Section 1 Questions: (submit as ‘myTigerID_cshw1.pdf’):

1. Train a bigram model on the corpus. Run it on the following commands and list the output:
    a. python myTigerID_cshw1_gram.py predict_ngram --word whale --nwords 100 --load PathToBigramModel.p

        ```
        whale ’ s more was this course steered straight flame and povelson , you jump when the wreck . but i ’ s the elephant in that in their outreaching comprehensiveness and that final consequence ? only fenced by a staid , and a spare spars and schoolmasters . the terrific ahab rushed to one . well , entitled cetology . “ the common usage had two seamen to you — that there , and so that , when stubb had led by one side , then , shipmates would answer me a little oil . ere forgetfulness altogether with the
        ```

    b. python myTigerID_cshw1_gram.py predict_ngram --word whale --nwords 100 --load PathToBigramModel.p --d

        ```
        whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale , and the whale
        ```

    c. python myTigerID_cshw1_gram.py predict_ngram --word tomato --nwords 100 --load PathToBigramModel.p

        ```
        Error: Word not found in training data.
        tomato
        ```

2. Train a trigram model on the corpus. Run it on the following commands and list the output:
    a. python myTigerID_cshw1_gram.py predict_ngram --word “the harpooneer” --nwords 100 --load PathToTrigramModel.p

        ```
        the harpooneer oar , he transports himself with boots at all . suddenly queequeg started to his wearied mates , and pine wood ; and no coffin and no coffin and went out of the harpoons and line . presently , catching sight of him . ” upon this strange affair myself . ” — _charles lamb ’ s beginning . ” — _job_ . “ pull up — pull up ! shiver her ! — aloft there ! there again ! — a savage , so long been bound . but no sooner trimmed the yards swung round ; starboard and
        ```

b. python myTigerID_cshw1_gram.py predict_ngram --word “the harpooneer” --nwords 100 --load PathToTrigramModel.p --d

   the harpooneer is a thing not to be the first time behold father mapple , so that the whale , and the

c. Run 2.b again without –d, 3 times in a row.

   Run1:
   the harpooneer sleepy at the girdling line of advance be strictly confined to its own unavoidable , straight out beyond the duodecimo , this omnitooled , open - mouthed at times his hate seemed almost theirs ; the masoned , walled - town , and victorious fowl , that at any time to become pursed together . such a sight strange to see him now — a whaleman too — shall we keep chasing this murderous fish till he couldn ’ t half like that just quoted from langsdorff , you might say a touch , and hailing the three - fourths

   Run2:
   the harpooneer oar , bethink him that at every motion of his wrinkles ; the better qualified and set on edge , for he might make himself at home , he led the chase was now wrapped in a horse - shoe stubbs , sir ? ” “ merchant service many captains never show themselves on deck , for ahab too is mad . yet when jonah was swallowed down and pulverize that subaltern ’ s boat , which he sailed from home ; all these things in his berth , and swear it now on negro hill or in tormented chase

   Run3:
   the harpooneer is it ye pull for your one lost leg ; while , drew out , and wag thy ears . you would take his hammer the heavy beating of his intentions , at times — that was sudden , as a dead comrade from the prairie . “ the things that you can ’ t you snap your oars , sir , ’ says jonah now , bulkington ? glimpses do ye do when ye come ? ” he added — “ gripe your oars , and various other parts of the sperm whale , so that he soon mounted

Section 2 Questions: (append to submit as ‘myTigerID_cshw1.pdf’):

1. Train a BPE tokenizer using the default value for k. What is the output of: python myTigerID_cshw1_bpe.py tokenize --text The bright green Norwegian avocado was eaten by the whale! --load PathToBPEModel.p

```
Tokens: ['the</w>', 'br', 'i', 'gh', 't</w>', 'gre', 'en</w>', 'no', 'r', 'we', 'gi', 'an</w>', 'a', 'vo',
'ca', 'do</w>', 'wa', 's</w>', 'ea', 'ten', '</w>', 'b', 'y</w>', 'the</w>', 'wh', 'al', 'e</w>',
'!</w>']'

Token IDs: [9, 469, -1, 85, 4, 230, 42, 95, -1, 191, 367, 57, -1, 410, 112, 362, 153, 2, 405,
288, -1, -1, 10, 9, 34, 20, 0, 91]
```

