I052

Reya Oberoi

NLP Assignment : N-gram modelling

```
import gradio as gr
import requests
import random
from collections import defaultdict
import nltk
from nltk.tokenize import word_tokenize
import re

nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

Dataet download, text pre-processing and Tokenization

```
url = "https://www.gutenberg.org/cache/epub/1342/pg1342.txt"
```

```
raw_text = requests.get(url).text.lower()

# Remove header and footer
start_marker = "*** start of the project gutenberg ebook"
end_marker = "end of the project gutenberg ebook"
raw_text = raw_text[raw_text.find(start_marker) + len(start_marker):raw_text.find(end_marker)]

# Tokenize and clean the text
tokens = word_tokenize(raw_text)
tokens = [token for token in tokens if re.match(r'^[a-z]+$', token)]
print("Data loaded and preprocessed.")
```

```
Data loaded and preprocessed.
```

N-GRAM MODELING

```
# Ngram modeling with laplace smoothing
def build_ngram_model(tokens, n, smoothing=1):
    model = defaultdict(lambda: defaultdict(lambda: smoothing))
    for i in range(len(tokens) - n):
        history = tuple(tokens[i:i + n])
        next_word = tokens[i + n]
        model[history][next_word] += 1

    for history in model:
        total_count = sum(model[history].values())
        for next_word in model[history]:
            model[history][next_word] /= total_count
    return model
```

Sentence Generation

```
# SENTENCE GENERATION WITH BACK-OFF
def generate_sentence_with_backoff(start_words, max_length=15):
    words = start_words.lower().split()

    # Pad history with None if start_words are fewer than 3
    if len(words) < 3:
        words = [None] * (3 - len(words)) + words

    for _ in range(max_length - len(words)):
        # Try 4-gram first
        history_4gram = tuple(words[-3:])
        if history_4gram in fourgram_model:
            next_word = random.choices(list(fourgram_model[history_4gram].keys()),
                                       list(fourgram_model[history_4gram].values()))[0]
        # Back off to Trigram
        elif len(words) >= 2:
            history_trigram = tuple(words[-2:])
            if history_trigram in trigram_model:
                next_word = random.choices(list(trigram_model[history_trigram].keys()),
                                           list(trigram_model[history_trigram].values()))[0]
```

```
            # Back off to Bigram
            else:
                history_bigram = tuple(words[-1:])
                next_word = random.choices(list(bigram_model[history_bigram].keys()),
                                           list(bigram_model[history_bigram].values()))[0]
        else: # Default to random word
            next_word = random.choice(tokens)

        words.append(next_word)

        # Stop at punctuation or word limit
        if next_word in ['.', '?', '!']:
            break

    # Remove leading None values before joining
    words = [w for w in words if w is not None]

    return ' '.join(words).capitalize() + '.'
```

```
# Build n-gram models
bigram_model = build_ngram_model(tokens, 1)
trigram_model = build_ngram_model(tokens, 2)
fourgram_model = build_ngram_model(tokens, 3)
print("N-gram models built successfully.")
```

```
N-gram models built successfully.
```

```
#Testing
sentence1 = generate_sentence_with_backoff("the man")
print(sentence1)

sentence2 = generate_sentence_with_backoff("her heart")
print(sentence2)
```

```
The man whom he so justly scorned from such a connection she could not.
Her heart to jane though suspicion was very far from dreading a rebuke either.
```

Gradio UI

```
# GRADIO UI
with gr.Blocks(
    theme=gr.themes.Soft(),
    css="""
    .gradio-container {
        background-color: #FFF5F2;
    }

    .gr-markdown h1 {
        color: #568F87 !important;
        font-family: Arial, sans-serif !important;
        font-weight: bold !important;
    }

    .gr-markdown p {
        color: #568F87 !important;
    }

    .gr-button {
        background-color: #F5BABB !important;
        color: #447D9B !important;
    }

    .gr-textbox input, .gr-textbox textarea {
        color: black !important;
    }

    .made-by-text {
        text-align: right;
        color: black;
    }
"""
) as demo:
    gr.Markdown("♡⋆｡˚ The Austen Engine📖")
    gr.Markdown("This language model generates sentences that mimic the writing style of Jane Austen using probabilistic n-gram models.")

    with gr.Row():
        start_words_input = gr.Textbox(
            lines=1,
            placeholder="e.g., The man",
            label="Starting Words"
```

```
        )

    generate_button = gr.Button("Generate Sentence")
    output_text = gr.Textbox(label="Generated Sentence")

    generate_button.click(
        fn=generate_sentence_with_backoff,
        inputs=start_words_input,
        outputs=output_text
    )

    gr.HTML("<p class='made-by-text'>Made by Reya Oberoi</p>")

demo.launch(share=True)
```

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://b7409a21f81fa39b5c.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working
```

♡‿✐ The Austen Engine

This language model generates sentences that mimic the writing style of Jane Austen using probabilistic n-gram models.

Starting Words

he wandered into the woods

Generate Sentence

Generated Sentence

He wandered into the woods to which they were the principal inhabitants they found bennet.

Made by Reya Oberoi

Use via API · Built with Gradio · Settings