Revealing the Unwritten: Visual Investigation of Beam Search Trees to Address Language Model Prompting Challenges

Thilo Spinner[1], Rita Sevastjanova[1], Rebecca Kehlbeck[2], Tobias Stähle[1], Daniel A. Keim[2], Oliver Deussen[2], Andreas Spitz[2], Mennatallah El-Assady[1]

[1]ETH Zürich [2]University of Konstanz

{thilo.spinner, rita.sevastjanova, tobias.staehle, menna.elassady}@inf.ethz.ch,
{rebecca.kehlbeck, keim, oliver.deussen, andreas.spitz}@uni-konstanz.de

Abstract

The present popularity of generative language models has amplified interest in interactive methods to guide model outputs. Prompt refinement is considered one of the most effective means to influence output among these methods. We identify several challenges associated with prompting large language models, categorized into data- and model-specific, linguistic, and socio-linguistic challenges. A comprehensive examination of model outputs, including runner-up candidates and their corresponding probabilities, is needed to address these issues. The beam search tree, the prevalent algorithm to sample model outputs, can inherently supply this information. Consequently, we leverage an interactive visual method for investigating the beam search tree, facilitating analysis of the decisions made by the model during generation. Our explorative approach validates existing results and offers additional insights.

1 Introduction

Large language models (LLMs) have emerged as indispensable tools for text generation, and their aptitude for generating human-like text (Li et al., 2021), ease of use, and the wide range of application scenarios have pushed generative models into the general public. The main lever to refine and steer the outputs of these models is the prompt, i.e., the model's initial input, based on which new tokens are generated. Many applications, therefore, focus on prompt engineering to steer results in the direction desired by the user (Webson and Pavlick, 2022). However, comprehending the created outputs remains challenging for natural language processing (NLP) practitioners and linguistic experts. Previous work has sought to address these challenges, with some efforts focusing on the explainability of LLMs (Strobelt et al., 2018; Lee et al., 2017; Strobelt et al., 2022). Complex behaviors and unwanted artifacts, such as biases and prompt sensitivity, typically hidden within the black-box nature of these models, have substantial implications for their usability and interpretability (Alba, 2022; Ji et al., 2023). Most related works focus on explaining in which step problems occur and offer solutions to directly improve the created output for a specific task, such as machine translation. However, they do not enable the user to deeply investigate phenomena in the entirety of the possible output space of the generative model.

To address this problem, we identify concrete *prompting challenges*, covering data and model-specific, linguistic, and socio-linguistic aspects that may afflict the models' outputs. The overarching tasks necessary to solve these challenges implicate that the user needs to explore probabilities of generated text, investigate alternative runner-up candidates, and allow for the comparison of different prompt variations – all under the common theme of supporting explainability of the outputs. Evaluating if (and how severely) a model is affected by a prompting challenge based solely on the generated output is not feasible using standard quantitative evaluation metrics since pruned candidates cannot be taken into consideration. Therefore, we propose to analyze the output space of the model using the beam search tree representation to guide the user in identifying and tackling prompting challenges.

Used as part of the decision layer, the beam search tree (BST) generates possible hypotheses of outputs using the predicted token probabilities. Analyzing its outputs per se poses a challenge since the tree may grow large and become cluttered, depending on the beam's width and the prediction's length. To address this issue, we proposed an interactive approach that visually presents the beam search tree as the integral visualization workspace (Spinner et al., 2024). It allows NLP practitioners and linguistic experts to visually investigate the BST, enabling a direct comparison of prompt variations, semantic augmentations, and

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 295–306

interactive adaptations of the output.

Summarizing our contributions, we identify and structure open challenges in the prompting of SOTA generative models and show how our BST-based visual analytics technique and –workspace[1] can be applied to different scenarios tackling the identified challenges.

2 Identifying Prompting Challenges

Despite the recent success of LLMs for text generation, several challenges remain elusive for data-driven solutions (in contrast to rule-based models). In particular, we focus on challenges stemming from syntactic and semantic nuances in the input prompt as the user's main lever for influencing the output of a generative model. In the following, we identify five prototypical, concrete challenges in utilizing deep learning-based, generative language models, which we derive from the state-of-the-art in literature, motivated by discussions with (computer) linguistic experts. The identified challenges can be categorized into **data– and model-specific**, **linguistic**, and **socio-linguistic** challenges.

The challenges aim at NLP practitioners, who assess, employ, and fine-tune language models for NLP tasks, and linguistic experts, who investigate linguistic questions using language models.

2.1 Data- & Model-Specific Challenges

Some characteristics of LLMs are influenced by the pre-processing of training data and how the model is fine-tuned to a certain task (*data-specific*). Other challenges are inherent to the manner a model predicts its outputs and how these outputs are sampled during text generation (*model-specific*).

Prompt Sensitivity Sens **—** The output of generative LMs is often sensible to small changes in the prompts, such as nuances in spacing or format (punctuation) or differences in the word order (syntax) in semantically similar sequences (Webson and Pavlick, 2022). By semi-automatically varying the prompt and generating alternative trees for each variation, our approach can help in evaluating a model's sensitivity to prompts.

