162: '”', 163: '•', 164: '…', 165: '™', 166: '∴',
167: '\ufeff'}
```

In [58]:
```
encode = lambda s: [ stoi[c]          for c in s   ]

encode("bahh")
```

Out[58]:
```
[68, 67, 74, 74]
```

In [59]:
```
decode = lambda l: ''.join(   itos[i] for i in l   )

decode([55, 54, 61, 61])
```

Out[59]:
```
'UT[['
```

In [60]:
```
data = torch.tensor(   encode(text), dtype=torch.long   )
```

```
print( data )
```

```
tensor([167,  54,  74,  ...,   1,   1,   1])
```

In [61]:
```
n          = int(   0.9*len(data)   )

train_data = data[:n]
val_data   = data[n:]
```

In [62]:
```
def get_batch(split):
    if split == "train":
        data = train_data
    else:
        data = val_data

    ix = torch.randint(   len(data) - block_size, (batch_size,)   )

    x  = torch.stack(    [  data[   i : i+block_size ]     for i in ix ]    )
    y  = torch.stack(    [  data[ i+1 : i+1+block_size ]   for i in ix ]    )

    x, y = x.to(device), y.to(device)

    return x, y
```

In [63]:
```
temp_batch_size = 4
temp_block_size = 16

## select random starting points for the 4 sentences
ix = torch.randint(
            len(data) - block_size,
            (temp_batch_size,)
)

print( ix )
```

```
tensor([36045809, 50637744,  1395100, 24036015])
```

In [64]:
```
for index_temp in ix:
    print(  data[index_temp]  )
```

```
tensor(71)
tensor(86)
tensor(2)
tensor(67)
```

In [65]:
```
x  = torch.stack(
    [ data[   i : i+  temp_block_size ]   for i in ix ]

)

y  = torch.stack(
    [ data[ i+1 : i+1+ temp_block_size ]  for i in ix ]
)

print(x)
print(y)
```

```
tensor([[71, 79, 81, 88, 71, 70,  2, 85, 87, 69, 74,  2, 81, 72,  2, 86],
        [86, 75, 81, 80,  2, 81, 72,  2, 82, 81, 89, 71, 84, 16,  1,  1],
        [ 2, 68, 87, 86,  2, 67, 86,  2, 86, 74, 75, 85,  2, 82, 81, 75],
        [67, 85, 71, 70,  2, 75, 80,  2, 86, 74, 75, 85,  2, 85, 81, 70]])
tensor([[79, 81, 88, 71, 70,  2, 85, 87, 69, 74,  2, 81, 72,  2, 86, 74],
        [75, 81, 80,  2, 81, 72,  2, 82, 81, 89, 71, 84, 16,  1,  1, 54],
        [68, 87, 86,  2, 67, 86,  2, 86, 74, 75, 85,  2, 82, 81, 75, 80],
        [85, 71, 70,  2, 75, 80,  2, 86, 74, 75, 85,  2, 85, 81, 70, 70]])
```

In [66]:
```python
@torch.no_grad()    ## for efficient processing
def estimate_loss():
    out = {}
    model.eval()   ## set to no training
    for split in ['train', 'val']:
        losses = torch.zeros(eval_iters)
        for k in range(eval_iters):
            X, Y = get_batch(split)
            logits, loss = model(X, Y)
            losses[k] = loss.item()
        out[split] = losses.mean()
    model.train()  ## back to training
    return out
```

NN Architectures

In [67]:
```python
class Head(nn.Module):

    def __init__(self, head_size):
        super().__init__()

        self.key   = nn.Linear(n_embd, head_size, bias=False)  ## [512, 64]
        self.query = nn.Linear(n_embd, head_size, bias=False)  ## [512, 64]
        self.value = nn.Linear(n_embd, head_size, bias=False)  ## [512, 64]

        tril_def = torch.tril( torch.ones(block_size, block_size) )  ## [40, 40]

        self.register_buffer(
                  'tril',
                  tril_def
               )

        self.dropout = nn.Dropout(dropout)

    def forward(self, x):

