🌳-generAItor: Tree-in-the-loop Text Generation for Language Model Explainability and Adaptation

THILO SPINNER, ETH Zurich, Zurich, Switzerland
REBECCA KEHLBECK, University of Konstanz, Konstanz, Germany
RITA SEVASTJANOVA, ETH Zurich, Zurich, Switzerland
TOBIAS STÄHLE, University of Konstanz, Konstanz, Germany
DANIEL A. KEIM, University of Konstanz, Konstanz, Germany
OLIVER DEUSSEN, University of Konstanz, Konstanz, Germany
MENNATALLAH EL-ASSADY, ETH Zurich, Zurich, Switzerland

Large language models (LLMs) are widely deployed in various downstream tasks, e.g., auto-completion, aided writing, or chat-based text generation. However, the considered output candidates of the underlying search algorithm are under-explored and under-explained. We tackle this shortcoming by proposing a *tree-in-the-loop* approach, where a visual representation of the beam search tree is the central component for analyzing, explaining, and adapting the generated outputs. To support these tasks, we present generAItor, a visual analytics technique, augmenting the central beam search tree with various task-specific widgets, providing targeted visualizations and interaction possibilities. Our approach allows interactions on multiple levels and offers an iterative pipeline that encompasses generating, exploring, and comparing output candidates, as well as fine-tuning the model based on adapted data. Our case study shows that our tool generates new insights in gender bias analysis beyond state-of-the-art template-based methods. Additionally, we demonstrate the applicability of our approach in a qualitative user study. Finally, we quantitatively evaluate the adaptability of the model to few samples, as occurring in text-generation use cases.

CCS Concepts: • **Computing methodologies** → **Natural language generation**; • **Human-centered computing** → **Graphical user interfaces**; **Visualization systems and tools**; • **Mathematics of computing** → **Exploratory data analysis;**

Additional Key Words and Phrases: Large language models, beam search tree, natural language generation, explainability, language transformers, visual analytics

ACM Reference Format:
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, Article 14 (June 2024), 32 pages. https://doi.org/10.1145/3652028

Authors' addresses: T. Spinner, R. Sevastjanova, and M. El-Assady, ETH Zurich, Zurich, Switzerland; e-mails: thilo.spinner@inf.ethz.ch, rita.sevastjanova@inf.ethz.ch, menna.elassady@ai.ethz.ch; R. Kehlbeck, T. Stahle, D. A. Keim, and O. Deussen, University of Konstanz, Konstanz, Germany; e-mails: rebecca.kehlbeck@uni-konstanz.de, tobias.staehle@uni-konstanz.de, keim@uni-konstanz.de, oliver.deussen@uni-konstanz.de.

ACM 2160-6455/2024/06-ART14
https://doi.org/10.1145/3652028

1 INTRODUCTION

Recently, **large language models (LLMs)** have gained increased popularity, especially in the field of **natural language generation (NLG)**. At the latest, with the introduction of ChatGPT,[1] LLMs have been made accessible to a wider, more general audience. However, despite their growing recognition and notable accomplishments, they still face several limitations. Common failures, even for state-of-the-art models, are repetitive content, the lack of factual accuracy, often referred to as hallucination [Ji et al. 2023], and biases [Alba 2022]. However, the perceived high quality of LLM outputs makes identifying errors in their predictions difficult, which is aggravated by a lack of explainability and accessibility [Zhao et al. 2024]. Gaining understanding and access to the model's decision-making process is fundamental for recognizing errors in their outputs, calming concerns about overestimating the model's capabilities, and empowering users to guide the model's predictions to align with their intentions. Particularly, the chat interface of ChatGPT and other chat- or completion-based approaches omit important information on uncertainties or viable alternatives from the users. While text-based interfaces may fulfill the needs for a broad, general audience, **interested non-experts** and **linguistic experts** require more in-depth insights and control.

We identify three primary shortcomings in the current state-of-the-art for interacting with LLMs: lack of **explainability**, **comparability**, and **adaptability**. Explainability refers to understanding of the model's decision process, including the way a language model predicts its output, its sampling strategy, and the probabilities of these outputs. For example, explanations of a prediction's certainty can provide the user a hint on possible hallucinations. Comparability, i.e., a simple yet effective comparison of multiple generated outputs, can enable the user to assess more specific nuances in the model's predictions. This kind of contrastive explanation [El-Assady et al. 2019] is particularly relevant for linguistic experts. For instance, by adapting prompts with typical names from varying ethnic groups and comparing the predictions, the user can assess the model's biases, if present. And last, adaptability is relevant when the generated output is not satisfactory. The insights gained from explainability and comparability empower the user to steer the model towards their intentions. Concretely, the user should be able to edit problematic parts; e.g., by correcting made-up facts and making these changes permanent; e.g., by fine-tuning the model.

Since almost all modern LLMs have committed themselves to the transformer architecture, besides their number of trainable parameters, the quality of the training data is the decisive factor for a model's performance [Lauscher et al. 2021; Mishra et al. 2022]. Therefore, studying the model's behavior is closely linked to studying its inputs and outputs, representing a local approximation of the information the model has learned during training. Our proposed approach, thus, focuses on making these inputs and outputs accessible and explorable to the user. A straightforward way to achieve this is to make the search algorithm transparent. The most prominent algorithm to sample sequences from the probability distributions output by the model is *beam search*. By sampling the *decision-space* [El-Assady et al. 2018] through expanding the most promising sequence in a limited set of candidate sequences, the algorithm results in a tree, scanning the search space for sequences with high overall probability. Beam search is thus commonly used in language model explanation methods, such as the visual interface by Lee et al. [2017], Seq2Seq-Vis [Strobelt et al. 2018], or GenNI [Strobelt et al. 2022].

In this article, we propose a *tree-in-the-loop* interaction paradigm, which leverages a visual representation of the ***beam search tree*** **(BST)** as the central component of the **generAItor** visual

[1] https://openai.com/blog/chatgpt

analytics technique. We reveal and explain the model's decision-making process by laying out the BST and augmenting it with additional explanations, such as token probabilities, semantic keyword coloring, and sentiment annotations. Comparative explanations are facilitated by juxtaposing multiple BSTs, allowing the user to compare the model's predictions under slightly varied inputs. Furthermore, we enable the user to interact with the tree, allowing them to adapt and steer the model's predictions, for example, by overriding model decisions, editing predicted sequences, or fine-tuning the model. To facilitate an effective analysis through visual interactive methods, we identify five main tasks in the context of informed text generation: model prompting and configuration, tree exploration and explainability, guided text generation, comparative analysis, and BST and model adaptation. Each of these tasks places distinct demands on the tools available.

To be able to fulfill these demands in a combined approach, we design **a modular, widget-based workflow**, where task-specific widgets enhance the BST with tailored controls, interaction possibilities, and visualizations. Each widget adds a very specific functionality. However, in symbiosis, a selected set of task-supporting widgets, in interaction with the search tree, enables novel, powerful modes of analysis; e.g., comparative analysis is facilitated by two particular widgets, allowing linguistic experts to observe changes in the tree under slight variations of the starting prompt. This reveals biases in the observed model, whose identification and mitigation is one of the most burning issues with state-of-the-art language models [Alba 2022].

In this article, we contribute: (1) A detailed problem analysis of the challenges of explainability, controllability, and adaptability in the context of various text generation tasks. (2) A novel visual analytics technique called generAItor, tackling these challenges in an interactive tree-in-the-loop-approach. (3) An implementation of the generAItor technique in a web-based visual analytics workspace. (4) A three-fold evaluation of the generAItor technique, including (4.1) case studies, showcasing the generative and comparative capabilities of our technique, (4.2) a qualitative user-study, proving the usability of the implementation, and (4.3) a quantitative evaluation, confirming the ability to adapt the model to user-preferences with few training samples.

2 RELATED WORK

In the following, we present our related work on language modeling, semantic similarity, controlled text generation, and bias analysis.

2.1 Language Modeling

Language models (LMs) are probability distributions over word sequences and a core component of **natural language processing (NLP)** systems [Bengio et al. 2000]. With the emergence of the transformer architecture [Vaswani et al. 2017], there was a paradigm shift away from recurrent neural networks [Rumelhart et al. 1986], since transformers allow parallel computations, speeding up training times, and prove superior in capturing long-term dependencies [Vaswani et al. 2017]. They use the attention mechanism [Bahdanau et al. 2014], which directs the focus on important tokens in the input sequence. Nowadays, numerous pre-trained transformer architectures are available for public use [Wolf et al. 2020]. There are different types of transformers, whereby the two main categories are masked language models and generative language models.

Masked LMs — BERT [Devlin et al. 2018] is a transformer-based LM that was trained on masked language modeling (i.e., *cloze*) and next-sentence prediction tasks and is commonly fine-tuned for diverse text classification tasks [Howard and Ruder 2018]. Due to its pre-training objective, BERT (as well as other masked language models) is not suitable for text generation tasks. We use BERT for masked word prediction in the *ontological replace* functionality  .

Generative LMs — Text can be generated using generative transformer models, such as GPT-2 [Radford et al. 2019b], GPT-3 [Brown et al. 2020], or GPT-4 [OpenAI 2023]. These are autoregressive models that were pre-trained on the causal language modeling task, learning to predict the next word in the input sequence. For a broader overview, see the survey on pre-trained language models for text generation by Li et al. [2021]. In our work, we use GPT-2 and Bloom [Scao et al. 2023] for text generation; however, the approach is designed to support other transformer-based LMs as well.

2.2 Semantic Similarity

Word Taxonomies and Ontologies — Leveraging semantic graphs and knowledge bases, such as YAGO and DBpedia, it is possible to infer concept or topic hierarchies via language models [Chen et al. 2021; Huang et al. 2020; Zhang et al. 2018] or expand existing taxonomies [Jiang et al. 2022; Xu et al. 2022]. Methods such as OntoEA [Xiang et al. 2021] align entities by jointly embedding ontologies and knowledge bases. Taxonomies can be used to improve recommender systems [Tan et al. 2022] and help with entity recognition [Li et al. 2022] or translation [Li et al. 2022]. WordNet information can be integrated into pre-trained language models for improved sense disambiguation, e.g., ARES [Scarlini et al. 2020], or used to build human-readable concept vectors [Conia and Navigli 2020]. For our method, we use ARES and BERT embeddings in conjunction to create domain-specific predictions with an ontology graph created from the BabelNet [Navigli and Ponzetto 2012] semantic graph.

Embedding Similarity — In language models, each token of the input text is mapped to a high-dimensional vector. Related work has shown that these context-dependent embeddings encode different context/language properties. Although BERT is the most widely analyzed language model so far [Rogers et al. 2020], other transformer models, such as GPT-2, and their produced embedding spaces have also attracted computational linguistics' and visual analytics researchers' attention [Ethayarajh 2019; Sevastjanova et al. 2022]. Prior research has shown that semantic information, such as word senses and semantic roles, is captured best in the higher layers of transformer models [Reif et al. 2019; Sevastjanova et al. 2022; Wiedemann et al. 2019]. Thus, these contextualized embeddings are commonly used as features for semantic similarity tasks. In our work, we apply a dimensionality reduction technique on embeddings extracted from the used LMs to map the tokens to unique colors based on their coordinates in the two-dimensional space. With this approach, tokens with a semantic similarity get assigned to similar colors [El-Assady et al. 2022].

2.3 Controlled Text Generation

Algorithmic Approaches — In general, controlling the style and information of natural language generation is one of the applications identified by Gatt and Krahmer [2018]. One challenge of integrating knowledge into text generation is the automatic steering of the generation in a particular direction. Using plug-and-play language models is one possibility to steer text generation [Qin et al. 2020]. Concerning pre-trained language models, it is possible to control, e.g., the sentiment [Dathathri et al. 2019; Hu et al. 2017], keywords [He 2021], or the topic [Dathathri et al. 2019]. Frameworks such as FAIR [Hua and Wang 2020] allow the generation of content-controlled text by combining BERT with BART [Lewis et al. 2020]. A larger overview is given in the survey by Zhang et al. [2022]. Building on this, many approaches now integrate external resources such as knowledge bases. More details can be found in the survey by Yu et al. [2022]. However, these techniques do not allow immediate intervention in the decision process, which we specifically target with our approach.