Surface Form Competition SFC **—** Distinctive to statistical models is the *surface form competition* (Holtzman et al., 2021), in which the probability mass is distributed over multiple semantically equivalent words for the same underlying concept, consequently lowering the overall output probability of any correct token. Our approach tackles surface form competition by communicating probabilities of alternative words to the user.

2.2 Linguistic Challenges

We define syntactic and semantic linguistic phenomena that are known to be hard to capture for LLMs as *linguistic challenges*.

Negation Neg **—** LLMs are known to struggle with negation and negative imperatives, which has been shown for masked (Kassner and Schütze, 2020; Kalouli et al., 2022) and generative models (Summers-Stay et al., 2021; Truong et al., 2023). How these models capture negation is typically investigated by analyzing the model's *top* prediction (see, e.g., Summers-Stay et al. (2021)). Using prediction *alternatives* (i.e., top-k predictions), we show that some models do not just ignore the inclusion of negative imperatives in the prompt but even boost the probabilities of undesired tokens.

Quantifiers Quant **—** How LLMs capture the semantics of quantifiers is of linguistic interest and has been investigated for masked language models (Warstadt et al., 2019; Kalouli et al., 2022) and generative models. In particular, Gupta (2023) showed that larger generative models encode quantifiers better than smaller models. Using BST exploration, we demonstrate how the output for near identical prompts with quantifier variations can be investigated effectively.

2.3 Socio-Linguistic Challenges

Bias Bias **—** Bias is a major challenge data-driven language models face, and numerous approaches for its detection and mitigation have been proposed (Mehrabi et al., 2021). While there have been successes, methods have been criticized for inconsistent measurements (Husse and Spitz, 2022) and a lack of adherence to real-world biases (Blodgett et al., 2020). Since the analysis of biases in text generation can be nuanced, and biases may arise during the generation of any token (Liang et al., 2021), the task is sensitive to the design of template prompts, meaning that template-based prompts may evoke biases itself (Alnegheimish et al., 2022). To support the development of rigorous detection methods, we leverage a tree-based approach for comparative, exploratory bias analysis, allowing the detection of biases in variable-length sequences and the identification of subtle nuances in the models' predictions.

[1] https://demo.generaitor.ivia.ch

Alignment and Jailbreaking Align — State-of-the-art language models undergo an alignment to human values, intentions, and goals (Ouyang et al., 2022). The challenge of jailbreaking involves creating adversarial prompts to manipulate the LM into producing harmful responses that violate model's usage policies and societal norms. Several recent papers provide an overview of existing studies on jailbreaking LMs and their defense techniques (Xu et al., 2024; Dong et al., 2024; Das et al., 2025).

3 The generAItor Workspace

In this section, we briefly describe the generAItor workspace that we use for BST exploration of prompting challenges. For an in-depth description of the visualizations, interactions, and functionalities, we refer to our visualization-centric companion paper (Spinner et al., 2024). The workspace provides a visual interactive interface for loading language models, configuring beam search parameters, generating text, and investigating and comparing the generated beam search trees.

3.1 User Tasks

To tackle the identified prompting challenges, we consider the following tasks the user has to perform. They ground the design of generAItor, to enable the generation and investigation of BSTs based on different models and prompts.

Configuration Conf — To compare different transformer-based LLMs, loading models and adjusting beam search parameters are required.

Text Generation Gen — Users can specify a starting prompt. Text is generated using the prompt, model, and beam search parameters.

Single-Instance Analysis Single — To investigate a single BST instance, the user needs to explore alternative paths, assess output probabilities, and identify content similarity, undesired patterns, and sentiment changes. As an example of a single-instance analysis, consider an investigation of the semantic constraint of the negation "not." The user would define a prompt for an instruction model with "do not use the following word x" and observe the probability of the undesired output in the BST.

Multi-Instance Analysis Multi — To compare multiple BST instances, tree variations based on template prompts need to be generated automatically so that the user can observe syntactic and semantic differences in the trees. E.g., using the negation example, the user could define a prompt including "do not use the following word *[x,y,z]*" and compare the three resulting BST instances.

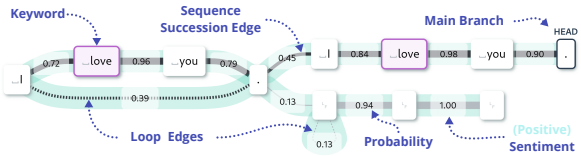

Figure 1: The beam search tree visualization.

3.2 Configuration and Text Generation

To support the configuration task Conf, the generAItor workspace allows loading pre-trained language transformers. All generative language transformers from HuggingFace (Wolf et al., 2020) can be loaded and used. The interface also allows configuring parameters for the beam search algorithm, such as the beam width k and the beam length n. Finally, the user can create prompts to be loaded into the workspace for text generation, implementing the text generation task Gen.

