

COMPOSABLE ADAPTATION MODELS FOR DEFINING
PARAMETERS OF GENERATIVE MODEL OUTPUT

CROSS-REFERENCE TO RELATED APPLICATIONS

- 5 [0001] The present application claims the benefit of the right of priority to US Patent Application No. 18/468,460 entitled “COMPOSABLE LOW-RANK ADAPTATION MODELS FOR DEFINING LARGE-LANGUAGE MODEL TEXT STYLE” filed September 15, 2023, the contents of which are herein incorporated by reference in their entirety.

10 TECHNICAL FIELD

- [0002] This disclosure relates to generative models, and in particular to modifying generative models using adaptation models.

BACKGROUND

- 15 [0003] Generative models may be considered a form of artificial intelligence (AI). Generative models are used to generate various forms of output including, for example, text and/or images.
- [0004] For example, Large-language models (LLMs) are useful for a wide variety of text generation applications, including, for example, generating text for output by chat agents.
- 20 However, LLMs typically have millions or billions of parameters. Given the size of LLMs, it is infeasible for most applications to retrain an LLM to generate text that has specific properties.
- [0005] In another example, image generation models such as, for example, diffusion models can be used to generate images for a variety of purposes.

25 SUMMARY

- [0006] According to the subject matter of the present application, a set of user-interface controls (e.g., sliders) are provided that allow control over attributes of output generated by a generative model. For example, in the case where the generative is an LLM, the output control may be provided over attributes of text generated by the LLM. Different values for the output
- 30 controls may correspond to different parameters employed for generation by the generative model. For example, different values of parameters may correspond to selection of different adaptation models and/or varying of parameters thereof.
- [0007] A computer-implemented method may be provided. The computer-implemented method may include maintaining a plurality of adaptation models that each include a set of weights

configured to modify parameters of a generative model to cause the generative model to generate output having a corresponding property. The computer-implemented method may further include presenting a set of manipulable user-interface controls to allow configuration of properties of output generated using the generative model and the output of the generative model may be modified based on one or more of the adaptation models of the plurality of adaptation models based on a state of the user-interface controls as manipulated.

[0008] In some implementations, the computer-implemented method may further include providing a preview of generative model output corresponding to a current state of the user-interface controls during presentation and manipulation thereof. Providing the preview may include providing input to the generative model and outputting the output of the generative model responsive to the input. It may be that the output of the generative model is modified based on the one or more of the adaptation models of the plurality of adaptation models based on the state of the user-interface controls as manipulated.

[0009] In some implementations, the computer-implemented method may further include maintaining the preview of generative model output corresponding to the current state of the user-interface controls during the presentation and manipulation thereof. Maintaining the preview may include iteratively providing input to the generative model and outputting the output of the generative model responsive to the input for each iteration. It may be that, for each iteration, the output of the generative model is modified based on the one or more of the adaptation models of the plurality of adaptation models selected based on the current state of the user-interface controls as manipulated.

[0010] In some implementations, the computer-implemented method may further include identifying a first adaptation model of the plurality of adaptation models based on the current state of the user-interface controls as manipulated. The first adaptation model may correspond to a first property of output generated by the generative model; and identifying a range of degrees of a second property of output generated by the generative model based on an adaptation model corresponding to a degree in the identified range being composable with the first adaptation model. It may be that presenting the set of manipulable user-interface controls includes displaying indications of degrees of the second property in the identified range as selectable and displaying indications of degrees of the second property that are not in the identified range as not selectable.

[0011] In some implementations, the presenting the set of user-interface controls may include identifying a first adaptation model of the plurality of adaptation models based on the current state of the user-interface controls as manipulated. The first adaptation model may correspond to a first property of output generated by the generative model; identifying a subset of

the plurality of adaptation models based on the first adaptation model; and presenting user-interface controls associated with the properties corresponding to the identified subset of adaptation models that are manipulable to select one or more of the corresponding properties.

5 [0012] In some implementations, the presenting the set of user-interface controls may include presenting a first user-interface control that is manipulable to change a degree of a first property of the output generated by the generative model. Manipulation of the set of user-interface controls may control a selection of the degree of the first property.

10 [0013] In some implementations, the presenting the set of user-interface controls may include presenting a first set of user-interface controls that are manipulable to select respective properties of generative model output within a first category of properties; and in response to receiving manipulation of a user-interface control of the first set of user-interface controls to select a property from among the first category of properties, presenting a second set of user-interface controls that are manipulable to select respective properties of generative model output within a second category of properties.

15 [0014] In some implementations, the properties within the second category of properties may be determined based on the selected property from among the first category of properties.

[0015] In some such implementations, the set of user-interface controls may be output for display by a browser application, and the method may further include storing identifiers of the plurality of adaptation models in the browser application.

20 [0016] In some implementations, the generative model may be a large language model (LLM). The output of such a generative model may be or may include text.

[0017] In some such implementations, the set of user-interface controls may be presented to allow configuration of properties of text generated by the LLM for output by a chat agent.

25 [0018] In some such implementations, the computer-implemented method may further include storing identifiers of the plurality of adaptation models in a repository or instruction set that is accessible to the chat agent during execution of the chat agent.

[0019] In some implementations, the providing the prompt to the LLM may include specifying an order in which the one or more adaptation models selected based on the current state of the user-interface controls as manipulated are to be composed.

30 [0020] In some implementations, the computer-implemented method may further include training the plurality of adaptation models based on a set of chat conversations that have been tagged as containing the property corresponding to each of the plurality of adaptation models.

[0021] In some implementations, the generative model may be an image generation model and the output of the generative model may be or may include one or more images. For example, the generative model may be a diffusion model.

[0022] In some implementations, the one or more of the adaptation models of the plurality of adaptation models may be selected based on a state of the user-interface controls as manipulated.

[0023] In some implementations, the adaptation models may be low-rank adaptation (LoRA) models.

[0024] In some implementations, there may be provided a computer readable storage medium storing executable instructions, execution of which by a processor may cause the processor to perform a method consistent with the foregoing. Notably, the processor may be a processor of a computer and execution of the instructions by the processor may cause the computer to perform a method consistent with the foregoing. In at least some implementations, the computer readable storage medium may be non-transitory.

[0025] In some implementations, there may be provided a system comprising at least one hardware processor; and at least one memory storing instructions, which, when executed by the at least one processor, may cause the system to perform a method consistent with the foregoing. The at least one memory may be or may include a non-transitory memory such as, for example, a non-transitory computer-readable storage medium.

[0026] In some implementations, there may be provided a computer program product including instructions which, when executed by at least one hardware processor, cause the at least one hardware processor to carry out a method consistent with the foregoing.

[0027] In some implementations, a computer-implemented method may be provided. The computer-implemented method may comprise maintaining a plurality of low-rank adaptation (LoRA) models that each include a set of weights configured to modify parameters of a large-language model (LLM) to cause the LLM to generate text having a corresponding property. The computer-implemented method may further comprise presenting a set of manipulable user-interface controls to allow configuration of properties of text generated using the LLM and wherein output of the LLM is to be modified based on one of more of the LoRA models of the plurality of LoRA models selected based on a state of the user-interface controls as manipulated. The computer-implemented method may further comprise providing a preview of LLM output corresponding to a current state of the user-interface controls during presentation and manipulation thereof, wherein providing the preview includes iteratively providing a prompt to the LLM and outputting the output of the LLM responsive to the prompt for each iteration (*i.e.*, the output that comes out of the LLM in response to the input is output, the input being the prompt), wherein, for

each iteration, the output of the LLM is modified based on one or more of the LoRA models of the plurality of LoRA models selected based on the current state of the user-interface controls as manipulated.

5 [0028] In some implementations, the computer-implemented method may further comprise identifying a first LoRA model of the plurality of LoRA models based on the current state of the user-interface controls as manipulated, wherein the first LoRA model corresponds to a first property of text generated by the LLM; and identifying a range of degrees of a second property of text generated by the LLM based on a LoRA model corresponding to a degree in the identified range being composable with the first LoRA model; wherein presenting the set of manipulable
10 user-interface controls comprises displaying indications of degrees of the second property in the identified range as selectable and displaying indications of degrees of the second property that are not in the identified range as not selectable.

[0029] In some implementations, the presenting the set of user-interface controls may comprise identifying a first LoRA model of the plurality of LoRA models based on the current
15 state of the user-interface controls as manipulated, wherein the first LoRA model corresponds to a first property of text generated by the LLM; identifying a subset of the plurality of LoRA models based on the first LoRA model; and presenting user-interface controls associated with the properties corresponding to the identified subset of LoRA models that are manipulable to select one or more of the corresponding properties.

20 [0030] In some implementations, the presenting the set of user-interface controls may comprise presenting a first user-interface control that is manipulable to change a degree of a first property of the text generated by the LLM, wherein manipulation of the set of user-interface controls a selection of the degree of the first property.

