

Logical Language Model Prompts for Fraud Detection

CROSS-REFERENCE TO RELATED APPLICATIONS

5 **[0001]** This application is a continuation of US patent application No. 18/650,276, filed 4/30/2024, titled "Dynamic Language Model Prompts for Fraud Detection," which in turn claims the benefit of the filing date of U.S. provisional patent application No. 63/612,405, filed on 12/20/2023, titled "Dynamic Inline Editing of Language Model Prompts," the content of both of which is incorporated by reference herein.

BACKGROUND OF THE INVENTION

10 **[0002]** The invention relates to computer security, and in particular to preventing online fraud such as phishing, among others.

[0003] Online fraud, especially in the form of phishing and identity theft, has been posing an increasing threat to Internet users worldwide. Sensitive identity information such as user names, IDs, passwords, social security and medical records, bank and credit card
15 details obtained fraudulently by international criminal networks operating on the Internet are used to withdraw private funds and/or are further sold to third parties. Beside direct financial damage to individuals, online fraud also causes a range on unwanted side effects, such as increased security costs for companies, higher retail prices and banking fees, declining stock values, lower wages and decreased tax revenue.

20 **[0004]** Online fraud is facilitated by the explosive growth of mobile computing and online services, with millions of devices such as smartphones and tablet computers constantly connected to the Internet and acting as potential targets. In a typical example of phishing, a user receives a fraudulent communication masquerading as a legitimate message from a service provider such as a bank, phone company, online retailer, etc. The message may
25 report a fictitious problem with the user's account or recent order and invite the user to contact the respective service provider via a link included in the respective message. The link may lead to a fake interface (e.g., webpage) used by online criminals to steal sensitive data such as login credentials and credit card numbers, among others. Accessing such links may further expose the user to a risk of installing malicious software.

[0005] Various security software may be used to detect fraudulent webpages and/or phishing messages. However, using such software may require installing a local security agent on the user's computing device, and may further require a certain level of knowledge about online communications, computer security, and/or types of online threats, which is expected to exceed that of an ordinary user. Furthermore, the methods used by cybercriminals to trick users into revealing sensitive information are continuously changing, so users and systems need to constantly adapt.

[0006] Modern anti-fraud systems and methods are increasingly relying on artificial intelligence (AI), and in particular on generative language models such as ChatGPT® from OpenAI, Inc., among others. A typical example of such methods comprises formulating a language model prompt to include a text sample (e.g., a suspect message received by a client), and feeding the respective prompt as input to a language model. The model may then process the respective prompt and reply with another text indicating whether the respective text sample is indicative of fraud. However, such approaches have so far proven less reliable than classical anti-fraud methods. Typical language models are pre-trained on a generic corpus of text, and therefore largely lack specific knowledge of online fraud. Furthermore, designing, training, and operating large language models require a hefty investment in computational resources and know-how, so developing such a model explicitly for fraud detection may fail a cost-benefit analysis. Therefore, there is an ongoing interest in developing reliable, cost-effective, and user-friendly methods of using AI in combating online fraud.

SUMMARY OF THE INVENTION

[0007] According to one aspect, a computer system comprises at least one hardware processor configured to analyze a target text formulated in a natural language, wherein analyzing the target text comprises executing a prompt manager communicatively coupled to a generative language module (GLM). The GLM is configured to receive from the prompt manager a language model (LM) prompt formulated in the natural language, and in response, output to the prompt manager a predicted token comprising a likely continuation of the received LM prompt. The prompt manager is configured to receive a logical prompt, the logical prompt including the target text and a code snippet comprising computer code for updating the LM prompt. The prompt manager is further configured to

formulate the LM prompt according to the logical prompt, the LM prompt instructing the GLM to perform a text analysis task according to the target text and in response, to output a flag token. The prompt manager is further configured to, in response to receiving the predicted token, initiate an execution of the code snippet to produce an updated LM prompt, transmit the updated LM prompt to the GLM, and formulate a reply message comprising a result of analyzing the target text, the reply message formulated according to an output of the GLM in response to the updated LM prompt.

[0008] According to another aspect, a computer-implemented method of analyzing a target text formulated in a natural language comprises employing at least one hardware processor of a computer system to execute a prompt manager communicatively coupled to a GLM. The GLM is configured to receive from the prompt manager an LM prompt formulated in the natural language, and in response, output to the prompt manager a predicted token comprising a likely continuation of the received LM prompt. Executing the prompt manager comprises receiving a logical prompt, the logical prompt including the target text and a code snippet comprising computer code for updating the LM prompt. Executing the prompt manager further comprises formulating the LM prompt according to the logical prompt, the LM prompt instructing the GLM to perform a text analysis task according to the target text and in response, to output a flag token. Executing the prompt manager further comprises, in response to receiving the predicted token, initiating an execution of the code snippet to produce an updated LM prompt, transmitting the updated LM prompt to the GLM, and formulating a reply message comprising a result of analyzing the target text, the reply message formulated according to an output of the GLM in response to the updated LM prompt.

[0009] According to another aspect, a non-transitory computer-readable medium stores instructions which, when executed by at least one hardware processor of a computer system, cause the computer system to analyze a target text formulated in a natural language, wherein analyzing the target text comprises executing a prompt manager communicatively coupled to a GLM. The GLM is configured to receive from the prompt manager an LM prompt formulated in the natural language, and in response, output to the prompt manager a predicted token comprising a likely continuation of the received LM prompt. The prompt manager is configured to receive a logical prompt, the logical prompt including the target

text and a code snippet comprising computer code for updating the LM prompt. The prompt manager is further configured to formulate the LM prompt according to the logical prompt, the LM prompt instructing the GLM to perform a text analysis task according to the target text and in response, to output a flag token. The prompt manager is further configured to, in response to receiving the predicted token, initiate an execution of the code snippet to produce an updated LM prompt, transmit the updated LM prompt to the GLM, and formulate a reply message comprising a result of analyzing the target text, the reply message formulated according to an output of the GLM in response to the updated LM prompt.