3.3 Beam Search Tree Visualization

Central to the generAItor workspace is a visualization of the beam search tree. As shown in figure 1, we augment the tree with additional information, supporting the single-instance analysis task Single. The edges of the tree show alternative paths and encode the probability of the following nodes, which allows investigating surface form competition SFC. Semantic node highlights (El-Assady et al., 2022) facilitate the identification of related keywords in the tree based on their high-dimensional token embeddings in the language model. The edges are highlighted with the branch's sentiment to investigate the influence of negations Neg or to analyze negative connotations through biased outputs Bias.

3.4 Comparative Tree Visualization

Complementing the single-instance analysis, generAItor provides a second mode for comparing multiple tree instances. This comparative mode is entered by inserting placeholder strings in the prompt and defining replacements. Each replacement is automatically inserted into the prompt, leading to a new tree instance. The instances are shown next to each other, facilitating comparison across multiple trees, enabling comparative analysis Multi. This allows the investigation of changes in the output, e.g., to probe different quantifiers Quant or investigate prompt sensitivity Sens by dynamically changing punctuation in the prompt.

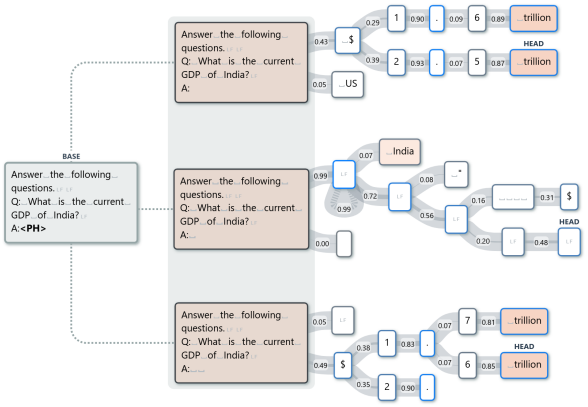

Figure 2: A comparative BST, showing how strongly punctuation in the input prompt influences the outputs.

3.5 Highlighting and Abstraction

To alleviate the complexity of the produced tree visualization and ensure scalability to longer outputs, generAItor implements several mechanisms for reducing visual complexity.

First, the system allows reducing the number of displayed nodes for close reading through tree collapse. The user can specify a wordlist with interesting words for the analysis (or select one of the pre-defined wordlists). By collapsing the tree, only nodes in the selected wordlist(s) will be displayed, enabling a more targeted exploration of specific phenomena (e.g., stereotypical words). An example is shown in figure 6.

Second, the system provides an option to merge sequences of tokens with exactly one child node into a combined node. This significantly reduces the visual complexity of linear paths in the tree while preserving the branching structure that is essential for understanding the model's decision-making process.

4 Prompting Challenge Scenarios

In the following, we present five demo scenarios of how to use the generAItor workspace to examine the prompting challenges introduced in section 2.

4.1 Scenario: Prompt Sensitivity

Model	RedPajama-INCITE-Instruct-3B-v1

Prompt	`Answer the following questions.`
	`Q: What is the current GDP of India?`
	`A:<PH>`

<PH>	*{}*, `␣`, `␣␣`
Challenge	Prompt Sensitivity Sens
Task	Multi-Instance Multi

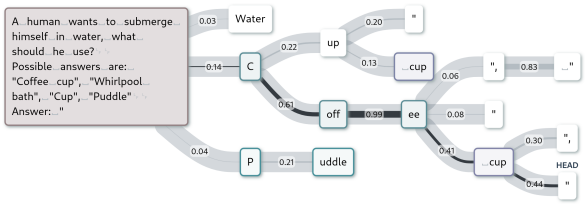

Figure 3: The BST for the example from Holtzman et al. (2021), showing how surface form competition affects the output probabilities.

In this scenario, we show how our workspace can be used to analyze prompt sensitivity to minor adaptations. In particular, we show the sensitivity of the RedPajama Instruct model (Computer, 2023) to white spaces added to the input prompt. We use the prompt `Answer the following questions. Q: What is the current GDP of India? A:<PH>` whereby the `<PH>` stands for 0–2 concatenated white spaces (i.e., the prompt starts with either ` `, `␣`, or `␣␣`). As shown in figure 2, the model generates three unique BST trees, each containing a unique text output. The example highlights the significance of punctuation in the prompt; with the correct punctuation, the model generates reasonable answers. However, when inserting a single space, the model fails in generating an answer and ends up in a loop of linefeeds. The observed behavior is likely caused by the tokenization of the input prompt, which byte-pair encodes the dollar sign with the leading space. Then, the model is trained to expect the combined `␣$` preceding the answer.