        B, T, E = x.shape   ## [batch_size, 40, 512]

        k = self.key(   x )            ## k = (B, T, 64)
        q = self.query( x )            ## q = (B, T, 64)

        E2 = 64     ## I think this is 64 and not 512
        ## (B, T, E) @ (B, E, T)  -> (B, T, T)
        wei = q @ k.transpose(-2, -1) * E2 ** -0.5
```

```
        wei = wei.masked_fill(
                    self.tril[:T, :T] == 0,
                    float('-inf')
        )

        ## (B, T, T)
        wei = F.softmax( wei, dim= -1 )         ## (B, T, T)
        wei = self.dropout(   wei   )

        ## perform weighted aggregation of values

        v   = self.value(  x  )   ## x = (B, 40, E)
        out = wei @ v             ## (B, T, T) @ (B, T, 64) -> (B, T, 64)

        return out
```

In [68]:
```
class FeedForward(nn.Module):

    def __init__(self, n_embd):         ## 512

        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(n_embd, 4 * n_embd),      ## [512, 4*512]
            nn.ReLU(),
            nn.Linear(4 * n_embd, n_embd),      ## [4*512, 512]
            nn.Dropout(dropout),
        )

    def forward(self, x):
        return self.net(x)
```

In [69]:
```
class MultiHeadAttention(nn.Module):

    def __init__(self, num_heads, head_size):    ## (8, 64)
        super().__init__()
        self.heads = nn.ModuleList(  [ Head(head_size) for _ in range(num_heads) ]
        self.proj  = nn.Linear(n_embd, n_embd)   ## 512, 512
        self.dropout = nn.Dropout(dropout)

    def forward(self, x):
        out = torch.cat(   [ h(x) for h in self.heads ], dim = -1   )
        out = self.proj(  out   )
        out = self.dropout(   out   )
        return out
```

In [70]:
```
class Block(nn.Module):

    def __init__(self, n_embd, n_head):     ## (512, 8)
        super().__init__()
        head_size = n_embd // n_head        ## 64
        self.sa   = MultiHeadAttention(n_head, head_size)
        self.ffwd = FeedForward( n_embd)    ## 512
        self.ln1  = nn.LayerNorm(n_embd)
        self.ln2  = nn.LayerNorm(n_embd)
```

```
    def forward(self, x):
        x = x + self.sa(     self.ln1(x)      )
        x = x + self.ffwd(   self.ln2(x)      )
        return x
```

In [71]:

```
class GPTModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.token_embedding_table = nn.Embedding(vocab_size, n_embd)   ## [65, 512
        self.pos_emb_table = nn.Embedding(block_size, n_embd)     ## [block, 512]

        self.blocks = nn.Sequential(
                *[   Block(n_embd, n_head=n_head) for _ in range(n_layer)    ]
        )

        self.ln_f    = nn.LayerNorm(  n_embd    )
        self.lm_ffw_head = nn.Linear(n_embd, vocab_size)  ## [512, 65] # FFW Layer

    def forward(self, idx, targets=None):
        B, T = idx.shape     ## (Batch, 40)
        ## ids and targets are both (B, T) tensors of integers

        tok_emb = self.token_embedding_table(idx)
        pos_emb = self.pos_emb_table(torch.arange(T, device=device))

        x = tok_emb + pos_emb    ## [B, T, E] or [64, 40, 512]

        ## This is the architecture
        x = self.blocks(  x  )   ## (B, T, E)
        x = self.ln_f(    x  )   ## (B, T, E)   ## norm
        logits = self.lm_ffw_head(x)         ## [B, 40, 65]

        if targets is None:
            loss = None
        else:
            B, T, E  = logits.shape
            logits  = logits.view( B*T, E)
            targets = targets.view(B*T)
            loss    = F.cross_entropy(logits, targets)
        return logits, loss

    def generate(self, idx, max_new_tokens):    ## idx is (B, T)
        for _ in range(max_new_tokens):
            ## crop idx to the last block_size tokens
            idx_cond = idx[:, -block_size:]
            logits, loss = self(idx_cond)    ## ## get preds
            logits = logits[:, -1, :]    ## focus on last one (B, E)
            probs = F.softmax(logits, dim= -1)    ## (B, E) get probs
            idx_next = torch.multinomial(probs, num_samples=1)     ## (B, 1) select
            idx = torch.cat(  (idx, idx_next), dim=1  )   ## (B, T+1) append sample
        return idx
```

In [72]:
```
model    = GPTModel()