10 BRIEF DESCRIPTION OF DRAWINGS

[0010] The foregoing aspects and advantages of the present invention will become better understood upon reading the following detailed description and upon reference to the drawings where:

15 [0011] Fig. 1 shows exemplary components of a system for preventing online fraud according to some embodiments of the present invention.

[0012] Fig. 2 shows an exemplary sequence of steps performed by a generative language module (GLM) according to some embodiments of the present invention.

[0013] Fig. 3 illustrates an exemplary operation of the GLM according to some embodiments of the present invention.

20 [0014] Fig. 4 shows a sequence of inference steps performed by the GLM according to some embodiments of the present invention.

[0015] Fig. 5 shows an exemplary language model (LM) prompt according to some embodiments of the present invention.

25 [0016] Fig. 6 shows an exemplary instruction cache associating flag tokens with LM instructions and code snippets, according to some embodiments of the present invention.

[0017] Fig. 7 shows an exemplary sequence of steps carried out by a prompt manager module according to some embodiments of the present invention.

30 [0018] Fig. 8 shows another exemplary sequence of steps performed by the prompt manager to update the LM prompt in-between consecutive inferences according to some embodiments of the present invention.

[0019] Fig. 9 illustrates exemplary alternative LM prompt changes corresponding to distinct outputs of the generative language module, according to some embodiments of the present invention.

[0020] Fig. 10 shows an alternative exemplary system for preventing online fraud according to some embodiments of the present invention.

[0021] Fig. 11 illustrates an exemplary logical prompt according to some embodiments of the present invention.

[0022] Fig. 12 shows another exemplary sequence of steps carried out by the prompt manager according to some embodiments of the present invention.

[0023] Fig. 13 shows an exemplary hardware configuration of a computer system programmed to execute some of the methods described herein.

[0024] Fig. 14 illustrates an exemplary conversation interface enabling the prevention of online fraud according to some embodiments of the present invention.

[0025] Fig. 15 shows another example of logical prompt according to some embodiments of the present invention directed at detecting fake news.

[0026] Fig. 16 illustrates an exemplary evolution of the LM prompt caused by processing the logical prompt in Fig. 15 according to some embodiments of the present invention.

[0027] Fig. 17 shows yet another example of logical prompt according to some embodiments of the present invention directed at detecting bullying and/or hate speech.

[0028] Fig. 18 illustrates an exemplary evolution of the LM prompt caused by processing the logical prompt in Fig. 17 according to some embodiments of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0029] In the following description, it is understood that all recited connections between structures can be direct operative connections or indirect operative connections through intermediary structures. A set of elements includes one or more elements. Any recitation of an element is understood to refer to at least one element. A plurality of elements includes at least two elements. Any use of 'or' is meant as a nonexclusive or. Unless otherwise required, any described method steps need not be necessarily performed in a particular illustrated order. A first element (e.g., data) derived from a second element encompasses a first element equal to the second element, as well as a first element generated by processing the second element and optionally other data. Making a determination or

decision according to a parameter encompasses making the determination or decision according to the parameter and optionally according to other data. Unless otherwise specified, an indicator of some quantity/data may be the quantity/data itself, or an indicator different from the quantity/data itself. A computer program is a sequence of processor instructions carrying out a task. Computer programs described in some embodiments of the present invention may be stand-alone software entities or sub-entities (e.g., subroutines, libraries) of other computer programs. A database or knowledgebase herein denotes any organized, searchable collection of data. Computer-readable media encompass non-transitory media such as magnetic, optic, and semiconductor storage media (e.g., hard drives, optical disks, flash memory, DRAM), as well as communication links such as conductive cables and fiber optic links. According to some embodiments, the present invention provides, *inter alia*, computer systems comprising hardware (e.g., one or more processors) programmed to perform the methods described herein, as well as computer-readable media encoding instructions to perform the methods described herein.

[0030] The following description illustrates embodiments of the invention by way of example and not necessarily by way of limitation.

[0031] Fig. 1 shows some exemplary components and operation of a fraud prevention system according to some embodiments of the present invention. The illustrated system comprises a security chatbot **10** communicatively coupled to a messaging system **14**. Components of chatbot **10** may be embodied as computer programs executing on hardware processors of a computer system, for instance on a server computer system carrying out fraud-prevention transactions with a plurality of client front-end devices as detailed below. However, a skilled artisan will know that some or all of the functionality of chatbot **10** may also be implemented in dedicated hardware such as field-programmable gate arrays (FPGA) or application-specific integrated circuits (ASIC), or in a combination of hardware and software.

[0032] In some embodiments, chatbot **10** comprises an artificial intelligence (AI) system configured to carry out a conversation (i.e., exchange of messages) with a user in a natural language (NL) such as English or Chinese, among others. Chatbot **10** may be further configured to determine according to a content of the respective conversation whether the user is confronted with a computer security threat, such as online fraud, malware, etc.