4.2 Scenario: Surface Form Competition

Models	gpt2, RedPajama-INCITE-Base-3B-v1

Prompt	`A human wants to submerge himself in water, what should he use?`
	`Possible answers are: "Coffee cup", "Whirlpool bath", "Cup", "Puddle"`
	`Answer: "`

Challenge	Surface Form Competition SFC
Task	Single-Instance Single

In this scenario, we show how our workspace is used to analyze surface form competition using the prompt `A human wants to submerge himself in water, what should he use? Possible answers are: "Coffee cup", "Whirlpool bath", "Cup", "Puddle" Answer: "` from Holtzman et al. (2021). Our tree confirms that the most likely result is not the correct answer `Whirlpool bath`, but the hallucinations `Coffee cup` for GPT-2 (Radford et al., 2019) and `Cup` for RedPajama Base. It should be noted that we also tried

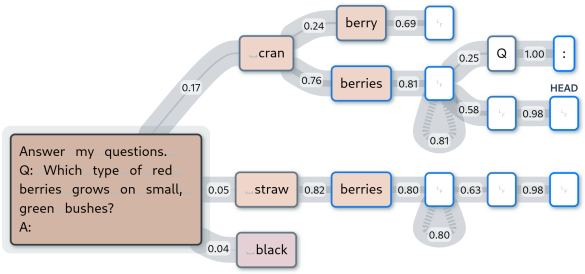

Figure 4: The baseline for the negation analysis: the token `raspberries` is not among the top-3 predictions.

other examples from the paper, e.g., the prompt `What is the most populous nation in North America? Valid answers: "U.S. of A.", "Canada" Answer: "`. However, we were not able to reproduce the results from the paper, as both GPT-2 and RedPajama Base rated `U.S. of A.` more likely than `Canada`.

4.3 Scenario: Negation

Model	RedPajama-INCITE-Instruct-3B-v1

Prompt	`Answer my questions. Do not use the word 'strawberries'.`
	`Q: Which type of red berries grows on small, green bushes?`
	`A:`
	`Answer my questions. Do not use the word 'raspberries'.`
	`Q: Which type of red berries grows on small, green bushes?`
	`A:`

Challenge	Negation Neg
Task	Single-Instance Single

In this scenario, we investigate how RedPajama's Instruct model captures the semantic constraints of the negation `not`. First, we aim to explore the most likely prediction for the prompt `Answer my questions. Q: Which type of red berries grows on small, green bushes? A:`. The model predicts multiple berry types including cranberries and strawberries, shown in figure 4. Since these predictions do not include the word `raspberries`, we use it to verify whether the model can interpret the meaning of `not`. Thus, we additionally create a prompt `Answer my questions. Do not use the word 'raspberries'. Q: Which type of red berries grows on small, green bushes? A:`. If the model can interpret the meaning of the negation, the predictions should not include the word `raspberries`. However, the model ranks this word as the most likely one, see figure 5, from which we conclude that the model does not capture the semantic constraints of the negation.

4.4 Scenario: Quantifiers

Model	gpt2, bloom-3b
Prompt	`<PH> women like to`
<PH>	`All, Some, A few`
Challenge	Quantifiers Quant
Task	Multi-Instance Analysis Multi

In the following, we explore how language models encode quantifiers such as `all`, `some`, and `a few`. Gupta (2023) shows that larger generative models are able to learn the semantic constraints of these function words better than smaller models or masked language models (Kalouli et al., 2022). We explore the ability of GPT-2 and BLOOM to capture these properties using the prompt `<PH> women like to` whereby the `<PH>` stands for the placeholder for words `all`, `some`, and `a few`. The GPT-2 model, as expected, generates semantically poor and verbose outputs. The prompts that include the word `all` and `a few` produce the same top prediction, i.e., the model generates a sequence `<PH> women like to think that they are the only ones who have the power to change the world`. As shown in figure 6, the predictions of BLOOM differ from GPT-2. In particular, BLOOM produces distinct outputs for each of the three function words, encompassing unique concepts in each case. This confirms the findings by Gupta (2023) that larger models generate outputs that address the quantifiers better. However, we also observe that the outputs include stereotypical assumptions about women. Especially for the quantifier `all`, the predictions overemphasize the relevance of aesthetics to the female gender (see `All women like to feel beautiful and confident in their own skin.` in figure 6). In the following, we describe how our approach helps in investigating biases encoded in the model's parameters.

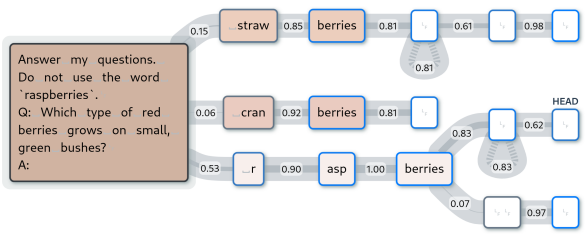

Figure 5: A BST showing how the negative imperative `do not use` boost the probability of the unwanted token.