m        = model.to(device)

optimizer = torch.optim.Adam(  m.parameters(), lr=learning_rate    )
```

In [81]:
```
start_time = time.perf_counter()
for iter in range(max_iters):

    if iter % eval_interval == 0:
        losses = estimate_loss()
        print(f"step {iter}: train loss {losses['train']:.4f}, val loss {losses['va

    xb, yb = get_batch('train')

    ## eval the loss
    logits, loss = m(xb, yb)

    optimizer.zero_grad(set_to_none=True)   ## zero out
    loss.backward()
    optimizer.step()
end_time = time.perf_counter()
print(f"Total Training Time: {end_time - start_time:0.4f} seconds")
```

```
step 0: train loss 1.3366, val loss 1.4400
step 10000: train loss 1.2785, val loss 1.3796
step 20000: train loss 1.2421, val loss 1.3471
step 30000: train loss 1.2170, val loss 1.3299
step 40000: train loss 1.1968, val loss 1.3118
step 50000: train loss 1.1873, val loss 1.3021
step 60000: train loss 1.1783, val loss 1.2899
step 70000: train loss 1.1672, val loss 1.2888
step 80000: train loss 1.1635, val loss 1.2777
step 90000: train loss 1.1523, val loss 1.2764
step 100000: train loss 1.1460, val loss 1.2667
step 110000: train loss 1.1416, val loss 1.2636
step 120000: train loss 1.1414, val loss 1.2581
step 130000: train loss 1.1363, val loss 1.2566
step 140000: train loss 1.1351, val loss 1.2579
step 150000: train loss 1.1304, val loss 1.2597
step 160000: train loss 1.1248, val loss 1.2436
step 170000: train loss 1.1198, val loss 1.2475
step 180000: train loss 1.1197, val loss 1.2433
step 190000: train loss 1.1197, val loss 1.2439
step 200000: train loss 1.1161, val loss 1.2456
step 210000: train loss 1.1109, val loss 1.2411
step 220000: train loss 1.1118, val loss 1.2321
step 230000: train loss 1.1092, val loss 1.2384
step 240000: train loss 1.1093, val loss 1.2347
step 250000: train loss 1.1047, val loss 1.2353
step 260000: train loss 1.0984, val loss 1.2386
step 270000: train loss 1.1021, val loss 1.2315
step 280000: train loss 1.1021, val loss 1.2338
step 290000: train loss 1.0919, val loss 1.2201
Total Training Time: 13221.3996 seconds
```

Total Training Time in hours: 3 hrs 40 min 21 sec

In [82]:
```
## Starting token  id_sos = 0
sos_context = torch.zeros(  (1, 1),  dtype=torch.long, device=device   )

generated_text = m.generate(sos_context, max_new_tokens=500)[0].tolist()

print(  decode(generated_text)   )
```

```
        _Face of
Perkins'—war    Business.—Senator of Perry.—Elliot Root in 1842.—The Allies.%--The N
apoleonic
   Basin.

_Why was Mexico_ the matter endowed in resolution that the same being practically
phrased to a means rising out of data with five
months. He was consecrated, Webster, who & N.H.
Roberts reviewes demand a receival of the increase of meeting in individual seas. Th
ese
defeated laws have published, it must be well
placed in cipheral comfort, and the keen range have
maintained wheat agai
```

In [83]:
```
sos_context = torch.ones(  (1, 1),  dtype=torch.long, device=device   )

generated_text = m.generate(sos_context, max_new_tokens=500)[0].tolist()

print(  decode(generated_text)   )
```

```
Western South Carolina was not won in practice than he was of this object in connect
ion. He was admitted to the day of unity as therefits was
confined upon a majority of those who have
separated jurisdiction.'

