Predicting Missing Story Endings with Generative and Classification Models

Matthew Frank

University of California, Berkeley

Abstract

Predicting the missing final sentence of a story is a challenging task. It requires understanding the narrative flow, reasoning, and external knowledge about the world. For this project, I used the ROCStories dataset and two generative models: Mistral-7B-Instruct and Llama-3.1-8B-Instruct. Both models generated the fifth sentence based on the first four sentences of each story. To evaluate their performance, I built a BERT-based classifier to predict which model's sentence better aligns with the original. Using cosine similarity, Mistral slightly outperformed Llama (0.4378 vs. 0.4284). The classifier achieved an accuracy of 63%. I also experimented with adding a dense layer to the BERT architecture but didn't see meaningful performance improvements. This project demonstrates how combining generative and classification models can help with narrative prediction tasks. It also opens up opportunities to further improve story completion systems.

Introduction

Motivation for the Work

Completing the final sentence of a story is a tough problem. It's more than just generating text—it requires understanding how a story flows, reasoning about the events, and knowing what makes sense in the real world. To predict a coherent and contextually appropriate ending, a model has to pick up on both the obvious and subtle patterns in the earlier parts of the story. This makes story completion a great test for evaluating how well generative models can handle complex tasks.

Relevance

Generative storytelling models are not just academic exercises—they have real-world applications. Platforms that thrive on user engagement could use these models to create interactive and engaging short stories. Think of applications like gaming, marketing, or even educational tools where AI helps generate content. This project focuses on story completion as a way to push the boundaries of what these systems can do.

Overview of the Work

For this project, I used the ROCStories dataset, which is a benchmark for understanding and predicting the flow of stories. I worked with two pre-trained generative models, Mistral-7B-Instruct and Llama-3.1-8B-Instruct, to generate the fifth sentence of a story given the first four. To evaluate these generated sentences, I built a BERT-based classifier that predicts which generated sentence better matches the original. Combining generative and classification models provided a structured approach to tackle this narrative prediction challenge.

Background

The ROCStories Dataset

The ROCStories dataset is a key resource for evaluating how well models understand and predict narratives. It has over 98,000 stories, with each story structured as five sentences. The goal is simple: test if models can predict the last sentence of a story based on the first four. These stories are designed to reflect realistic, everyday situations, which makes them a good test for commonsense reasoning. Tasks like the Story Cloze Test use this dataset to push models to go beyond just understanding syntax and infer deeper relationships and motivations.

Previous Work on the Story Cloze Test

The Story Cloze Test has been a popular way to benchmark models using the ROCStories dataset. Early methods were straightforward, using embeddings in vector space to guess the best ending. As NLP evolved, pre-trained models like BERT and GPT started showing major improvements. Recent transformer-based models have taken things even further, generating story endings that are both coherent and contextually appropriate.

Advances in Generative Modeling

Models like GPT, BERT, and now Mistral and Llama have pushed generative text tasks to new levels.

These models are great at generating fluent and coherent text, but they often hit a wall when deeper reasoning or maintaining narrative flow is required. Research into techniques like multi-task pre-training (e.g., MVP: Multi-task Supervised Pre-training for Natural Language Generation) and few-shot learning with multilingual models has shown how to push these models further. That said, integrating these capabilities into rigorous evaluation frameworks remains a challenge.

Gaps and Limitations in Prior Approaches

Even with these advancements, there's room for improvement. Most studies stop at evaluating the quality of the generated sentences without going deeper into how well they align with the original narrative. Metrics like cosine similarity are useful but don't capture everything about narrative coherence. By combining generative outputs with a classification model, this project offers a new way to bridge that gap and improve both evaluation and performance.

Methods

Dataset

For this project, I used the ROCStories dataset, which contains about 100,000 five-sentence stories. Each story has a clear structure: the first four sentences set up the narrative, and the fifth sentence serves as the conclusion. I focused on the first 40,000 examples to keep things computationally manageable while still capturing a wide variety of story patterns.