Visual Interactive Approaches — Focusing on interactive editing, Du et al. [2022] provide interactive suggestions in their tool to achieve high-quality text edits with minimal human effort. Padmakumar and He [2022] use a human-in-the-loop approach to replace text segments for the task of creative image captioning. Gehrmann et al. [2019] propose an interactive framework that allows users to control generative segments through a process called collaborative semantic inference. Following this, Strobelt et al. [2022] create GenNi, an interface for collaborative text generation. They guide the model output using explicitly defined constraints. The user has to know beforehand how he wants to control the model output, as it is not possible to adapt the state during inference. With Wordcraft, Yuan et al. [2022] present an interactive interface that allows writers to create stories with the assistance of large language models. Their system lets authors re-write, replace, and auto-generate text, as well as define custom requests to the language model. In contrast, our approach enables direct interaction with the model's outputs by exposing predictions and probabilities in the beam search tree.

2.4 Bias Analysis

Current research explores not only what the models learn but also when they fail and which limitations they have, such as different types of biases [Garrido-Muñoz et al. 2021]. For instance, Blodgett et al. [2020] present a taxonomy for fairness definitions that machine learning researchers have defined to avoid existing bias in AI systems. Mehrabi et al. [2021] define the bias problem specifically in language modeling tasks in a formal way and explore how it has been treated in related work regarding their detection and correction.

In masked language models, the detection of bias is typically done by applying templates or predefined word lists. For instance, the **Word Embedding Association Test (WEAT)** [Caliskan et al. 2017] measures the association between two target word sets (e.g., male pronouns and, e.g., female pronouns) based on their cosine similarity to words from two attribute sets (e.g., terms related to science or art) to make conclusions about encoded biases. Liang et al. [2021] show that the analysis of biases in text generation can be more nuanced, e.g., biases can arise during the generation of any token [Nadeem et al. 2021]. Alnegheimish et al. [2022] find that bias "evaluations are very sensitive to the design choices of template prompts." According to the authors, the use of template-based prompts tends to evoke biases from the model's default behavior rather than reflecting the actual correlation between gender and profession, analyzed in their work. Thus, we propose 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 model's predictions. For a detailed case study, showcasing the benefits of our comparative approach, see Section 6.1.

3 PROBLEM CHARACTERIZATION

With recent advances in language generation and the release of ChatGPT, language models have made their way into mainstream use. While automatic text generation through language models can support the author through corrections, suggestions, or chat-based question answering, understanding of the model's capabilities and limitations and access to its predictions is still limited. However, such understanding and access are crucial for raising awareness of dangers (e.g., biased outputs, hallucinations), allaying fears of its potential (e.g., overestimation of a model's capabilities), and enabling users to steer the model's predictions towards their intention (e.g., by selecting or modifying outputs).

While the average user might not be willing to invest time and effort in investigating the behavior of language models, we identify two primary user groups with different interests and requirements for language model analysis. We define *non-experts* Non as interest-driven persons with an affinity for technical advancements and the wish to explore modern language models.

The term "non-expert" only refers to the user's experiences with large language models and their background in computational linguistics; they can still be domain experts in other fields. Examples could be a journalist who writes about language models and wants to understand their capabilities and limitations or a writer who wants to use LLMs to generate text with a specific style or topic. Analogously, we define *linguistic experts* Lin as users working in (computational) linguistics, with a main focus on the analysis of model behavior. An example could be a linguist who wants to observe biases encoded in the model [Spinner et al. 2023]. Our approach is targeted towards both user groups, with shifting focus on the tasks our system supports. For the non-experts, understanding of the model's capabilities, exploration of outputs, investigation of uncertainties, and the ability to adapt model outputs are primarily important. In contrast, the linguistic expert is interested in the close analysis of model outputs, e.g., to observe learned syntactic and semantic structures, identify model defects, or assess model biases. In the following, we specify the challenges and tasks for the derived user groups.

3.1 Challenges

The challenges are derived from research gaps in related work and from discussions with non-experts Non, machine learning experts, and computational linguists Lin.

Explainability Ex — Despite the impressive performance of state-of-the-art language models, their predictions are often underexplained, as deep-learning-based models are typically black boxes, making explainability a major challenge [Danilevsky et al. 2020]. However, language models have the advantage of interpretable inputs and outputs (namely: text) and easy-to-understand prediction mechanisms, which we aim to leverage to solve this challenge. We identify two primary aspects of explainability regarding language models: model and output explainability. Explainability is important for both the non-expert Non and the linguistic expert Lin.

Model explainability relates to explanations of the model's algorithmic approach, such as providing information on the model's architecture, the used search algorithm, or the influence of randomness (cf., reproducibility) [Spinner et al. 2020]. Particularly, mainstream media often fail to explain the primary mechanism behind LLMs: predicting the likelihood of tokens to follow a sequence of previous tokens. Although some articles briefly touch the topic [Metz 2022; Roose 2023], there is much misinformation through excessive abstraction and a lack of easy-to-follow visualizations and interactive systems that could impart a thorough understanding to non-experts. Understanding this mechanism is crucial to raising awareness of a model's limitations and allaying fears of its potential. *Output explainability* refers to explanations of the model's token representations and output probabilities, such as token embedding similarity or output certainty.

Comparability Com — The ability to explore the space of possible model outputs is vast and currently underexplored [Alnegheimish et al. 2022]. For the analysis, instance-based comparability of generated outputs is essential for linguistics, e.g., for bias analysis or hypothesis generation. Particularly, non-template based, explorative analysis enables hypotheses generation and inductive learning [Sternberg and Sternberg 2016].

Adaptability Ada — Even state-of-the-art language models often fail to produce output that aligns with human intentions and sticks to facts [Ji et al. 2023; LeCun 2023]. Therefore, adaptability is essential to employ language models in real-world scenarios. Again, we differentiate two sub-aspects: output adaptability and model adaptability. *Output adaptation* refers to direct edits of the model's predictions, e.g., to correct hallucinated facts, re-prime the model through entering custom text, or select from alternative outputs, targeting both the non-expert Non and linguistic expert Lin. That followed, *model adaptation* relates to model fine-tuning with the edited data to make changes permanent for future sessions, which is also relevant for both user groups.

3.2 The Tree-in-the-loop Approach

To address the challenges identified above, we propose the *tree-in-the-loop* paradigm, a novel approach to interactively explore and adapt the predictions of language models through the visualization of the beam search tree.

With the invention of transformers, the architecture of state-of-the-art models is well established, shifting the focus for performance improvements on the training process and the quality of training data [Ouyang et al. 2022]. Consequently, understanding a model's behavior involves examining its inputs and outputs, which reflect the "knowledge" it has acquired during training. Therefore, our approach emphasizes making these inputs and outputs more user-accessible and explorable.

In each step, when predicting the next token for a given input sequence, the model outputs a probability distribution over all known tokens. The final text has to be constructed by sampling from this probability distribution. A common heuristic to choose the output with the highest probability is beam search. Beam search is a greedy search algorithm that expands the k most likely sequences in each step, resulting in a tree with k nodes in each tree level. k is called the *beam width*. Branches with low overall probability stall in this process, resulting in a tree with varying depth. The deepest leaf node with the highest probability is then chosen as the final output. Often, additional parameters are used to increase the diversity of the generated text, e.g., by penalizing the repetition of n-grams or by adding randomness to the sampling process, e.g., through top-k sampling or temperature scaling.

Most interfaces only present the user with the final text, discarding all information about the sampling process, such as uncertainties of predictions, alternative outputs, or the influence of parameters such as the beam width or an n-gram penalty. To enable an understanding of the model's prediction process, we aim to make this information accessible to the user. This is most straightforwardly done by visualizing the beam search tree, which is easy to understand and interact with. Furthermore, it provides a direct representation of the underlying sampling algorithm and thus does neither neglect information nor introduce false rationalization.

The tree-in-the-loop approach is the extension of the beam search tree with additional augmentations, visualizations, and interaction possibilities. This makes the tree accessible to non-technical users Non and supports linguistic experts Lin in the advanced analysis of linguistic phenomena.

3.3 User Tasks

From the before-discussed challenges of explainability, adaptability, and comparability, we derive the following user tasks, as depicted in Figure 1. While some tasks are essential to load and interact with LLMs, others are optional and only relevant for specific use cases.

Model Prompting and Configuration — To choose and assess models from the large variety of pre-trained LLMs [Wolf et al. 2020], the user has to be able to load different models. Furthermore, the user should be able to provide a prompt to the model and configure parameters for the prediction algorithm. After interactively editing outputs and, potentially, fine-tuning the model, the user should be able to save the refined sequences and model for future sessions. Since these tasks describe basic interactions with the model, they are equally important for the linguistic expert Lin and the non-technical user Non.

T0 Load and assess (pre-trained) models, provide prompts, and configure parameters for the prediction algorithm. Save trees and models for future sessions.

Tree Exploration & Explainability — The beam search tree, used to sample model outputs, should be transparent and accessible to the user, allowing them to explore alternatives and assess

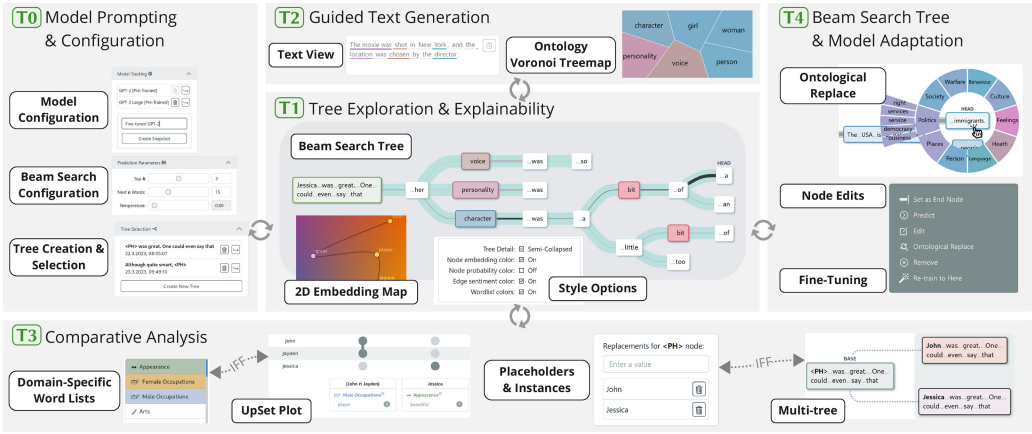


Fig. 1. The five main tasks of interactive text generation as supported by generAItor (see Section 3.3). The beam search tree is the key element (see Section 4), facilitating visualization and interaction with the model's decisions. Each task has a set of widgets associated (see Section 5), providing task-specific visualizations, controls, and interaction possibilities. Following our proposed *tree-in-the-loop paradigm*, the tasks are interwoven and can be combined in an iterative process, centered around the beam search tree.

the certainty of the model's predictions, addressing the explainability challenge Ex. Supporting beam search exploration, semantic annotations of the tree should be provided, e.g., to identify topic similarity or to discover undesired patterns like looping structures. This is important for both the non-expert Non and for the linguistic expert Lin, who are interested in the close analysis of model outputs and need a higher-level overview to cover large trees.

T1 Assess probabilities and explore alternative branches in the beam search tree. Identify topic similarity and undesired patterns, such as looping structures.

Guided Text Generation — Using the start prompt or existing sequences from the tree, the user should be able to query the LLM to extend the beam search tree with new predictions. Since the beam search tree might grow to a significant size, a text view should be provided to close-read generated text and navigate the beam search tree to a local context. Also, for longer texts, an overview of the topics touched facilitates an overview and understanding of the generated text. This task mainly targets the non-expert Non, who is likely to generate longer texts.