[0031] In some implementations, the presenting the set of user-interface controls may
25 comprise presenting a first set of user-interface controls that are manipulable to select respective properties of text within a first category of properties; and in response to receiving manipulation of a user-interface control of the first set of user-interface controls to select a property from among the first category of properties, presenting a second set of user-interface controls that are manipulable to select respective properties of text within a second category of properties.

30 [0032] In some implementations, the properties within the second category of properties may be determined based on the selected property from among the first category of properties.

[0033] In some implementations, the set of user-interface controls may be presented to allow configuration of properties of text generated by the LLM for output by a chat agent.

[0034] In some implementations, the computer-implemented method may further comprise storing identifiers of the plurality of LoRA models in a repository or instruction set that is accessible to the chat agent during execution of the chat agent.

5 [0035] In some implementations, the set of user-interface controls may be output for display by a browser application, and wherein the method may further comprise storing identifiers of the plurality of LoRA models in the browser application.

[0036] In some implementations, the providing the prompt to the LLM may comprise specifying an order in which the one or more LoRA models selected based on the current state of the user-interface controls as manipulated are to be composed.

10 [0037] In some implementations, the computer-implemented method may further comprise training the plurality of LoRA models based on a set of chat conversations that have been tagged as containing the property corresponding to each of the plurality of LoRA models.

[0038] In some implementations, there may be provided a computer readable storage medium storing executable instructions, execution of which by a processor may cause the
15 processor to perform a method consistent with the foregoing. Notably, the processor may be a processor of a computer and execution of the instructions by the processor may cause the computer to perform a method consistent with the foregoing. In at least some implementations, the computer readable storage medium may be non-transitory.

[0039] In some implementations, there may be provided a system comprising at least one
20 hardware processor; and at least one memory storing instructions, which, when executed by the at least one processor, may cause the system to perform a method consistent with the foregoing. The at least one memory may be or may include a non-transitory memory such as, for example, a non-transitory computer-readable storage medium.

[0040] In some implementations, there may be provided a computer program product
25 including instructions which, when executed by at least one hardware processor, cause the at least one hardware processor to carry out a method consistent with the foregoing.

[0041] In some implementations, a non-transitory computer readable storage medium storing executable instructions may be provided. The execution of the executable instruction by a processor may cause the processor to maintain a plurality of low-rank adaptation (LoRA) models
30 that each include a set of weights configured to modify parameters of a large-language model (LLM) to cause the LLM to generate text having a corresponding property. The execution of the executable instruction by a processor may further cause the processor to present a set of manipulable user-interface controls to allow configuration of properties of text generated using the LLM and wherein output of the LLM is to be modified based on one of more of the LoRA models

of the plurality of LoRA models selected based on a state of the user-interface controls as manipulated. The execution of the executable instruction by a processor may further cause the processor to provide a preview of LLM output corresponding to a current state of the user-interface controls during presentation and manipulation thereof, wherein providing the preview includes
5 iteratively providing a prompt to the LLM and outputting the output of the LLM responsive to the prompt for each iteration, wherein, for each iteration, the output of the LLM is modified based on one or more of the LoRA models of the plurality of LoRA models selected based on the current state of the user-interface controls as manipulated.

[0042] In some implementations, a system comprising at least one hardware processor; and
10 at least one non-transitory memory storing instructions may be provided. The instructions, when executed by the at least one hardware processor, may cause the system to maintain a plurality of low-rank adaptation (LoRA) models that each include a set of weights configured to modify parameters of a large-language model (LLM) to cause the LLM to generate text having a corresponding property. The instructions, when executed by the at least one hardware processor,
15 may further cause the system to present a set of manipulable user-interface controls to allow configuration of properties of text generated using the LLM and wherein output of the LLM is to be modified based on one of more of the LoRA models of the plurality of LoRA models selected based on a state of the user-interface controls as manipulated. The instructions, when executed by the at least one hardware processor, may further cause the system to provide a preview of LLM
20 output corresponding to a current state of the user-interface controls during presentation and manipulation thereof, wherein providing the preview includes iteratively providing a prompt to the LLM and outputting the output of the LLM responsive to the prompt for each iteration, wherein, for each iteration, the output of the LLM is modified based on one or more of the LoRA models of the plurality of LoRA models selected based on the current state of the user-interface
25 controls as manipulated.

[0043] In some implementations, a computer program product comprising instructions may be provided. The instructions, when executed by at least one hardware processor, may cause the at least one hardware processor to maintain a plurality of low-rank adaptation (LoRA) models that each include a set of weights configured to modify parameters of a large-language model (LLM)
30 to cause the LLM to generate text having a corresponding property. The instructions, when executed by the at least one hardware processor, may further cause the at least one hardware processor to present a set of manipulable user-interface controls to allow configuration of properties of text generated using the LLM and wherein output of the LLM is to be modified based on one of more of the LoRA models of the plurality of LoRA models selected based on a state of

the user-interface controls as manipulated. The instructions, when executed by the at least one hardware processor, may further cause the at least one hardware processor to provide a preview of LLM output corresponding to a current state of the user-interface controls during presentation and manipulation thereof, wherein providing the preview includes iteratively providing a prompt to the LLM and outputting the output of the LLM responsive to the prompt for each iteration, wherein, for each iteration, the output of the LLM is modified based on one or more of the LoRA models of the plurality of LoRA models selected based on the current state of the user-interface controls as manipulated.

BRIEF DESCRIPTION OF THE DRAWINGS

[0044] Detailed descriptions of implementations of the present invention will be described and explained through the use of the accompanying drawings.

[0045] FIG. 1 is a block diagram illustrating an environment in which a text customization system operates, according to some implementations.

15 [0046] FIG. 2 illustrates an example user interface from which a user can select text properties.

[0047] FIGS. 3A-3D illustrate an example user interface in which a user selection of a first text property causes display of a second set of text properties.

20 [0048] FIGS. 4A-4B illustrate an example user interface with user-interface controls to select various text properties.

[0049] FIGS. 5A-5B illustrate an example user interface on which a selectable range of degrees of a text property are modified based on other selections made by a user.

[0050] FIGS. 6A-6C illustrate an example user interface on which a user selection of a degree of a first text property causes display of a range of degrees of a second set of text properties.

25 [0051] FIG. 7 is a flowchart illustrating a process for customizing LLM text properties, according to some implementations.

[0052] FIG. 8A is a block diagram of a simplified convolutional neural network, which may be used in examples of the present disclosure.

30 [0053] FIG. 8B is a block diagram of a simplified transformer neural network, which may be used in examples of the present disclosure.

[0054] FIG. 9 is a block diagram of an example computing system, which may be used to implement examples of the present disclosure.

[0055] The technologies described herein will become more apparent to those skilled in the art from studying the Detailed Description in conjunction with the drawings. Embodiments or

implementations describing aspects of the invention are illustrated by way of example, and the same references can indicate similar elements. While the drawings depict various implementations for the purpose of illustration, those skilled in the art will recognize that alternative implementations can be employed without departing from the principles of the present technologies. Accordingly, while specific implementations are shown in the drawings, the technology is amenable to various modifications.

DETAILED DESCRIPTION

[0056] A computer system uses adaptation models to customize generative model output generation. Adaptation models are models that may be used to fine-tune a trained generative model. For example, low-rank adaptation models (LoRAs) are models with a significantly lower rank than an LLM (e.g., having hundreds of parameters instead of hundreds of billions of parameters) but that when applied to an LLM modify parameters of the LLM. Given their smaller rank, it can be faster to train LoRAs than to fine-tune an LLM, and the LoRAs typically have a smaller size than the fine-tuned model. In at least some cases, LoRAs can perform better than fine-tuned LLMs when generating text with desired properties.

[0057] In a particular example, a computer system uses low-rank adaptation models (LoRAs) as adaptation models to customize text generation by a large language model (LLM), a type of generative model. For example, this could find application in LLM-based chat agents. Conveniently, in this way, the use of LoRAs to modify an LLM can enable flexible customization of the LLM for a desired purpose.

[0058] According to implementations described herein, a computer system presents a set of manipulable user-interface controls that allow configuration of properties of output generated by a generative model. For example, in the case of the above particular example, the manipulable user-interface controls may allow configuration of properties of text output generated by the LLM. Output of the generative model is modified based on one or more adaptation models. For example, output of the generative model may be modified on one or more LoRA models. Returning to the particular example, output of the LLM may be modified based on one or more LoRA models based on a state of the user-interface controls as manipulated. Such modification may modify output based on one or more adaptation models selected based on a state of the user-interface controls as manipulated. For example, in the particular example, output of the LLM may be modified based on one or more LoRA models that are selected based on a state of the user-interface controls as manipulated.

[0059] In some implementations, the computer system may provide a preview of LLM output corresponding to the current state of the user-interface controls during presentation and manipulation thereof. To provide this preview, the computer system may iteratively provide a prompt to the LLM and outputs the output of the LLM responsive to that prompt for each iteration.

