Comparing Neural Approaches for Short Story Generation

Final Project Proposal

Team: Abhilasha Singh, Pranjal Wakpaijan, Shabnam Rafat Ummey Sawda
DATS-6312: Natural Language Processing for Data Science
Professor: Amir Jafari

Problem Selection and Motivation:

The automatic generation of coherent and engaging short stories remains a significant challenge in natural language processing, requiring models to understand narrative structure, maintain character consistency, and produce contextually appropriate continuations. This project aims to explore and compare different neural approaches for generating story continuations from given story beginnings.

Working with a dataset of synthetically generated children's stories characterized by simple vocabulary and clear narrative structures, This project will compare different neural approaches for generating story continuations from given beginnings. Using a dataset of simple, child-friendly stories, the study will evaluate three types of models:

1. **Transformer-based models** that learn language patterns from data.
2. **Memory-augmented transformers** that can remember information across longer story contexts.
3. **Structured models** that explicitly learn story patterns using graph and latent representations.

By systematically comparing these approaches across evaluation metrics including coherence, creativity, and vocabulary adherence, this project seeks to identify which strategies are most effective for narrative generation. The findings will advance understanding of automated story generation and inform future research in creative text generation systems.

Dataset:

We will use the TinyStories dataset, a collection of synthetically generated short stories created specifically for training and evaluating language models on simple, coherent narratives.

The dataset contains approximately 2.12 million stories, each averaging 5,500 characters (around 200–300 words). This large-scale corpus provides a strong foundation for training neural story generation models by offering:

- Ample data for multiple training splits (80% training, 10% validation, 10% testing)
- Diverse narrative patterns across a variety of themes and character interactions
- Support for transfer learning, allowing the use of pretrained transformer models such as GPT-2 or BERT as backbones

The stories are written with a limited vocabulary suitable for 3–4-year-old children, which simplifies language complexity while preserving challenges related to narrative coherence, character consistency, and plot flow.

Dataset URL:**https://huggingface.co/datasets/roneneldan/TinyStories**

NLP Model:

We will use following approaches:

1. **Transformer-based Autoregressive Model (e.g., GPT-2 / GPT-Neo fine-tuning):**
   This approach utilizes a pretrained autoregressive transformer model such as GPT-2 or GPT-Neo, fine-tuned on the TinyStories dataset to learn narrative continuation patterns. The model generates each token conditioned on preceding context, enabling fluent and coherent story generation within short narrative spans. By leveraging transfer learning from large-scale language modeling, this approach provides a strong baseline for narrative fluency and local coherence without explicit structural control.

2. **Memory-Augmented Transformer (e.g. Transformer-XL or Recurrent Memory Transformer):** This approach employs a Transformer-XL architecture that extends the self-attention mechanism with a recurrent memory module, allowing the model to retain contextual information across longer story segments. The memory-based design facilitates consistent character references and plot continuity over extended narratives. This architecture enhances long-term coherence and narrative consistency, addressing limitations of standard transformers in modeling dependencies beyond fixed context windows.

3. **VAE + GNN with Hierarchical Transformer Decoder:**
   This approach combines a Graph Neural Network to encode story structure into latent graph representations, a Variational Autoencoder to learn the distribution of narrative patterns, and a hierarchical transformer decoder that generates text from the learned latent codes, enabling controllable story continuation with explicit structural modeling.

Packages:

The following Python libraries will be used to build, train, and evaluate the models:
PyTorch, PyTorch Geometric, Transformers, spaCy, NLTK, sentence-transformers, NetworkX, DGL (Deep Graph Library), pandas, NumPy, evaluate, BERTScore, sacreBLEU, matplotlib, seaborn, TensorBoard, tqdm, and Weights & Biases (wandb).

Performance Metrics:

We will evaluate our approaches using the following metrics:

- **BLEU (Bilingual Evaluation Understudy):** Measures n-gram overlap between generated and reference continuations.
- **Perplexity:** Measures how well the model predicts the text, with lower values indicating better language modeling.
- **BERT Score:** Evaluates semantic similarity between generated and reference texts using contextual embeddings.
- **Human Evaluation:** Subjective ratings on narrative coherence, creativity, character consistency, and overall quality.

Project Schedule:

- **Week 1:** Conduct data exploration, perform preprocessing, and establish the baseline model setup.
- **Week 2:** Fine-tune Transformer models, experimenting with both memory-enabled and standard configurations.
- **Week 3:** Implement and train integrated architecture comprising VAE, GNN, and Hierarchical Transformer components.
- **Week 4:** Develop a Streamlit-based demonstration and update associated Git repository files.
- **Week 5:** Perform result analysis and finalize the project report.