CMP711 Natural Language Processing (Aselsan Akademi)

HW01 – Basic Text Processing & Language Models
Due Date: November 3, 2025

You are required to write a Python program that implements a variation of the **Ngram** (**Bigram**) Language Model. Your program should contain a method named **trainFromFile(fn)** which should learn the Ngram language model (both bigram and unigram) from the training data provided in a given input text file whose file name is **fn**. Your python program may represent the learned Ngram language model an instance of class. When the top-level method **trainFromFile(fn)** is invoked, your Python program should learn a Ngram language model from the given training file **fn** by performing the following tasks:

- Your program should read the given training file and create a list of tokenized (and lowercased) sentences.
    - The training file is encoded in UTF-8 and may contain both Turkish and English text.
    - A token in the training file is a word containing only Turkish-English letters (lowercase or uppercase letters) or an end-of-sentence token.
    - You should assume that each sentence ends with one of the following end-of-sentence tokens: ".", "?", or "!". Each tokenized sentence should be padded with a special sentence-start token "<s>" and a special sentence-end token "</s>". This means that the first token of each tokenized sentence should be "<s>", and the last token should be "</s>". All words in a tokenized sentence should be lowercased. Before converting other characters to lowercase, explicitly replace the Turkish uppercase letters "I" and "İ" with "ı" and "i", respectively. Each token in a tokenized sentence can be a lower-cased word (containing only lowercased Turkish and English letters), an end-of-sentence token, <s> or </s>.
- While a list of tokenized sentences is created, the following information about the training file and the created Ngram language model should be also collected.
    - The tokenized sentence list is a list of sentences and each sentence is a list of tokens. Each token in a tokenized sentence can be a lower-cased word (containing only lowercased Turkish and English letters), an end-of-sentence token, <s> or </s>. The first token of a tokenized sentence should be the special sentence-start token "<s>", and the last token should be the special sentence-end token "</s>".
    - The total number of tokens in the train file.
    - The size of the vocabulary, i.e., the number of unique tokens in the training data.
    - The total number of sentences in the train file.
    - The vocabulary list (unigrams) is the list of unique tokens in the list of tokenized sentences (including special tokens <s> and </s>). Each item in this vocabulary list (the list of unigrams) must be a tuple **(word,frequency)** where **frequency** is the count of occurrences of the word (unigram).
    - The list of bigrams is the list bigrams appearing in the sentences of the tokenized sentence list. Each item in the list must be a tuple **((word1, word2),frequency)** where **frequency** is the count of occurrences of the bigram **(word1,word2)**.
- Then, your program should print the following information into an output text file named as ***fn*_Result.txt** where ***fn*** is the name of the training file.

  - The number of sentences in the tokenized sentence list.
  - The total number of tokens in the tokenized sentence list (Corpus Size).
  - The number of unique words in the tokenized sentence list (Vocabulary Size).
  - The tokenized sentence list (all tokenized sentences).
  - Unigrams: Your program should print all sorted unigrams (sorted wrt their frequencies in descending order) together with their frequencies.
  - Bigrams: Your program should print all sorted bigrams (sorted wrt their frequencies in descending order) together with their frequencies. You should only print bigrams with non-zero frequencies.
- After the printing the above values into the output file (**fn_Result.txt** where **fn** is the name of the training file), your program should print the smoothed bigram probability values together with the original (unsmoothed) bigram probability values.
  - The **unsmoothed probability** value of a bigram (w1,w2), (i.e., **P(w2|w1)**) can be found using the following formula.

    ```
    P((w1,w2)) = freq((w1,w2)) / freq(w1)
    ```

    when the both frequencies are non-zero values. Otherwise, it is equal to zero when at least one of them is zero. Remember that the unsmoothed probability value of a bigram is equal to 0 when at least one of the words is an unknown word.
  - The **smoothed probability** value of the **bigram** (where **bigram** is a bigram **(w1,w2)**) should be computed using add-1 smoothing method and unknown words also should be handled by add-1 smoothing method. Remember, the smoothed probability of a bigram can be computed as follows with this approach:

    ```
    P’((w1,w2)) = (freq((w1,w2))+1)/(freq(w1)+(sizeOfVoc+1))
    ```

    Since `sizeOfVocab` does not include unknown tokens, we add 1 to the size of the vocabulary in the denominator to account for unknown tokens. The smoothed bigram probability of a bigram where `w2` is an unknown word and `w1` is not an unknown word must be equal to `1/(freq(w1)+(sizeOfVocab+1))`, while the smoothed bigram probability of a bigram where `w1` is an unknown word must be equal to `1/(sizeOfVocab+1)`.
  - Your program should print unsmoothed and smoothed probability values of sorted bigrams (*sorted wrt their frequencies in descending order*) together with their frequencies. Remember that you should also include the probability values for the unknown token **<unk>**. For each bigram your program should print a line in the following form.