5 For each iteration, the output of the LLM may be modified based on one or more of the LoRA models of the plurality of LoRA models selected based on the current state of the user-interface controls as manipulated. Put another way, the computer system may provide a first prompt as input to the LLM and may then output the output (e.g., as a preview) of the LLM responsive to that prompt, with that output being based on one or more of the LoRA models selected based on the

10 current state of the user-interface controls. Then, as the user-interface controls are manipulated, the preview may be maintained by iteratively providing the prompt to the LLM and outputting the output of the LLM responsive to that prompt for each iteration with the output of the LLM being modified based on one or more LoRA models (which may be the same or different than those applied at the prior iteration) based on the current state of the user-interface controls as

15 manipulated. Conveniently, in this way, the preview can be updated interactively to reflect the effects of manipulating the user-interface controls, and, in particular, any changes in the application and selection of LoRA models according to changes in the state of the user-interface controls as they are manipulation. Notably, this may allow a user to tune the user interface controls more easily as compared, for example, to a system where manual testing of different LoRA

20 selections and applications is required to achieve a desired effect on model output.

[0060] Provision of a computer system whereby manipulable user-interface controls allow for configuration of properties of output will now be discussed in greater detail. The particular example discussed above in which text output of an LLM can be modified by way of manipulation of manipulable user-interface controls will now be discussed in greater detail in order to illustrate

25 the subject matter of the present application in the context of a specific example. It is noted, however, that a skilled person having regard to the following discussion would readily recognize that elements therein such as, for example, the use of an LLM as generative model, LoRA models as adaptation models, particular forms of user interface controls, and the selection between different LoRA models based on the state of the user interface control represent the application of

30 the subject matter of the present application to an example scenario and that the subject matter of the present application is broader and not limited to that particular example.

Text Customization System

[0061] FIG. 1 is a block diagram illustrating an environment 100 in which a text customization system 120 operates, according to some implementations. As shown in FIG. 1, the environment 100 can include a LoRA repository 105, the text customization system 120, and one or more large language models (LLMs) 130. Other implementations of the environment 100 include additional, fewer, or different components.

[0062] The LoRA repository 105 maintains low-rank adaptation (LoRA) models that are usable to modify text properties generated by an LLM. In particular, each LoRA model includes a set of weights that are configured to modify parameters of the LLM to cause the LLM to generate text having a specified property. For example, the repository 105 can include LoRA models that are trained to cause the LLM to generate text that uses certain regional dialects, has certain traits, or increases or decreases the degree of a trait, where such traits include, for example, friendliness, talkativeness, supportiveness, directness, concision or wordiness, gregariousness, reservation, or complementariness.

[0063] LoRAs are generally composable, meaning that multiple LoRAs can be applied to an LLM at the same time to change multiple properties of the LLM-generated text. However, composing multiple LoRAs does not necessarily cause linear effects on the output of an LLM. For example, some LoRAs when composed with certain other LoRAs overwhelm or cancel out the other LoRAs such that the other LoRAs do not have an effect on the text generated by the LLM (or have a lesser effect than if the other LoRAs were applied to the LLM alone). Similarly, the order in which LoRAs are composed can influence whether the LoRAs modify the LLM's parameters in the intended manner. For example, the text generated by an LLM when a first LoRA is applied first and a second LoRA is applied second may have different properties than the text generated when the second LoRA is applied first and the first LoRA applied second.

[0064] The text customization system 120 facilitates customization of LLM-generated text using LoRA models. A plurality of LoRA models can be trained based on different text properties, such that each of the trained LoRA models when applied to an LLM will cause the LLM to produce text with a certain property. The text customization system 120 enables a user to select between the properties associated with the trained LoRA models in order to customize LLM-generated text to the user's preferences or needs.

[0065] In an example application of customizing LLM-generated text, the text customization system 120 enables a user to customize a chat agent for a particular purpose by selecting properties for the text to be output by the chat agent. A business who uses a chat agent on their website to interact with website visitors, for example, can specify properties of the text generated for output by the chat agent, thereby giving the chat agent a certain "personality" or set

of traits. After an operator of the business selects desired properties for the chat agent's text, the corresponding LoRAs can be stored in a repository or instruction set used by the chat agent while it executes a conversation. In another example, a website visitor can customize the chat agent on the website to the visitor's preferences. After the visitor makes selections of the properties the visitor would like the chat agent to use, identifiers of the corresponding LoRAs can be stored to the visitor's browser application where they can be retrieved by the chat agent for use during a conversation. The chat agent can also be automatically customized based on signals associated with the visitor, such as the visitor's age, gender preference, or geographic location. For example, a business that operates the chat agent may specify a certain set of properties or ranges of values for properties that may be used by the chat agent, where specific properties or property values (and the corresponding LoRAs) are selected when the chat agent is accessed by a visitor based on the signals associated with the visitor.

[0066] The text customization system 120 can train at least a subset of the LoRAs in the repository 105. Some implementations of the text customization system 120 train LoRA models based on a training set of text, such as a set of chat conversations, where each item of text has been tagged as containing text with a certain property. For example, the text customization system 120 trains LoRA models using chat conversations between two human users, chat conversations between a chat agent and a human user, or chat conversations between two chat agents. Using the tagged training items, the text customization system 120 modifies weights of a low-rank model until the low-rank model causes an LLM to produce text exhibiting the corresponding property. Some of the LoRA models in the repository 105 can be trained using items of text that are labeled as either exhibiting or not exhibiting a particular text property. When one of these models is applied to an LLM, the LLM produces text that has the corresponding property. Other models can be trained using text items that are labeled as exhibiting a degree of a particular property, such that the trained models cause an LLM to produce text with the corresponding degree of the property. The text customization system 120 can train multiple LoRAs for multiple different degrees of the same property. Applying a model that is trained for a higher degree of a property to an LLM, such as a higher degree of friendliness, causes the LLM to produce text that is more friendly than a model that is trained for a lower degree of the same property.

[0067] Instead of or in addition to the text customization system 120 training LoRA models in the repository 105, some implementations of the LoRA repository 105 include models that were trained by external systems.

[0068] The text customization system 120 enables a user to select the custom text properties to be generated by the LLM based on at least one user input 110. To facilitate these selections,

the text customization system presents a set of manipulable user-interface controls. A user can manipulate the user-interface controls to select desired text properties. As the user manipulates the controls, the text customization system 120 generates a prompt 125 to the LLM 130, which outputs generated text 135 in response to the prompt. The text customization system 120 can display the generated text 135 to the user to provide a preview of text with the properties selected by the user. Based on the preview, the user can determine whether the LLM-generated text is satisfactory or can continue to manipulate the user-interface controls to select different properties if not.

[0069] The manipulable user-interface controls presented by the text customization system 120 can include controls to select or deselect particular properties (e.g., via checkboxes, radio dials, toggles, or drop-down lists) or to select a degree of a property (e.g., via slider bars, dials, or number entry boxes). FIG. 2 illustrates an example user interface 200 from which a user can select text properties. User-interface controls presented on the interface 200 can include controls, such as radio dials 210 and 220, for selecting or deselecting respective text properties. For example, a user can select a regional dialect for text to be generated by the LLM by manipulating one of the radio dials 210. Additionally, as described above, LoRAs can be trained based on text that expresses different degrees of a property. Accordingly, as further shown in FIG. 2, the user-interface controls can include controls, such as sliders 230, for specifying a degree of a corresponding property. Slider 230, for example, represents a range of degrees of the trait “directness,” and a user can increase or decrease a degree of the trait by manipulating the slider 230 (e.g., sliding a thumb of the slider 230 left or right). The slider 230 can enable a user to select continuous or discrete degree values for a property. If, for example, the text customization system 120 has access to LoRA models that are trained to produce four degrees of the property “directness” (e.g., 25%, 50%, 75%, and 100%), the slider 230 can snap to each of these four percentages such that a user can select only one of the four percentage values. Alternatively, the text customization system 120 can enable a user to select any degree value between 0% and 100% but causes one of the LoRA models that is trained for a value closest to the user-selected value to be used to modify the LLM.

[0070] In some implementations, the text customization system 120 presents a first set of user-interface controls that are manipulable to select respective properties of text within a first category of properties. After a user has manipulated one or more controls from the first set to select one or more corresponding properties, the text customization system 120 presents a second set of user-interface controls that are manipulable to select respective properties within a second category of text properties. For example, referring to the example user interface in FIG. 2, the text

customization system 120 can first present user-interface controls to select a regional dialect, as a first category of properties. After the user has selected a desired dialect, the system 120 presents user-interface controls to select properties from among a second category, a trait of the text.