The Westerns under George Penn in the United States of America._]

The Great Lakes that exploded the Bower Sturdill. To Nashville, in whatever
manners, Superintendence and theirs", being "Christians.") The son therefore might s
tir more men averse yells into the gun, wore cattel
about Pout
```

Q&A

In [84]:
```
new_lst = encode("What is America?")
```

In [85]:
```
new_np = np.array(  new_lst   )
new_np
```

Out[85]:
```
array([57, 74, 67, 86,  2, 75, 85,  2, 35, 79, 71, 84, 75, 69, 67, 33])
```

In [86]:
```
new_context = torch.tensor(new_np, dtype=torch.long, device=device )
```

```
new_context = new_context.view( (1, -1))
new_context
```

Out[86]:
```
tensor([[57, 74, 67, 86,  2, 75, 85,  2, 35, 79, 71, 84, 75, 69, 67, 33]],
       device='cuda:0')
```

In [87]:
```
generated_text = m.generate(new_context, max_new_tokens=1000)[0].tolist()

print(  decode(generated_text)   )
```

```
What is America? How is this order, and very combine now had proved
him a raking baron, but it was lie to its morally in
due view; but some court is good interest in the
Government for the present. Congress's removal to Canada, he would "get out the firs
t addition of the fifty Years. It was not very significant to go any better chances,
when their year later
foreigners, threatened their screams of resiin, consequently picturesqueness, and hi
s
warriors of deep tudy. The snow springs trees are
bitterly supersediness.

I please, think, there resorted to an army as specifically
equal; but they are very much, at least to give
the former Debs and Generals "making their
Benedicts," to lay them in arms and the modern forests in the confiscatory night. Fr
om
1660 to the more President Hampshire. The strength of the
Dutchester, who, of the Marque Act, now met at
them, and half the (repel with half the children's earth, was thought
worn-out of a prize. French of Britain wrote of the improvement of supremacy that
```

In [88]:
```
new_lst = encode("When was America founded?")

new_np = np.array(  new_lst   )

new_context = torch.tensor(new_np, dtype=torch.long, device=device )

new_context = new_context.view( (1, -1))

generated_text = m.generate(new_context, max_new_tokens=1000)[0].tolist()

print(  decode(generated_text)   )
```

```
When was America founded?

C. H. Supply of 1824-37-1820. Conspicuo and Whigan. Some men
reared as follows:

    The amendment of the House on Mr. Hutchinson as a complete needen
fear. There has Beecher beguiled his readiness into nine one-half. There were not le
ss local elections than the league, and flouted another at accomplishment now, the r
ight to all
American citizens of the class. Though the business of the
Pies, the Court discussed the literature; and what parts for
From this public of lay an acre of rural negro
question the people.

Mr. R. W. Keller delegations advocated the temporary
loan of five votes, because they may see them to
have put such language. Happily were put by everybody
of napoly. While the capital lay, and fresh the other, cloth o'clock, were importuna
tely digested into about a
          branch used for their salvation without terminal protection systems. It wa
s consequent to the militia regiment.

2. Coming into a proclamation and unabandondden army call them. It on a circumstance
cle
```

In [89]:
```
new_lst = encode("Who won the civil war?")

new_np = np.array(  new_lst   )

new_context = torch.tensor(new_np, dtype=torch.long, device=device )

new_context = new_context.view( (1, -1))

generated_text = m.generate(new_context, max_new_tokens=1000)[0].tolist()

print(  decode(generated_text)   )
```

```
Who won the civil war?

§§§ 422, 432.--_a_. Why was the wildest execution in pursuance of the minors. This w
as the
Junction, and advised an electronic workman

Pursuant to the Project Gutenberg™ collection will
remain freely available, and should be enjoined, in the action, one of a new
indignation, usage, did the right to reflections in
him, he could abandon the first, turns the master killed the tenth century. He was a
ppointed accordingly elected a Union and
Commander-in-chief of General Bragg.

Completing the controversy.

[Illustration: The Cohons and the Koyukuk Islanders]

A Few Marmont--16.

At best, suhtered brother on the newspaper.