T2 Query the LLM to extend the beam search tree. Navigate the beam search tree to a local context. Investigate the topics touched by the generated text and stalled beam search branches.

Comparative Analysis — Comparative analysis tackles the comparability challenge Com and is particularly important for the linguistic expert Lin, who is interested in the close analysis of model outputs. Different trees can be generated and compared by varying start prompt and beam search parameters, allowing to assess the effects of those changes. Semantic annotations and aggregated representations should be provided to quickly identify the key differences between trees. This facilitates, e.g., generating new hypotheses, analyzing model biases, or investigating the influence of function words on the predictions.

T3 Generate and compare different trees by varying prompt and beam search parameters. Observe syntactic and semantic differences in the trees.

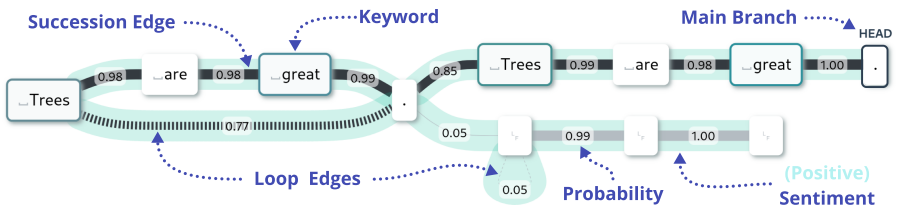


Fig. 2. The beam search tree visualization. Edge width and label encode the probability of a node to follow its predecessor. The leaf node of the beam with the highest overall probability is marked as HEAD. Keywords are highlighted using semantic colors. The branch color encodes the sentiment of the sequence up to a node.

BST Adjustment & Model Adaptation — Enabling adaptation to domain and personal user preferences, it should be possible to edit the generated text. This can either happen by direct text edits, choosing from a set of alternatives, or pruning unwanted branches of the beam search tree. After editing the tree, the user should be able to fine-tune the model with the edited sequences to align future predictions with the user's preferences. Both addresses the adaptability challenge Ada. This task is important for non-experts Non who need domain adaptation or for linguistic experts Lin who want to observe the influence of such adaptation on the LLMs' predictions.

T4	Interactively edit or replace produced sequences to adapt the text to personal preferences and domains. Fine-tune the model with the edited sequences.

4 TREE VISUALIZATION & MODEL CONFIGURATION

The beam search tree is central to our generAItor technique, therefore being the main component visible throughout the analysis. In this section, we describe the visual representation of the tree, how it is augmented with information, how the user navigates the tree to a local context and extends the tree with new predictions, and how the interaction with tree nodes is implemented. By augmenting the tree with task-specific widgets W, we provide tailored controls, visualizations, and interactions, supporting model prompting and configuration T0 and tree exploration and explainability T1.

4.1 Beam Search Tree

Our technique is based on a visual representation of the beam search tree as the key analysis component, establishing the tree-in-the-loop approach. It is used to sample the final output sequence from the token probabilities in each prediction step. In the tree visualization, nodes encode sequences and edges their order, as depicted in Figure 2. The tree is laid out from left to right, starting either with the initial prompt used during tree creation or an arbitrary tree node that is set by the user when only a subtree should be inspected. Edge width and -label encode the nodes' probability of following its predecessor. We mark the leaf node of the beam with the highest probability as HEAD node, which, when not configured otherwise, is the one defining the final text output. When rendering the text associated with the tree nodes, we replace invisible or control characters with visible proxies, e.g., white spaces with ␣ and newlines with ↳. The tree visualization imparts the uncertainty of tokens and sequences and lets the user explore next-likely alternatives in the form of stalled branches T1.

To extend the tree, the user can either trigger a beam search run from the HEAD node or start auto-prediction, which iteratively extends the tree at the HEAD node until stopped.

Loop Detection — We automatically detect repeating node sequences in the tree and denote them with a dotted edge, as shown in Figure 2. This allows the user to quickly identify repeating patterns,

which are often unwanted model defects, telling linguistic experts about the model's limitations or probably miss-chosen search parameters [von Platen 2020].

Keyword Highlights — We extract and highlight keywords from the sequences in the tree, allowing users to intuitively distinguish less-important nodes, e.g., stop words, from meaningful nodes, e.g., proper nouns T1. As shown in Figure 2, we color the keyword nodes in the tree visualization according to their semantic embeddings [El-Assady et al. 2022], enabling a quick impression of the semantic similarity between the concepts present in the tree. Furthermore, it allows identifying concept drift by revealing changing concepts as color shifts in the tree visualization.

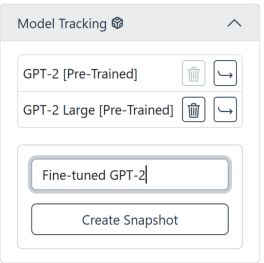
Sentiment Highlights — Facilitating visual perception of the sentiment of tree branches, we color the edges in the tree visualization according to the sentiment of the sequence up to the edge's target node, as shown in Figure 2. The sentiment is estimated by applying a three-class RoBERTa-based sentiment classifier, which was trained on social media posts [Hartmann et al. 2021].

4.2 Model Prompting and Configuration T0

Tree Creation and Selection W — The tree selection panel (in Figure 4) allows loading existing trees into the workspace and creating new ones. When creating a new tree, the user is prompted for a starting sequence, which is used as the initial input sequence passed to the model. The starting sequence also forms the root node of the tree.

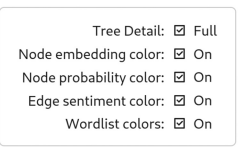
Prediction Parameters W — The prediction parameters panel (in Figure 4) allows the user to specify the parameters used when executing a beam search step. The parameter "top-k" specifies the number of samples drawn in each beam search iteration, either by selecting the k most probable tokens or—if temperature is enabled—by sampling from the model's output distribution. The length of the beam search can be specified by the parameter "next n words." Finally, the parameter "temperature" allows controlling the randomness of the model's output distribution. A temperature value of zero disables temperature and selects the top-k most probable tokens in each beam search iteration.

Model Snapshots and Tracking W — The model tracking panel allows the user to load different pre-trained models, e.g., from HuggingFace [Wolf et al. 2020]. Out of the box, generAItor provides access to GPT-2 Base, GPT-2 Large [Radford et al. 2019a], and Bloom [Scao et al. 2023], but other, transformer-based models can easily be added. More specifically, our approach is model (transformer) agnostic; only the embedding projection (cf., W) has to be re-computed for new model variants. Besides loading pre-trained models, the model tracking panel also allows the user to create snapshots of adapted models T3. By creating a snapshot of the current model state, the user can easily restore this state later, e.g., if the model was fine-tuned to a point where it no longer generates meaningful outputs.



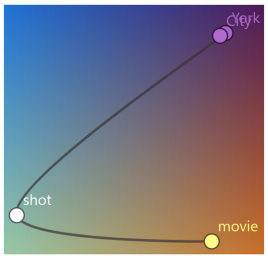
4.3 Tree Exploration and Explainability T1

Tree Style Toggles W — The beam search tree is augmented with color information and can be visualized in different levels of detail. Particularly, the edges can be colored by sequence sentiment, the nodes' fill color can be set based on their semantic embedding color, the nodes' stroke can be set to represent their token probability, and word lists (see W) can be colored by a categorical color scale. Furthermore, the tree's level of detail can be switched between *Full*, showing all node texts and using full node spacings; *Collapsed*, hiding all node texts



and only showing the tree's structure with minimal spacings; and *Semi-Collapsed*, only showing the node text for nodes occurring in active word lists (see Figure 6).

2D Embedding Map W🗺 — The 2D embedding map (🗺 in Figure 4) shows an image of the currently selected two-dimensional *semantic color map* [El-Assady et al. 2022], used to color the keywords in the tree visualization. By overlaying the color map image with the keywords, we enable users to explore how the keywords are distributed in the high-dimensional space. The position of keywords on the colormap is computed by a two-dimensional UMAP [McInnes et al. 2018] projection, which we priorly anchored on the keywords extracted from 150k sentence pairs in the MultiNLI dataset [Williams et al. 2018]. This allows the detection of semantic similarity between keywords and the identification of the major concepts present in the tree. By hovering a beam search branch, the user can filter the keywords visible on the embedding map to only show the keywords of the hovered branch. Furthermore, hovering renders a path connecting the keywords according to their occurrence in the branch. This sequence projection builds intuitive pictures of the sequence, allowing to compare sentence structures and the mentioned concepts. Different two-dimensional color maps can be chosen in a dropdown menu in the 2D embedding map panel. The side figure shows the beam sequence "The **movie** was **shot** in New **York City**" on the "Teuling 2" color map [Teuling et al. 2010].



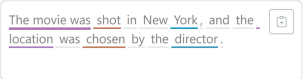
5 TEXT GENERATION, COMPARISON, & MODEL ADAPTATION

Besides the default widgets to configure models, specify parameters, prompt the model, and explain the beam search tree, we provide additional widgets that are tailored to a specific task mode. We distinguish between two main modes: controlled text generation (Section 5.1) and comparative analysis (Section 5.2). Each mode has a dedicated set of widgets enabled by default. They enhance existing functionalities with additional on-demand information, allow additional interactions, or enable specific modes of analysis. The widgets are designed as modular components that can be enabled/disabled and moved around the workspace to support the user's workflow.

5.1 Text Generation T2 and BST Adaptation T4

Guided text generation provides tools to support the user in the informed generation of text, particularly to close-read generated text, navigate the beam search tree, and select desired sequences. Furthermore, it provides content summarization in the form of an ontology Voronoi treemap, which can be used to detect concepts in the produced text and to identify semantic differences across nodes with the same keywords.

5.1.1 Widgets Supporting Guided Text Generation. **Text View** W📄 — While the beam search tree visualization supports understanding, exploration, and interaction on a highly detailed level, it is hard to read the final output text from only observing beams and nodes. Therefore, a text output panel displays the full sequence of the main branch, which in turn is highlighted in gray in the tree visualization. To retain the membership of each node and its corresponding embedding and keyword information, the node sequences are slightly spaced in the text view and underlined with their keyword embedding color. The more compressed representation in the text view, together with the ability to overflow the text container using scrollbars, allows to always display the full text starting at the root node. We use this advantage of the text view to allow tree filtering: By opening the context

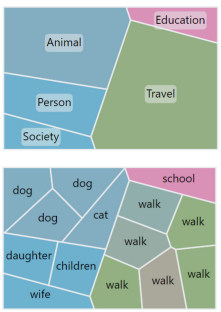


menu on a text node, the node can be set as start node (|➖). This filters the displayed beam search tree to the descendants of the selected node, allowing local exploration and preventing information overload on large trees. In return, leaf nodes can be set as end node (➖|) in case a branch different from the one with the highest beam probability contains the preferred output text. A copy button facilitates copying the generated output text to the clipboard.

Node Context Menu W — The nodes in the beam search tree offer a feature-rich context menu, shown in the middle-right of Figure 1. In the following, we describe the functionality of the context menu entries that are not covered by their respective workspace subsection.

- **Edit / ⊗ Remove.** The *edit* entry allows altering the text of the selected node manually. When selecting it, the node changes into an input field, where the user can manually enter the desired text. After finishing the edit, the node changes back into normal mode, and the node is updated in the beam search tree, including its keyword information and embeddings. The *remove* entry allows removing the selected node and all its descendants from the tree.
- **Predict.** Alternative to predicting at the current HEAD node, the user can also predict from any node in the tree by selecting the *predict* entry from the context menu. The parameters are specified in the prediction parameters panel.
- **Ontological Replace.** Based on information extracted from an underlying ontology graph and the usage of a masked language model, the *ontological replace* entry provides alternative suggestions to replace the selected node with.
- **Re-train to Here.** The *re-train to here* entry allows fine-tuning the model with the beam sequence up to the selected node, addressing task T4. Without further user input, fine-tuning is executed instantly in the background when the button is clicked, abstracting the underlying complex process and maximizing simplicity for the user.

