SentiGen: Sentiment-Driven AI Text Generator

Author: Krish Lakhani
GitHub: @Krish-Lakhani19

Email: krishlakhani46767@gmail.com

1. Introduction

SentiGen is an intelligent text generation system that produces sentiment-aligned content. It integrates **state-of-the-art NLP models** to analyze user-provided prompts, detect sentiment, and generate coherent text matching the detected sentiment. Its **Streamlit-based interface** provides a user-friendly platform for real-time experimentation and text generation.

The system is designed for developers, researchers, and content creators seeking **automated sentiment-aware text generation**.

2. Objectives

The primary objectives of SentiGen are:

- 1. **Automatic Sentiment Detection**: Identify whether a user input is positive, negative, or neutral using DistilBERT.
- 2. **Sentiment-Aware Text Generation**: Generate contextually relevant text aligned with the detected sentiment using GPT-2 Medium.
- 3. **Interactive User Interface**: Provide adjustable settings for text length, creativity, and sentiment selection.
- 4. **Export Functionality**: Allow users to download generated text as .txt files.

3. Methodology

SentiGen combines sentiment analysis with prompt-conditioned text generation:

3.1 Workflow

 $\textit{User Input} \rightarrow \textit{Sentiment Analyzer} \rightarrow \textit{Prompt Engineering} \rightarrow \textit{Text Generator} \rightarrow \textit{Output Display}$

- 1. **Input Processing**: User submits a text prompt.
- 2. **Sentiment Analysis**: DistilBERT classifies the input as positive, negative, or neutral, with a confidence score.
- 3. **Prompt Engineering**: The system applies **sentiment-specific templates** to guide text generation.
- 4. **Text Generation**: GPT-2 Medium generates coherent text conditioned by the sentiment template.
- 5. **Output Display**: Streamlit displays the generated text along with metrics: word count, character count, and sentiment classification.

3.2 Sentiment Conditioning

Sentiment	Prompt Template
Positive	"This is wonderful because"
Negative	"This is disappointing because"
Neutral	"From an objective perspective"

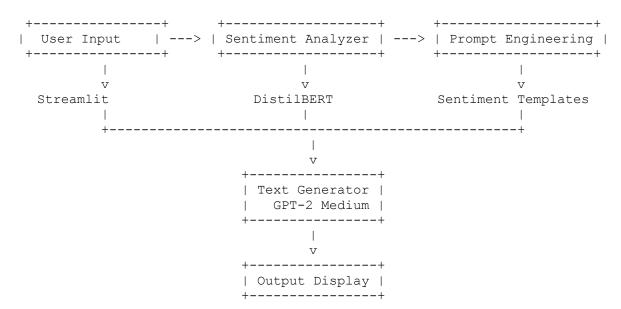
Example:

- Input: "summer vacation at the beach"
- Detected Sentiment: Positive

• Output: "This is wonderful because summer vacation at the beach brings endless joy and relaxation..."

4. Architecture Diagram

Here's the system architecture of SentiGen:



5. Implementation

Technology Stack:

Component Technology

Sentiment Analysis DistilBERT (distilbert-base-uncased-finetuned-sst-2-english)

Text Generation GPT-2 Medium

ML Framework PyTorch + Hugging Face Transformers

Frontend Streamlit Language Python 3.10

Setup Steps:

```
Step 1: Clone the repository
git clone https://github.com/yourusername/sentigen.git
cd sentigen

Step 2: Create and activate virtual environment
conda create -n sentigen python=3.10 -y
conda activate sentigen
OR using venv
python3 -m venv venv
source venv/bin/activate # Mac/Linux

Step 3: Install dependencies
conda install pytorch torchvision -c pytorch -y
pip install -r requirements.txt

Step 4: Run the application
streamlit run app.py
```

6. Results & Discussion

SentiGen produces coherent, sentiment-aligned outputs:

Input	Detected	Generated Text (Excerpt)
	Sentiment	
"summer vacation at the beach"	Positive	"This is wonderful because summer vacation at the beach brings endless joy and relaxation"
"dealing with Monday morning traffic"	Negative	"This is disappointing because dealing with Monday morning traffic drains energy and patience"

Performance Metrics:

Model	Task	Accuracy	Speed
DistilBERT	Sentiment Analysis	~95%	<1s
GPT-2 Medium	Text Generation	N/A	10–20s

Challenges & Solutions:

Challenge	Solution
Sentiment-text alignment	Sentiment-specific prompt templates
Generation coherence	Adjusted temperature and top-p sampling parameters
Slow model loading	Streamlit caching (@st.cache_resource)
Neutral detection	Introduced confidence threshold (<0.6)

7. Conclusion

SentiGen successfully integrates sentiment analysis and text generation into a **cohesive**, **interactive system**. It demonstrates the value of **prompt engineering** in controlling generative outputs and provides a **user-friendly interface** for experimentation. Future enhancements include:

- Multi-language support
- Emotion detection beyond sentiment
- Long-form content generation (articles, stories)
- Personalized user history and authentication

8. References

- 1. Hugging Face Transformers Library: https://huggingface.co/transformers
- 2. PyTorch Documentation: https://pytorch.org/docs/stable/index.html
- 3. Streamlit Documentation: https://docs.streamlit.io