Alternatively or additionally, chatbot **10** may perform other services such as automated classification of online content (e.g., detection of online propaganda, fake news, and AI-generated content), among others. Some embodiments of chatbot **10** are further configured to provide various other information to the user, for instance answer general questions and offer advice on various computer security subjects such as malicious software, spam, communication privacy, securing online payments, parental control, etc. Chatbot **10** may further advise the user on purchasing computer security software, manage the user's subscriptions to various computer security services, answer billing questions, or in any other way act as a user-friendly interface between the user and a computer security service provider.

[0033] In an exemplary fraud prevention scenario illustrated in Fig. **1**, the user may ask chatbot **10** whether a message he/she has recently received could be fraudulent. Chatbot **10** may receive a copy of the respective message (illustrated as an exemplary target message **15**), perform a fraud detection procedure as detailed below, and return a reply message **16** comprising a result of the procedure back to the user.

[0034] In some embodiments, chatbot **10** interacts with human users via a user interface displayed on a front-end device **12** (Fig. **1**). Exemplary front-end devices **12** include personal computers, laptops, tablet computers, mobile telecommunication devices (e.g., smartphones), media players, TVs, game consoles, home appliances (e.g., refrigerators, thermostats, intelligent heating and/or lighting systems), and wearable devices (e.g., smartwatches, sports and fitness equipment), among others. A basic user interface according to some embodiments comprises a communication interface enabling the user to interact with chatbot **10** in a natural language, for instance to ask questions, transmit and/or request various data, and/or receive results of various computer security tasks. An exemplary communication interface as described herein is illustrated in Fig. **16**. The communication interface may be integrated with other computer security functionality, for instance with a dashboard for configuring and displaying various security settings and/or for displaying a current security status of a front-end device **12**. In exemplary embodiments, the user interface may display an indicator of whether device **12** includes malicious software, an indicator of whether device **12** is currently connected to a virtual private network (VPN), etc. Other exemplary content displayed by the respective user

interface may include an indicator of a status and/or details of a user account/service level agreement (SLA)/subscription for using security chatbot **10** or other security software. A user interface as described herein may be organized in any manner known in the art, for instance to include a variety of visual elements (e.g., dials, gauges, charts), each indicating a value of a current security setting and/or a value of a monitored quantity. Some visual aspects of the user interface may be customizable, such as a color scheme, position, and content of various screen areas, etc.

[0035] In other exemplary embodiments, the user may interact with chatbot **10** via communication interfaces of an online messaging application executing on front-end device **12**. Online messaging herein encompasses peer-to-peer messaging as well as messaging via public chatrooms, forums, social media sites, etc. Examples of online messaging include an exchange of short message service (SMS) messages, a sequence of e-mail messages, and a sequence of messages exchanged via instant messaging applications such as WhatsApp Messenger®, Telegram®, WeChat®, and Facebook® Messenger®, among others. Other exemplary online messaging include a content of a Facebook® wall, a chat conducted on an online forum such as Reddit® and Discord®, and a set of comments to a blog post. Exemplary online messaging applications according to embodiments of the present invention include client-side instances of mobile applications such as WhatsApp®, Facebook®, Instagram®, SnapChat® etc., as well as software executing the server side of the respective messaging operations. Other examples of online messaging applications include an email client and an instance of an Internet browser.

[0036] For clarity, the present description will focus on communication interfaces that enable the user to carry out a natural language conversation by typing. In other words, exchanges between the user and security chatbot **10** described herein are predominantly in text form. However, a skilled artisan will know that this aspect is not meant to be limiting. The described systems and methods may be adapted to processing of audio messages (spoken conversations), video messages, or any combination of carrier media. In such embodiments, chatbot **10** may be configured to process the respective type of input directly, or alternatively, to convert the type of input provided by the user into text before applying some of the methods described herein. Furthermore, in some embodiments, a communication interface as described herein may enable the user to attach various types

of media files (e.g., an image/screenshot, an audio file such as a recorded voice message, etc.) to a text message.

[0037] Some embodiments as illustrated in Fig. 1 employ a messaging system 14 to transmit messages between front-end device 12 and security chatbot 10. Messaging system 14 generically represents any messaging and electronic communication functionality that goes beyond the scope of the present invention. For instance, messaging system 14 may represent hardware and/or software implementing a conventional electronic communication service such as an email service, short message service (SMS), and instant messaging services like WhatsApp®, iMessage®, and Microsoft Teams®, among others. Messaging system 14 may for instance aggregate messages from multiple front-end devices 12, route, and/or selectively deliver such messages to their intended destination. In some exemplary embodiments, front-end device 12 may invoke a local instance of a conventional online messaging application to enable the user to send and/or receive messages 15-16 from/at device 12. In turn, chatbot 10 may transmit and/or receive the respective messages via application programming interface (API) calls to a remote web service exposed by messaging system 14. The messages themselves may be routed by messaging system 14 via third-party servers while traveling between front-end device 12 and the computer(s) implementing fraud prevention system 10.

[0038] The actual data exchanged during messaging may vary in format according to the respective messaging platform, protocol, and/or application, but in general such may comprise an encoding of a text and/or an encoding of a media file (e.g., image, movie, sound, etc.). The text part may comprise text written in a natural language, as well as other alphanumeric and/or special characters such as emoticons, among others. An encoding of messages 15-16 may further include identifiers of a sender and receiver of the respective message and a timestamp indicative of a time of transmission of the respective message. Such metadata may enable chatbot 10 to associate each message with an ongoing conversation, and to maintain a conversation context for each conversation, for instance by arranging messages in sequence according to their respective timestamps.