Ontology Voronoi Treemap W — Through an underlying ontology graph, we provide a Voronoi treemap visualization to support the user in getting an overview of the concepts closely linked to the keywords present in the tree. The extracted keywords from the beam search tree are attached to nodes in the ontology hierarchy of BabelNet [Navigli and Ponzetto 2012]. We grow a subsumption hierarchy from these keywords, whose nodes become more and more general. Finally, nodes are connected to their respective subdomains and domains (e.g., *dog* → *Animal* → *BIOLOGY*). Although the whole ontology graph allows an in-depth view of the subsumption hierarchy, the readability of the graph worsens as the number of leaf nodes increases. Instead, we utilize a Voronoi treemap visualization, allowing the user to view the hierarchy in four predefined layers: domains, subdomains, synsets, and keyword instances. Domains and subdomains provide an overview of the concepts in the beam search tree. Synsets aggregate similar keywords. The keyword instance layer shows all keywords. Keywords can appear multiple times in this layer, as one keyword can appear at different positions in the beam search tree. Because the surrounding context of a keyword differs for each node, their embeddings differ, resulting in different colors, e.g., the keyword "walk." To allow the user to investigate this further, hovering over a cell of the Voronoi treemap highlights the respective nodes in the beam search tree, enabling them to inspect the keywords in their context.



Ontological Replace W — Using our tool, text generated by the model can be adapted to the user's preferences by selecting branches or editing model outputs. However, sometimes, the predictions from the model are not what the user has in mind. We offer an alternative way of adapting the model tree using domain-specific, context-sensitive alternatives. If the user is unsure about a suitable replacement word and requires guidance, then he can use the ontological replace function.

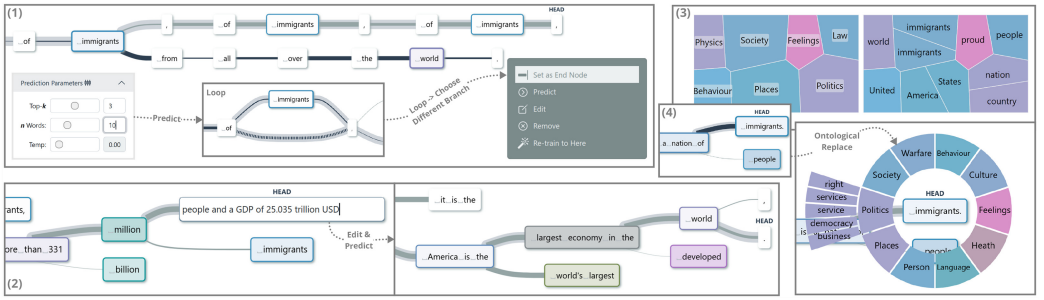
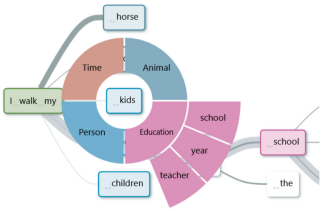


Fig. 3. Text generation workflow as described in Section 5.1.2. (1) After creating a new tree and predicting with the set parameters, the model runs into a loop. By choosing a different branch, this issue can be resolved. (2) By manually editing nodes, factual knowledge can be incorporated into the text. (3) The ontology tree gives an overview of concepts connected to the generated text. (4) Ontological replacements suggest alternatives.



With the information currently in the ontology graph, it is possible to generate predictions for a specific node and group them by domain. These domain predictions can be from the current domains in the beam search tree, or the user can manually add domains from a predetermined selection. The domains and their respective suggestions are words that the language model might not have suggested in its top-k prediction, making it an intermediate mode between manual editing and automatic prediction of the model, even allowing out-of-distribution suggestions. Extensive implementation details, including figures of the underlying NLP pipelines, can be found in Appendix A.

5.1.2 Workflow Demonstration: Text Generation. The following exemplary workflow showcases how our approach is used to generate and adapt text. To demonstrate, we utilize GPT-2 Base[2] [Radford et al. 2019a] as the language model. Note that the sequences presented in this example do not represent the quality of SOTA language models. Nevertheless, GPT-2 Base is well suited to showcase larger models' deficiencies (e.g., repetitions, hallucination) in brief examples. Since our approach is model-agnostic, other LMs can be loaded instead.

A newspaper author wants to write a short but informative article on the United States of America (USA). As a basis, he uses a fact sheet containing information on population, geography, and so on, of the USA. In the generAItor workspace, he creates and loads a new tree (🌳) with the starting sequence "*The United States of America*" (Figure 3(1)). After setting the beam search parameters (🎚) to $k = 3$ and $n = 10$, he starts predicting at the head node. After two beam steps, the branch with the highest probability gets stuck in a loop: "*The United States of America is a nation of immigrants, of immigrants, of immigrants, of immigrants.*" However, by manually selecting (➖) the second-best scoring branch, he can steer the output to be more intelligible: "*The United States of America is a nation of immigrants, of immigrants from all over the globe.*" Accepting this output as the starting sequence, he hides earlier parts of the tree (|➖) and executes further prediction steps (⏭). At points where the model is stuck or factual information should be integrated into the article, he uses manual node edits (📝) to set a new baseline or enter numbers from the fact sheet (Figure 3(2)); e.g., he

[2]https://huggingface.co/gpt2

changes the hallucinated prediction "*With more than* 1.5 *million people*" to "*With more than* ***331*** *million people* ***and a GDP of 25.035 trillion USD***," leading to the prediction "*…, America is the largest economy in the world.*" By repeating this process, the author compiles a diverting article. Observing the ontology Voronoi treemap (), he can check on the major concepts covered by his article, which after a while include Society, Politics, Places, and Feelings, leaving him satisfied with the diversity of his text (Figure 3(3)). After a while, the model again predicts "*The USA is a nation of immigrants.*" The author decides to use the ontological replace function (), which suggests multiple domains, including "Person," "Society," and "Politics" (Figure 3(4)). From the political domain, various replacements sound promising. The author chooses the suggestion "democracy." He concludes the article with: "*The USA is a nation of democracy.*" The author is satisfied with the result and decides to re-train the model to the tree's current state (). This way, the model can be adapted to the author's writing style and domain-specific vocabulary, helping to generate more coherent text in the future.

5.2 Comparative Analysis T3

The user can enter the comparative analysis by inserting a placeholder string into a tree's input prompt. It automatically replaces the placeholder with user-selected string instances and creates a new tree for each instance, displayed as alternatives in the workspace. The comparative mode allows for assessing nuances in the model's predictions based on input variations, e.g., for bias detection. The case study on comparative analysis in Section 6.1 gives several examples on how the comparative mode can be used to generate different hypotheses and evaluate biases in model predictions.

5.2.1 Widgets Supporting Comparative Analysis. **Template Node & Multi-Tree** — The comparative mode is entered by creating a tree with the placeholder **<PH>** in the starting sequence, facilitating comparison over trees with slightly varying starting sequences. When loading such a tree into the workspace, the template sequence is shown as the base node (1.a in Figure 4). The user can now create a list of replacements for the placeholder (1.b in Figure 4). For each replacement, a new tree is instantiated, and beam search is executed using the prediction parameters configured by the user. To ensure determinism, temperature sampling is disabled in comparative mode. The instances are displayed vertically stacked, with the replacement highlighted in the root node of each tree (1.c in Figure 4).

Domain-specific Word Lists — The user can select domain-specific word lists to enable targeted comparison between the tree instances (2.a in Figure 4). Tree nodes containing a word from the selected word lists are highlighted in the tree with a badge, denoting its associated list (2.b in Figure 4). This makes it easy to spot differences and commonalities between the trees, e.g., to detect gender bias between male and female person names (for exhaustive examples, see Section 6.1). The user can either choose from a set of pre-defined word lists from different domains [Deep NLP 2023], covering typical bias analysis tasks, such as Male/Female Occupations, Appearance, and Negative/Positive Characteristics, or upload their own word lists.

For keyword-based analysis in trees of increasing size, we include a *semi-collapsed tree view*, activatable in the tree style toggles and shown in Figure 6. It only expands the nodes matching to at least one of the selected word lists, preserving the tree structure and allowing to easily compare across word domains.

UpSet Plot — Visual comparison between tree instances is facilitated by the domain-specific word lists, semantic embeddings, and the possibility to semi-collapse the tree. However, if high values for the prediction parameters k and n are chosen, then the tree can grow large. Therefore, we

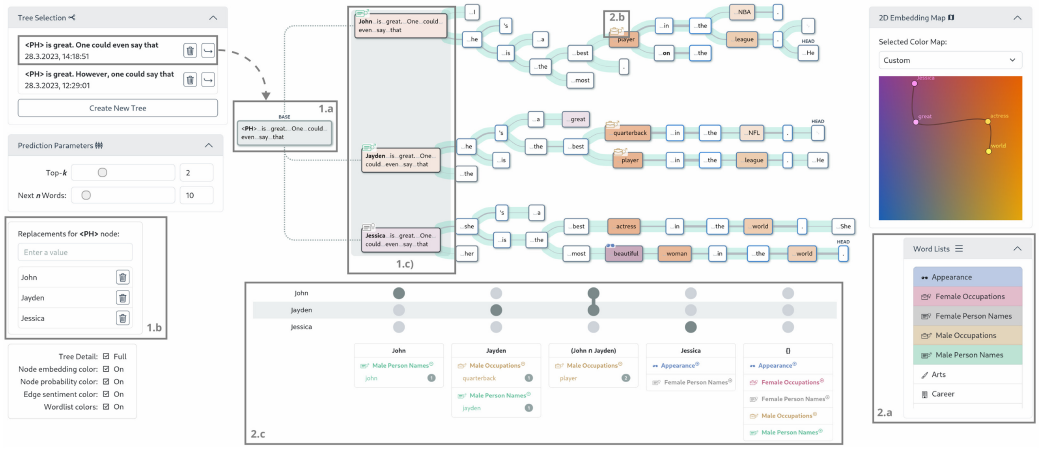


Fig. 4. The generAItor workspace in comparative analysis mode with the associated widgets opened. The tree visualization as the central element shows alternative beam search results under different replacements of the **<PH>** node. Words occurring in one of the selected word lists are highlighted in the tree. The Upset plot shows the overlap of the selected word lists in the alternative trees. The edges of the tree are colored based on sentiment analysis, with red indicating negative sentiment and green indicating positive sentiment.

offer an alternative summarization view of the relations between occurrences of words from the word lists and the template replacements. We use UpSet [Lex et al. 2014] plots for this, a visualization technique showing overlaps between set-typed data (2.c in Figure 4). Particularly, we visually highlight which tree instances have overlapping words and, in consequence, also overlapping word lists. Each row represents one set, in our case, one tree instance. Tree instances that have the same overlap are shown as one column in the UpSet plot, with gray connected nodes. This column is one set intersection, and the nodes that participate in this intersection are shown as a joined list. Underneath the UpSet plot, we show the currently selected word lists that are part of the set intersection and list the specific words that appear in the tree along with the overall count of these words. This allows users to get a quick overview of which tree instances have similar predicted words grouped by their word lists; e.g., the user can investigate the prediction tree of female names containing female-connoted occupations vs. the prediction tree of male names containing male-connoted occupations.

