Exploring Neural Approaches for Tiny 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:

Automated story generation is a challenging task requiring models to maintain narrative coherence, vocabulary alignment, and contextual consistency. This project investigates how different neural architectures perform in the task of generating short children's stories based on a given prompt and a set of required keywords.

Using a simplified children's story dataset (TinyStories), we plan to compare four distinct modeling paradigms:

1. **Autoregressive Transformer (GPT-2 Baseline)**
   Evaluate a decoder-only transformer's ability to generate fluent stories conditioned on prompts and keywords.
2. **LSTM–Attention Seq2Seq Model**
   Tests whether classical recurrent models with attention can better enforce structural alignment and keyword adherence.
3. **Flan-T5 (Encoder–Decoder Transformer)**
   Investigate instruction-tuned architectures for controlled story continuation using cross-attention.
4. **Graph Neural Network + Small Language Model (GNN-SLM)**
   Explore whether graph-based narrative representations improve structural consistency and semantic control.
5. **Standalone Small Language Model (SLM):**
   Evaluates a lightweight transformer's ability to generate coherent, keyword-aligned stories with minimal computational overhead.

By comparing these varied approaches under identical training and evaluation settings, this study aims to identify the strengths and limitations of each model and determine which architecture best supports coherent, keyword-aligned story generation.

Dataset:

We will use the TinyStories dataset from Hugging Face, 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. **Baseline Model: GPT-2 Causal Language Model**
   The baseline model is a decoder-only transformer trained to predict the next token based on preceding context, enabling fluent and contextually coherent story continuation. By fine-tuning GPT-2 on the TinyStories dataset, we establish a foundational benchmark for narrative quality, fluency, and semantic consistency. This model serves as a reference point against which the performance of more structured or enhanced architecture can be compared.

2. **LSTM–Attention Seq2Seq Model**
   This model employs a classical encoder–decoder architecture, where an LSTM encoder processes the input prompt and Bahdanau attention enables the decoder to selectively focus on relevant encoder states during generation. This approach allows for stronger alignment between input prompts and generated text, potentially improving keyword incorporation and structural coherence. This model provides a contrast to transformer-based architectures by evaluating the effectiveness of recurrent networks with attention for narrative generation.

3. **Flan-T5 (Encoder–Decoder Transformer)**
   Flan-T5 is an instruction-tuned encoder–decoder transformer designed to follow natural language prompts with high accuracy. By encoding the input prompt and generating continuations through cross-attention mechanisms, the model produces structured and semantically aligned story outputs. Its strong generalization and instruction-following capabilities make it well-suited for controlled narrative generation and comparison against both autoregressive and recurrent architectures.

4. **GNN + SLM (Graph-Enhanced Story Modeling)**
   This approach integrates a Graph Neural Network to encode relational and structural aspects of the narrative into expressive graph-based embeddings. These representations are then passed to a Small Language Model (SLM) that decodes the encoded structure into coherent story continuations. By explicitly modeling narrative relationships, this hybrid architecture aims to enhance structural consistency, character linkage, and controlled generation beyond what traditional sequence models provide.

5. **Standalone Small Language Model (SLM)**
   This model serves as a lightweight transformer-based language model designed to operate with significantly fewer parameters than GPT-2, enabling efficient training and rapid experimentation on limited resources. The SLM will be fine-tuned directly on the TinyStories dataset and conditioned on prompts and required keywords. By isolating the SLM without graph-based enhancements, this model provides a clean baseline for evaluating how smaller architectures perform in terms of fluency, coherence, and keyword incorporation. Comparing the SLM to larger transformers and hybrid GNN-SLM architectures will help determine whether compact models can effectively generate narrative text while maintaining semantic control and structural consistency.

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.
- **Keyword-Based Metrics:** Evaluates how accurately models incorporate required keywords (strict accuracy and average coverage).

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.