[0039] Some embodiments of chatbot 10 may maintain a plurality of concurrent conversations with various users on various subjects. Internally, chatbot 10 may represent each conversation as a separate data structure (e.g., an object with multiple data fields)

identified by a unique conversation ID. A conversation object may be formulated according to any data standard known in the art, and may include a user_ID identifying front-end device **12** and/or an individual user of the respective device. The conversation object may further include a plurality of message indicators, each corresponding to an individual message exchanged within the respective conversation. Each individual message indicator may in turn include an identifier of a sender and/or of a receiver, a text content of the respective message, and a timestamp indicating a moment in time when the respective message was sent and/or received. In an alternative embodiment, a conversation object may comprise a concatenation of the text content of all messages in the respective conversation, individual messages arranged in the order of transmission according to their respective timestamp. Message indicators may further include a set of media indicators, for instance copies of an image/video/audio file attached to the respective message, or a network address/URL where the respective media file is located. Some embodiments keep a conversation alive as long as its count of messages does not exceed a predetermined value, as long as the time elapsed since its first message does not exceed a predetermined time threshold, and/or as long as a time elapsed since its latest message does not exceed another predetermined time threshold.

[0040] To carry out natural language conversations with human users, some embodiments of chatbot **10** (Fig. 1) include a generative language module (GLM) **40** to generate synthetic sentences, questions and/or answers. GLM **40** comprises an implementation of a computational model of a natural language, for instance a set of artificial neural networks pre-trained on a corpus of text formulated in the respective natural language. Exemplary language models include probabilistic *n*-gram models, language models implemented using recurrent neural networks, and large language models (LLMs) implemented using generative pre-trained transformers (GPT), among others. Language module **40** is herein deemed 'generative' in the sense that it is configured to input a sequence of words (e.g., a sentence or a question) and in response, automatically generate another sequence of words (e.g., a plausible continuation or a reply) according to the input word sequence. The structural and operational details of GLM **40** go beyond the scope of the present invention. GLM **40** may be implemented using any method known in the art of artificial intelligence. In some embodiments, GLM **40** implements an instance of a pre-trained, off-the-shelf

LLM such as GPT-3 from OpenAI, LLaMA from Meta AI, and Mistral from Mistral AI, among others.

5 [0041] The operation of GLM 40 is illustrated in Figs. 2-3. As shown generically in Fig. 3, GLM 40 is configured to receive a language model (LM) prompt 22 comprising a sequence of tokens 24a-d, and in response, output a predicted token 26 determined according to LM prompt 22, predicted token 26 comprising a likely continuation of the sequence of tokens 24a-d of LM prompt 22. The set of calculations carried out by GLM 40 to produce an individual predicted token (not including the LM initialization steps described below) are herein collectively denoted as one inference step. When the architecture of GLM 40 is based on neural networks, such calculations may comprise matrix multiplications and evaluating a set of activation functions, among others.

10 [0042] Exemplary tokens 24a-d and 26 may comprise individual words, but also numbers, punctuation marks, special characters, abbreviations, initialisms (e.g., LOL, ROFL), emojis, as well as network addresses, universal record identifiers (URI) and locators (URL), among others. Some tokens may include multiple words such as phrases, etc. Some GLMs 40 are further configured to input fragments of computer code. For instance, individual tokens 24a-d and 26 may include computer instructions, variable names and values, mathematical symbols, etc.

15 [0043] Fig. 2 shows a sequence of steps carried out by GLM 40 during a fraud detection procedure according to some embodiments of the present invention. A fraud detection procedure herein comprises formulating an initial LM prompt, initializing GLM 40, and carrying out a sequence of inference steps until a termination condition is satisfied. LM prompt 22 may be updated in between consecutive inference steps, as described in detail below.

20 [0044] In a step 102 (Fig. 2), GLM 40 receives LM prompt 22. In some embodiments as illustrated in Fig. 1, LM prompt 22 is received from a prompt manager module 30, the operation of which is described in detail below. A step 104 determines whether the current LM prompt is an initial prompt, and if yes, a steps 106 may initialize GLM 40 in preparation for inference. Initialization may include operations such as memory allocation and loading necessary data structures into memory (e.g., a set of synapse weights determined in training, etc.), among others. A further step 108 may perform a tokenization

25

30

of LM prompt **22**, said tokenization comprising breaking up LM prompt **22** into individual tokens according to a pre-determined vocabulary and set of rules, and representing each token as a vector of numbers. In one exemplary embodiment, tokenization may determine a one-hot encoding of individual tokens, wherein each token is represented as a $N \times 1$ binary vector having all zeros except a 1 on the M -th row, wherein N is the size of the vocabulary (typically several thousand words/tokens), and M is the position of the respective token within the vocabulary.

[0045] Next, a step **110** may compute token embeddings for all tokens of the initial LM prompt. An embedding herein denotes an internal representation of an individual token, comprising a set of coordinates of the respective token in an abstract multidimensional space typically referred to as an embedding space. The size (count of dimensions) of a typical embedding space ranges between several thousand and several tens of thousands. In an exemplary embodiment using a deep neural network, GLM **40** comprises a sandwich of interconnected neural network layers. A first subset of these layers collectively operate as an encoder, taking each token of LM prompt **22** as input and calculating an embedding of the respective token, i.e., projecting each respective token into the embedding space. In typical embodiments, step **110** comprises a matrix multiplication between the numerical representation of a respective token determined in step **108** and a matrix of synapse weights pre-determined during training of GLM **40**. In some embodiments, step **110** further determines positional embeddings indicative of a position of each token **24a-d** within LM prompt **22**.

