VisuaLLM: Easy Web-based Visualization for Neural Language Generation

František Trebuňa and **Ondřej Dušek**
Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Prague, Czechia
ferotre@gmail.com, odusek@ufal.mff.cuni.cz

Abstract

VisuaLLM is a Python library that enables interactive visualization of common tasks in natural language generation with pretrained language models (using HuggingFace's model API), with tight integration of benchmark datasets and fine-grained generation control. The system runs as a local generation backend server and features a web-based frontend, allowing simple interface configuration by minimal Python code. The currently implemented views include data visualization, next-token prediction with probability distributions, and decoding parameter control, with simple extension to additional tasks.

1 Introduction

While pretrained language models (PLMs) reached state-of-the-art performance on many natural language generation (NLG) benchmarks (Kale and Rastogi, 2020; Ribeiro et al., 2020; Xiang et al., 2022), they are hard to control directly and are often used as a black-box architecture, where the developer feeds linearized training data and retrieves outputs in a batch-wise manner, without access to fine-grained model behavior. Interfaces for interactive PLM testing (such as the OpenAI playground[1] or the HuggingFace website)[2] typically only allow very basic operation, do not show any information beyond inputs/prompts and outputs, and are not connected to benchmark datasets.

This paper presents VisuaLLM, a simple, extensible library for visualization of PLM generation processes, built as a web-based frontend on top of the HuggingFace Transformers and Datasets frameworks (Wolf et al., 2020), taking inspiration from generic analysis tools such as TensorBoard[3] or WandB.[4] The current version allows to visualize tabular NLG datasets and the processes of low-level next-token prediction and high-level generation, with easily adjustable settings and tight benchmark dataset and metrics integration. The user can easily explore and evaluate PLMs in an interactive way, thus gaining insight into model behavior and being able to tune models more effectively. VisuaLLM is designed to be easily modified for different NLG tasks, where the user only picks a choice of datasets, models, adjustable decoding parameters and metrics. The whole interface is easily customizable via minimal Python code and can be extended to other NLG tasks. VisuaLLM can be installed by running `pip install visuallm`.[5]

```
ntp = NextTokenPredictionComponent(
    model=model,    # HuggingFace model object
    dataset=dataset # Huggingface dataset object
)
gen = InteractiveGenerationComponent(
    model=model,
    dataset=dataset,
    selectors={"num_beams": (1, 20)},
    metrics={"perplexity": Perplexity}
)
vis = DatasetVisualizationComponent(
    dataset=dataset
)
app = Server(
    __name__,
    components=[ntp, gen, vis]
).app

>> flask run
```

Figure 1: Example setup code for VisuaLLM.

2 System Architecture

We build on HuggingFace Transformers (Wolf et al., 2020) as the most commonly used PLM framework. We expect the programmer to load HuggingFace model and dataset objects and pass them to our framework as shown in Figure 1. VisuaLLM is used as a local server running on the user's machine, similar to e.g. TensorBoard. The code is divided into a web-based frontend (written

[1] https://platform.openai.com/
[2] https://huggingface.co/models
[3] https://www.tensorflow.org/tensorboard
[4] https://wandb.ai/

[5] Source code is available on GitHub at https://github.com/gortibaldik/visuallm. A demonstration screencast is shown at https://youtu.be/RMFEEW-Iu-4.

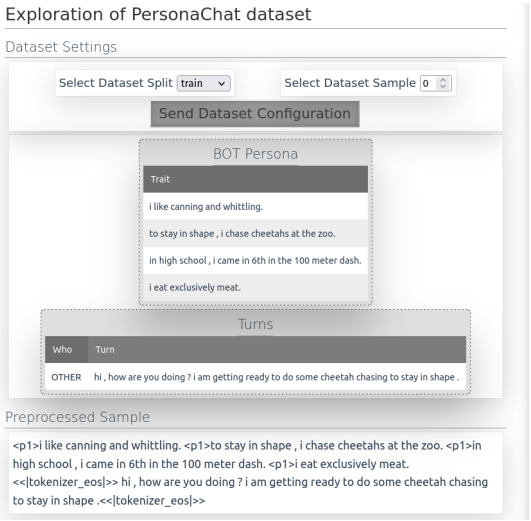

Figure 2: A view of a data sample from PersonaChat.

