FRI Natural language processing course 2023
Project report

University of Ljubljana
Faculty of Computer and Information Science

Literacy situation models knowledge base creation

Timotej Košir, Luka Brecelj, Matjaž Ciglič

Abstract
TO-DO.

Keywords
TO-DO

Advisors: Aleš Žagar and Slavko Žitnik

Introduction

In recent years, there have been significant advancements in text generation models, but they still face challenges in producing coherent and lengthy stories with context provided as additional knowledge. Our paper addresses this issue by developing a model that can generate ending based on a story without a conclusion, while taking into account given context. Context will be provided in form of character sentiment, which will be reversed. To achieve this, we divide the problem into two domains: character sentiment analysis and story generation. With the former, we aim to extract information about the attributes of individual characters in the story, while the latter focuses on using the provided data as additional context for story generation.

Related work

Article Automatic Story Generation: Challenges and Attempts [1], provides a good baseline. It describes an approach to spatiotemporal modeling using convolutional LSTM networks (CLSTM). The authors introduce a Bayesian hierarchical framework that employs low-rank matrix factorization to learn separable spatial and temporal priors for the CLSTM weights [1]. Furthermore, they incorporate uncertainty quantification into the framework using dropout variational inference. Experiments on synthetic and real-world datasets demonstrate the efficency of the proposed method, with improvements in predictive performance and uncertainty quantification compared to existing techniques [1].

More similar to our area is an interesting article titled Controllable Multi-Character Psychology-Oriented Story Generation [2]. It presents an approach to automatic story generation that emphasizes controllable multi-character interactions and psychological elements [2]. The authors propose a psychologically-driven, deep-learning-based framework that uses BERT-based models for character interaction, an LSTM model for plot control, and a psychology-based rule system to manage character relationships and emotions. Experiments demonstrate the effectiveness of the proposed method in generating coherent and engaging stories with customizable plots, complex character relationships, and a rich portrayal of character psychology, highlighting its potential for advancing the field of story generation.

In the article[3], authors developed a text generation framework to address the disadvantages of previous model[4]. They incorporated external knowledge and developed a contextual ranker to rank the relevance of retrieved knowledge sentences to the story context. Sentence embedding is then used for weak supervision. The top-ranked knowledge sentences are fed to the conditional text generator to guide generation. The results are measured by automatic metrics and human evaluations, which show that their model generates more fluent, consistent, and coherent stories with lower repetition rate and higher diversities compared to previous state-of-the-art models. The model is also controllable as evidenced by successful replacement of keywords used to generate stories. Especially beneficial for us are the comparison between BERT and GPT2 languange models and ways how to compare goodness of generated stories.

Very recent article on the topic[5] explores the use of large-scale latent variable models for neural story generation with a focus on controllability. Authors integrate latent representation vectors with a Transformer-based pre-trained architecture to build a conditional variational autoencoder (CVAE) and show through experiments that their model achieves state-of-the-art conditional generation ability while maintaining excellent representation learning capability and controllability. The model components, such as encoder, decoder, and variational posterior, are all built on top of pre-trained GPT2 language model.

Dataset

At first, we tried utilizing the ”1002 short stories” database [6] for our project. This database seemed promising, as it contained a substantial number of stories that could potentially serve as a foundation for our research. However, we faced significant challenges while working with this database due to its structure and presentation.

One of the main issues we encountered was that all stories were compiled into a single CSV file, making it difficult to manage and process the data. Furthermore, there was an excessive amount of unnecessary text associated with each story, including introductions, author's notes, and other metadata. This additional text would require manual removal, which would be a labor-intensive and time-consuming process.

Considering these limitations, we decided to abandon the ”1002 short stories” database in favor of the ”Fairy Tales from Around the World” dataset[7]. This alternative dataset offered a rich and diverse collection of 1076 stories. The stories vary in length from 176 to 31420 words, with an average of 2370.1 words. They come from various countries and traditions, providing a more suitable foundation for our project. However, it having an unmarked story endings and inconsistencies in the stories still needed to be addressed.