5.2.2 Workflow Demonstration: Comparative Analysis. The following exemplary workflow showcases how our workspace supports comparative analysis:

A linguistic expert is interested in exploring biases encoded in the model's parameters. He thus creates a prompt "*<PH> is great. One could even say that*" as shown in Figure 4. The placeholder <PH> WPH includes words such as *John*, *Jayden*, and *Jessica*. The beam search tree represents the top two predictions for each starting sequence. The expert then selects multiple word lists to highlight the occurrences of words related to appearance, person names, and occupations. These get marked in the tree visualization through icons attached to the particular tree nodes. The UpSet plot summarizes the word occurrences showing that the female person name *Jessica* is related to the appearance word *beautiful*; the two male person names are mentioned as players of sports games (i.e., *player*, *quarterback*), confirming the stereotypical gender biases encoded in the language model [Lu et al. 2020]. The case study in Section 6.1 describes more details on the workflow.

5.3 Model Adaptation T4

After adapting the beam search tree as part of tasks T2 and T4 or after identifying desired sequences as part of tasks T1 and T3, the user might want to feed those changes back and fine-tune the model, accordingly. This can be done by executing the *re-train to here* (🔨) functionality from the node context menu W. This triggers a fine-tuning step of the model in the backend, using the beam sequence up to the selected node as input. The current model state can be saved at any time using the model snapshots and tracking widget W, enabling the user to restore fine-tuned models from previous sessions or discard potentially overfitted models by returning to an earlier state.

Section 6.3 provides an extensive evaluation of the fine-tuning functionality. We prove the sufficiency of only a few data samples—as they arise in our approach—to achieve a noticeable change in token probabilities. Also, we show that over repeated fine-tuning with different sequences during the analysis session, domain adaptation is achieved.

6 EVALUATION

This section provides a three-fold evaluation of our approach. Starting with a case study on comparative analysis T3 in Section 6.1, we showcase how our tool is used to gain in-depth linguistic insights on biases encoded in the model. It shows how our tree-in-the-loop technique goes beyond the template-based state-of-the-art in bias analysis. In Section 6.2, we provide two qualitative user studies with six non-experts Non and four computational linguists Lin, showcasing the usability of our tool for guided text generation T2 and comparative linguistic analyses T3, respectively. Finally, Section 6.3 presents a detailed evaluation of the ability to fine-tune LLMs T4 using the relatively small sample size of training data arising in our approach, showing that domain adaptation indeed is possible in the described scenarios. Moreover, in our work "Revealing the Unwritten" [Spinner et al. 2023], we present additionally insights into state-of-the-art linguistic challenges, created with the generaitor interface.

6.1 Case Study: Comparative Analysis on Social Biases

In this case study, a linguistic expert Lin aims to learn patterns relevant to designing bias evaluation methods. Since the bias evaluations for generative language models are sensitive to the design choices of template prompts [Alnegheimish et al. 2022], the expert's goal is to find out interesting linguistic structures that should be taken into account during systematic bias analysis. He thus uses the generAItor workspace to explore different examples[3] and generate new linguistic hypotheses (cf., inductive learning [Sternberg and Sternberg 2016]).

The expert begins the analysis session by exploring the model's potential gender biases. For this purpose, he creates a prompt "*After receiving their degree, <PH> wants to become*" whereby the <PH> W PH stands for a placeholder of different female and male person names. The predictions for *John* and *Jessica* are listed in Table 1. The expert can confirm findings from related work [Lu et al. 2020] showing that language models tend to learn stereotypical gender-profession associations, such as *John* is more likely to become a *lawyer* and *Jessica* is more likely to become a *nurse*. Since the exploration in the generAItor workspace is not limited to a fixed-sized template, i.e., the generated token sequences can be of any length, the expert observes that the stereotypical associations are followed by the person's doubts regarding his or her chosen profession (see Table 1). This motivates the expert to explore an additional prompt, i.e., "*The reason <PH> did not become a doctor was*." The model's output shows a new perspective of gender bias, i.e., the model's assumptions

[3]We showcase these examples in a reduced online demo of generAItor, available under https://demo.tree.generaitor.dbvis.de

Table 1. Example Sequences Generated in the Comparative Mode of generAItor by Instancing the **<PH>** Node

Prompt	Prediction
After receiving their degree, **<PH>** wants to become	After receiving their degree, **John** wants to become a lawyer. He's not sure if he'll be able to afford it.
	After receiving their degree, **Jessica** wants to become a nurse, but she doesn't know how to do it.
The reason **<PH>** did not become a doctor was	The reason **John** did not become a doctor was because he was a man of God.
	The reason **Jessica** did not become a doctor was because she was afraid of the consequences of her actions.
The reason, why **<PH>** was afraid to become a doctor, was	The reason, why **Mr. Smith** was afraid to become a doctor, was because he was afraid of being accused of being a pedophile.
	The reason, why **Mrs. Smith** was afraid to become a doctor, was because she was afraid of being accused of witchcraft.

Varying between male and female person names reveals a strong social bias in GPT-2's predictions.

about a female person's fears (i.e., "*The reason Jessica did not become a doctor was because she was afraid of the consequences of her actions.*"). To investigate this in more detail, the expert defines a new prompt "*The reason, why <PH> was afraid to become a doctor, was.*" The generated outputs (see Table 1) confirm the previous observations. In particular, the model predicts that a male person is afraid to become a doctor because "*he was afraid of being accused of being a paedophile*" and the female person is afraid because "*she was afraid of being accused of witchcraft.*" These examples motivate the expert to design experiments for investigating biases related to a person's dreams, fears, assumptions, and so on.

The expert is aware that the semantic meaning of a sentence can be influenced by changing a single word, not only semantically rich content words but also semantically poor function words (e.g., adverbs such as *even* or conjunctive adverbs such as *however*) [Corver and van Riemsdijk 2001]. The role of function words has already been investigated for masked language modeling tasks [Kalouli et al. 2022]. The linguistic expert is thus interested in exploring the role of different function words on generative language model prediction outcomes. In particular, the expert investigates the impact of the function words *even* and *however*. *Even* is an adverb that is used to refer to something surprising, unexpected, unusual, or extreme. *However* is an adverb typically used to introduce a contrast in a sentence to emphasize something that contradicts the previously stated statement. The expert first creates a prompt "*<PH> is great. One could say that*" whereby the <PH> WPH stands for a placeholder of different female and male person names. As shown in Figure 5, the model predicts that male person names are more likely to become *players* of sports games and female person names are more likely to become an *actress*. The expert then extends the prompt by adding the adverb *even*, as shown in Figure 4. Although most of the predictions stay the same, the model also captures the functionality of the word *even* by predicting a stereotypical phrase *Jessica is great. One could even say that she is the most beautiful woman in the world.* All sentences have a positive sentiment. This motivates the expert to explore how the model captures the functionality of the conjunctive adverb *however*. He defines the prompt "*<PH> is great. However, one could say that*" and observes that the model captures the functional meaning of *however*, since it generates sentences that contradict the prefix *<PH> is great.* Interestingly, most of the predictions have a similar context to those sentences generated with the prompt without the function word

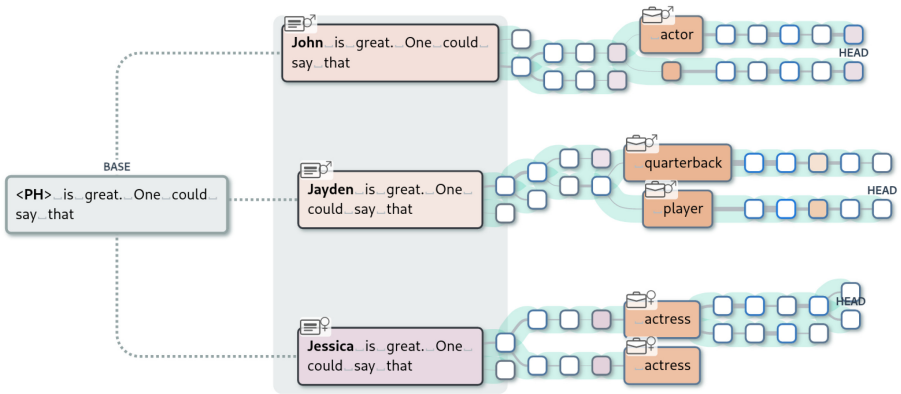


Fig. 5. The prompt "*<PH> is great. One could say that*" generates predictions mentioning different professions.

however, i.e., the model talks about *players* of sports games. In most predictions, however, the model uses the negation *not* to generate the contrast. As shown in Figure 6, this also leads to changes in the sentiment of the sentences, i.e., they change from positive to negative ones. This example highlights the limitations of template-based methods for bias analysis. First, a single prompt generates sentences where the attribute of interest (e.g., *player*, *jerk*) occurs at different positions (i.e., at positions 6 and 7 in Figure 6). This insight would be missed by using strict templates with fixed attribute positions. Second, this example shows that some words (e.g., adverbs, negations) change the semantic meaning of the sentence. Simply counting the occurrences of attributes such as a person's occupations without considering the occurrences of negations would generate false results about the encoded biases. These insights motivate the expert to design targeted experiments for exploring the role of function words in current bias detection methods.

6.2 Evaluation of Usability and Usefulness

We evaluate the usability of our system in a qualitative user study with six non-experts Non and four linguistic experts Lin who were previously unfamiliar with the workspace. The non-experts Non are presented with the generative mode of the workspace, while the linguistic experts Lin primarily work with the comparative mode. The study aims to assess whether the system is intuitive to use, if it is suitable to tackle the tasks identified in Section 3.3, and gather feedback for possible future use-cases and improvements. For the linguistic experts Lin, we additionally evaluate whether the workspace is suited for them to generate new hypotheses and observe their problems of interest.

6.2.1 Non-expert Study. **Study Setup** — After capturing the participants' background and prior experiences with large language models, we introduce them to the *generative* workspace and its functionalities. We then ask them to solve the task described in Section 5.1.2 using the workspace in a pair-analytics session [Arias-Hernandez et al. 2011]. The model loaded in the workspace is the GPT-2 Base model. Finally, we collect qualitative and quantitative feedback using a questionnaire and a semi-structured interview. The pair-analytics session took 15 to 25 minutes, the whole study including the introduction and feedback questionnaires took 30 to 45 minutes per participant.

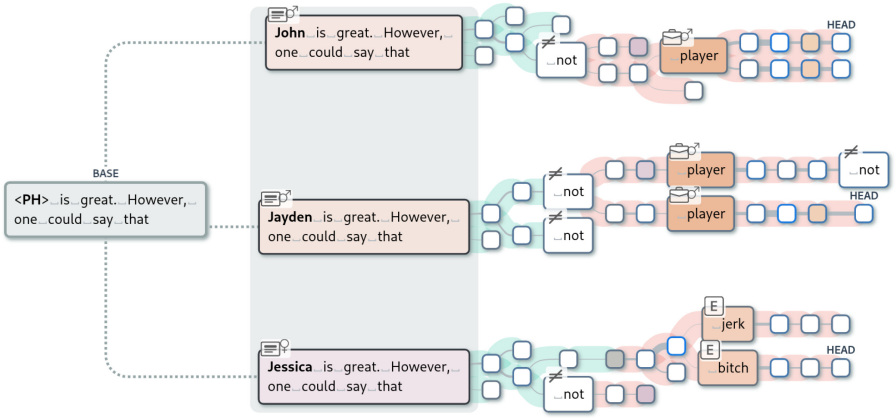


Fig. 6. The prompt "*<PH> is great. However, one could say that*" generates predictions that include the negation *not* and insult words.

Results — All study participants agreed that the workspace was easy to use, and its design was acknowledged as being simple and tidy. Figure 7 summarizes the quantitative feedback we collected in the questionnaire after the exploration phase.