The dataset was in good shape and required minimal preprocessing. Sentences were already tokenized and cleaned, so I could jump straight into model integration. An important note is that the labels (whether a generated sentence from Mistral or Llama better matches the original) are approximately balanced, as shown in Figure 1. This ensures that our evaluation isn't skewed by class imbalance.

Modeling

Generative Models I worked with two pre-trained generative models: Mistral-7B-Instruct and Llama-3.1-8B-Instruct. These were chosen because they're known for their size and strong performance on narrative tasks. Here's how the inputs and outputs were structured:

- **Input:** The first four sentences of each story concatenated into a single input.
- **Output:** A generated candidate for the missing fifth sentence.

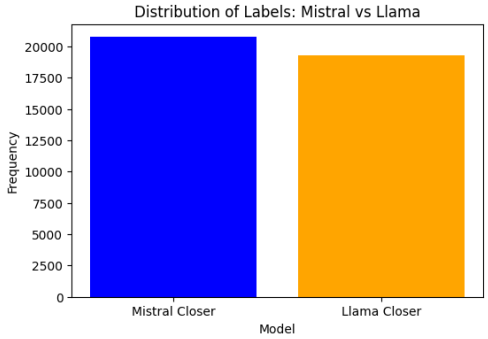

Figure 1: *Label Distribution: Mistral vs. Llama. Classes are balanced, ensuring fairness in evaluation.*

These generated sentences were then stored for evaluation and fed into a classification model to assess how well they aligned with the original.

Model Flow Diagrams To better explain the process, I created diagrams to show how the baseline and novel approaches process the data.

Baseline Model Flow. In the baseline, I used a pre-trained BERT model. It takes the story context and the generated sentences (from both Mistral and Llama) as inputs and directly predicts which one is closer to the original. The flow for this is shown in Figure 2.

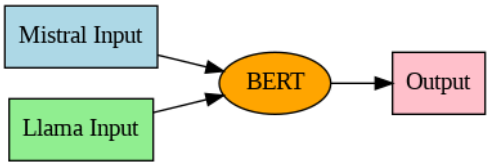

Figure 2: *Baseline Model Flow: Inputs from Mistral and Llama are processed by BERT to make a prediction.*

Novel Approach Model Flow. To improve on the baseline, I added a dense layer after the BERT model. The goal was to give the model extra capacity to understand more complex relationships between the inputs. Figure 3 illustrates this updated architecture.

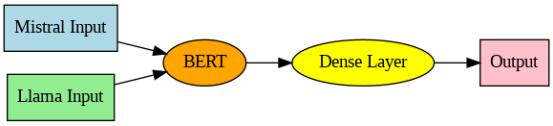

Figure 3: *Novel Approach Model Flow: BERT is followed by a dense layer before making a prediction.*

Baseline For the baseline, I used cosine similarity as a quick metric to compare how well Mistral and Llama-generated sentences matched the originals. Here's what I did:

- **Cosine Similarity:** I used the all-MiniLM-L6-v2 model to get sentence embeddings and compute similarity scores. Mistral performed slightly better (0.4378) compared to Llama (0.4284).
- **Grid Search:** I fine-tuned a BERT classifier by testing different learning rates (1e-5, 2e-5) and batch sizes (8, 16, 32). The best configuration achieved 63% accuracy.

Classification The classification model was a fine-tuned version of `bert-base-uncased`. Here's how the inputs were structured:

- **Mistral Input:** The first four sentences combined with the Mistral-generated fifth sentence.
- **Llama Input:** The first four sentences combined with the Llama-generated fifth sentence.

The goal was to predict which generated sentence better aligned with the original story.

Novelty To see if I could improve performance, I added a dense layer on top of BERT. I experimented with different hidden dimensions (64, 128, 256, 512, and 1024). However, these changes didn't lead to meaningful accuracy gains. Here's what I learned:

- Adding dense layers didn't improve the results, likely because BERT already captures the necessary features.
- Increasing the model's capacity may have caused redundancy or overfitting without adding value.