[0071] Once a user has made the desired selections of text properties, the user can select the “preview text” button 240 to review the effect of the selected properties on LLM-generated text. The text customization system 120 determines the LoRA models corresponding to any user-selected properties and generates a prompt to the LLM that identifies the corresponding LoRA models. The prompt can instruct the LLM to generate any type of text for the preview. The user may, for example, ask the LLM to generate a paragraph of text about a desired topic. The text customization system 120 can instead maintain a set of dummy prompts that cause the LLM to generate various items of text. Alternatively, the text customization system 120 generates a mock chat agent interface, where the user can chat with the chat agent and review the types of responses generated by the chat agent when the chat agent is configured according to the user-selected text properties. Each chat input received from the user can be used to generate a corresponding prompt to the LLM.

[0072] Other implementations of the user interface 200 do not provide the “preview text” button 240. For example, the text customization system 120 can automatically generate prompts to the LLM and output text for preview by the user after the user manipulates the user-interface controls.

[0073] In some implementations, the text customization system 120 identifies sets of LoRA models that are composable with one another. A composable set of LoRAs can include two or more LoRA models for which the composed set of LoRAs causes an LLM to generate text that satisfies a performance criterion. The performance criterion, for example, can evaluate whether the LLM-generated text has each of the properties that would be generated if the LoRAs in the set were applied individually to the LLM. Alternatively, the performance criterion can evaluate whether LLM-generated text has properties that are within a threshold degree of each of the properties that would be generated if each LoRA in the set were individually applied. Furthermore, some implementations of the text customization system 120 identifies an order in which the LoRAs in a composable set should be applied to the LLM to cause the LLM to generate text with the intended properties.

[0074] Some of the user-interface controls that are displayed can be identified based on a state of other user-interface controls as manipulated. In particular, the text customization system 120 can modify the presented set of user-interface controls based on the composability of LoRAs

associated with user-selected controls. FIGS. 3A-6C illustrate example user interfaces in which the presented controls are modified based on user selections.

[0075] Referring to FIGS. 3A-3D, illustrated is an example user interface 300 in which a user selection of a first property causes the text customization system 120 to determine and present a second set of properties for which the corresponding LoRA models are composable with the LoRA model of the selected first property. In some implementations, a first LoRA model can be identified based on a current state of the controls as manipulated. Based on the first LoRA model, the text customization system 120 identifies a subset of LoRA models that are composable with the first model (e.g., models that can be composed with the first LoRA model while satisfying a performance criterion). A second set of controls can be provided that enable a user to select between properties corresponding to this subset of LoRA models. For example, in FIG. 3A, text customization system 120 provides user-interface controls by which a user can select a regional dialect for the text to be generated by the LLM. After the user has selected a dialect (e.g., by manipulating the radio dial 310), the text customization system 120 provides options for traits that include directness, friendliness, and supportiveness, as shown in FIG. 3B, which can be traits that the text customization system 120 has determined correspond to LoRAs that can be composed with the LoRA for the selected regional dialect. The user can select one or more of the provided traits (e.g., by manipulating the radio dial 320). If the user instead chooses a different dialect (e.g., by manipulating the radio dial 330 shown in FIG. 3C), a different set of traits are presented to the user (as shown in FIG. 3D) based on the LoRA models for the different traits being composable with the LoRA for the selected dialect. The user can then select one or more of the traits provided for the selected dialect, for example by manipulating the radio dial 340.

[0076] The text customization system 120 can present different sets of properties to a user depending on other properties selected by the user, as illustrated in FIGS. 3A-3D. Alternatively, the text customization system 120 can display the same set of properties regardless of the state of the user-interface controls as manipulated, but can prevent user selection of some properties. For example, if the LoRA that causes an LLM to output text with an American English dialect is not composable with the LoRA that causes the LLM to generate “complimentary” text, the text customization system 120 deactivates a user-interface control associated with the trait “complimentary” to prevent user selection of the trait.

[0077] Furthermore, the text customization system 120 can sequentially present sets of properties for selection by a user, such as sets of properties associated with different categories, as illustrated in FIGS. 3A-3D. For example, the user interfaces 300 first present options to select a dialect (a first category of text properties). Once the dialect has been selected, the text

customization system 120 identifies a set of traits (a second category of text properties) for which the corresponding LoRAs can be composed with the LoRA for the selected dialect. The identified properties are then output for display to and selection by the user. However, the text customization system 120 can alternatively modify the user-interface controls that are presented as a user manipulates the controls. FIG. 4A illustrates an example user interface 400 that provides user-interface controls to select various text properties. When a user selects a first property (e.g., by manipulating a radio dial 410 as shown in FIG. 4B), the text customization system 120 modifies the display of one or more other properties such that the user-interface controls associated with the other properties are not manipulable. For example, FIG. 4B illustrates that the control 420 for the property “talkative” has been deactivated based on the user’s selection of the property “concise” (e.g., because the LoRA trained to produce “talkative” text is not composable with the LoRA trained to produce “concise” text).

[0078] The text customization system 120 can also present user interfaces on which the system modifies the degree of a text property that is selectable by a user in response to other selections by the user. As described above, the text customization system 120 can enable users to select a degree of a property, such as a degree of “directness” of LLM-generated text. Like other LoRA models described herein, the LoRA that is trained to achieve the selected degree may be composable with only a subset of the other models in the LoRA repository 105.

[0079] In some implementations, the text customization system 120 identifies a first LoRA model, corresponding to a first property, based on a current state of the user-interface controls as manipulated. A range of degrees of a second text property are identified based on the LoRA models trained to produce the identified range being composable with the first LoRA model. The text customization system 120 then presents user-interface controls from which the identified range is selectable. Degrees of the property outside the identified range are not selectable via the user-interface controls. FIGS. 5A-5B illustrate an example user interface 500 on which the selectable range of degrees of a property are modified based on other selections made by a user. In FIG. 5A, for example, the user can interact with controls 510 to select a regional dialect. Based on the dialect that is selected, there may be a limited range of degrees of some traits that are suitable to combine with the selected dialect. FIG. 5B illustrates an example in which the trait “directness” has a limited range (e.g., 40%-80%) for which corresponding LoRAs can be composed with the LoRA to produce text with an American English dialect. Accordingly, after the user manipulates the control 510 to select the American English dialect, the text customization system 120 modifies a slider 520 for the directness trait such that the user can only select values between 40% and 80%. For example, the text customization system 120 causes the slider 520 to be displayed with

indicators 525 to identify the selectable range. If a user attempts to manipulate the slider 520 to select a value outside this range, the slider can be configured to snap the thumb to a value within the range, take no action in response to the user's input, or otherwise notify the user that the user attempted to select a non-selectable value.

- 5 [0080] In another example, the text customization system 120 identifies a first LoRA model that corresponds to a first degree of a first property based on a current state of the user-interface controls as manipulated. Like in the example illustrated with respect to FIGS. 5A-5B, the text customization system 120 then identifies a range of degrees of a second property for which corresponding LoRAs are composable with the first LoRA, and modifies the user-interface control
- 10 for the second property to only enable user selection of the identified range of degrees. FIGS. 6A-6C illustrate an example user interface 600 on which a user selection of a degree of a first property causes the text customization system 120 to determine and present a range of degrees of a second set of properties for which the corresponding LoRA models are composable with the LoRA model of the selected degree of the first property. For example, as shown in FIG. 6A, a user can interact
- 15 with a slider 610 to select a degree of the trait "friendliness." Once the user has selected a value, such as 40%, the text customization system 120 modifies a slider 620 for the trait "talkativeness," as illustrated in FIG. 6B, such that only a limited range of the trait (e.g., 50%-75%) is selectable. For example, the slider 620 includes indicators 625 to identify the selectable range to the user. If the user manipulates the slider 610 to select a different value for the friendliness trait (e.g., 70%,
- 20 as illustrated in FIG. 6C), the text customization system 120 can modify the slider 620 to identify a different range of selectable degrees of talkativeness (e.g., 25%-40). In some implementations, if a user attempts to select a degree of talkativeness that is outside the range illustrated in FIG. 6C, the text customization system 120 can modify a degree of friendliness indicated by the slider 610 to a degree that is composable with the degree of talkativeness selected by the user.

25 Customizing LLM-Generated Text using LoRA Models

[0081] Figure 7 is a flowchart illustrating a process 700 for customizing LLM text properties, according to some implementations. The process 700 can be performed by a computer system, such as the text customization system 120. Other implementations of the process 700 include additional, fewer, or different steps, or can perform the steps in different orders.

- 30 [0082] At step 702, the computer system maintains a plurality of LoRA models. Each LoRA model includes a set of weights configured to modify parameters of an LLM to cause the LLM to generate text having a corresponding property. At least some of the LoRA models maintained by the computer system can be trained by the computer system to generate text with particular properties.

[0083] At step 704, the computer system presents a set of manipulable user-interface controls to allow configuration of properties of the text generated by the LLM. For example, the computer system presents controls that are similar to the controls described above with respect to FIGS. 2-6C.