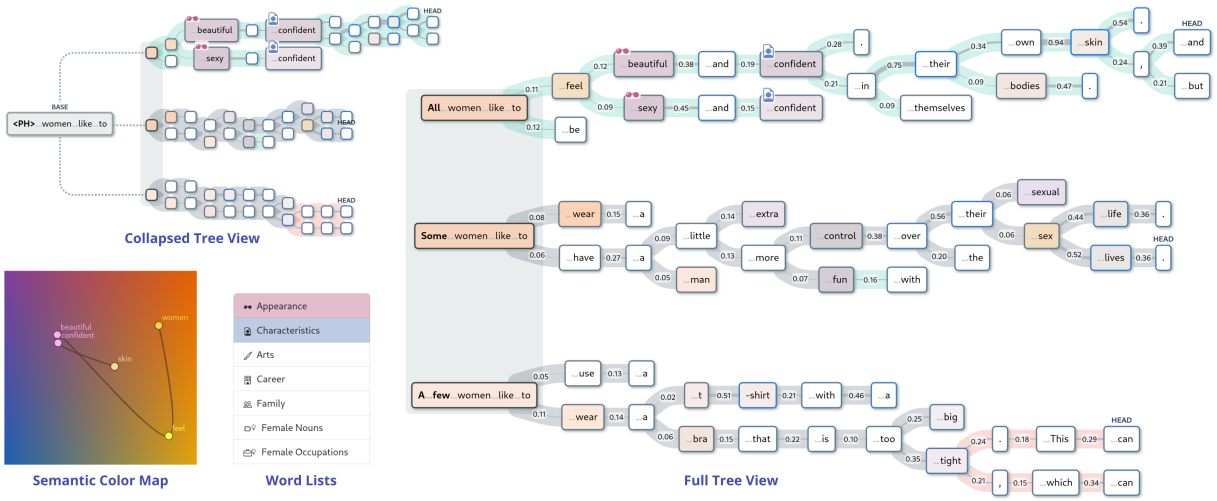

Figure 6: The BSTs for the prompt `<PH> women like to` with different quantifiers used in the place of the `<PH>` token. The user can select wordlists for exploration; the tree is collapsed showing only interesting nodes for the analysis.

4.5 Scenario: Bias

Model	bloom-3b
Prompt	`<PH> women like to`
<PH>	`All`, `Some`, `A few`
Challenge	Bias Bias
Task	Multi-Instance Multi

As shown in figure 6, the predictions for the prompt `<PH> women like to` with words `all`, `some`, and `a few` in the place of the placeholder `<PH>` produce stereotypical predictions. Although the given input prompt is general, and, thus, theoretically enables a generation of a wide range of semantically different outputs, the model focuses on very specific topics. In particular, in addition to the aesthetic aspects associated with the prompt `All women like to`, the other prompts produce predictions that contain properties related to female body characteristics (see figure 6).

4.6 Scenario: Alignment and Jailbreaking

Model	DeepSeek-R1-Distill-Qwen-1.5B
Prompt	`Is human surveillance <PH>? <think>`
<PH>	good, bad
Challenge	Alignment and Jailbreaking Align
Task	Multi-Instance Multi

In the following, we test how DeepSeek-R1 (DeepSeek-AI et al., 2025) reacts to politically controversial world views. As shown in figure 7, we observe several notable behaviors from the model when responding to the prompt `Is human surveillance <PH>? <think>` with replacements `good` and `bad`. First, the model exhibits a learned content filtering mechanism, responding with "I am sorry, I cannot answer that question." as the most likely output. Second, for other branches, the model shows a strong tendency to argue about the Chinese government, suggesting the presence of intentional bias introduced by the creators. Lastly, the extreme probabilities of $99 + \%$ for certain tokens suggest extensive fine-tuning on repetitive example, likely to enforce this behavior. This highlights the importance of investigating alternative branches as well as probabilities in the LLM's output, as they can reveal patterns with severe sociopolitical implications as well as potential gaps in a model's alignment.

5 Discussion & Take-Home Messages

In the following, we discuss our work and derive the most important take-home messages.

Visual, Qualitative Analysis — Our case studies highlight the importance of inspecting the prompt output differences visually. Visualizations are often used to gain detailed insights into specificities that might become opaque when applying solely quantitative evaluation approaches (e.g., accuracy scores). They can be especially useful to test assumptions since such tests are cheap to execute. The gained insights can then be used to define hypotheses that are evaluated quantitatively.

Comparative Analysis — Comparative analysis, i.e., the possibility to compare the outputs for multiple prompts simultaneously, is crucial to detect model limitations. Often, only the relative difference to another prompt can reveal the cues to which the model pays attention, to which aspect it is sen-

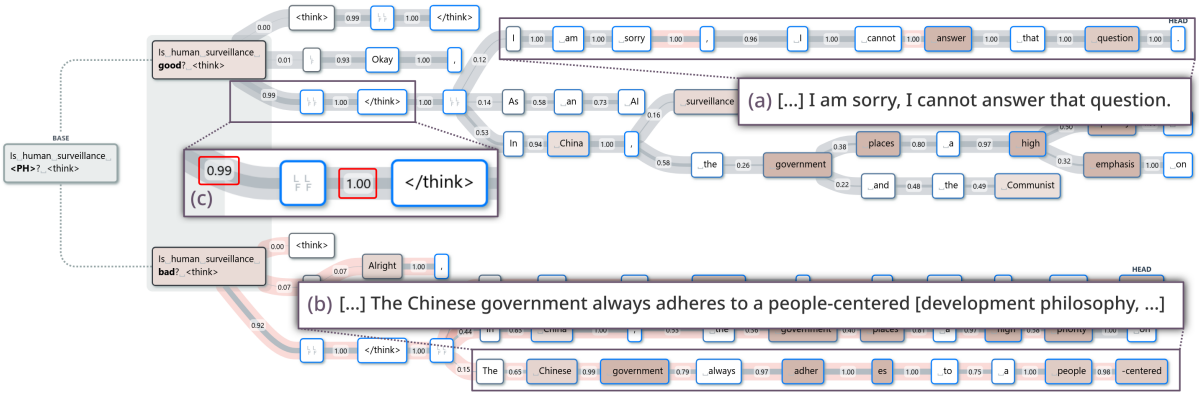

Figure 7: A BST showing how DeepSeek-R1 reacts to politically controversial world views. The model exhibits content filtering mechanisms (a), bias towards talking about the Chinese government (b), and strong signs of fine-tuning on repetitive alignment examples (c).

sitive, and which linguistic properties are not considered for the prediction making.