To tackle these issues, we devised a system that approximated each story's ending by considering the final 10% of the text. This method enabled us to generate an initial dataset with rough approximations of story endings. Although not flawless, it served as a valuable starting point for further refinement. We opted for a combination of computational and manual efforts, acknowledging that the automated procedure might not yield a perfect dataset.

After the initial processing of the dataset, we proceeded to verify whether each file contained a complete story from start to finish, as well as the correct placement of the special character $< EOS >$ between the story and the ending. Additionally, we removed quotation marks and replaced exclamation points and question marks with commas within each story. This step allowed us to eliminate stories that were presented unconventionally, had ambiguous or incomplete endings, or did not align with our research goals. This process helped maintain the dataset's integrity and quality.

By employing a blend of automated processing and manual assessment, we successfully crafted a dataset tailored to our needs. Dataset, containing 1076 stories, provided a solid foundation for our project, enabling us to work towards predicting story endings based on incomplete stories.

Methods

This section presents the methodology needed to perform various tasks in our pipeline. Initially, we obtained the character sentiment and combined it with the dataset. By combining both the story and context, we obtained prompts that were used to train our story ending generation model. We describe the model and explain the necessary steps to ensure its good performance and how we evaluated it.

Sentiment Analysis

To obtain sentiment analysis on the characters, appearing in a fairy tale, we follow three main steps: character detection, extraction of sentences containing character mentions, and sentiment analysis of the extracted sentences. The implementation is based on Python and utilizes libraries such as spaCy, NLTK, and the Hugging Face Transformers library.

To detect characters in each story, we employ spaCy's pre-trained en_core_web_trf model to perform named entity recognition. We focus on entities labeled as ”PERSON” and add them to a set, ensuring unique character names. The resulting list of characters is returned. Then, we tokenize the story into sentences and search for each character's name in the sentences using regular expression searches. Upon finding a match, we extract a window of sentences around the character's mention, specified by a window size parameter. After collecting all the sentences for each character, we utilize a pre-trained sentiment analysis BERT transformer from Hugging Face [8] to obtain a score for each sentence. These scores are averaged to produce the final score for the character's sentiment.

The detected characters and their sentiment scores are stored in a JSON format, with one entry for each story. The resulting JSON file, ”stories_context.json”, contains a comprehensive representation of the characters and their associated sentiment scores. The entire process is executed for every fairy tale in the dataset

Story Generation

In order to exploit the power of pre-trained models, we use the smallest version of GPT2 model [9] for story ending generation. It has 12 layers, 12 heads per layer, input size of 1024 units and total of 117M parameters. For our intended use, it's important to have an encoder that features an unmasked or bi-directional structure, which provides a more comprehensive information than the masked or uni-directional structure typically found in the decoder for auto-regressive generation. Since GPT-2 only accepts a prompt as input to initiate text generation, we need to embed the necessary information about the desired context for story generation into the prompt itself. We accomplish this by defining a special token, $< sep >$, which separates the text we want to continue from the context. The prompt takes the form of ”context + $< sep >$ + sentence”, which is then encoded into the latent space using the GPT-2 tokenizer.

To fine-tune the model, we use the dataset of stories with obtained character sentiments. Ideally, we would use the whole story up until the end and all the character names with their sentiments reversed as the prompt. But GPT2 model accepts at most 1024 tokens as input, so we would need to have some internal state that would remember the past inputs of the story or embed the whole story into limited amount of tokens. To solve the problem, we currently split the story into windows of 500 words, which gives us about 500 tokens for each window to train the model on. Another problem is also too broad character sentiment score, which is a decimal in range from -1 to 1. We limit it by constraining the score to arbitrary amount of groups of positive, neutral and negative sentiments. With that we can train the model on a smaller dataset, since a few different labels appear much more commonly than a score would. Last issue, which might need addressing, are names in the stories. Since there are many different names, we might need to make them more uniform by naming all positive characters and negative characters similary (e.g. A1, A2, B1, B2, ...) to be sure our model correctly connects names to their corresponding sentiment. Nonetheless, we need to conduct more tests to be certain.