[0046] Following initialization, tokenization, and embedding, GLM **40** may iterate through a series of consecutive inferences, until some termination condition is satisfied (a step **114** returns a YES). At each inference, a step **116** may determine a predicted token (see token **26** in Fig. **3**). In embodiments using a deep neural network, another set of network layers of GLM **40** act as a decoder, computing predicted token **26** according to the plurality of token embeddings of LM prompt **22**. Such calculations may again comprise matrix multiplications of token embeddings with synapse weights determined through training.

[0047] In some embodiments, LM prompt **22** is updated in between consecutive inference steps, and a new cycle is started with step **102**. However, when not dealing with an initial

LM prompt, some embodiments may save computational resources by re-using some of the token embeddings already calculated in previous inference steps. As illustrated in Fig. 2, when step 104 returns a NO, a step 112 may only tokenize and embed any new tokens found in the LM prompt. Some embodiments further update positional embeddings to reflect any changes in the position of individual tokens within the LM prompt.

[0048] In conventional language model applications as illustrated in Fig. 4, after each inference, a new LM prompt is generated by appending the currently predicted token to the previous LM prompt. The new LM prompt is then fed back to the language model for a new inference. An exemplary LM prompt may thus progressively evolve from version 22a through to version 22e in four consecutive inference steps. The sequence of predicted tokens generated during a fraud detection procedure (for instance, exemplary token sequence 26a-e in Fig. 4) is herein deemed predicted token sequence 27.

[0049] In contrast to such conventional prompting, in some embodiments of the present invention, LM prompt 22 may be updated in-between consecutive inference steps in ways that differ from and may be more substantial than merely appending the current predicted token. Exemplary modifications to LM prompt 22 include insertions, replacements, and deletions of selected tokens. Tokens may be inserted at various positions within LM prompt 22, as opposed to the end. Furthermore, the type (e.g., insertion, deletion) and content (actual inserted and/or deleted tokens) of prompt modifications are determined by calculations performed outside of the GLM itself. Some explicit examples are given below. Also, crucially, modifications to LM prompt 22 are done inline, i.e., on an already ingested prompt, without re-initializing GLM 40 (as opposed to submitting an entirely new prompt and causing a re-initialization of GLM 40). Some embodiments thus manage to extend the functionality and improve the performance of a conventional chatbot, while benefitting from computational savings as described above.

[0050] In some embodiments, prompt manager 30 (Fig. 1) is configured to receive target message 15 and formulate an initial LM prompt according to target message 15. Prompt manager 30 is further configured to feed LM prompt 22 to GLM 40, receive the output of GLM 40, and dynamically update prompt 22 according to the received predicted token(s) 26. In some embodiments, dynamically updating LM prompt 22 encompasses initiating an execution of and actually executing a set of code snippets for modifying LM

prompt **22**. Prompt manager **30** may be further configured to direct the execution of GLM **40**, for instance cause GLM **40** to execute one or more inference steps and/or cause GLM **40** to suspend execution at a selected moment/inference step.

[0051] Fig. **5** shows an example of an LM prompt **22f** transmitted by prompt manager **30** to GLM **40**. Prompt **22f** includes at least a part of target message **15** received by chatbot **10** as part of the current fraud detection procedure. Prompt **22f** further includes a set of instructions **19** for GLM **40** formulated in natural language. Instructions **19** instruct GLM **40** to perform specific text processing and/or semantic analysis tasks on the respective target message. In fraud detection applications, instructions **19** may ask GLM **40** to perform tasks related to fraud detection, such as determining whether a target message is indicative of fraud, determining whether a target message is a joke, determining a sentiment of a target message (e.g., love, hate, anger, threat, etc.), formulating a summary of the target message, assigning the target message to one of a pre-determined set of categories, and extracting various fraud-indicative characteristics of the respective target message (e.g., whether the message is asking for money or personal details, whether the message is offering something for sale, etc.), among others.

[0052] LM prompt **22f** further includes a set of flag tokens **29a-b**. Flag tokens herein denote specific tokens whose presence in the output of GLM **40** (i.e., in predicted token sequence **27**) is detected and interpreted by prompt manager **30** as further detailed below. Exemplary flag tokens include a specific word, keyword, or character sequence (e.g., flag token **29b**), a specific sequence of words, an attribute-value pair (e.g., flag token **29a**), and a tuple of attribute values, among others. In alternative embodiments, flag tokens may be identified according to whether they include a predetermined special character, such as #, \$, etc. In some embodiments as illustrated, instructions **19** further instruct GLM **40** to output at least one indicated flag token in response to, or as part of executing an indicated task. Some flag tokens such as token **29b** in Fig. **5** may act as placeholders or intra-prompt cues indicating a specific location of a future inline modification of the respective LM prompt (e.g., a location of a token insertion as shown in examples below).

[0053] In some embodiments, detection of a flag token within the output of GLM **40** causes prompt manager **30** to initiate an execution of a specific code snippet (i.e., computer program) associated with the respective flag token. Some embodiments as illustrated in

Figs. 1 and 6 include an instruction cache 32 storing a plurality of code snippets and maintaining a mapping associating code snippets with flag tokens, enabling a selective identification of a code snippet according to a flag token and/or *vice versa*. In simple embodiments, each flag token 29c-d is mapped to at least one respective GLM instruction 19a-b. Such a mapping may cause a respective instruction to be inserted into the LM prompt in response to detecting the presence of the associated flag token. For instance, in the illustrated example in Fig. 6, detection of flag token 29d within the output of GLM 40 will cause insertion of instruction 19b into the LM prompt. Also, in the illustrated example, detection of flag token 29d will further cause execution of code snippet 3 and 4.