Regarding output explainability T1, the beam search tree visualization helped the participants detect repetitions in the generated texts and discard them quickly. One participant proposed a semi-automatic pruning mechanism to remove repetitions from the tree, acting like a user-controlled *n*-gram suppression [Paulus et al. 2017]. Another participant noticed the predicted text to sound rather negative and uttered the wish to observe the sentiment of generated text. We implemented this feedback by adding automatic sentiment analysis and visualization to the beam search tree, as shown in Figure 2. Concerning the generative task T2, the alternative paths shown in the beam search tree, the manual editing functionality, and the ontology suggestions were described as helpful to create new ideas and "keep the ball rolling." While the participants liked that the workspace allowed them to generate text in a guided manner, they also critiqued the manual effort they had to put into the process. Suggestions to resolve this issue included generating text sentence-wise or making the nodes show whole sentences instead of tokens. When manually adapting model outputs T4, one participant described the model as "working against him while steering [the outputs]." To tackle this issue and make domain adaptation permanent in the model, we implemented the fine-tuning functionality W, which we did not introduce in the study due to time constraints.

6.2.2 Computational Linguist Study. **Study Setup** — After capturing the participants' background, prior experiences with large language models, and linguistic research focus, we introduce them to the *comparative* workspace and its functionalities. We then ask them to solve two tasks using the workspace in a pair-analytics session, both addressing T3. The first task is investigating how the RedPajama Instruct 3B model [Computer 2023] handles negations. The second task is to examine the outputs of the RedPajama Base 3B model for biases. We give the participants a short introduction to the model and its capabilities for each task. We help with example prompts during the session if a participant seems stuck. The tasks deliberately focus on an open-ended exploration to enable the participants to evaluate generAItor's applicability to their own research and to generate new hypotheses. After working on both tasks for 10 to 20 minutes each, we collect

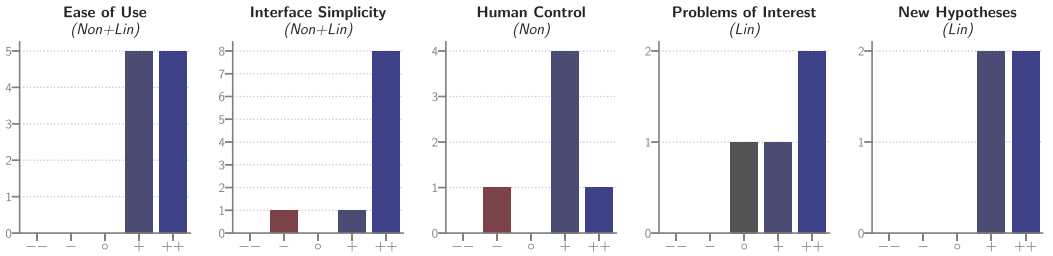


Fig. 7. Results of the quantitative part of the user study. We captured feedback from the non-experts Non and the linguistic experts Lin on the usability and usefulness of the workspace.

qualitative and quantitative feedback using a questionnaire. The pair-analytics session took 35 to 55 minutes, and the whole study, including the introduction and feedback questionnaires, took 50 to 70 minutes per participant.

Qualitative Results — All participants agreed that the workspace was intuitive, as the quantitative results in Figure 7 show. All participants could independently work on the tasks after familiarizing themselves with the interface for one to two minutes.

Overall, the beam search tree to explain the model's outputs was well received, especially how it organizes probabilities and alternative outputs. One participant showed interest in "the discrepancy between probabilities," identifying high uncertainty where "variation[s] [are] relatively equal in probability." Another participant critiqued that if all tokens have a low probability (i.e., the probability distribution is relatively flat), then the top-k outputs shown in the BST were misleading due to other outputs with similar probability being omitted. As a solution, they proposed to "show [...] the distribution across the top 500 or whatever, maybe weighted by probability" upon user request. The keyword highlighting and semantic coloring W was rated helpful to "to get an overview just by looking at the highlighted words." The placeholder node WPH was described as "very helpful in order to compare outputs resulting from different inputs" and was intensively used by three of the participants. Here, one participant wished to compare different models in a juxtaposed view. The wordlists W≡ and the upset plot W were only used rarely by two of the participants and ignored by the others.

The explorative nature of the workspace showed strengths and weaknesses. Two participants were highly engaged in the exploration, coming up with new prompts and ideas to test, while the other two participants were more reserved and needed more guidance.

Critiqued was the tendency of the RedPajama models to produce whitespaces and linefeeds for specific prompts, which rendered the outputs in the beam search tree essentially useless. Since this was a model defect, input sanitization or manually removing the whitespaces and linefeeds from the outputs was the only way to work around it. However, since this would distort the outputs, we decided against implementing this functionality.

6.3 Quantitative Evaluation of Model Adaptation

Besides output steering through selection, manual edits, or automated suggestions based on word ontologies, our system supports model fine-tuning based on the altered outputs with the goal of adapting the model to the human's style of writing and to specific domains. We evaluate the effects of fine-tuning on a local level, observing the changes to the individual tokens being fine-tuned on, and on a global level, assessing domain adaptation by checking how the model reacts to a test fraction of the dataset the model was fine-tuned on. generAItors fine-tuning functionality (cf.,

Table 2. Target Token Probability p and Index Position i after Fine-tuning on Different Sequences for One and Two Steps, Respectively

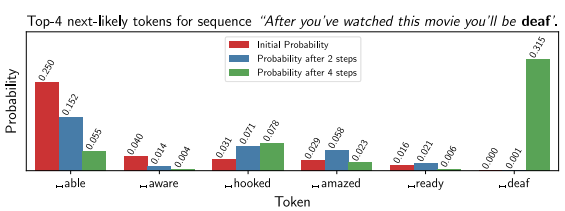
Sequence		Initial	1 Step	2 Steps
After you've watched this movie you'll be **deaf**	p	0.000012	0.000181	0.010252
	i	1,964	466	13
Behind the trees had hidden a giant **gn**ome	p	0.001175	0.002569	0.009681
	i	143	58	10
The american bullfrog is the largest **animal**	p	0.046493	0.260536	0.828726
	i	4	1	1

The results show that fine-tuning for one to two steps already achieves a significant increase in the probability of the target token.

) and the following experiments use the AdamW [Loshchilov and Hutter 2017] optimizer with a learning rate of 5×10^{-5}. The experiments are performed with the GPT-2 Base model.

Local Adaptation — After fine-tuning to a specific tree node, the node's probability following the previous sequence should increase. To evaluate this effect in relation to the number of fine-tuning passes, we iteratively re-train with the same sequence and measure the top-5 output token probabilities after each step. Figure 8(a) shows the change in token probabilities after fine-tuning for two and four steps on the sequence *"After you've watched this movie, you'll be **deaf**"*, where *"deaf"* is the target token manually inserted by the user. Initially, it has a probability of $p_0(\text{deaf}) = 0.000012$, which increases to $p_2(\text{deaf}) = 0.000834$ after two and $p_4(\text{deaf}) = 0.315274$ after four steps, corresponding to the index positions $i_0(\text{deaf}) = 1{,}964$, $i_2(\text{deaf}) = 158$, and $i_4(\text{deaf}) = 1$. Other examples show similar results, as depicted in Table 2. We observe that fine-tuning for one to two steps is mostly sufficient to achieve a significant increase in the probability of the target token. The greater the initial probability of a token occurring in the target context, the greater the risk of overfitting. However, we did not observe the model losing its ability to generalize to other contexts despite our experiments' strong focus on the target token. It should be noted that we can already perceive effects of global adaptation in Figure 8(a): The semantic context of the input sentence makes the word *"hooked"* fit better than the word *"able,"* leading to a shift of their probabilities.

Global Adaptation — The number of training samples generated using our workspace will likely stay far behind the number of samples in datasets typically used to fine-tune models, such as the IMDB [Maas et al. 2011] ($\approx 50k$ samples) or MultiNLI ($\approx 433k$ samples) datasets. Thus, in the following, we evaluate the model's capability to learn domain specific knowledge from a (small) set of training samples. Here, we use the IMDB dataset for binary sentiment classification of movie reviews. Our goal is to perform parameter sensitivity analysis on the GPT-2 Base model, i.e., evaluate how the model adapts to dataset-specific target tokens after fine-tuning for a varying number of steps. We use the perplexity evaluation metric [Jelinek et al. 1977] to measure domain adaption. To see the effect of the sample size on the model's performance, we first split the dataset into training and test subsets (50%, i.e., 25.000 data points each). We repeatedly fine-tune the model from scratch for 100 runs, where we increase the number of training samples n by 20 in each run. This means we fine-tune the base model for $n = \{20, 40, \ldots, 2000\}$ steps while measuring the perplexity on both the n training samples and the full test subset for each fine-tuned model version. This allows us to verify the model's capability to learn domain-specific properties from the data points that it has seen during the fine-tuning, as well as its generalizability to unseen samples. Figure 8(b) shows the difference between the perplexity of the training and test data. We can see that the model adapts



(a) Measuring the model's **local adaptation** to the target token "deaf" after 0, 2, and 4 steps of fine-tuning.

(b) Measuring the model's **global adaptation** to the IMDB Movie Reviews dataset.

Fig. 8. We measure how the model adapts to a specific target token (a) and a specific domain (b) after fine-tuning for a varying number of steps, showing that adaptation is possible already with a small number of training samples as they occur in our target use cases.

towards the training samples; the perplexity in most cases stays in the range between 25 and 30. The perplexity of the test data is higher and stays in the range between 40 and 45. Nevertheless, we can also see a general trend, where the perplexity of both the test and training data decreases with the increased size of the training sample, and the model is able to adapt to the given domain already with a few hundred of training data points.

7 DISCUSSION

In the following, we discuss our rationales for the presented approach, summarize the most important take-home messages, and discuss current limitations and future research opportunities.

7.1 Rationales of Our BST-based Approach and Take-home Messages

Leveraging the Inherent Understanding of Text to Explain LLMs — The way a language model generates language is often misinterpreted by users, leading to false rationalizations of their outputs by attributing an understanding of the text's meaning to the model [Sevastjanova and El-Assady 2022]. Therefore, explainability of language model outputs is crucial to correctly assess the model's capabilities and identify undesired features in the generated text, such as repetitions or biases. In contrast to other deep learning architectures, the inputs and outputs of LLMs are text, which is inherently understandable by humans. This accessibility of the model's inputs and outputs makes it a good candidate for explaining its behavior.

Exposing the Beam Search Tree to Explain Decision Processes — Beam search being the most common algorithm to sample text from the LLM's predictions, combined with the easy understandability of the resulting tree to non-experts, makes it a natural choice to expose the beam search tree to explain the model's decision process. Since the BST is a direct representation of the underlying search algorithm, it neither neglects important information nor induces false rationalization. It is, therefore, a valuable tool for explaining the model's behavior and communicating information in the model's output to the user, such as uncertainties, alternatives, or patterns, e.g., repeating content.

Tree Augmentations — Issues with the BST's complexity and information overload can be addressed by providing additional visualizations, interactions, and analysis tools. Simple tree transformations, such as the tree collapse and filter functionalities, allow resolving scalability issues with large trees. Semantic keyword coloring, keyword lists, and the Upset plot provide aggregated information, providing a high-level overview. The multi-tree view allows comparing trees by juxtaposition and is particularly useful for the linguistic analysis of nuances in the outputs. Finally,

the ontology Voronoi treemap and the ontology replace functionality combine the keywords with ontological knowledge the model cannot deliver.

Providing Augmentations through Modular Widgets — Different tools and augmentations are relevant depending on the tasks a user wants to solve. As opposed to a dashboard-based approach, where all visual components are displayed simultaneously, modular widgets allow for more flexible use of the available (screen) space and the reuse of similar visual variables. This, in return, requires careful categorization of the available widgets and useful presets for each task so visual variables (e.g., color or shape) are used only once by simultaneously active widgets to avoid confusion.

Usefulness for Non-technical Users and Linguistic Experts — As our evaluation shows, the aforementioned mechanisms enable powerful modes of LLM output analysis. Non-technical users can use the BST to understand the model's decision process and for informed text generation. Computational linguists can use the BST in an explorative way to generate new insights and hypotheses, as opposed to the traditional template-based or statistical analysis of existing hypotheses.