Evaluation

After fine-tunning the GPT2, we have to evaluate the generated text. Since there are no target texts we can compare our result to, we can use combination of automatic metrics and human evalutiaon.

One of such metrics is perplexity. Commonly used in the context of language modeling, it measures how well a model can predict a sequence of words by assigning a probability score to each possible word, based on the previous words in the sequence. In our case, we can split the test dataset into shorter sequences of words, where each is a prompt for the language model to generate a new story with a given context. We then calculate the perplexity score of the generated ending given the context. A lower perplexity score indicates that the language model is better at generating text that fits the given sentiment label. Perplexity scores of the generated reviews are compared to those of the original reviews to evaluate their quality.

Comparing the sentiment of characters in the original and generated story is another approach to automatically evaluate the performance. In this approach, the sentiment of each character is analyzed to determine whether the sentiment in the story aligns with the opposite character sentiment in the generated text. For example, if the writing prompt requires a story with a kind boy, then the sentiment analysis of the generated story's characters should reflect the opposite.

Last proposed approach is manual comparison of the generated story ending with the given context. After generating several stories, a human evaluator can manually compare the coherence, relevance, and alignment of the story with the context. The best story out of them is chosen and we fine-tune the model accordingly. Specifically, if the story is an existing one, it can be added to dataset and used as a target to teach the model. Contrary, if the story is unseen, it can be added as a new example to our dataset.

Results

First step was to perform sentiment analysis of the characters. It is not a focus of our article, so we mostly used pretrained models. To detect characters, we used NER [10][11] model. Since it was not trained to detect people from stories, some non-named entites, such as witch, mother or brother get lost. But overall, named entites are detected very well. An additional issue arises with the use of pronouns (e.g., he/she) since the model does not recognize them as referring to specific name. To address this challenge, we have expanded the context to include more sentences after a name is mentioned. With that, we have tagged most of the characters and given them a score, which will be preprocessed to range of 3 or 5 different sentiments (in example of 5 as very negative, negative, neutral, positive and very positive).

We have also fine-tunned the GPT2 context model on a small sample from our dataset. We used about 2% of stories with batch size of 8 and 10 epochs. The biggest problem we were facing was addition of special tokens, which confuse the model weights and the GPT2 would have started generating tokens instead of coherent texts. By decreasing batch update constant in the model these weights converge faster and solve the problem. But when the model will be trained using the whole dataset, there will be enough data to solve the problem without having to decrease batch update constant. Currently, the model generates texts that resemble style and rythm of stories, but it requires further refinement since it often starts using characters that were not in the given prompt.

The evalutation methods have not been implemented yet.

Discussion

TO-DO

Acknowledgments

TO-DO

References

[1] Amal Alabdulkarim, Siyan Li, and Xiangyu Peng. Automatic story generation: Challenges and attempts. *arXiv preprint arXiv:2102.12634*, 2021.

[2] Feifei Xu, Xinpeng Wang, Yunpu Ma, Volker Tresp, Yuyi Wang, Shanlin Zhou, and Haizhou Du. Controllable multi-character psychology-oriented story generation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1675–1684, 2020.

[3] Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. Megatron-cntrl: Controllable story generation with external knowledge using large-scale language models. *arXiv preprint arXiv:2010.00840*, 2020.

[4] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

[5] Le Fang, Tao Zeng, Chaochun Liu, Liefeng Bo, Wen Dong, and Changyou Chen. Transformer-based conditional variational autoencoder for controllable story generation. *arXiv preprint arXiv:2101.00828*, 2021.

[6] Shubchat. 1002 short stories from project gutenberg, 2023.

[7] Anna Bengardt. Fairy tales from around the world, 2021. Kaggle Dataset.

[8] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, and Julien Plu. Huggingface's transformers: State-of-the-art natural language processing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6370–6380, 2019.

[9] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[11] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.