Day64_NLP_2_NLG_Text_Generation_and_Visualization

August 14, 2025

Natural Language Generation (NLG) Basics & Visualization

Today, we continued our journey into Natural Language Processing (NLP) by shifting our focus from understanding language (NLU) to producing it — Natural Language Generation (NLG). We explored how machines can create meaningful, human-like text and visual representations from data. Our session covered basic and creative generation techniques, starting with visualizing word frequency using WordClouds, moving into template-based sentence generation, and experimenting with n-gram models for pattern-based text creation. We also discussed where NLG fits in real-world applications like chatbots, automated report writing, and story generation.

1 Introduction to NLG

Definition: Natural Language Generation is the process of creating human-like text from structured or unstructured data.

Applications:

- Chatbots & Virtual Assistants
- Automated Report Writing
- Summarization
- Story Generation

2 WordCloud – Visualizing Words

A **WordCloud** is a visual representation of text where the size of each word reflects its frequency or importance.

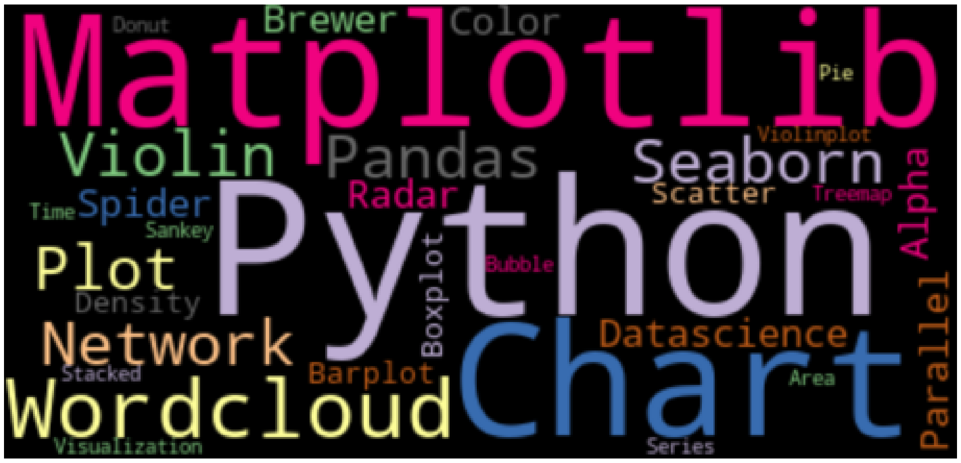
[1]:
```
# Import required libraries
import nltk
import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

[2]:
```
# Sample text
text = (
    "Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin␣
 ↪Chart "
    "Pandas Datascience Wordcloud Spider Radar Parallel Alpha Color Brewer␣
 ↪Density "
```

```
    "Scatter Barplot Boxplot Violinplot Treemap Stacked Area Chart␣
↪Visualization "
    "Donut Pie Time-Series Wordcloud Sankey Bubble"
)
```

[3]:
```
# Create WordCloud
wordcloud = WordCloud(
    width=420,
    height=200,
    margin=2,
    background_color='black',
    colormap='Accent',
    mode='RGBA'
).generate(text)
```

[4]:
```
# Plot WordCloud
plt.imshow(wordcloud, interpolation='quadric')
plt.axis('off')  # Remove axis
plt.margins(x=0, y=0)  # Remove margins
plt.show()
```



Explanation:

- **width & height:** Size of the WordCloud image.
- **background_color:** Sets the background color.
- **colormap:** Changes the color theme.
- **.generate(text):** Creates the WordCloud from the given text.

3 Generating Text with Templates

Template-based generation is one of the simplest NLG methods.

[5]:
```
templates = [
    "Today is a {} day.",
    "I feel {} about learning NLP.",
    "The weather is {} and perfect for coding."
]

words = ["beautiful", "exciting", "sunny"]

for t, w in zip(templates, words):
    print(t.format(w))
```

```
Today is a beautiful day.
I feel exciting about learning NLP.
The weather is sunny and perfect for coding.
```

4 N-gram Based Text Generation

You can use n-grams to generate sentences based on word frequency patterns.

[7]:
```
from nltk import word_tokenize, bigrams
import random

# Sample corpus
corpus = "Natural Language Processing is fun and Natural Language Generation␣
 ↪makes it even more fun."
corpus
```

[7]:
```
'Natural Language Processing is fun and Natural Language Generation makes it
even more fun.'
```

[8]:
```
tokens = word_tokenize(corpus)
bi_grams = list(bigrams(tokens))

current_word = random.choice(tokens)
result = [current_word]

for _ in range(10):
    candidates = [b[1] for b in bi_grams if b[0] == current_word]
    if candidates:
        current_word = random.choice(candidates)
        result.append(current_word)
    else:
        break

print(" ".join(result))
```

```
is fun .
```

5 Next Steps in NLG

- Use Markov Chains for probabilistic text generation.
- Use Hugging Face Transformers to generate human-like sentences with GPT-2 or GPT-3.
- Evaluate generated text with BLEU or ROUGE scores.