2. Train a BPE tokenizer using k=3000. What is the output of: python myTigerID_cshw1_bpe.py tokenize --text The bright green Norwegian avocado was eaten by the whale! --load PathToBPEModel.p

```
Tokens: ['the</w>', 'bri', 'gh', 't</w>', 'gre', 'en</w>', 'nor', 'we', 'gi', 'an</w>', 'a', 'voc', 'ad',
'o</w>', 'wa', 's</w>', 'ea', 'ten</w>', 'b', 'y</w>', 'the</w>', 'wh', 'al', 'e</w>', '!</w>']

Token IDs: [9, 942, 85, 4, 230, 42, 1716, 191, 367, 57, -1, 2714, 194, 14, 153, 2, 405,
675, -1, 10, 9, 34, 20, 0, 91]
```

3. Train a BPE tokenizer using k=10. What is the output of: python myTigerID_cshw1_bpe.py tokenize --text The bright green Norwegian avocado was eaten by the whale! --load PathToBPEModel.p

```
Tokens: ['th', 'e</w>', 'b', 'r', 'i', 'g', 'h', 't</w>', 'g', 'r', 'e', 'e', 'n', '</w>', 'n', 'o', 'r', 'w', 'e', 'g',
'i', 'an', '</w>', 'a', 'v', 'o', 'c', 'a', 'd', 'o', '</w>', 'w', 'a', 's</w>', 'e', 'a', 't', 'e', 'n', '</w>', 'b',
'y', '</w>', 'th', 'e</w>', 'w', 'h', 'a', 'l', 'e</w>', '!', '</w>']

Token IDs: [1, 0, -1, -1, -1, -1, -1, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 8, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 0, -1, -1, -1, -1, 0, -1, -1]
```

// AI assisted

Section 3 Questions: Tokenization in Real-World Scenarios

Google SentencePiece:

1. What are the classes required to import a trainable BPE tokenizer? What would the import statement look like?
   The classes required to import a trainable BPE tokenizer is SentencePieceProcessor. SentencePiece provides a python wrapper around a C++ implementation and training is done using SentencePiece trainer. The import will be: import sentencepiece as spm
2. Assume we have a string_corpus object. How would we train a bpe_model object?
   spm_train --input=corpus.txt --model_type=bpe --vocab_size=8000
3. Assume the same string_corpus object. How to perform inference using our bpe_model?
   sp = spm.SentencePieceProcessor(model_file="bpe.model")

```
sp.encode("some text")
```

4. Does this library differ from our implementation in Section 2 in terms of arguments for how BPE trains? If so, how?

    Yes it does differ from our implementation in section 2 because SentencePiece uses a fixed vocabulary size while we use a fixed number of merges k. Sentence piece also trains from raw text.

5. What backend(s), e.g., programming languages besides Python does this library utilize?

    C++ is the backend programming language

HuggingFace Tokenizers:

1. What are the classes required to import a trainable BPE tokenizer? What would the import statement look like?

    ```
    from tokenizers import Tokenizer
    from tokenizers.models import BPE
    from tokenizers.trainers import BpeTrainer
    from tokenizers.pre_tokenizers import Whitespace
    ```

2. Assume we have a string_corpus object. How would we train a bpe_model object?

    ```
    tokenizer = Tokenizer(BPE())
    tokenizer.pre_tokenizer = Whitespace()
    trainer = BpeTrainer(special_tokens=["[UNK]"])
    tokenizer.train(files=["string_corpus.txt"], trainer=trainer)
    ```

3. Assume the same string_corpus object. How to perform inference using our bpe_model?

    ```
    output = tokenizer.encode("How to perform inference using our bpe_model")
    tokens = output.tokens
    token_ids = output.ids
    ```

4. Does this library differ from our implementation in Section 2 in terms of arguments for how BPE trains? If so, how?

    Yes, HuggingFace Tokenizers differ from our implementation in section 2 for training BPE. HuggingFace Tokenizers uses a fixed vocabulary size to control training while we use k merge.

5. What backend(s), e.g., programming languages besides Python does this library utilize?

    They use Rust, Node.js, and Ruby

// End of AI assisted