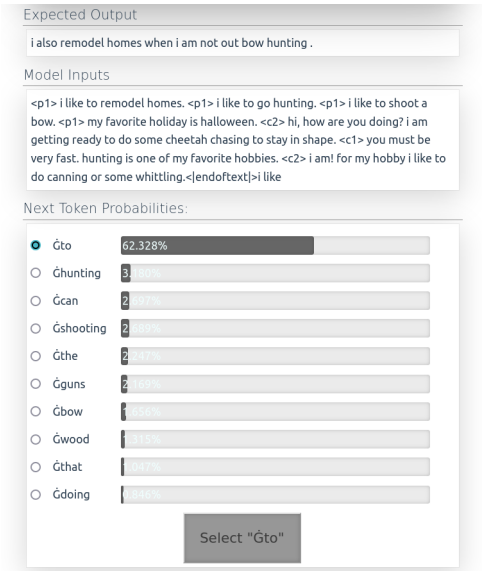

Figure 3: Next-token prediction visualization.

in Vue.js) and a Python backend (based on Flask). The frontend is built in a modular fashion, 100% configurable from Python code on the backend. Any frontend-backend communication is therefore abstracted away from the user. The whole setup is designed to use as little code as possible; customizing for any new HuggingFace-based model or task takes typically just a few dozen lines of code.

3 Usecases

We demonstrate VisuaLLM by visualizing dialogue generation on the PersonaChat benchmark (Zhang et al., 2018). The presented views can easily be extended or modified for other NLG tasks. For instance, we are currently working on an extension to allow interactive user input.

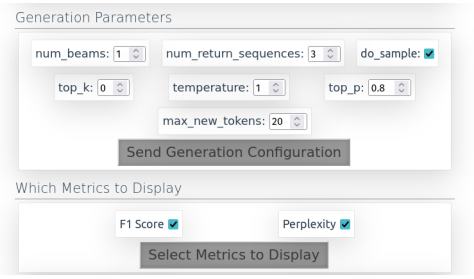

Figure 4: Generation parameters and metrics control.

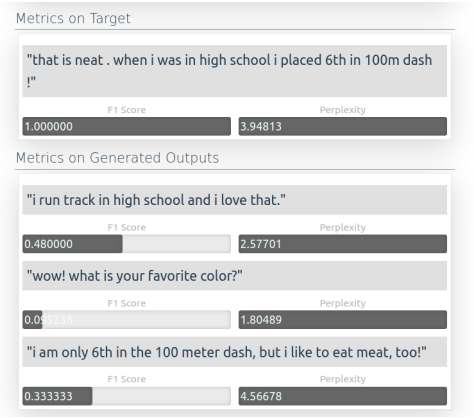

Figure 5: Outputs visualization with metrics.

Dataset Visualization Figure 2 shows a single data instance from PersonaChat. The dataset assumes a short persona description and preceding dialogue context as inputs for response generation. The interface shows both a tabular human-readable representation and a low-level linearized model input (configured for a specific trained model).

Next Token Prediction Visualization Visualizing next-token probability distributions is another common debugging task for PLMs. In Figure 3, we show that VisuaLLM allows the user to interactively step through sequence generation and control which token is selected, showing next-token probability distributions at each step. This view is also configured to include low-level model inputs and human reference outputs.

Generation Visualization This view is more high-level than the previous, exploring outputs generated with different decoder settings and their automatic metric scores. Controls in Figure 4 directly translate to HuggingFace's `generate()` method parameters and allow any callable Python metrics to be used, with configurable display of the metric values. Multiple generated outputs (using different settings) can then be compared to the reference, as shown in Figure 5.

Acknowledgements

This work was supported by the project TL05000236 *AI asistent pro žáky a učitele* co-financed by the Technological Agency of the Czech Republic within the ÉTA 5 Programme, and by the European Research Council (Grant agreement No. 101039303 NG-NLG). It used resources provided by the LINDAT/CLARIAH-CZ Research Infrastructure (Czech Ministry of Education, Youth and Sports project No. LM2018101).

References

Mihir Kale and Abhinav Rastogi. 2020. Text-to-Text Pre-Training for Data-to-Text Tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating Pretrained Language Models for Graph-to-Text Generation. *arXiv:2007.08426 [cs]*. ArXiv: 2007.08426.

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, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771 [cs]*. ArXiv: 1910.03771.

Jiannan Xiang, Zhengzhong Liu, Yucheng Zhou, Eric P. Xing, and Zhiting Hu. 2022. ASDOT: Any-Shot Data-to-Text Generation with Pretrained Language Models. In *Findings of EMNLP*, Abu Dhabi, UAE. arXiv. ArXiv:2210.04325 [cs].

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing Dialogue Agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.