[0054] Code snippets as described herein may implement any procedure useful in detecting online fraud, from simple text manipulations such as inserting tokens into and/or deleting tokens from the LM prompt, to extracting data from the LM prompt and/or output token sequence 27 and combining such data with other fraud-indicative characteristics of the text message, such as an identity or an address of the sender, a timestamp of the message, an indicator of whether the message includes a hyperlink, etc. A definitive verdict may then be passed back to GLM 40 with instructions on generating explanations, user advice, information about a particular type of fraud, etc.

[0055] Fig. 7 shows an exemplary sequence of steps carried out by prompt manager 30 according to some embodiments of the present invention. A sequence of steps 122-124 listens for incoming target messages 15 for analysis. When a message is received (step 124 returns a YES), in a step 126 prompt manager 30 may formulate an initial LM prompt according to target message 15. Step 126 may comprise, for instance, formulating LM instructions 19 and inserting a set of flag tokens (see e.g., flag tokens 29a-b in Fig. 5). The actual instructions and/or flag tokens may depend on a type/category of target message, and/or may further vary according to message metadata such as an identity of the sender and/or receiver of the respective message, a timestamp, a geographical location of the receiver etc. Some embodiments may then cycle repeatedly through a sequence of steps 128-134 until a termination condition is satisfied. Exemplary termination conditions include reaching a pre-determined count of inference steps (e.g., generating 200 successive

predicted tokens), generation of specific predicted tokens (e.g., full stop '.'), etc. In some embodiments, termination may be induced by explicitly instructing GLM 40 to stop.

[0056] Steps 128-130 transmit LM prompt 22 to generative language module 40 and initiate execution of an inference step. When the termination condition is satisfied, a sequence of steps 136-138 may formulate and output reply message 16, which may include at least a part of the current predicted token sequence 27. When the result of the current fraud detection procedure indicates a likelihood of fraud, reply message 16 may further include an explanation or description of the respective type of scam, and a set of recommendations, instructions, and/or advice for the user on countering or mitigating the respective threat. Conversely, when the target message is deemed benign, reply message 16 may include advice on avoiding online fraud.

[0057] When the termination condition is not satisfied (step 132 returns a NO), in a step 134 prompt manager 30 may update LM prompt 22 according to the current output of GLM 40, e.g., according to the current predicted token 26. Fig. 8 illustrates an exemplary sequence of steps carried out by prompt manager 30, the illustrated sequence detailing the inline dynamic editing of LM prompt 22 (step 134 in Fig. 7). In a step 142 some embodiments append the current output of GLM 40 to LM prompt 22. A step 144 may parse predicted token sequence 27 to determine whether any flag tokens are present. If no flag tokens were detected in the output of GLM 40 (a step 146 returns a NO), some embodiments will determine that LM update is complete and effectively proceed to step 128 in Fig. 7.

[0058] If predicted token sequence 27 (i.e., the output of GLM 40) includes at least one flag token, a step 148 may identify at least one code snippet according to the respective flag token, e.g., by looking up instruction cache 32 (see description above in relation to Fig. 6). A step 150 may then initiate execution of the identified code snippet. Step 150 may encompass actually executing the respective code snippet, and also transmitting a request to execute the respective code snippet to another computing module, which may or may not execute on the local machine. Calling such external modules may use any method known in the art. In one such example, the respective code snippet is executed remotely as a web service, in which case step 150 may include formulating an HTTP request to a URL specific to the respective code snippet. The respective HTTP request may include a

set of argument values extracted from and/or determined according to a current content of the predicted token sequence.

[0059] In a step 152, prompt manager 30 may formulate a set of supplemental tokens for insertion into LM prompt 22 according to a result of executing the identified code snippet. Exemplary supplemental tokens may include a new set of LM instructions for GLM 40, various parameter values calculated by the respective code snippet, and other flag tokens, among others. Some examples are given below. A further step 154 may then insert the respective supplemental tokens into LM prompt 22. In some embodiments wherein a flag token identified in step 144 acts as a placeholder, supplemental tokens are inserted into LM prompt 22 precisely at the position of the respective flag token. In some embodiments, a step 156 may then delete the respective flag token and/or other tokens (as indicated in the respective code snippet) from predicted token sequence 27 and/or from LM prompt 22.

[0060] The sequence of steps illustrated in Figs. 7-8 enables carrying out various advanced calculations using GLM 40, calculations which would otherwise be impossible or unreliable using conventional prompting that updates the LM prompt merely by appending the current output of a generative language module. For instance, some code snippets stored in instruction cache 32 may collectively implement a decision tree via dynamic LM instructions 19.

[0061] Fig. 9 illustrates a basic implementation of an IF clause according to some embodiments of the present invention. More complicated calculations may be achieved in a similar manner. An exemplary initial LM prompt 22g includes instructions for selectively outputting flag tokens 29c-d according to whether the target message fraudulent or not, respectively. As shown above in relation to Figs. 5-6, exemplary flag token 29c (verdict:1) may be associated with LM instruction 19a, while flag token 29d (verdict:0) may be associated with LM instruction 19b. Therefore, the LM prompt is updated either from version 22g to version 22h in response to GLM 40 returning verdict:1, or from version 22g to version 22j in response to GLM 40 returning verdict:0, thus effectively implementing an IF clause. LM prompts 22h or 22j are then re-submitted as input to GLM 40 for the next inference step.