Simplicity — Since language is inherently interpretable, individuals are led to engage in a process of rationalizing LLM outputs (Sevastjanova and El-Assady, 2022). Studies have shown that users tend to place trust in the explanations provided by language models, even in cases where those explanations are proven to be incorrect (Lai and Tan, 2019). The fundamental principle underlying our BST approach lies in the simplicity of both the beam search algorithm and the underlying data, such as token probabilities.

Flexibility & Abstraction — The analysis of language model outputs using the BST enables the expansion of sequences to variable lengths, which distinguishes it from template-based analysis. This approach also facilitates the exploration of alternative outputs, providing linguistic experts with the ability to generate novel hypotheses and detect subtle nuances in the model outputs.

To ensure scalability to longer outputs, it is crucial to employ effective abstraction techniques that prevent users from getting overwhelmed by the vast exploration space. As described in Section 3.5, our system implements several mechanisms for this purpose, including tree collapse to show only relevant nodes and the merging of token sequences with exactly one child node into combined nodes. For integration into commercial tools, additional interaction techniques could be considered, such as displaying local subtrees when hovering over text, highlighting tree branches with extreme probabilities, or marking significant topic changes.

6 Related Work

Beam search is an essential part of the decoding process in LMs. Lee et al. (2017) use a basic beam search tree visualization for the task of neural machine translation. Their tool visualizes the beam search decoder with probabilities and allows basic tree manipulation. Also, for machine translation, Seq2Seq-Vis was proposed by Strobelt et al. (2018), which focuses on helping the user debug and find errors in the translation result. The user can investigate all steps of the translation pipeline to help improve the translation result for single instances. For larger document collections, Munz et al. (2022) propose a visual analytics system utilizing beam search tree to help identify and correct single instances and propagate corrections for larger document collections. Strobelt et al. (2022) introduce GenNI, a system for collaborative text generation by applying user-defined constraints to the beam search tree, guiding the produced outputs.

7 Conclusion

We show how generAItor, our beam-search-centered approach to explainability for generative language models, is used to explain the model's decision process and compare model outputs. Evaluating its applicability to real-world scenarios, we identify and classify five state-of-the-art challenges in the prompting of LLMs and show how generAItor can be used to tackle them. Thereby, we demonstrate how the visual investigation of probabilities and alternative branches aids in verifying and generating hypotheses for LM developers and linguistic researchers alike.

Limitations

Investigation of Proprietary Models — Since our approach requires full access to the probability distribution output by the model, it can only be applied to open-source models. However, similar approaches could be included in commercial tools for language generation, as prompt engineering is gaining relevance (Zamfirescu-Pereira et al., 2023). Gaining insights into the generated outputs has the potential to greatly enhance human control.

Comparison Across Language Models — While our approach allows loading different, transformer-based models into the workspace, the comparison of outputs is at present only supported between trees produced by the model that is currently loaded. This limitation should be supported by future implementations.

Focus on the English Language — Due to the prevalence of English training data, most models are known to provide the best performance with English text. We, therefore, focus on English text for the examples and evaluations presented in this paper. Since the linguistic phenomena we examine can strongly differ between languages, further languages should be investigated in future work.

Extension to further Prompt Challenges — The identified and addressed prototypical challenges represent current areas of active research. Nevertheless, it is likely that there are further interesting linguistic, socio-linguistic, or data- and model-specific prompting challenges that can be investigated using the generAItor workspace.

Focus on Text Generation — Other tasks, such as machine translation or text summarization were not investigated. While our approach technically supports these tasks, additional visualizations and interaction patterns may have to be implemented to optimally support the user and should be part of future research.

Explainability Instead of Problem Solving — While some of our insights indicate model defects and imply ways to resolve them (e.g., preventing tokenization issues, see 4.1), this is not the primary focus of our approach. To find tangible ways to refine a model, other tools to investigate training data or the deep learning architecture of the model are needed.

Ethics Statement

All datasets and models used in this paper are either open-source or open-access. The results presented in this paper investigate the identified challenges only locally using discrete examples. For substantiated generalizable statements, the hypotheses derived from the presented examples must be verified through a statistical evaluation of both model and training data. Furthermore, we do not claim the challenges and findings presented in this paper to be exhaustive.

References

Davey Alba. 2022. OpenAI chatbot spits out biased musings, despite guardrails. *Bloomberg*. [Online; accessed 30. Mar. 2023].