Implementation Details

Tools and Frameworks I used the Hugging Face Transformers library for model implementation and PyTorch for training and evaluation. These tools made it easier to experiment with different setups and track results.

Training and Validation Splits The dataset was split 80/20 into training and validation sets. This split provided enough data to train the model while keeping a good portion for evaluation.

Hardware All experiments were run on a GPU-enabled environment to handle the computational load.

Hyperparameters Here are the key hyperparameters I tested:

- **Learning rates:** 1e-5 and 2e-5.
- **Batch sizes:** 8, 16, 32.
- **Epochs:** 5.

Results

Baseline Results

The baseline evaluation included both generative and classification tasks. Below are the results for each.

Generative Models Using cosine similarity to compare generated sentences to the original, Mistral slightly outperformed Llama with a mean score of 0.4378 compared to 0.4284. This indicates that Mistral's outputs were more aligned with the original fifth sentences. The standard deviations (0.19 for Mistral and 0.17 for Llama) show variability, meaning some outputs were much closer to the original, while others were far off.

Classification Model For the classification task, I performed a grid search to tune hyperparameters for the `bert-base-uncased` model. I tested different combinations of learning rates and batch sizes. Table 1 summarizes the results.

Learning Rate	Batch Size	Loss	Accuracy
0.00001	8	0.64399	0.632625
0.00001	16	0.63505	0.622750
0.00001	32	0.64250	0.626500
0.00002	8	0.64204	0.627375
0.00002	16	0.91045	0.621750
0.00002	32	0.68278	0.623375

Table 1: *Grid Search Results for Classification Model.*

The best configuration was a learning rate of 0.00001 and a batch size of 8, resulting in an accuracy of 63.26%.

Dense Layer Experiments

I also tested whether adding a dense layer to the BERT model would improve classification performance. I tried different hidden layer dimensions (64, 128, 256, 512, and 1024). Unfortunately, none of these configurations improved accuracy. Table 2 shows the results.

HiddenDim	Loss	Accuracy
64	0.63491	0.629625
128	0.65062	0.615625
256	0.66249	0.624875
512	0.70591	0.621875
1024	0.65440	0.615375

Table 2: *Performance with Added Dense Layers.*

These results suggest that BERT's pre-trained architecture already captures most of the necessary

features for this task. Adding dense layers didn't help and might have introduced redundancy or overfitting.

Discussion

Analysis of Results

Generative Models Mistral performed slightly better than Llama with a cosine similarity score of 0.4378 compared to 0.4284. This shows that Mistral generally produces outputs that are closer to the original story endings. However, the variability in the scores (standard deviation of 0.19 for Mistral vs. 0.17 for Llama) indicates that both models had inconsistent results across the dataset. Some outputs were a good match, while others missed the mark.

Classification Model The classification model achieved an accuracy of about 63%, which sets a reasonable baseline. However, the results indicate a performance ceiling with the current configuration. Grid search helped find the optimal hyperparameters, but there wasn't significant room for improvement beyond this baseline.

Dense Layer Experiments Adding dense layers to the classification model didn't improve accuracy. The results suggest that BERT's pre-trained architecture already captures the necessary relationships between inputs, and the additional layers may have introduced redundancy or even overfitting. This outcome highlights that bigger isn't always better, especially with a well-tuned baseline model.

Strengths and Weaknesses

Strengths:

- **Generative and Classification Integration:** The combined approach leverages the strengths of generative and classification models to tackle a challenging narrative task.
- **Quantitative Evaluation:** Using cosine similarity provides a clear, interpretable way to measure how well the generated sentences align with the original.

Weaknesses:

- **Classification Model Performance:** The classifier plateaued at 63% accuracy, which leaves room for improvement with either better features or a different architecture.
- **Dense Layer Additions:** Experiments with dense layers didn't yield better results, showing the difficulty of enhancing pre-trained models without careful adjustments.