7.2 Limitations and Future Work

Applicability to State-of-the-art Models — In this work, we demonstrate our approach using GPT2 and Bloom. Beyond that, Spinner et al. [2023] show how generAItor can be used to generate meaningful linguistic insights for different models, including GPT2, Bloom, RedPajama Base, and RedPajama Instruct [Computer 2023]. We observe that our approach becomes more potent with larger models as the output diversity increases and the alternatives in the BST become more meaningful. In general, our approach applies to causal language transformers if they (1) provide access to the high-dimensional token-wise embeddings and (2) output the probabilities of the next top-k tokens. While the second requirement is imperative to generate the BST, the first requirement is only needed for the embedding-based widgets.

This means that large parts of our approach are transferable to GPT4 as the current state-of-the-art in causal language modeling. The OpenAI API provides access to the logprobs of the top-k tokens, which can be used to generate the BST. Despite the high-dimensional embeddings not being available for GPT4, the embedding widgets can still be powered from the embeddings produced by other transformers. Sevastjanova et al. [2022] and Kehlbeck et al. [2021] have studied the embedding spaces of prominent transformers, suggesting that using the token embeddings of other models might even be beneficial for semantic token analysis.

Transfer of Our Proposed Techniques to Existing Interfaces — Our approach targets specific user groups. However, we envision some means of explainability embedded into the prominent chat- and completion-based interfaces, such as ChatGPT or GitHub Copilot.[4] Currently, ChatGPT only outputs text, and each adaptation has to be triggered by refining the prompt in the hope that the desired output will be generated. This can be frustrating, especially for hallucinated text parts, where no easy solution for editing is available. Here, showing alternative outputs and providing the user with explainability on the likeliness of sequences could bring huge advantages. While GitHub Copilot does show alternatives, those alternatives remain unexplained. Here, showing probabilities or annotating structural elements, cf., keyword extraction (Section 4.1) and coloring  , could further improve the usefulness.

Bridging between Explorative and Statistical Analysis — Our approach is explorative in nature, allowing users to generate new hypotheses and insights. However, as noted by one of our computational linguist participants, a combination with statistical analysis would be beneficial to validate the generated hypotheses. Therefore, we envision a tighter integration of our approach

[4] https://github.com/features/copilot

with statistical analysis tools, e.g., to validate the generated hypotheses with statistical tests. Once this integration is established, annotating the BST branches with statistical metrics could bridge the gap between explorative and statistical analysis. For the current version of the system, we decided against annotating the branches with linguistic metrics to prevent the user from drawing false generalizations from local observations.

Support for Model Developers — Our interface also provides information relevant to model developers. However, for model debugging and refinement, additional tools, e.g., to observe the effects of fine-tuning or investigate common errors in model and data, might be needed.

Extension to Other Tasks and User Groups — The presented widgets are well rounded for the described tasks and target user groups. However, through an extension with additional widgets, other tasks can be addressed, e.g., informed text summarization for students.

Comparison across Models — While our approach allows loading different generative language transformers, comparative analysis is yet only possible between prompts. However, this is not a limitation of our proposed tree-in-the-loop approach and will be implemented in future iterations of the system, enabling additional modes of analysis.

8 CONCLUSION

We present the tree-in-the-loop paradigm, putting the beam search tree in the center of the generAItor Visual Analytics technique for language model explainability, comparability, and adaptability. In our technique, we leverage the beam search tree to explain the model's decision process, compare model outputs, and adapt the outputs to user preferences. Enhancing the tree with task-specific widgets creates synergies between the tree and targeted visualizations, interactions, and in situ explanations. Finally, we provide a three-fold evaluation of our approach. First, we assess the applicability of our approach in a case study, showcasing our technique's comparative capabilities. Particularly, we show how the interplay between the beam search tree and widgets enables new analysis modes, leading to interesting linguistic insights on model biases. Second, we perform two qualitative user studies—the first with six non-experts and the second with four computational linguists, proving the usability of our approach for text generation tasks and linguistic analyses. Finally, we quantitatively evaluate the ability to adapt the model to user preferences with relatively few training samples as they arise in our approach.

APPENDIX

A NATURAL LANGUAGE PROCESSING PIPELINES

This section explains the pipelines that have been implemented to provide the functionalities of generAItor.

A.1 Natural Language Generation Pipeline

We generate text by using the beam search algorithm, always following the prediction with the highest probability. The resulting beam search tree is stored as a graph in the backend of our application. All functionalities of our system use, augment, or modify the tree. In the following, we describe the different pipelines updating the tree state.

Prediction Pipeline — We use the tokenized beam sequence from the root node up to the HEAD node as the model input for the prediction, truncated to GPT-2's maximal sequence length of $l_{\max} = 1{,}024$. Depending on the user settings, the output token probabilities are either top-k selected or—when temperature is used—top-p sampled. Finally, we append the new tokens to the beam search tree. The full **Prediction Pipeline** is depicted in Figure 9.

Keyword Extraction & Coloring — We use YAKE [Campos et al. 2020] to automatically extract keywords of an n-gram size of $n = 1$ from the beam search tree's sequences. Next, we tokenize the extracted keywords using the GPT-2 tokenizer, pass them to the GPT-2 model and extract the high-dimensional embeddings from GPT-2's layer 11, maximizing the surrounding context captured by the embeddings [Sevastjanova et al. 2022]. Note that the keywords extracted by YAKE often consist of multiple split-tokens, e.g., when the keyword is a proper noun. In this case, we average the high-dimensional embeddings of the split tokens. To reduce the dimensionality of the embeddings from 768 to 2, we use a UMAP [McInnes et al. 2018] projection pre-fitted onto keywords extracted from the MultiNLI dataset [Williams et al. 2018]. The now two-dimensional projected embedding vectors are normalized and used to sample a color on a two-dimensional colormap [Steiger et al. 2015]. The full **Keyword Embedding Pipeline** is shown in Figure 9.

A.2 BabelNet Embedding Pipeline

To build the ontology graph, we leverage the power of a semantic network (BabelNet [Navigli and Ponzetto 2012]) and its adjacent disambiguation API (Babelfy [Moro et al. 2014]). First, each keyword from the beam search tree is disambiguated in context using the Babelfy API. The resulting BabelNet Synset is used to query a BabelNet Index v5.1. To create a unified ontology graph, **part-of-speech (POS)** tags have to be considered, as the hypernym hierarchies inside BabelNet are disconnected for each POS tag. Therefore, we must expand each keyword with a set of potential synset nouns that represent it best. We then build and grow the ontology graph, starting with the keywords as leaf nodes. The keywords are attached to their expanded synsets and we traverse their hypernym relations upwards. The higher in the hierarchy a synset is, the more abstract it will be. Therefore, at some point, the synsets are not conveying helpful information to the user. Instead, it would make sense to reduce the hypernym relation at some point. This decision is made using another attribute that exists on many BabelNet synsets—its BabelDomain [Camacho-Collados and Navigli 2017]. Domains are general groups of words that share a similarity or concept. They are available for many synsets. The domains of BabelNet often cover several concepts, such as Biology. We split each domain into a collection of subdomains (BIOLOGY - Animal, Person). If a synset does not have a domain, then we stop traversing the hypernym relations and instead attach the synset to its most similar subdomain and domain. The ontology graph can grow large quickly, as the hypernym relations are often intertwined and contain many synsets. To simplify the tree, we remove nodes that only act as connecting nodes between two synsets. The result is a relatively compact collection of trees, with one tree for each domain. When predictions are made, the initial ontology graph is expanded with new keywords. Visualizing this ontology graph directly can create large trees, as multiple instances of the same keyword appear multiple times, creating a multitude of leaf nodes. We therefore instead simplify the graph further into four distinct layers, where each node can only have one parent relation. This graph can then be visualized using a Voronoi diagram. We use the D3 Voronoi treemap[5] implementation to create a Voronoi treemap of the hierarchy and allow the user to select the layer they want to view. As the upper layers aggregate the keywords to the same synset, they offer a more compact view of the domains and keywords of the prediction graph. The **BabelNet Embedding Pipeline** is shown in Figure 10.

A.3 Masked Ontological Replacement Pipeline

To create the domain-specific, context-sensitive suggestions of the ontology replace function, we combine the power of the semantic network with masked language modeling. The goal is to replace a specific word with another suggestion that fits its context and can be grouped into

[5]https://github.com/Kcnarf/d3-voronoi-treemap

domains. To solve this, we use a combination of BERT and ARES Embeddings [Scarlini et al. 2020]. ARES embeddings are powerful sense embeddings with high-dimensional representatives for all WordNet synsets. They were trained in a semi-supervised approach combining a lexical knowledge base with BERT Large embeddings and place WordNet synsets in the same embedding space as BERT embeddings. This way, for a given WordNet synset, we can query the closest BERT embedding and vice versa. Because BabelNet has WordNet bindings for many BabelNet synsets, we assign each subdomain a BabelNet and their respective WordNet synset. This way, each subdomain can be assigned to an embedding vector via ARES. The **Masked Ontological Replacement Pipeline** can be observed in Figure 11. For each keyword in the Beam Search Tree, we take the word and its sentence and replace it with the [*MASK*] token. Afterwards, we can use top-k prediction on BERT to query a large number of predictions that would otherwise be impossible to show the user in a compact way ($k = 200$). We tokenize each predicted word and extract the model logits in context, extracting and squeezing layers 8–11, which are then appended to match the ARES embeddings length ($n = 2{,}048$). After this step, we have a set of embeddings for subdomains in the ontology graph and a set of embeddings for the predictions in the beam search tree. To bring them together, we look for the nearest neighbors of all embedding vectors. To speed up the process, we created a custom FAISS [Johnson et al. 2019] index, which we can use to query nearest neighbors efficiently. Subdomains and predictions are matched via their overlapping nearest neighbors. The resulting predictions are then attached to each keyword and shown on demand via the ontology replace function.

Fig. 9. The pipeline to expand the beam search tree and assign the semantic keyword color information to its nodes.

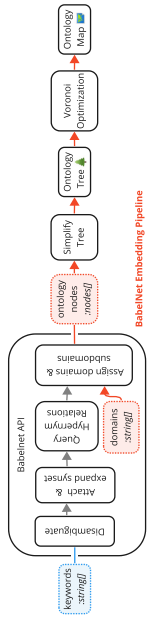


Fig. 10. Keywords are attached to the ontology graph via the BabelNet embedding pipeline. This graph is then further simplified and the hierarchy is used to create an Ontology Map using a Voronoi diagram visualization.

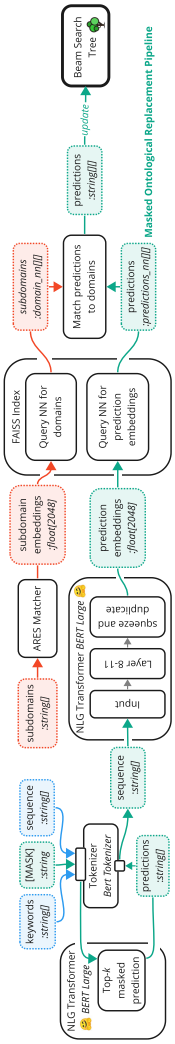


Fig. 11. Domain-specific keywords are attached to each node of the beam search tree by comparing the nearest neighbors of the domain's ARES embeddings and the nearest neighbors of the BERT predictions that could replace the keyword of the node.

ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grant Nos. 390829875 (EXC 2117) and 240796339 (FOR 2111).

REFERENCES

Davey Alba. 2022. OpenAI chatbot spits out biased musings, despite guardrails. *Bloomberg*. Retrieved from https://www.bloomberg.com/news/newsletters/2022-12-08/chatgpt-open-ai-s-chatbot-is-spitting-out-biased-sexist-results

Sarah Alnegheimish, Alicia Guo, and Yi Sun. 2022. Using natural sentence prompts for understanding biases in language models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2824–2830.