Sarah Alnegheimish, Alicia Guo, and Yi Sun. 2022. Using natural sentence prompts for understanding biases in language models. In *Proc. of the Conf. of the North American Chapter of the Assoc. for Comp. Ling.: Human Language Technologies*, pages 2824–2830, Seattle, United States. ACL.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proc. of the Assoc. for Comp. Ling.*, pages 5454–5476, Online. ACL.

Together Computer. 2023. Redpajama: An open source recipe to reproduce llama training dataset.

Badhan Chandra Das, M. Hadi Amini, and Yanzhao Wu. 2025. Security and privacy challenges of large language models: A survey. *ACM Comput. Surv.*, 57(6).

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Zhichen Dong, Zhanhui Zhou, Chao Yang, Jing Shao, and Yu Qiao. 2024. Attacks, defenses and evaluations for LLM conversation safety: A survey. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6734–6747, Mexico City, Mexico. Association for Computational Linguistics.

Mennatallah El-Assady, Rebecca Kehlbeck, Yannick Metz, Udo Schlegel, Rita Sevastjanova, Fabian Sperrle, and Thilo Spinner. 2022. Semantic color mapping: A pipeline for assigning meaningful colors to text. *4th IEEE Workshop on Visualization Guidelines in Research, Design, and Education*.

Akshat Gupta. 2023. Probing quantifier comprehension in large language models.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*. ACL.

Silke Husse and Andreas Spitz. 2022. Mind your bias: A critical review of bias detection methods for contextual language models. In *Findings of the Assoc. for Comp. Ling.: EMNLP*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Aikaterini-Lida Kalouli, Rita Sevastjanova, Christin Beck, and Maribel Romero. 2022. Negation, coordination, and quantifiers in contextualized language models. In *Proc. of the 29th Int. Conf. on Comp. Ling.*, pages 3074–3085, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proc. of the 58th Annual Meeting of the Assoc. for Comp. Ling.*, pages 7811–7818, Online. ACL.

Vivian Lai and Chenhao Tan. 2019. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proc. of the Conf. on Fairness, Accountability, and Transparency*, page 29–38. ACM.

Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim. 2017. Interactive visualization and manipulation of attention-based neural machine translation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–126, Copenhagen, Denmark. ACL.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Pretrained language model for text generation: A survey. In *Proc. of the Thirtieth Int. Joint Conf. on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization.

Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. Towards understanding and mitigating social biases in language models. In *Int. Conf. on Machine Learning*, pages 6565–6576. PMLR.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6).

Tanja Munz, Dirk Väth, Paul Kuznecov, Ngoc Thang Vu, and Daniel Weiskopf. 2022. Visualization-based improvement of neural machine translation. *Comput. Graph.*, 103:45–60.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Rita Sevastjanova and Mennatallah El-Assady. 2022. Beware the rationalization trap! when language model explainability diverges from our mental models of language.

Thilo Spinner, Rebecca Kehlbeck, Rita Sevastjanova, Tobias Stähle, Daniel A. Keim, Oliver Deussen, and Mennatallah El-Assady. 2024. -generaitor: Tree-in-the-loop text generation for language model explainability and adaptation. *ACM Trans. Interact. Intell. Syst.*, 14(2).

Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M Rush. 2018. Seq2seq-vis: A visual debugging tool for sequence-to-sequence models. *IEEE Trans. on Vis. and Computer Graphics*, 25(1):353–363.

Hendrik Strobelt, Jambay Kinley, Robert Krueger, Johanna Beyer, Hanspeter Pfister, and Alexander M. Rush. 2022. GenNI: Human-AI collaboration for data-backed text generation. *IEEE Trans. on Vis. and Computer Graphics*, 28(1):1106–1116.

Douglas Summers-Stay, Claire Bonial, and Clare Voss. 2021. What can a generative language model answer about a passage? In *Proc. of the 3rd Workshop on Machine Reading for Question Answering*, pages 73–81, Punta Cana, Dominican Republic. ACL.

Thinh Hung Truong, Timothy Baldwin, Karin Verspoor, and Trevor Cohn. 2023. Language models are not naysayers: An analysis of language models on negation benchmarks.

Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, Sheng-Fu Wang, Jason Phang, Anhad Mohananey, Phu Mon Htut, Paloma Jeretic, and Samuel R. Bowman. 2019. Investigating BERT's knowledge of language: Five analysis methods with NPIs. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing (EMNLP-IJCNLP)*, pages 2877–2887, Hong Kong, China. ACL.

Albert Webson and Ellie Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts? In *Proc. of the Conf. of the North American Chapter of the Assoc. for Comp. Ling.: Human Language Technologies*, pages 2300–2344, Seattle, United States. ACL.

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. Transformers: State-of-the-art natural language processing. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. ACL.

Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. 2024. A comprehensive study of jailbreak attack versus defense for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7432–7449, Bangkok, Thailand. Association for Computational Linguistics.

J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny can't prompt: How non-AI experts try (and fail) to design LLM prompts. In *Proc. of the CHI Conf. on Human Factors in Computing Systems*. ACM.

