```
import zipfile

zip_path = "/content/archive (6).zip"   # Updated zip file path
extract_path = "/content/wikitext"

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print("Dataset unzipped successfully!")
```

```
Dataset unzipped successfully!
```

```
import os

os.listdir("/content/wikitext")
```

```
['wikitext-2']
```

```
import re
import math
import nltk
from collections import Counter
```

```
with open("/content/wikitext/wikitext-2/wiki.train.tokens", "r", encoding="utf-8") as f:
    train_text = f.read()

with open("/content/wikitext/wikitext-2/wiki.test.tokens", "r", encoding="utf-8") as f:
    test_text = f.read()

print(train_text[:500])
```

```
 = Valkyria Chronicles III =

 Senjō no Valkyria 3 : <unk> Chronicles ( Japanese : 戦場のヴァルキュリア3 , lit . Valkyria of the Battlefield 3 ) , commonly
```

The dataset used is the WikiText-2 dataset obtained from Kaggle. It consists of cleaned Wikipedia articles designed for language modeling tasks. The dataset is provided as plain text files containing natural language sentences. The training file is used to build N-gram models, while the test file is used for evaluation. The dataset contains more than 1500 words, making it suitable for this lab.

```
def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^a-z\s]', '', text)
    tokens = text.split()
    return tokens

train_tokens = preprocess(train_text)
test_tokens = preprocess(test_text)

print(train_tokens[:20])
```

```
['valkyria', 'chronicles', 'iii', 'senj', 'no', 'valkyria', 'unk', 'chronicles', 'japanese', 'lit', 'valkyria', 'of', 'the',
```

```
unigram_counts = Counter(train_tokens)
total_words = sum(unigram_counts.values())
vocab_size = len(unigram_counts)
```

```
bigrams = list(zip(train_tokens[:-1], train_tokens[1:]))
bigram_counts = Counter(bigrams)
```

```
trigrams = list(zip(train_tokens[:-2], train_tokens[1:-1], train_tokens[2:]))
trigram_counts = Counter(trigrams)
```

```
def unigram_prob(word):
    return (unigram_counts[word] + 1) / (total_words + vocab_size)

def bigram_prob(w1, w2):
    return (bigram_counts[(w1, w2)] + 1) / (unigram_counts[w1] + vocab_size)
```

```
def trigram_prob(w1, w2, w3):
    return (trigram_counts[(w1, w2, w3)] + 1) / (bigram_counts[(w1, w2)] + vocab_size)
```

```
sentences = [
    "the government passed a new law",
    "this is a language modeling task",
    "machine learning models are powerful",
    "the article discusses economic growth",
    "this sentence is unseen completely"
]
```

```
def sentence_probability(sentence, model="unigram"):
    words = sentence.lower().split()
    prob = 1

    if model == "unigram":
        for w in words:
            prob *= unigram_prob(w)

    elif model == "bigram":
        for i in range(len(words)-1):
            prob *= bigram_prob(words[i], words[i+1])

    elif model == "trigram":
        for i in range(len(words)-2):
            prob *= trigram_prob(words[i], words[i+1], words[i+2])

    return prob
```

```
def perplexity(sentence, model="unigram"):
    words = sentence.lower().split()
    N = len(words)
    prob = sentence_probability(sentence, model)
    return pow(1/prob, 1/N)
```

```
for s in sentences:
    print("\nSentence:", s)
    print("Unigram Perplexity:", perplexity(s, "unigram"))
    print("Bigram Perplexity:", perplexity(s, "bigram"))
    print("Trigram Perplexity:", perplexity(s, "trigram"))
```

```
Sentence: the government passed a new law
Unigram Perplexity: 570.3494623125696
Bigram Perplexity: 468.6392751636427
Trigram Perplexity: 643.5481395885191

Sentence: this is a language modeling task
Unigram Perplexity: 1863.491255204366
Bigram Perplexity: 553.2022903152201
Trigram Perplexity: 506.01173420300324

Sentence: machine learning models are powerful
Unigram Perplexity: 7274.517343730462
Bigram Perplexity: 2774.1701871321065
Trigram Perplexity: 458.6295570734411

Sentence: the article discusses economic growth
Unigram Perplexity: 5110.8919053109485
Bigram Perplexity: 2140.412985838345
Trigram Perplexity: 458.6631809416231

Sentence: this sentence is unseen completely
Unigram Perplexity: 4173.233385232522
Bigram Perplexity: 2385.3570726588887
Trigram Perplexity: 458.64300854362637
```