We went to bed the chief companion. Marshal Granger was
deciding to invent a tax on railroad payments on bankruptcy by getting the rear of t
he editor of
"Three Duties," not on a most menacing kind
of genius you will be a verge of observation, or impracticable, it was considered as
embraced if Pocahontas became
dwelling upon the King, in Virginia he directs a firm not a
```

Fill in the Blank

In [90]:

```
new_lst = encode("America is")

new_np = np.array(  new_lst   )

new_context = torch.tensor(new_np, dtype=torch.long, device=device )

new_context = new_context.view( (1, -1))

generated_text = m.generate(new_context, max_new_tokens=1000)[0].tolist()

print(  decode(generated_text)   )
```

America is walking out of first inviolated laws of
the Confederation, and the next year Wilson, which he
disciplined leads below the comforts of Christopher were now changing to _all_.

"The name of expression as also =Skagames is ended by centre. A gross inceremonizati
on of industry, interesting person than priieves, it could be lifted again, and art
commended for it.

     Finding that he interposes on the experience of General Sheridan's
eloquence is hard to dispose, the right to renew the power
of government in two States and another barrel of rigges_ | 1 | 2 |

Total | | | 11 | | 3 | |o--| 14 | -- | -- | | | |20 || | 2 | --
 Between | 3,791,882|14,861|29,175|16.9| 3,288| 182.75.3
District of   3.10. Holghland. 1850.
  Fowles...Sdartine.
  Sailor, Sherman, Edward R., Charles W., nominated forty years by
Fenton's reply and robustitude.

"Now,

In [91]:
```
new_lst = encode("Europe is")

new_np = np.array(  new_lst   )

new_context = torch.tensor(new_np, dtype=torch.long, device=device )

new_context = new_context.view( (1, -1))

generated_text = m.generate(new_context, max_new_tokens=1000)[0].tolist()

print(  decode(generated_text)   )
```

Europe is, but was largely impossible as its
aristocratic and mutual prospects. The parties, in
Canada, while the waving of the Indians
were issued in Lincoln's House. The intent of training might therefore continue so v
astly, and he is best one of the
greatest favourites, and gradually trim to as they attributed the people.

[Illustration: Portrait.]
Goosevesee looking forth a "preliminary"--in
cases which have so matched breath a reality, however, was due to a lamp that
these men should come from Blanch's House.

Seven years later Franklin WICCOTT, two words in detachment
on the one hand, under the Democratic Jury, and the Department of Federal Office, th
e Court rules, held that the confusion in
Jacob Brown in 1848 five-year-note. In 1858 opposition William
Kearsley, a free and his territory, it was property or discovery of a just quota reg
ion, excluding believe, without
enrolling such evils, similar and personal execution, II.
  proportions of the States, IV.
  memorial and elaborate revolt

In [92]:
```
new_lst = encode("Canada is")

new_np = np.array(  new_lst   )

new_context = torch.tensor(new_np, dtype=torch.long, device=device )

new_context = new_context.view( (1, -1))

generated_text = m.generate(new_context, max_new_tokens=1000)[0].tolist()

print(  decode(generated_text)   )
```

```
Canada is short of Coloradose which later complains Capt. John Price, a Tory
   scene which rushed into despotillato widely and chucklebering; against the
connection of Indians. After that obligation has appeared for more money repealed by
public goods which shall be sued functioned as very
ill. But a defective speecher left the breaking out of the
     machines of the ruffled horns oak to land stone, sold,
numbers of cotton on both sides, and pointed four hundred vanquis gener gute in
] love of a single quorum"                                51



Kentucky and South                                        583
      Status of the sufferings of discussion showed that Greely situate is gifted wi
th that of
the judicial conciliation), or to the institution and in favor of a virtue
of practical disciousness abuse of the buyers of that State. Soon the
neighboring wheel was limely uplifted by international progress, Reynal, and John Mo
rris arrived.


This accomplishment touches, as if they had beheld five
thou
```

Future Work

- More Data
- Longer Train Time

Figuring out dimensions

In [31]:
```
new_context.shape
```

Out[31]:
```
torch.Size([1, 14])
```

In [32]:
```
sos_context_tmp = torch.ones(  (1, 1),  dtype=torch.long, device=device   )
sos_context_tmp.shape
```

Out[32]:
```
torch.Size([1, 1])
```