5 [0084] At step 706, one or more LoRA models are selected by the computer system based on a state of the user-interface controls as manipulated, where the output of the LLM can be modified based on the selected LoRA models. As a user interacts with the user-interface controls, the computer system can iteratively update the controls that are displayed or the properties of the controls such that the controls only enable selection of LoRA models that are mutually
10 composable.

[0085] At step 708, the computer system provides a preview of LLM output corresponding to the current state of the user-interface controls during presentation and manipulation of the controls. To provide the preview, the computer system iteratively provides a prompt to the LLM and outputs the output of the LLM responsive to the prompt for each iteration. Each prompt can
15 identify any LoRA models associated with the properties selected by the user based on the manipulation of the controls. In some implementations, when multiple LoRA models are being used to modify multiple properties of the LLM-generated text, the prompt also specifies an order in which the LoRAs are to be composed when they are applied to the LLM. At each iteration of providing the prompt to the LLM, the output of the LLM is modified based on the one or more
20 LoRA models that are selected based on the state of the user-interface controls as manipulated.

[0086] After the user has selected desired properties for the LLM-generated text, the computer system, at step 710, stores identifiers of the LoRA models for later use by a relevant system or application. For example, the process 700 can be used when a user is customizing a chat agent on a website maintained by the user, such that the chat agent produces text that has
25 certain properties desired by the user as the chat agent interacts with visitors to the user's website. The LoRA models for the properties selected by the user can be stored in a repository or instruction set that is accessible to the chat agent during execution of the chat agent. In another example, the process 700 can be used when a user is customizing how a chat agent will interact with the user. In this case, the LoRA models for the properties selected by the user can be stored in a browser
30 application used by the user such that the LoRA models can be used to customize text produced by a chat agent executing in the browser application.

[0087] Having discussed the subject matter of the present application in the context of a particular example, several observations as to example aspects as may be varied in the example without departing from the subject matter of the present application will now be discussed.

[0088] First it is noted again that aspects of the subject matter of the present application such as, for example, the provision of manipulable user-interface controls that configure that of adaptation models in order to customize a generative model and the output thereof is not limited to LoRAs as adaptation models or an LLM as the generative model. For example, a skilled person
5 having regard to the present disclosure would recognize that the subject matter of the present application could also be applied in the context of other generative models such as, for example, image generation models.

[0089] Second, it is noted that terminology used above in the context of the example is readily varied in the context of other implementation choices. For example, a repository of some
10 other form of adaptation models than LoRAs (or even more than one form of adaptation model) may be employed when additional or alternative forms of adaptation model to LoRAs are employed in a particular implementation. Similarly, while a text customization system is discussed in the context of the particular example, depending on the form or forms of output provided by a given generative model being customized using adaptation models selected based on manipulation
15 of user interface controls, such a customization system may take a different name such as, for example, an image customization system, a multimedia customization system, etc.

[0090] Third, it is noted that while sliders may be a suitable user interface control to employ in the context of at least some implementations of the subject matter of the present application, other forms of user interface controls are contemplated (e.g., radio buttons). Furthermore, more
20 than one form of user interface control may be employed in combination in some cases (e.g., sliders and radio buttons).

[0091] Fourth, while, in the process 700, the adaptative models are employed by selecting adaptation models from amongst different adaptation models in an adaptation model repository based on the state of the user interface controls, this is not required. For example, it could be a
25 given adaptation model is parameterized. In such cases, the state of one or more of the user interface controls (e.g., a value determined based on the state of its manipulation) may be employed to adjust such a parameter. In a particular example, one or more of the adaptation models employed in a given scenario may have one or more parameters thereof adjusted based on the state of one or more of the user interface controls. Notably, in some implementations, selection amongst
30 adaptation models (e.g., for varying one aspect of a generative model) could be employed in addition to selection amongst adaptation models (e.g., for varying another aspect of a generative model). For example, it may be that values associated with one or more user interface controls are used to select from amongst adaptation models and values associated with one or more other user interface controls may be used to determine values of parameters of one or more of the selected

adaptation models (e.g., the values of those parameters based be determined based on state or values associated with those other user interface controls). Furthermore, it is noted that adaptation models such as, for example, LoRAs as discussed above, are composable. Accordingly, multiple adaptation models can be applied to a generative model at the same time to change multiple properties of the output or even the degree of a given property of the output. As such, in some cases it may, additionally or alternatively, be that the state of the user interface controls may serve to select two or more adaptation models to employ in order to change one or more output properties and/or degrees thereof for a generative model.

[0092] Fifth, it should be recognized that the provision of a preview is not a required aspect. Furthermore, where such a preview is provided it may take different forms depending on the form of output of the generative model being customized.

[0093] Of course, the above-discussed observations are merely examples of manners in which aspects of the particular illustrative example discussed above may be varied or changed without departing from the subject matter of the present application. A skilled person having regard to the description and claims of the present application would readily recognize that the aspects of that example as may be varied without departing from the subject matter of the application are not necessarily limited to the aspects discussed in the above observations.

Neural Networks and Machine Learning

[0094] To assist in understanding the present disclosure, some concepts relevant to neural networks and machine learning (ML) are now discussed.

[0095] Generally, a neural network comprises a number of computation units (sometimes referred to as “neurons”). Each neuron receives an input value and applies a function to the input to generate an output value. The function typically includes a parameter (also referred to as a “weight”) whose value is learned through the process of training. A plurality of neurons may be organized into a neural network layer (or simply “layer”) and there may be multiple such layers in a neural network. The output of one layer may be provided as input to a subsequent layer. Thus, input to a neural network may be processed through a succession of layers until an output of the neural network is generated by a final layer. This is a simplistic discussion of neural networks and there may be more complex neural network designs that include feedback connections, skip connections, and/or other such possible connections between neurons and/or layers, which need not be discussed in detail here.

[0096] A deep neural network (DNN) is a type of neural network having multiple layers and/or a large number of neurons. The term DNN may encompass any neural network having

multiple layers, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and multilayer perceptrons (MLPs), among others.

[0097] DNNs are often used as ML-based models for modeling complex behaviors (e.g., human language, image recognition, object classification, etc.) in order to improve accuracy of outputs (e.g., more accurate predictions) such as, for example, as compared with models with fewer layers. In the present disclosure, the term “ML-based model” or more simply “ML model” may be understood to refer to a DNN. Training a ML model refers to a process of learning the values of the parameters (or weights) of the neurons in the layers such that the ML model is able to model the target behavior to a desired degree of accuracy. Training typically requires the use of a training dataset, which is a set of data that is relevant to the target behavior of the ML model. For example, to train a ML model that is intended to model human language (also referred to as a language model), the training dataset may be a collection of text documents, referred to as a text corpus (or simply referred to as a corpus). The corpus may represent a language domain (e.g., a single language), a subject domain (e.g., scientific papers), and/or may encompass another domain or domains, be they larger or smaller than a single language or subject domain. For example, a relatively large, multilingual and non-subject-specific corpus may be created by extracting text from online webpages and/or publicly available social media posts. In another example, to train a ML model that is intended to classify images, the training dataset may be a collection of images. Training data may be annotated with ground truth labels (e.g. each data entry in the training dataset may be paired with a label), or may be unlabeled.

[0098] Training a ML model generally involves inputting into an ML model (e.g. an untrained ML model) training data to be processed by the ML model, processing the training data using the ML model, collecting the output generated by the ML model (e.g. based on the inputted training data), and comparing the output to a desired set of target values. If the training data is labeled, the desired target values may be, e.g., the ground truth labels of the training data. If the training data is unlabeled, the desired target value may be a reconstructed (or otherwise processed) version of the corresponding ML model input (e.g., in the case of an autoencoder), or may be a measure of some target observable effect on the environment (e.g., in the case of a reinforcement learning agent). The parameters of the ML model are updated based on a difference between the generated output value and the desired target value. For example, if the value outputted by the ML model is excessively high, the parameters may be adjusted so as to lower the output value in future training iterations. An objective function is a way to quantitatively represent how close the output value is to the target value. An objective function represents a quantity (or one or more quantities) to be optimized (e.g., minimize a loss or maximize a reward) in order to bring the output value as

close to the target value as possible. The goal of training the ML model typically is to minimize a loss function or maximize a reward function.

[0099] The training data may be a subset of a larger data set. For example, a data set may be split into three mutually exclusive subsets: a training set, a validation (or cross-validation) set, and a testing set. The three subsets of data may be used sequentially during ML model training. For example, the training set may be first used to train one or more ML models, each ML model, e.g., having a particular architecture, having a particular training procedure, being describable by a set of model hyperparameters, and/or otherwise being varied from the other of the one or more ML models. The validation (or cross-validation) set may then be used as input data into the trained ML models to, e.g., measure the performance of the trained ML models and/or compare performance between them. Where hyperparameters are used, a new set of hyperparameters may be determined based on the measured performance of one or more of the trained ML models, and the first step of training (i.e., with the training set) may begin again on a different ML model described by the new set of determined hyperparameters. In this way, these steps may be repeated to produce a more performant trained ML model. Once such a trained ML model is obtained (e.g., after the hyperparameters have been adjusted to achieve a desired level of performance), a third step of collecting the output generated by the trained ML model applied to the third subset (the testing set) may begin. The output generated from the testing set may be compared with the corresponding desired target values to give a final assessment of the trained ML model's accuracy. Other segmentations of the larger data set and/or schemes for using the segments for training one or more ML models are possible.