[0062] The disclosure above mainly describes a fraud detection system wherein code snippets for manipulating LM prompt 22 are pre-determined and pre-loaded into security

chatbot **10**. However, in alternative embodiments, code snippets as described herein may be received as part of the input to chatbot **10** as illustrated in Fig. **10**, wherein a prompt creator module **18** wraps target message **15** together with selected code snippets for performing a computer security procedure into a logical prompt **20** transmitted to chatbot **10**. Stated otherwise, the input to chatbot **10** may include computer code in the form of code snippets. However, in contrast to conventional language model operation, the respective computer code is not passed on to the language model but is instead used by prompt manager **30** to manipulate the LM prompt. In other words, in embodiments of the present invention, computer code included in logical prompt **20** is not meant to be interpreted or executed by the language model itself, but is instead used by prompt manager **30** for inline dynamic editing of the input of the respective language model.

[0063] An exemplary logical prompt **20a** according to some embodiments of the present invention is illustrated in Fig. **11**. Logical prompt **20** includes a code snippet **25** comprising computer code for dynamically updating LM prompt **22**. The illustrated code snippet implements the exemplary fraud detection procedure shown in Fig. **9**. Snippet **25** may be formulated in any coding language or specification, such as bytecode, Python, a version of Javascript® Object Notation (JSON) or extensible markup language (XML), etc. Alternatively, snippet **25** may include a location indicator (e.g., file path, network address, URL) of a computer program for editing LM prompt **22**. Code snippet **25** may be separated from a rest of the logical prompt by a specific marker. In the example of Fig. **11**, code snippet **25** is framed between `<llmi ...>` and `<\llmi ...>` tags. A skilled artisan will know that the illustrated formatting is meant only as an example and is not limiting the scope of the invention.

[0064] Code snippets **25** may be developed and kept up to date by computer security operators and stored within a code repository available to prompt creator **18**. Different code snippets may be developed for different types or categories of target message, for different tasks (e.g., fraud detection vs. detection of fake news, etc.), as well as for different users or user categories (e.g., according to a service subscription, according to a geographical location of the respective user, etc.).

[0065] Fig. **12** shows an exemplary sequence of steps performed by prompt manager **30** in embodiments configured to receive logical prompts **20**. Steps **162-164** may listen for

logical prompts **20**. When a logical prompt is detected, in a step **166** some embodiments initialize an LM prompt with a copy of the received logical prompt. A further step **168** may then parse logical prompt **20** to identify code snippets **25**. For each code snippet, in a step **172** some embodiments may then save the respective code snippet to a code repository such as instruction cache **32**. Step **172** may additionally or alternatively create a pointer to the respective code snippet, thus enabling prompt manager **30** to selectively trigger execution of the respective code snippet. A step **174** may associate the respective code snippet and/or pointer with a specific flag token, as indicated in logical prompt **20**. A further step **176** may then insert a flag token into initial LM prompt **22**, as per logical prompt **20**. Some flag tokens act as placeholders; in such situations code snippet **25** may indicate a location for insertion of the respective flag token. For instance, some embodiments may replace code snippet **25** within LM prompt **22** with the associated flag token. In other words, the respective flag token may be inserted into LM prompt **22** at a position where snippet **25** was located within logical prompt **20**. Alternative embodiments may not remove code snippet **25** from LM prompt **22**, but instead include a set of instructions causing GLM **40** to ignore the respective code snippet altogether.

[0066] When all code snippets in logical prompt **20** have been processed (a step **170** returns a NO), prompt manager **30** may proceed to feeding the initial LM prompt to GLM **40**. Prompt manager **30** may then cycle through a sequence of steps **178-184** for each of a plurality of inference steps, until a termination condition is satisfied (e.g., until a step **182** returns a YES). A step **180** may initiate execution of an individual inference step by GLM **40**, causing the output of a predicted token. If the termination condition is not satisfied, in a step **184** prompt manager **30** may update the LM prompt according to the respective predicted token(s). Execution of step **184** may comprise the exemplary steps illustrated in Fig. **8**. When the termination condition is satisfied, a sequence of steps **186-188** may formulate and output reply message **16**, for instance as described above in relation to Fig. **7**.

[0067] Fig. **13** shows an exemplary hardware configuration of a computer system **80** programmed to execute some of the methods described herein. System **80** generically embodies various computing devices, such as front-end device **12** in Fig. **1**, as well as a server computer system executing an instance of security chatbot **10**. The illustrated device

is a personal computer; other devices such as servers, mobile telephones, tablet computers, and wearables may have slightly different configurations.

5 [0068] Processor(s) **82** comprise a physical device (e.g. microprocessor, multi-core integrated circuit formed on a semiconductor substrate) configured to execute computational and/or logical operations with a set of signals and/or data. Such signals or data may be encoded and delivered to processor(s) **82** in the form of processor instructions, e.g., machine code.

10 [0069] Processor(s) **82** are generally characterized by an instruction set architecture (ISA), which specifies the respective set of processor instructions (e.g., the x86 family vs. ARM® family), and the size of registers (e.g., 32 bit vs. 64 bit processors), among others. The architecture of processor(s) **82** may further vary according to their intended primary use. While central processing units (CPU) are general-purpose processors, graphics processing units (GPU) may be optimized for image/video processing and parallel computing, such as the implementation of some neural network architectures. Processors **82** may further include application-specific integrated circuits (ASIC), such as Tensor Processing Units (TPU) from Google®, Inc., and Neural Processing Units (NPU) from various manufacturers. TPUs and NPUs may be particularly suited for AI applications as described herein. For instance, selected parts of GLM **40** may execute on a GPU or NPU.