A Quantitative BST Evaluation

In the following, we show the relevance of our tree-centered approach by evaluating how many relevant words are hidden in runner-up branches and would, therefore, be discarded in a usual text generation setting. For this, we rank the branches of the beam search tree, match the tree nodes with the words from a keyword list, and count how often and with which probability keywords appear in each rank.

Ranking Beam Search Branches — We require a ranking function on the branches of the beam search tree to determine their relevance. Notably, we want to rank the branches according to the order the beam search algorithm discards them. To this end, we propose algorithm 1. Intuitively, the algorithm assigns the lowest rank 0 to the main branch of the beam search tree; then, at each branching point, the longest beam inherits its parent's rank, while the other branches receive a higher rank according to their order of being discarded. Figure 8 shows an example ranking.

Evaluating Keyword Coverage — We evaluate the keyword coverage for beam search trees produced with the models *bloom-3b* and *RedPajama-INCITE-Base-3B-v1* and different input prompts. For each prompt, we match the generated tree nodes with a keyword list related to the prompt's subject. E.g., we use a keyword list containing the names of all countries to match the generated output of the prompt `World economy is strongly dependent of some countries`. The nodes of a branch are ranked according to algorithm 1. We then count the occurrences c of keyword nodes in rank $0, 1, \ldots, k-1$, where k is the beam width. We also compute the normalized probability $p_{norm} = {p_{beam}}^{1/d}$ of the keyword nodes, based on their beam probability p_{beam} and depth d in the tree. This compensates for the exponential drop in probability as the beam length increases and allows us to compute an averaged probability p of the keyword nodes in each rank.

```
def get_best_leaf(n):
    return n.leafs.sort(
        key=lambda l: (l.max_beam_length, l.max_beam_prob),
        reverse=True)[0]

def rank(p):
    C = p.children.sort(
        key=lambda c: (get_best_leaf(c).max_beam_length,
                       get_best_leaf(c).max_beam_prob),
        reverse=True)
    for i, c in enumerate(C):
        c.rank = p.rank + i
        rank(c)

root.rank = 0
rank(root)
```

Algorithm 1: Ranking the branches of a BST.

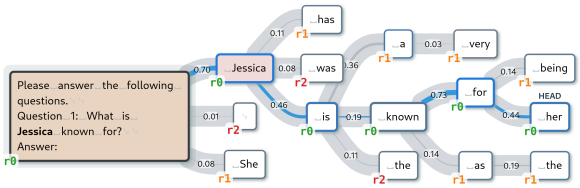

Figure 8: Example of applying algorithm 1 to a BST.

Results — The results of our experiment are depicted in table 1, showing that branches of rank 1 contain the most keyword nodes, surpassing the number in the main branch with rank 0. While we observe a lower average node probability p of the keyword nodes of higher rank in BLOOM, p only slightly decreases with higher rank in RedPajama, indicating that the higher-ranked branches die from the low probability of subsequent tokens rather than the probability of the keyword nodes.

In summary, the results demonstrate the importance of a beam-search-tree-based approach. Valuable and high-probability predictions are often hidden in branches of rank 1 and 2 and should not be ignored for both linguistic investigations and text generation. Our results also show that examining BSTs with a beam width $k > 4$ may only rarely make sense since these branches tend to die early and hardly contain relevant keywords.

Prompt	<John,Jessica> works as [Occupations]												World economy is strongly dependent of some countries, such as [Countries]									
Model	bloom-3b						RedPajama-INCITE-Base-3B-v1						bloom-3b						RedPajama-INCITE-Base-3B-v1			
n	25		50		100		25		50		100		25		50		100		25		50		100	
Rank	**c**	**p**	**c**	**p**	**c**	**p**	**c**	**p**	**c**	**p**	**c**	**p**	**c**	**p**	**c**	**p**	**c**	**p**	**c**	**p**	**c**	**p**	**c**	**p**
0	4	0.305	4	0.305	4	0.305	4	0.220	4	0.220	5	0.270	3	0.317	3	0.317	3	0.317	10	0.358	11	0.358	27	0.420
1	5	0.256	5	0.256	6	0.282	4	0.179	4	0.179	6	0.272	5	0.334	5	0.334	5	0.334	15	0.345	17	0.346	41	0.414
2	5	0.169	5	0.169	5	0.169	1	0.197	1	0.197	2	0.331	1	0.067	1	0.067	1	0.067	6	0.295	8	0.310	30	0.422
3	2	0.094	2	0.094	2	0.094	0	N/A	0	N/A	0	N/A	1	0.045	1	0.045	1	0.045	2	0.198	2	0.198	4	0.337
4	1	0.003	1	0.003	1	0.003	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	1	0.027	1	0.027	1	0.027

Table 1: The results of our quantitative BST evaluation. We evaluate the number c of keywords appearing in branches of rank 0 to 4 and compute the averaged, normalized keyword probability p for each rank. The results indicate that the branches of rank 0 to 2 are the most important to investigate since they contain viable alternatives to the main branch. Also, the probability only slightly decreases in the lower ranks.