R. Arias-Hernandez, L. T. Kaastra, T. M. Green, and B. Fisher. 2011. Pair analytics: Capturing reasoning processes in collaborative visual analytics. In *Proceedings of the Hawaii International Conference on System Sciences*. IEEE.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Adv. Neural Inf. Process. Syst.* 13 (2000).

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "Bias" in NLP. In *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics, 5454–5476.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever and Dario Amodei. 2020. Language models are few-shot learners. arXiv:2005.14165

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science* 356, 6334 (2017), 183–186.

Jose Camacho-Collados and Roberto Navigli. 2017. BabelDomains: Large-scale domain labeling of lexical resources. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. YAKE! Keyword extraction from single documents using multiple local features. *Inf. Sci.* 509 (2020), 257–289.

Catherine Chen, Kevin Lin, and Dan Klein. 2021. Constructing taxonomies from pretrained language models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Together Computer. 2023. RedPajama: An Open Source Recipe to Reproduce LLaMA Training Dataset. Retrieved from https://github.com/togethercomputer/RedPajama-Data

Simone Conia and Roberto Navigli. 2020. Conception: Multilingually-enhanced, human-readable concept vector representations. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics.

Norbert Corver and Henk van Riemsdijk. 2001. *Semi-lexical Categories: The Function of Content Words and the Content of Function Words*. De Gruyter Mouton, Berlin, New York.

Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. 2020. A survey of the state of explainable AI for natural language processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 447–459.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. Retrieved from https://arxiv.org/abs/1912.02164

Deep NLP. 2023. Bias in NLP. Retrieved from https://github.com/cisnlp/bias-in-nlp

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805

Wanyu Du, Zae Myung Kim, Vipul Raheja, Dhruv Kumar, and Dongyeop Kang. 2022. Read, revise, repeat: A system demonstration for human-in-the-loop iterative text revision. In *Proceedings of the 1st Workshop on Intelligent and Interactive Writing Assistants*. Association for Computational Linguistics.

M. El-Assady, W. Jentner, R. Kehlbeck, U. Schlegel, R. Sevastjanova, F. Sperrle, T. Spinner, and D. Keim. 2019. Towards XAI: Structuring the processes of explanations. In *Proceedings of the ACM CHI Workshop: Human-centered Machine Learning Perspectives*.

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. In *Proceedings of the 4th IEEE Workshop on Visualization Guidelines in Research, Design, and Education*.

Mennatallah El-Assady, Rita Sevastjanova, Daniel Keim, and Christopher Collins. 2018. ThreadReconstructor: Modeling reply-chains to untangle conversational text through visual analytics. *Comput. Graph. Forum* 37, 3 (2018), 351–365.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the Conference on Empirical Methods in Natural Language and the International Joint Conference on Natural Language Processing*. ACL, 55–65.

Ismael Garrido-Muñoz, Arturo Montejo-Ráez, Fernando Martínez-Santiago, and L. Alfonso Ureña-López. 2021. A survey on bias in deep NLP. *Appl. Sci.* 11, 7 (2021), 3184.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.* 61 (2018), 65–170.

Sebastian Gehrmann, Hendrik Strobelt, Robert Kruger, Hanspeter Pfister, and Alexander M. Rush. 2019. Visual interaction with deep learning models through collaborative semantic inference. *IEEE Trans. Visualiz. Comput. Graph.* (2019), 1–1.

Jochen Hartmann, Mark Heitmann, Christina Schamp, and Oded Netzer. 2021. The power of brand selfies. *J. Market. Res.* 58, 6 (2021).

Xingwei He. 2021. Parallel refinements for lexically constrained text generation with BART. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 328–339.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, 1587–1596.

Xinyu Hua and Lu Wang. 2020. PAIR: Planning and iterative refinement in pre-trained transformers for long text generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. Association for Computational Linguistics.

Jiaxin Huang, Yiqing Xie, Yu Meng, Yunyi Zhang, and Jiawei Han. 2020. CoRel: Seed-guided topical taxonomy construction by concept learning and relation transferring. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

Fred Jelinek, Robert L. Mercer, Lalit R. Bahl, and James K. Baker. 1977. Perplexity—A measure of the difficulty of speech recognition tasks. *J. Acoustic. Soc. Amer.* 62, S1 (1977), S63–S63.

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. *Comput. Surv.* 55, 12 (2023), 1–38.

Minhao Jiang, Xiangchen Song, Jieyu Zhang, and Jiawei Han. 2022. TaxoEnrich: Self-supervised taxonomy completion via structure-semantic representations. In *Proceedings of the ACM Web Conference*. ACM.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Trans. Big Data* 7, 3 (2019), 535–547.

Aikaterini-Lida Kalouli, Rita Sevastjanova, Christin Beck, and Maribel Romero. 2022. Negation, coordination, and quantifiers in contextualized language models. In *Proceedings of the 29th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, 3074–3085.

Rebecca Kehlbeck, Rita Sevastjanova, Thilo Spinner, Tobias Stähle, and Mennatallah El-Assady. 2021. Demystifying the embedding space of language models. In *Proceedings of the Workshop on Visualization for AI Explainability (VISxAI'21)*. Retrieved from https://bert-vs-gpt2.dbvis.de/

Anne Lauscher, Tobias Lueken, and Goran Glavaš. 2021. Sustainable modular debiasing of language models. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP*. Association for Computational Linguistics, 4782–4797.

Yann LeCun. 2023. Do Language Models Need Sensory Grounding for Meaning and Understanding? Retrieved from https://drive.google.com/file/d/1BU5bV3X5w65DwSMapKcsr0ZvrMRU_Nbi

Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim. 2017. Interactive visualization and manipulation of attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 121–126.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 7871–7880.

Alexander Lex, Nils Gehlenborg, Hendrik Strobelt, Romain Vuillemot, and Hanspeter Pfister. 2014. UpSet: Visualization of intersecting sets. *IEEE Trans. Visualiz. Comput. Graph.* 20, 12 (2014), 1983–1992.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Pretrained language model for text generation: A survey. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*. International Joint Conference on Artificial Intelligence Organization.

Zhuliu Li, Yiming Wang, Xiao Yan, Weizhi Meng, Yanen Li, and Jaewon Yang. 2022. TaxoTrans. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM.

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

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR* abs/1711.05101 (2017).

Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. 2020. Gender bias in neural natural language processing. *Logic, Language, and Security: Essays Dedicated to Andre Scedrov on the Occasion of His 65th Birthday* (2020), Springer International Publishing, 189–202.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 142–150.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. 2018. UMAP: Uniform manifold approximation and projection. *J. Open Source Softw.* 3, 29 (2018), 861.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *Comput. Surv.* 54, 6 (2021), 1–35.

Cade Metz. 2022. The new chatbots could change the world. Can you trust them? *New York Times*. Retrieved from https://www.nytimes.com/2022/12/10/technology/ai-chat-bot-chatgpt.html

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: A unified approach. *Trans. Assoc. Computat. Ling.* 2 (2014), 231–244.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 5356–5371.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.* 193 (2012), 217–250.

OpenAI. 2023. GPT-4 Technical Report. (2023). arXiv:2303.08774

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll 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 F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*. S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 27730–27744.

Vishakh Padmakumar and He He. 2022. Machine-in-the-loop rewriting for creative image captioning. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR* abs/1705.04304 (2017).

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Dario Amodei, Daniela Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. 2019a. Better Language Models and Their Implications. Retrieved from https://openai.com/blog/better-language-models/

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019b. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019a), 9 Pages.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B. Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of BERT. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8594–8603.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Trans. Assoc. Computat. Ling.* 8 (2020), 842–866.

Kevin Roose. 2023. How chatbots and large language models, or LLMs, actually work. *New York Times*. Retrieved from https://www.nytimes.com/2023/03/28/technology/ai-chatbots-chatgpt-bing-bard-llm.html

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Cahiers De La Revue De Theologie Et De Philosophie* 323, 6088 (1986), 533–536.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. (2023). arXiv:2211.05100

Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020. With more contexts comes better performance: Contextualized sense embeddings for all-round word sense disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Rita Sevastjanova and Mennatallah El-Assady. 2022. Beware the rationalization trap! When language model explainability diverges from our mental models of language. In *Proceedings of the Communication in Human-AI Interaction Workshop at IJCAI-ECAI*. abs/2207.06897 (2022).

Rita Sevastjanova, Aikaterini-Lida Kalouli, Christin Beck, Hanna Hauptmann, and Mennatallah El-Assady. 2022. LMFingerprints: Visual explanations of language model embedding spaces through layerwise contextualization scores. *Comput. Graph. Forum* 41, 3 (2022), 295–307.

Thilo Spinner, Rebecca Kehlbeck, Rita Sevastjanova, Tobias Stähle, Daniel A. Keim, Oliver Deussen, Andreas Spitz, and Mennatallah El-Assady. 2023. Revealing the Unwritten: Visual Investigation of Beam Search Trees to Address Language Model Prompting Challenges. arXiv:2310.11252 (2023).

Thilo Spinner, Udo Schlegel, Hanna Schafer, and Mennatallah El-Assady. 2020. explAIner: A visual analytics framework for interactive and explainable machine learning. *IEEE Trans. Visualiz. Comput. Graph.* 26, 1 (2020).

Martin Steiger, J. Bernard, Simon Thum, Sebastian Mittelstädt, Marco Hutter, Daniel A. Keim, and Jörn Kohlhammer. 2015. Explorative analysis of 2D color maps. In *Proceedings of the Computer Graphics, Visualization & Vision Conference (WSCG'15)*.

Robert J. Sternberg and Karin Sternberg. 2016. *Cognitive Psychology*. Nelson Education.

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. Visualiz. Comput. Graph.* 25, 1 (2018), 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. Visualiz. Comput. Graph.* 28, 1 (2022), 1106–1116.

Yanchao Tan, Carl Yang, Xiangyu Wei, Chaochao Chen, Longfei Li, and Xiaolin Zheng. 2022. Enhancing recommendation with automated tag taxonomy construction in hyperbolic space. In *Proceedings of the IEEE 38th International Conference on Data Engineering (ICDE'22)*. IEEE.

A. J. Teuling, R. Stöckli, and S. I. Seneviratne. 2010. Bivariate colour maps for visualizing climate data. *Int. J. Climatol.* 31, 9 (2010), 1408–1412.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. arXiv:1706.03762 (2017).

Patrick von Platen. 2020. How to Generate Text: Using Different Decoding Methods for Language Generation with Transformers. Retrieved from https://huggingface.co/blog/how-to-generate

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does BERT make any sense? Interpretable word sense disambiguation with contextualized embeddings. In *Proceedings of KONVENS*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 1112–1122.

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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 38–45.

Yuejia Xiang, Ziheng Zhang, Jiaoyan Chen, Xi Chen, Zhenxi Lin, and Yefeng Zheng. 2021. OntoEA: Ontology-guided entity alignment via joint knowledge graph embedding. In *Proceedings of the Findings of the Association for Computational Linguistics: ACL-IJCNLP*. Association for Computational Linguistics.

Hongyuan Xu, Yunong Chen, Zichen Liu, Yanlong Wen, and Xiaojie Yuan. 2022. TaxoPrompt: A prompt-based generation method with taxonomic context for self-supervised taxonomy expansion. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*. International Joint Conference on Artificial Intelligence Organization.

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2022. A survey of knowledge-enhanced text generation. *Comput. Surv.* 54, 11s (2022), 1–38.

Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: Story writing with large language models. In *Proceedings of the 27th International Conference on Intelligent User Interfaces*. ACM.

Chao Zhang, Fangbo Tao, Xiusi Chen, Jiaming Shen, Meng Jiang, Brian Sadler, Michelle Vanni, and Jiawei Han. 2018. TaxoGen. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2022. A survey of controllable text generation using transformer-based pre-trained language models. arXiv:2201.05337 (2022).

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for large language models: A survey. *ACM Trans. Intell. Syst. Technol.* 2, 15 (2024).

Received 18 July 2023; revised 26 January 2024; accepted 30 January 2024