    Bigram Frequency UnsmoothedProbability SmoothedProbability

- In the last step, your program should compute the probabilities of at least two sentences using the smoothed bigram probability values. One of the sentences should contain at least one unknown word. If a sentence contains a word which is not in the training text, your program should use the probability values of the unknown token **<unk>** for that word. Then it should print those sentences into **fn_Result.txt** where **fn** is the name of the training file together with their computed probabilities.

You should test your program with at least given two sample files (`hw01_tiny.txt`, `hw01_tinytr.txt`). You will submit the produced result files for each file together with your program file (`hw01_`*`yourname`*`.py` – Make sure that this file contains only your python program). You should create a result file for each test file (such as `hw01_tinytr_Result.txt` and `hw01_tiny_Result.txt`) to submit together with your program file. **The content of each result file should be as follows:**

Number of Sentences in File: ....

Number of Total Tokens (Corpus Size): ....

Number of Unique Words (Vocabulary Size): ....

Tokenized Sentences:

> *Sentence1*
>
> *Sentence2*
>
> ...

Unigrams Sorted wrt Frequencies (from Higher to Lower Frequencies):

Unigram1	*ItsFrequency*
Unigram2	*ItsFrequency*
...	

Bigrams Sorted wrt Frequencies (from Higher to Lower Frequencies):

Bigram1	*ItsFrequency*
Bigram2	*ItsFrequency*
...	

Bigram Probability Values Sorted wrt Frequencies (from Higher to Lower Frequencies):

Bigram1	*ItsFrequency*	*UnsmoothedProbability*	*SmoothedProbability*
Bigram2	*ItsFrequency*	*UnsmoothedProbability*	*SmoothedProbability*
...			

SampleSentence1	*ItsComputedProbability*
SampleSentence2	*ItsComputedProbability*
...	

Example:

- The provided training file **hw01_tiny.txt** contains the following lines:

```
a b c d. a b b c d. a c d. a b c f. e c f. e c d. e b b c d.
```

- The following result file (**hw01_tiny_Result.txt**) with this training file

```
Number of Sentences in File: 7
Number of Total Tokens (Corpus Size): 48
Number of Unique Words (Vocabulary Size): 9
Tokenized Sentences:
    ['<s>', 'a', 'b', 'c', 'd', '.', '</s>'],
    ['<s>', 'a', 'b', 'b', 'c', 'd', '.', '</s>'],
    ['<s>', 'a', 'c', 'd', '.', '</s>'],
    ['<s>', 'a', 'b', 'c', 'f', '.', '</s>'],
    ['<s>', 'e', 'c', 'f', '.', '</s>'],
    ['<s>', 'e', 'c', 'd', '.', '</s>'],
    ['<s>', 'e', 'b', 'b', 'c', 'd', '.', '</s>']

Unigrams Sorted wrt Frequencies (from Higher to Lower Frequencies):
     ('.', 7)
     ('</s>', 7)
     ('<s>', 7)
     ('c', 7)
     ('b', 6)
     ('d', 5)
     ('a', 4)
     ('e', 3)
     ('f', 2)

Bigrams Sorted wrt Frequencies (from Higher to Lower Frequencies):
     (('.', '</s>'), 7)
     (('c', 'd'), 5)
     (('d', '.'), 5)
     (('<s>', 'a'), 4)
     (('b', 'c'), 4)
     (('<s>', 'e'), 3)
     (('a', 'b'), 3)
     (('b', 'b'), 2)
     (('c', 'f'), 2)
     (('e', 'c'), 2)
     (('f', '.'), 2)
     (('a', 'c'), 1)
     (('e', 'b'), 1)
```

```
Bigram Probability Values Sorted wrt Frequencies (from Higher to Lower
Frequencies):
     (('.', '</s>'), 7)       1.0       0.470588
     (('c', 'd'), 5)          0.714285  0.352941
     …
     (('e', 'b'), 1)          0.333333  0.153846
     …
     (('.', '<unk>'), 0)      0         0.058823
     …
     (('<unk>', '<unk>'), 0)  0         0.1


(['<s>','a','f','d','.','</s>'])      ItsComputedProbability
(['<s>','a','g','d','.','</s>'])      ItsComputedProbability
```

Hand in:

- You will submit your homework by sending an email to me (**ilyas@cs.hacettepe.edu.tr**). Make sure that the subject of email is “CMP711 HW01”. You have to send a single zip file (`*yourname*.zip`, `*yourname*.gzip` or `*yourname.rar*`) holding three files (`hw01_*yourname*.py`, `hw01_tiny_Result.txt`, `hw01_tinytr_Result.txt`) with your email.