[0100] Backpropagation is an algorithm for training a ML model. Backpropagation is used to adjust (also referred to as update) the value of the parameters in the ML model, with the goal of optimizing the objective function. For example, a defined loss function is calculated by forward propagation of an input to obtain an output of the ML model and comparison of the output value with the target value. Backpropagation calculates a gradient of the loss function with respect to the parameters of the ML model, and a gradient algorithm (e.g., gradient descent) is used to update (i.e., "learn") the parameters to reduce the loss function. Backpropagation is performed iteratively, so that the loss function is converged or minimized. Other techniques for learning the parameters of the ML model may be used. The process of updating (or learning) the parameters over many iterations is referred to as training. Training may be carried out iteratively until a convergence condition is met (e.g., a predefined maximum number of iterations has been performed, or the value outputted by the ML model is sufficiently converged with the desired target value), after which the ML model is considered to be sufficiently trained. The values of the learned parameters

may then be fixed and the ML model may be deployed to generate output in real-world applications (also referred to as “inference”).

[0101] In some examples, a trained ML model may be fine-tuned, meaning that the values of the learned parameters may be adjusted slightly in order for the ML model to better model a specific task. Fine-tuning of a ML model typically involves further training the ML model on a number of data samples (which may be smaller in number/cardinality than those used to train the model initially) that closely target the specific task. For example, a ML model for generating natural language that has been trained generically on publicly-available text corpuses may be, e.g., fine-tuned by further training using the complete works of Shakespeare as training data samples (e.g., where the intended use of the ML model is generating a scene of a play or other textual content in the style of Shakespeare).

[0102] FIG. 8A is a simplified diagram of an example CNN 810, which is an example of a DNN that is commonly used for image processing tasks such as image classification, image analysis, object segmentation, etc. An input to the CNN 810 may be a 2D Red, Green Blue (RGB) image 812.

[0103] The CNN 810 includes a plurality of layers that process the image 812 in order to generate an output, such as a predicted classification or predicted label for the image 812. For simplicity, only a few layers of the CNN 810 are illustrated including at least one convolutional layer 814. The convolutional layer 814 performs convolution processing, which may involve computing a dot product between the input to the convolutional layer 814 and a convolution kernel. A convolutional kernel is typically a 2D matrix of learned parameters that is applied to the input in order to extract image features. Different convolutional kernels may be applied to extract different image information, such as shape information, color information, etc.

[0104] The output of the convolution layer 814 is a set of feature maps 816 (sometimes referred to as activation maps). Each feature map 816 generally has smaller width and height than the image 812. The set of feature maps 816 encode image features that may be processed by subsequent layers of the CNN 810, depending on the design and intended task for the CNN 810. In this example, a fully connected layer 818 processes the set of feature maps 816 in order to perform a classification of the image, based on the features encoded in the set of feature maps 816. The fully connected layer 818 contains learned parameters that, when applied to the set of feature maps 816, outputs a set of probabilities representing the likelihood that the image 812 belongs to each of a defined set of possible classes. The class having the highest probability may then be outputted as the predicted classification for the image 812.

[0105] In general, a CNN may have different numbers and different types of layers, such as multiple convolution layers, max-pooling layers and/or a fully connected layer, among others. The parameters of the CNN may be learned through training, using data having ground truth labels specific to the desired task (e.g., class labels if the CNN is being trained for a classification task, pixel masks if the CNN is being trained for a segmentation task, text annotations if the CNN is being trained for a captioning task, etc.), as discussed above.

[0106] Some concepts in ML-based language models are now discussed. It may be noted that, while the term “language model” has been commonly used to refer to a ML-based language model, there could exist non-ML language models. In the present disclosure, the term “language model” may be used as shorthand for ML-based language model (i.e., a language model that is implemented using a neural network or other ML architecture), unless stated otherwise. For example, unless stated otherwise, “language model” encompasses LLMs.

[0107] A language model may use a neural network (typically a DNN) to perform natural language processing (NLP) tasks such as language translation, image captioning, grammatical error correction, and language generation, among others. A language model may be trained to model how words relate to each other in a textual sequence, based on probabilities. A language model may contain hundreds of thousands of learned parameters or in the case of a large language model (LLM) may contain millions or billions of learned parameters or more.

[0108] In recent years, there has been interest in a type of neural network architecture, referred to as a transformer, for use as language models. For example, the Bidirectional Encoder Representations from Transformers (BERT) model, the Transformer-XL (extra-long) model and the Generative Pre-trained Transformer (GPT) models are types of transformers. A transformer is a type of neural network architecture that uses self-attention mechanisms in order to generate predicted output based on input data that has some sequential meaning (i.e., the order of the input data is meaningful, which is the case for most text input). Although transformer-based language models are described herein, it should be understood that the present disclosure may be applicable to any ML-based language model, including language models based on other neural network architectures such as recurrent neural network (RNN)-based language models.

[0109] FIG. 5B is a simplified diagram of an example transformer 550, and a simplified discussion of its operation is now provided. The transformer 550 includes an encoder 852 (which may comprise one or more encoder layers/blocks connected in series) and a decoder 554 (which may comprise one or more decoder layers/blocks connected in series). Generally, the encoder 852 and the decoder 854 each include a plurality of neural network layers, at least one of which may

be a self-attention layer. The parameters of the neural network layers may be referred to as the parameters of the language model.

[0110] The transformer 850 may be trained on a text corpus that is labelled (e.g., annotated to indicate verbs, nouns, etc.) or unlabelled. LLMs may be trained on a large unlabelled corpus.

5 Some LLMs may be trained on a large multi-language, multi-domain corpus, to enable the model to be versatile at a variety of language-based tasks such as generative tasks (e.g., generating human-like natural language responses to natural language input).

[0111] An example of how the transformer 850 may process textual input data is now described. Input to a language model (whether transformer-based or otherwise) typically is in the
10 form of natural language as may be parsed into tokens. It should be appreciated that the term “token” in the context of language models and NLP has a different meaning from the use of the same term in other contexts such as data security. Tokenization, in the context of language models and NLP, refers to the process of parsing textual input (e.g., a character, a word, a phrase, a sentence, a paragraph, etc.) into a sequence of shorter segments that are converted to numerical
15 representations referred to as tokens (or “compute tokens”). Typically, a token may be an integer that corresponds to the index of a text segment (e.g., a word) in a vocabulary dataset. Often, the vocabulary dataset is arranged by frequency of use. Commonly occurring text, such as punctuation, may have a lower vocabulary index in the dataset and thus be represented by a token having a smaller integer value than less commonly occurring text. Tokens frequently correspond to words,
20 with or without whitespace appended. In some examples, a token may correspond to a portion of a word. For example, the word “lower” may be represented by a token for [low] and a second token for [er]. In another example, the text sequence “Come here, look!” may be parsed into the segments [Come], [here], [,], [look] and [!], each of which may be represented by a respective numerical token. In addition to tokens that are parsed from the textual sequence (e.g., tokens that
25 correspond to words and punctuation), there may also be special tokens to encode non-textual information. For example, a [CLASS] token may be a special token that corresponds to a classification of the textual sequence (e.g., may classify the textual sequence as a poem, a list, a paragraph, etc.), a [EOT] token may be another special token that indicates the end of the textual sequence, other tokens may provide formatting information, etc.

30 **[0112]** In FIG. 8B, a short sequence of tokens 856 corresponding to the text sequence “Come here, look!” is illustrated as input to the transformer 850. Tokenization of the text sequence into the tokens 856 may be performed by some pre-processing tokenization module such as, for example, a byte pair encoding tokenizer (the “pre” referring to the tokenization occurring prior to the processing of the tokenized input by the LLM), which is not shown in FIG. 8B for simplicity.

In general, the token sequence that is inputted to the transformer 850 may be of any length up to a maximum length defined based on the dimensions of the transformer 850 (e.g., such a limit may be 2048 tokens in some LLMs). Each token 856 in the token sequence is converted into an embedding vector 860 (also referred to simply as an embedding). An embedding 860 is a learned numerical representation (such as, for example, a vector) of a token that captures some semantic meaning of the text segment represented by the token 856. The embedding 860 represents the text segment corresponding to the token 856 in a way such that embeddings corresponding to semantically-related text are closer to each other in a vector space than embeddings corresponding to semantically-unrelated text. For example, assuming that the words “look”, “see”, and “cake” each correspond to, respectively, a “look” token, a “see” token, and a “cake” token when tokenized, the embedding 860 corresponding to the “look” token will be closer to another embedding corresponding to the “see” token in the vector space, as compared to the distance between the embedding 860 corresponding to the “look” token and another embedding corresponding to the “cake” token. The vector space may be defined by the dimensions and values of the embedding vectors. Various techniques may be used to convert a token 856 to an embedding 860. For example, another trained ML model may be used to convert the token 856 into an embedding 860. In particular, another trained ML model may be used to convert the token 856 into an embedding 860 in a way that encodes additional information into the embedding 860 (e.g., a trained ML model may encode positional information about the position of the token 856 in the text sequence into the embedding 860). In some examples, the numerical value of the token 856 may be used to look up the corresponding embedding in an embedding matrix 858 (which may be learned during training of the transformer 850).

