```
import re

def process_text(file_path):
    """
    Reads a text file, cleans the text by removing special characters,
    converting to lowercase, and adds sentence start and end markers.

    Args:
        file_path (str): The path to the input text file.

    Returns:
        str: The processed text.
    """
    with open(file_path, 'r', encoding='utf-8') as f:
        text = f.read()

    # Remove special characters and convert to lowercase, and remove leading numbers
    cleaned_text = re.sub(r'^\d+\s*', '', text, flags=re.MULTILINE) # Remove leading numbers
    cleaned_text = re.sub(r'[^\w\s.]', '', cleaned_text).lower()


    # Add sentence start and end markers
    sentences = cleaned_text.split('.')
    processed_sentences = [f"<s> {sentence.strip()} </s>" for sentence in sentences if sentence.strip()]
    processed_text = " ".join(processed_sentences)

    return processed_text

# Specify the path to your text file
file_path = '/content/lab_2_truyen_kieu.txt'

# Process the text
processed_content = process_text(file_path)

# Print the processed text (or you can save it to a new file)
print(processed_content)
```

```
quanh tường ra ý tìm tòi ngẩn ngơ </s> <s> sinh đà có ý đợi chờ
```

```
cách tường lên tiếng xa đưa ướm lòng </s> <s> thoa này bắt được hư không
biết đâu hợp phố mà mong châu về
tiếng kiều nghe lọt bên kia
ơn lòng quân tử sá gì của rơi
chiếc thoa nào của mấy mươi </s> <s> mà lòng trọng nghĩa khinh tài xiết bao
sinh rằng lân lý ra vào
gần đây nào phải người nào xa xôi
rày nhờ được chút thơm rơi
kể đà thiếu não lòng người bấy nay </s> <s> bấy lâu mới được một ngày
dừng chân gạn chút niềm tây gọi là
vội về thêm lấy của nhà
khuyến vàng đôi chiếc khăn là một vuông
thang mây qua bước ngọn tường </s> <s> phải người hôm nọ rõ ràng chẳng như </s> <s> sượng sùng giữ ý rụt rè
kẻ nhìn rõ mặt người e cúi đầu
nẻng từ ngẫu nhỉ gặp nhau
```

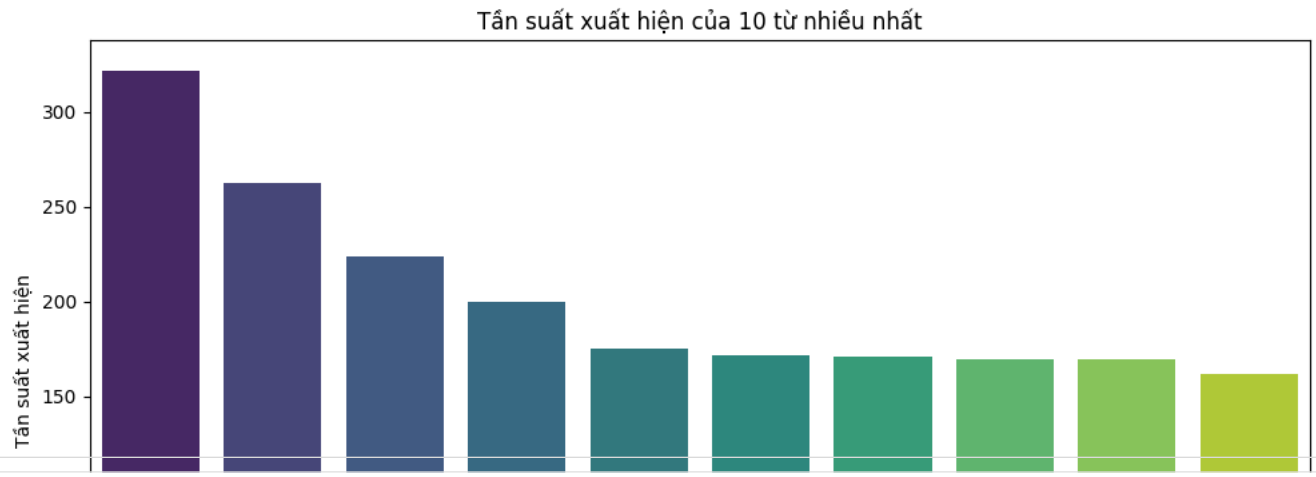
Bắt đầu lập trình hoặc tạo mã bằng trí tuệ nhân tạo (AI).