Comparison to Literature

These results align with prior work that highlights the effectiveness of transformer-based models for narrative tasks. One distinction of this project is the combination of generation and classification, which provides a more structured way to evaluate the models. This addresses gaps in earlier studies that relied on simpler metrics or qualitative comparisons.

Model Comparison

Mistral and Llama outputs were compared using cosine similarity:

- **Mistral:** Generated sentences often aligned more closely with the original and included richer details.
- **Llama:** Outputs were concise and contextually appropriate but lacked the depth seen in Mistral's sentences.

Example Comparison Here's a side-by-side example from the dataset:

Mistral: "Frank called an Uber and was able to make it to his date on time, apologizing for his tardiness."
Llama: "He called a taxi to pick him up, but it took a long time to arrive."

This highlights how Mistral produced a more detailed and aligned ending, while Llama's response was simpler but less specific.

Conclusion

Findings and Contributions

This project tackled the challenge of predicting a missing story ending using the ROCStories dataset. We focused on combining generative models, Mistral-7B-Instruct and Llama-3.1-8B-Instruct, with a BERT-based classifier to evaluate the quality of the generated sentences.

Here's what we found:

- **Mistral vs. Llama:** Mistral outperformed Llama in cosine similarity (0.4378 vs. 0.4284), showing it produced outputs more aligned with the original story endings.
- **Classifier Baseline:** The BERT-based classifier gave a reasonable baseline with 63% accuracy. While not perfect, it provides a starting point for automated evaluation of generative outputs.
- **Dense Layers:** Adding dense layers didn't improve classification accuracy, showing that BERT's architecture was already effective for this task.

Broader Implications

This work highlights how generative and classification models can work together to tackle complex narrative tasks. Beyond academic research, here's where this approach could be applied:

- **Content Platforms:** Automatically creating engaging short stories for apps like Instagram or TikTok.
- **Interactive Storytelling:** Helping users co-create stories in gaming or educational tools.
- **Personalized Narratives:** Tailoring story endings to individual preferences for better user engagement.

Lessons Learned

Here are the key takeaways:

- **Metrics Matter:** Cosine similarity worked well for evaluating generated outputs, but adding metrics like BLEU or ROUGE could provide more insights.
- **BERT Strengths and Limits:** While BERT handled this task effectively, trying to improve it with dense layers didn't help. Sometimes, simpler approaches are the most reliable.
- **Classifier Challenges:** The classifier plateaued at 63%, which shows there's room to improve with new features or better architectures.

Future Directions

There are several ways to build on this work:

- **Better Metrics:** Explore new ways to evaluate generated sentences beyond cosine similarity to capture more nuanced aspects of storytelling.
- **Model Improvements:** Test hybrid models or task-specific modifications to improve classification performance.
- **Dataset Expansion:** Use more diverse datasets to generalize findings and refine the models.
- **Multimedia Applications:** Extend this approach to generate videos or audio narratives, aligning with trends in multimedia content creation.

Overall, this project shows what current AI systems can do and where they still struggle with understanding and generating human-like narratives. There's plenty of potential here to push storytelling systems even further.

References

1. Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Michel Galley, and Lucy Vanderwende. A Corpus and Evaluation Framework for Deeper Understanding of Commonsense Stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016. Available at: `https://arxiv.org/abs/1604.01696`.
2. Jiwei Li and Eduard Hovy. Story Cloze Evaluator: Vector Space Representation Evaluation by Predicting What Happens Next. In *Proceedings of the 2016 Conference on Computational Linguistics*, 2016. Available at: `https://aclanthology.org/W16-2505.pdf`.
3. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 2020. Available at: `https://arxiv.org/abs/1910.10683`.
4. Jonas Pfeiffer, Aishwarya Kamath, Anna Rücklé, Kyunghyun Cho, and Iryna Gurevych. Few-shot Learning with Multilingual Language Models. In *Proceedings of ACL 2021*, 2021. Available at: `https://arxiv.org/abs/2109.04650`.