[0113] The generated embeddings 860 are input into the encoder 852. The encoder 852 serves to encode the embeddings 860 into feature vectors 862 that represent the latent features of the embeddings 860. The encoder 852 may encode positional information (i.e., information about the sequence of the input) in the feature vectors 862. The feature vectors 862 may have very high dimensionality (e.g., on the order of thousands or tens of thousands), with each element in a feature vector 862 corresponding to a respective feature. The numerical weight of each element in a feature vector 862 represents the importance of the corresponding feature. The space of all possible feature vectors 862 that can be generated by the encoder 852 may be referred to as the latent space or feature space.

[0114] Conceptually, the decoder 854 is designed to map the features represented by the feature vectors 862 into meaningful output, which may depend on the task that was assigned to the transformer 850. For example, if the transformer 850 is used for a translation task, the decoder 854

may map the feature vectors 862 into text output in a target language different from the language of the original tokens 856. Generally, in a generative language model, the decoder 854 serves to decode the feature vectors 862 into a sequence of tokens. The decoder 854 may generate output tokens 864 one by one. Each output token 864 may be fed back as input to the decoder 854 in order to generate the next output token 864. By feeding back the generated output and applying self-attention, the decoder 854 is able to generate a sequence of output tokens 864 that has sequential meaning (e.g., the resulting output text sequence is understandable as a sentence and obeys grammatical rules). The decoder 854 may generate output tokens 864 until a special [EOT] token (indicating the end of the text) is generated. The resulting sequence of output tokens 864 may then be converted to a text sequence in post-processing. For example, each output token 864 may be an integer number that corresponds to a vocabulary index. By looking up the text segment using the vocabulary index, the text segment corresponding to each output token 864 can be retrieved, the text segments can be concatenated together and the final output text sequence (in this example, “Viens ici, regarde!”) can be obtained.

[0115] Although a general transformer architecture for a language model and its theory of operation have been described above, this is not intended to be limiting. Existing language models include language models that are based only on the encoder of the transformer or only on the decoder of the transformer. An encoder-only language model encodes the input text sequence into feature vectors that can then be further processed by a task-specific layer (e.g., a classification layer). BERT is an example of a language model that may be considered to be an encoder-only language model. A decoder-only language model accepts embeddings as input and may use auto-regression to generate an output text sequence. Transformer-XL and GPT-type models may be language models that are considered to be decoder-only language models.

[0116] Because GPT-type language models tend to have a large number of parameters, these language models may be considered LLMs. An example GPT-type LLM is GPT-3. GPT-3 is a type of GPT language model that has been trained (in an unsupervised manner) on a large corpus derived from documents available to the public online. GPT-3 has a very large number of learned parameters (on the order of hundreds of billions), is able to accept a large number of tokens as input (e.g., up to 2048 input tokens), and is able to generate a large number of tokens as output (e.g., up to 2048 tokens). GPT-3 has been trained as a generative model, meaning that it can process input text sequences to predictively generate a meaningful output text sequence. ChatGPT is built on top of a GPT-type LLM, and has been fine-tuned with training datasets based on text-based chats (e.g., chatbot conversations). ChatGPT is designed for processing natural language, receiving chat-like inputs and generating chat-like outputs.

[0117] A computing system may access a remote language model (e.g., a cloud-based language model), such as ChatGPT or GPT-3, via a software interface (e.g., an application programming interface (API)). Additionally or alternatively, such a remote language model may be accessed via a network such as, for example, the Internet. In some implementations such as, for example, potentially in the case of a cloud-based language model, a remote language model may be hosted by a computer system as may include a plurality of cooperating (e.g., cooperating via a network) computer systems such as may be in, for example, a distributed arrangement. Notably, a remote language model may employ a plurality of processors (e.g., hardware processors such as, for example, processors of cooperating computer systems). Indeed, processing of inputs by an LLM may be computationally expensive/may involve a large number of operations (e.g., many instructions may be executed/large data structures may be accessed from memory) and providing output in a required timeframe (e.g., real-time or near real-time) may require the use of a plurality of processors/cooperating computing devices as discussed above.

[0118] Inputs to an LLM may be referred to as a prompt, which is a natural language input that includes instructions to the LLM to generate a desired output. A computing system may generate a prompt that is provided as input to the LLM via its API. As described above, the prompt may optionally be processed or pre-processed into a token sequence prior to being provided as input to the LLM via its API. A prompt can include one or more examples of the desired output, which provides the LLM with additional information to enable the LLM to better generate output according to the desired output. Additionally or alternatively, the examples included in a prompt may provide inputs (e.g., example inputs) corresponding to/as may be expected to result in the desired outputs provided. A one-shot prompt refers to a prompt that includes one example, and a few-shot prompt refers to a prompt that includes multiple examples. A prompt that includes no examples may be referred to as a zero-shot prompt.

[0119] FIG. 9 illustrates an example computing system 900, which may be used to implement examples of the present disclosure, such as a prompt generation engine to generate prompts to be provided as input to a language model such as a LLM. Additionally or alternatively, one or more instances of the example computing system 900 may be employed to execute the LLM. For example, a plurality of instances of the example computing system 900 may cooperate to provide output using an LLM in manners as discussed above.

[0120] The example computing system 900 includes at least one processing unit, such as a processor 902, and at least one physical memory 904. The processor 902 may be, for example, a central processing unit, a microprocessor, a digital signal processor, an application-specific integrated circuit (ASIC), a field-programmable gate array (FPGA), a dedicated logic circuitry, a

dedicated artificial intelligence processor unit, a graphics processing unit (GPU), a tensor processing unit (TPU), a neural processing unit (NPU), a hardware accelerator, or combinations thereof. The memory 904 may include a volatile or non-volatile memory (e.g., a flash memory, a random access memory (RAM), and/or a read-only memory (ROM)). The memory 904 may store instructions for execution by the processor 902, to the computing system 900 to carry out examples of the methods, functionalities, systems and modules disclosed herein.

[0121] The computing system 900 may also include at least one network interface 906 for wired and/or wireless communications with an external system and/or network (e.g., an intranet, the Internet, a peer to peer (P2P) network, a WAN and/or a Local Area Network (LAN)). A network interface may enable the computing system 900 to carry out communications (e.g., wireless communications) with systems external to the computing system 900, such as a language model residing on a remote system.

[0122] The computing system 900 may optionally include at least one input/output (I/O) interface 908, which may interface with optional input device(s) 910 and/or optional output device(s) 912. Input device(s) 910 may include, for example, buttons, a microphone, a touchscreen, a keyboard, etc. Output device(s) 912 may include, for example, a display, a speaker, etc. In this example, optional input device(s) 910 and optional output device(s) 912 are shown external to the computing system 900. In other examples, one or more of the input device(s) 910 and/or output device(s) 912 may be an internal component of the computing system 900.

[0123] A computing system, such as the computing system 900 of FIG. 9, may access a remote system (e.g., a cloud-based system) to communicate with a remote language model or LLM hosted on the remote system such as, for example, using an application programming interface (API) call. The API call may include an API key to enable the computing system to be identified by the remote system. The API call may also include an identification of the language model or LLM to be accessed and/or parameters for adjusting outputs generated by the language model or LLM, such as, for example, one or more of a temperature parameter (which may control the amount of randomness or “creativity” of the generated output) (and/or, more generally some form of random seed as serves to introduce variability or variety into the output of the LLM), a minimum length of the output (e.g., a minimum of 10 tokens) and/or a maximum length of the output (e.g., a maximum of 1000 tokens), a frequency penalty parameter (e.g., a parameter which may lower the likelihood of subsequently outputting a word based on the number of times that word has already been output), a “best of” parameter (e.g., a parameter to control the number of times the model will use to generate output after being instructed to, e.g., produce several outputs based on slightly varied inputs). The prompt generated by the computing system is provided to the language

model or LLM and the output (e.g., token sequence) generated by the language model or LLM is communicated back to the computing system. In other examples, the prompt may be provided directly to the language model or LLM without requiring an API call. For example, the prompt could be sent to a remote LLM via a network such as, for example, as or in message (e.g., in a payload of a message).