```
from collections import Counter
words = processed_content.split()

words = [word for word in words if word not in ['<s>', '</s>']]

word_counts = Counter(words)

vocabulary = list(word_counts.keys())

print(f"Kích thước tập từ vựng: {len(vocabulary)}")
print("\n10 từ xuất hiện nhiều nhất:")
for word, count in word_counts.most_common(10):
    print(f"{word}: {count}")

print(f"\nTần suất của từ 'trăm': {word_counts['trăm']}")
```

```
Kích thước tập từ vựng: 2393

10 từ xuất hiện nhiều nhất:
một: 322
đã: 263
người: 224
nàng: 200
lòng: 175
lời: 172
cho: 171
là: 170
cũng: 170
có: 162

Tần suất của từ 'trăm': 32
```

```
import matplotlib.pyplot as plt
import seaborn as sns

top_10_words = word_counts.most_common(10)
words, counts = zip(*top_10_words)

plt.figure(figsize=(10, 6))
sns.barplot(x=list(words), y=list(counts), palette='viridis')
plt.xlabel("Từ")
plt.ylabel("Tần suất xuất hiện")
plt.title("Tần suất xuất hiện của 10 từ nhiều nhất")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-1360157378.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

  sns.barplot(x=list(words), y=list(counts), palette='viridis')
```

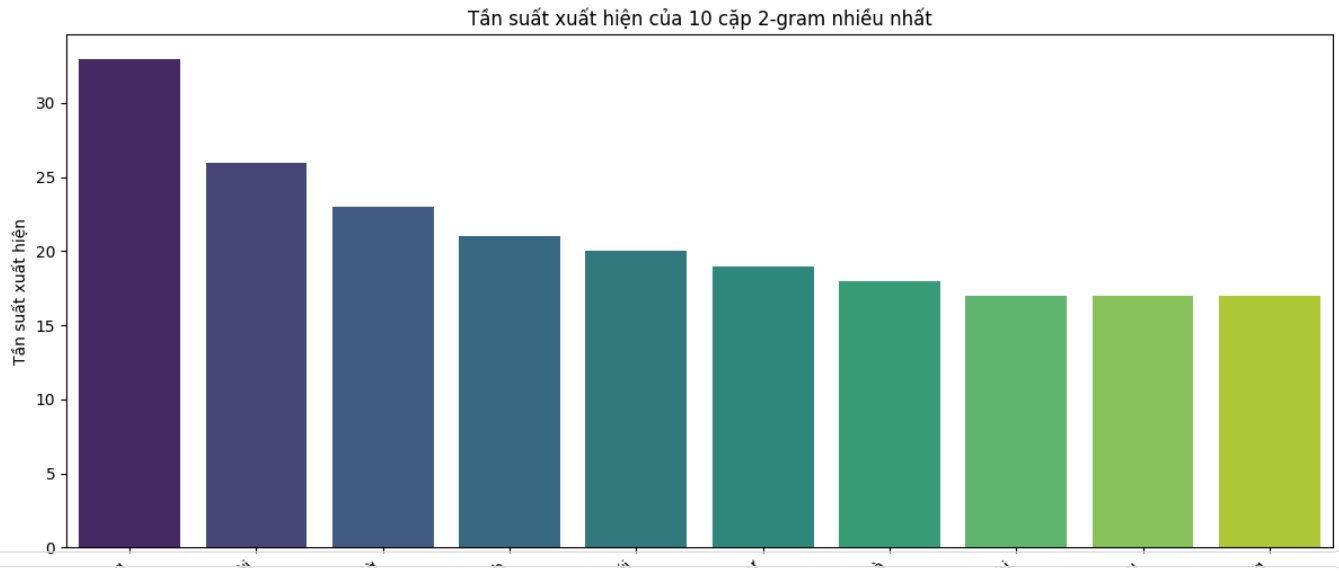



```
from collections import Counter

words_without_markers = [word for word in processed_content.split() if word not in ['<s>', '</s>']]

bigrams = [(words_without_markers[i], words_without_markers[i+1]) for i in range(len(words_without_markers)-1)]

bigram_counts = Counter(bigrams)

print("10 cặp 2-gram xuất hiện nhiều nhất:")
for bigram, count in bigram_counts.most_common(10):
    print(f"{bigram}: {count}")
```

```
10 cặp 2-gram xuất hiện nhiều nhất:
('nàng', 'rằng'): 33
('một', 'lời'): 26
('bây', 'giờ'): 23
('một', 'mình'): 21
('nàng', 'mới'): 20
('tiểu', 'thư'): 19
('một', 'nhà'): 18
('làm', 'chi'): 17
('bấy', 'lâu'): 17
('vội', 'vàng'): 17
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Get the top 10 most common bigrams and their counts
top_10_bigrams = bigram_counts.most_common(10)
bigrams, counts = zip(*top_10_bigrams)

# Format the bigrams for display on the plot
bigram_labels = [" ".join(bigram) for bigram in bigrams]

# Create a bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x=bigram_labels, y=list(counts), palette='viridis')
plt.xlabel("2-gram")
plt.ylabel("Tần suất xuất hiện")
plt.title("Tần suất xuất hiện của 10 cặp 2-gram nhiều nhất")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-4027048582.py:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

  sns.barplot(x=bigram_labels, y=list(counts), palette='viridis')
```




```
def calculate_conditional_probability(word1, word2, word_counts, bigram_counts):
    bigram = (word1, word2)
    bigram_count = bigram_counts.get(bigram, 0)
    word1_count = word_counts.get(word1, 0)

    if word1_count == 0:
        return 0
    else:
        return bigram_count / word1_count
word1 = 'nàng'
word2 = 'rằng'
conditional_prob = calculate_conditional_probability(word1, word2, word_counts, bigram_counts)

print(f"Xác suất có điều kiện p({word2} | {word1}): {conditional_prob:.4f}")

word1 = 'một'
word2 = 'lời'
conditional_prob = calculate_conditional_probability(word1, word2, word_counts, bigram_counts)

print(f"Xác suất có điều kiện p({word2} | {word1}): {conditional_prob:.4f}")
```

```
Xác suất có điều kiện p(rằng | nàng): 0.1650
Xác suất có điều kiện p(lời | một): 0.0807
```

```
ngram_model = {}
for word1, word2 in bigrams:
    if word1 not in ngram_model:
        ngram_model[word1] = {word2: 1}
    else:
        if word2 not in ngram_model[word1]:
            ngram_model[word1][word2] = 1
        else:
            ngram_model[word1][word2] += 1

for word1, next_words in ngram_model.items():
    total_count = sum(next_words.values())
    for word2 in next_words:
        ngram_model[word1][word2] /= total_count

print("2-gram language model (sample):")
for word, prob_dict in list(ngram_model.items())[:5]:
    print(f"'{word}': {prob_dict}")
```

```
2-gram language model (sample):
'nàng': {'rằng': 0.5, 'mới': 0.5}
'một': {'lời': 0.3333333333333333, 'mình': 0.3333333333333333, 'nhà': 0.3333333333333333}
'bây': {'giờ': 1.0}
'tiểu': {'thư': 1.0}
'làm': {'chi': 1.0}
```

```
sentences = processed_content.split(" </s>")

six_word_sentences = []
eight_word_sentences = []

for sentence in sentences:
    cleaned_sentence = sentence.replace("<s> ", "").strip()

    words = cleaned_sentence.split()

    if len(words) == 6:
        six_word_sentences.append(cleaned_sentence)
    elif len(words) == 8:
        eight_word_sentences.append(cleaned_sentence)

print(f"Số lượng câu 6 chữ: {len(six_word_sentences)}")
print(f"Số lượng câu 8 chữ: {len(eight_word_sentences)}")
```

```
Số lượng câu 6 chữ: 248
Số lượng câu 8 chữ: 246
```

```
import random
def generate_six_word_sentence():
    current_word = random.choice(vocabulary)
    sentence = [current_word]

    while len(sentence) < 6:
        next_word_probs = ngram_model.get(current_word, {})

        if not next_word_probs:
            next_word = random.choice(vocabulary)
        else:
            next_word = random.choices(list(next_word_probs.keys()), weights=list(next_word_probs.values()), k=1)[0]

        sentence.append(next_word)
        current_word = next_word

    return " ".join(sentence)

sample_six_word_sentence = generate_six_word_sentence()
print(f"Câu 6 chữ ngẫu nhiên: {sample_six_word_sentence}")
```

```
Câu 6 chữ ngẫu nhiên: hắt huấn thua ấm côn về
```

```
def generate_eight_word_sentence():
    current_word = random.choice(vocabulary)
    sentence = [current_word]

    while len(sentence) < 8:
        next_word_probs = ngram_model.get(current_word, {})

        if not next_word_probs:
            next_word = random.choice(vocabulary)
        else:
            next_word = random.choices(list(next_word_probs.keys()), weights=list(next_word_probs.values()), k=1)[0]

        sentence.append(next_word)
        current_word = next_word

    return " ".join(sentence)

sample_eight_word_sentence = generate_eight_word_sentence()
print(f"Câu 8 chữ ngẫu nhiên: {sample_eight_word_sentence}")
```

```
Câu 8 chữ ngẫu nhiên: trươc hòng sờ uốn sắp viễn chục ngấn
```

```
def generate_luc_bat_couplet():
    six_word_sentence = generate_six_word_sentence()
    eight_word_sentence = generate_eight_word_sentence()

    return f"{six_word_sentence}\n{eight_word_sentence}"
sample_couplet = generate_luc_bat_couplet()
print("Cặp lục bát ngẫu nhiên:")
print(sample_couplet)
```

```
Cặp lục bát ngẫu nhiên:
đèo dương sau tót giòn cờ
nhạt điếc nghi sắp quên vẽ giục tám
```

```
num_couplets = int(input("Nhập số lượng cặp câu lục bát cần sinh: "))

print("\nCác cặp lục bát ngẫu nhiên đã sinh:")
for i in range(num_couplets):
    couplet = generate_luc_bat_couplet()
    print(f"Cặp {i+1}:\n{couplet}\n")
```

```
Nhập số lượng cặp câu lục bát cần sinh: 2

Các cặp lục bát ngẫu nhiên đã sinh:
Cặp 1:
lưu đao ve kề tiếng chẩy
nham nực kém chép phù dàn đài xử

Cặp 2:
chẩy não khư làn mắn xem
dĩ ba khoảng đớn lạy thủ thong chước
```