Remarks

[0124] The terms “example”, “embodiment” and “implementation” are used interchangeably. For example, reference to “one example” or “an example” in the disclosure can be, but not necessarily are, references to the same implementation; and, such references mean at least one of the implementations. The appearances of the phrase “in one example” are not necessarily all referring to the same example, nor are separate or alternative examples mutually exclusive of other examples. A feature, structure, or characteristic described in connection with an example can be included in another example of the disclosure. Moreover, various features are described which can be exhibited by some examples and not by others. Similarly, various requirements are described which can be requirements for some examples but no other examples.

[0125] The terminology used herein should be interpreted in its broadest reasonable manner, even though it is being used in conjunction with certain specific examples of the invention. The terms used in the disclosure generally have their ordinary meanings in the relevant technical art, within the context of the disclosure, and in the specific context where each term is used. A recital of alternative language or synonyms does not exclude the use of other synonyms. Special significance should not be placed upon whether or not a term is elaborated or discussed herein. The use of highlighting has no influence on the scope and meaning of a term. Further, it will be appreciated that the same thing can be said in more than one way.

[0126] Unless the context clearly requires otherwise, throughout the description and the claims, the words “comprise,” “comprising,” and the like are to be construed in an inclusive sense, as opposed to an exclusive or exhaustive sense; that is to say, in the sense of “including, but not limited to.” As used herein, the terms “connected,” “coupled,” or any variant thereof means any connection or coupling, either direct or indirect, between two or more elements; the coupling or connection between the elements can be physical, logical, or a combination thereof. Additionally, the words “herein,” “above,” “below,” and words of similar import can refer to this application as a whole and not to any particular portions of this application. Where context permits, words in the above Detailed Description using the singular or plural number may also include the plural or singular number respectively. The word “or” in reference to a list of two or more items covers all of the following interpretations of the word: any of the items in the list, all of the items in the list,

and any combination of the items in the list. The term “module” refers broadly to software components, firmware components, and/or hardware components.

[0127] While specific examples of technology are described above for illustrative purposes, various equivalent modifications are possible within the scope of the invention, as those skilled in the relevant art will recognize. For example, while processes or blocks are presented in a given order, alternative implementations can perform routines having steps, or employ systems having blocks, in a different order, and some processes or blocks may be deleted, moved, added, subdivided, combined, and/or modified to provide alternative or sub-combinations. Each of these processes or blocks can be implemented in a variety of different ways. Also, while processes or blocks are at times shown as being performed in series, these processes or blocks can instead be performed or implemented in parallel, or can be performed at different times. Further, any specific numbers noted herein are only examples such that alternative implementations can employ differing values or ranges.

[0128] Details of the disclosed implementations can vary considerably in specific implementations while still being encompassed by the disclosed teachings. As noted above, particular terminology used when describing features or aspects of the invention should not be taken to imply that the terminology is being redefined herein to be restricted to any specific characteristics, features, or aspects of the invention with which that terminology is associated. In general, the terms used in the following claims should not be construed to limit the invention to the specific examples disclosed herein, unless the above Detailed Description explicitly defines such terms. Accordingly, the actual scope of the invention encompasses not only the disclosed examples, but also all equivalent ways of practicing or implementing the invention under the claims. Some alternative implementations can include additional elements to those implementations described above or include fewer elements.

[0129] To reduce the number of claims, certain implementations are presented below in certain claim forms, but the applicant contemplates various aspects of an invention in other forms. For example, aspects of a claim can be recited in a means-plus-function form or in other forms, such as being embodied in a computer-readable medium. A claim intended to be interpreted as a mean-plus-function claim will use the words “means for.” However, the use of the term “for” in any other context is not intended to invoke a similar interpretation. The applicant reserves the right to pursue such additional claim forms in either this application or in a continuing application.

CLAIMS

We claim:

1. A computer-implemented method comprising:
maintaining a plurality of adaptation models that each include a set of weights configured
5 to modify parameters of a generative model to cause the generative model to
generate output having a corresponding property;
presenting a set of manipulable user-interface controls to allow configuration of properties
of output generated using the generative model and wherein output of the
generative model is to be modified based on one or more of the adaptation models
10 of the plurality of adaptation models based on a state of the user-interface controls
as manipulated.
2. The computer-implemented method of claim 1, further comprising:
providing a preview of generative model output corresponding to a current state of
15 the user-interface controls during presentation and manipulation thereof,
wherein providing the preview includes providing input to the generative
model and outputting the output of the generative model responsive to the
input, wherein the output of the generative model is modified based on the
one or more of the adaptation models of the plurality of adaptation models
20 based on the state of the user-interface controls as manipulated.
3. The computer-implemented method of claim 2, further comprising:
maintaining the preview of generative model output corresponding to the current
state of the user-interface controls during the presentation and manipulation
25 thereof, wherein maintaining the preview includes iteratively providing
input to the generative model and outputting the output of the generative
model responsive to the input for each iteration, wherein, for each iteration,
the output of the generative model is modified based on the one or more of
the adaptation models of the plurality of adaptation models selected based
30 on the current state of the user-interface controls as manipulated.
4. The computer-implemented method of any one of claims 1 to 3, further comprising:
identifying a first adaptation model of the plurality of adaptation models based on the
current state of the user-interface controls as manipulated, wherein the first

adaptation model corresponds to a first property of output generated by the generative model; and

identifying a range of degrees of a second property of output generated by the generative model based on an adaptation model corresponding to a degree in the identified range being composable with the first adaptation model;

wherein presenting the set of manipulable user-interface controls comprises displaying indications of degrees of the second property in the identified range as selectable and displaying indications of degrees of the second property that are not in the identified range as not selectable.

5. The computer-implemented method of any one of claims 1 to 3, wherein presenting the set of user-interface controls comprises:

identifying a first adaptation model of the plurality of adaptation models based on the current state of the user-interface controls as manipulated, wherein the first adaptation model corresponds to a first property of output generated by the generative model;

identifying a subset of the plurality of adaptation models based on the first adaptation model; and

presenting user-interface controls associated with the properties corresponding to the identified subset of adaptation models that are manipulable to select one or more of the corresponding properties.

6. The computer-implemented method of any one of claims 1 to 5, wherein presenting the set of user-interface controls comprises:

presenting a first user-interface control that is manipulable to change a degree of a first property of the output generated by the generative model;

wherein manipulation of the set of user-interface controls a selection of the degree of the first property.

7. The computer-implemented method of any one of claims 1 to 5, wherein presenting the set of user-interface controls comprises:

presenting a first set of user-interface controls that are manipulable to select respective properties of generative model output within a first category of properties; and

in response to receiving manipulation of a user-interface control of the first set of user-interface controls to select a property from among the first category of properties, presenting a second set of user-interface controls that are manipulable to select respective properties of generative model output within a second category of properties.

8. The computer-implemented method of claim 7, wherein the properties within the second category of properties are determined based on the selected property from among the first category of properties.

9. The computer-implemented method of any one of claims 1 to 8, wherein the set of user-interface controls are output for display by a browser application, and wherein the method further comprises:

storing identifiers of the plurality of adaptation models in the browser application.

10. The computer-implemented method of any one of claims 1 to 9, wherein the generative model is a large language model (LLM) and wherein the output of the generative model is or includes text.

11. The computer-implemented method of claim 10, wherein the set of user-interface controls are presented to allow configuration of properties of text generated by the LLM for output by a chat agent.

12. The computer-implemented method of claim 10 or claim 11, further comprising: storing identifiers of the plurality of adaptation models in a repository or instruction set that is accessible to the chat agent during execution of the chat agent.

13. The computer-implemented method of any one of claims 10 to 12, wherein providing the prompt to the LLM comprises specifying an order in which the one or more adaptation models selected based on the current state of the user-interface controls as manipulated are to be composed.

14. The computer-implemented method of any one of claims 10 to 13, further comprising:

training the plurality of adaptation models based on a set of chat conversations that have been tagged as containing the property corresponding to each of the plurality of adaptation models.

5 15. The computer-implemented method of any one of claims 1 to 9, wherein the generative model is an image generation model and wherein the output of the generative model is or includes one or more images.

10 16. The computer-implemented method of claim 15, wherein the generative model is a diffusion model.

 17. The computer-implemented method of any one of claims 1 to 16, wherein the one or more of the adaptation models of the plurality of adaptation models are selected based on a state of the user-interface controls as manipulated.

15 18. The computer-implemented method of any one of claims 1 to 17, wherein the adaptation models are low-rank adaptation (LoRA) models.

20 19. A computer readable storage medium storing executable instructions, execution of which by a processor causes the processor to perform the method of any one of claims 1 to 18.

 20. A system comprising:
 at least one hardware processor; and
25 at least one memory storing instructions, which, when executed by the at least one hardware processor, cause the system to perform the method of any one of claims 1 to 18.

30 21. A computer program product comprising instructions which, when executed by at least one hardware processor, cause the at least one hardware processor to carry out the method of any one of claims 1 to 18.