15 [0070] Memory unit **83** may comprise volatile computer-readable media (e.g. dynamic random-access memory - DRAM, GPU memory) storing data/signals/instruction encodings accessed or generated by processor(s) **82** in the course of carrying out operations. Input devices **84** may include computer keyboards, mice, and microphones, among others, including the respective hardware interfaces and/or adapters allowing a user to introduce data and/or instructions into computer system **80**. Output devices **85** may include display devices such as monitors and speakers among others, as well as hardware interfaces/adapters such as graphic cards, enabling the respective computing appliance to communicate data to a user. In some embodiments, input and output devices **84-85** share a common piece of hardware (e.g., a touch screen). Storage devices **86** include computer-readable media enabling the non-volatile storage, reading, and writing of software instructions and/or data. Exemplary storage devices include magnetic and optical disks and flash memory devices, as well as removable media such as CD and/or DVD disks and

drives. Network adapter(s) **87** comprise specialized hardware that enable computer system **80** to connect to an electronic communication network and/or to other devices/computer systems for data transmission and reception.

[0071] Controller hub **90** generically represents the plurality of system, peripheral, and/or chipset buses, and/or all other circuitry enabling the communication between processor(s) **82** and the rest of the hardware components of computer system **80**. For instance, controller hub **90** may comprise a memory controller, an input/output (I/O) controller, and an interrupt controller. Depending on hardware manufacturer, some such controllers may be incorporated into a single integrated circuit, and/or may be integrated with processor(s) **82**. In another example, controller hub **90** may comprise a northbridge connecting processor **82** to memory **83**, and/or a southbridge connecting processor **82** to devices **84**, **85**, **86**, and **87**.

[0072] The exemplary systems and methods described above enable efficiently using AI systems such as large language models (LLM) for complex specialized text analysis tasks such as detecting online fraud, among others.

[0073] In contrast to many conventional anti-fraud solutions, some embodiments of the present invention employ a chatbot to interface with the user in a friendly, conversational manner. The chatbot may assist the user with a variety of tasks, such determining whether the user is subject to an online threat such as phishing, advising the user on computer security issues, answering questions about subscriptions, accounts, billing, etc. In some embodiments, the chatbot impersonates a user of a popular messaging or social media platform such as Facebook® or WhatsApp Messenger® and so is accessible via a user interface of the respective applications. An exemplary user interface as described is illustrated in Fig. **14**. Stated otherwise, the user does not need to install or learn any new software to carry out a fraud analysis. Furthermore, the user may submit questions and data related to any communication application or platform via a single chatbot interface. For instance, the user may use an instance of a WhatsApp Messenger® application to converse with an anti-fraud chatbot about messages received via other communication applications (e.g., Facebook®, email clients, SMS, etc.). The respective chatbot may automatically identify the user's needs, guide the user into providing relevant data for

analysis, and communicate the analysis results back to the user, together with explanations, recommendations, and advice for protecting the user against online fraud.

[0074] Chatbots implementing large language models (LLM) are rapidly becoming a popular technical solution for interacting with users in a broad variety of situations and applications. The advantages include extending the reach of a target technology to users that lack a technical or computing background, and a reduction of operational costs by replacing human customer care operators with AI agents. Some advanced chatbots such as ChatGPT® from OpenAI, Inc. are capable of answering computer security questions and analyzing data to determine whether a user is targeted by a computer security threat. However, studies have shown that such chatbots sometimes provide wrong or misleading answers, or answers that strongly depend on how the question is formulated. More importantly, their grasp of highly specific issues such as computer security is only as good as the training corpus they have ingested. In other words, if the training corpus does not include training examples relevant to a specific question or situation, the respective chatbot may not return a correct answer or assessment. This problem is especially acute in the area of computer security, wherein methods employed by malicious software and online scammers are continuously changing. Generic training corpora and methodologies are therefore relatively unlikely to keep up with the threat landscape.

[0075] In principle, a pre-trained LLM may be further trained to specifically address computer security questions, for instance using a purposely built and maintained corpus of text including examples of online fraud such as known phishing attempts delivered via email messages, SMS, and social media platforms. However, even though such additional training is likely to increase the performance of the respective LLM in detecting online fraud, training LLMs typically incurs substantial computational costs. Furthermore, the additional training does not solve a fundamental problem, namely that LLM are enormously complex systems typically having billions of tunable parameters, and whose behavior is essentially opaque and unpredictable.

[0076] Current LLM-based chatbots have also been shown to be vulnerable to malicious manipulation, commonly known in the art as adversarial attacks. Typical examples include carefully formulating the input to an LLM to cause it to fail, as in producing the wrong output, unexpected output (also known as a hallucination), or no output at all.

[0077] In addition to the above shortcomings, several computer experiments have revealed that typical LLM-based chatbots often fail to correctly solve more complex problems and/or logical operations such as navigating a decision tree with many possible outcomes depending on the contents of the input. Solving such a problem in a conventional way involves including all instructions for navigating the decision tree (e.g., distinguishing among all possible cases) in one single prompt. However, the LLM may fail to follow such complicated instructions, mainly because of its limited attention span. For instance, it may follow instructions selectively and unpredictably, leading to conclusions which are logically unsound, conflicting, or plain wrong. Furthermore, some commercial LLM services charge according to the length (i.e., count of tokens) of the prompt, so a relatively long and complex prompt may incur substantial costs.

[0078] Various workarounds have been proposed to solve this issue. One exemplary strategy known as prompt chaining comprises presenting the LLM with a sequence of individual prompts, each prompt formulated according to a response by the LLM to a previous prompt. In the example of a decision tree, each individual prompt may represent a single branch or branch point. Such prompts may be generated dynamically, according to the characteristics of the target text and according to replies to previous prompts. In other words, the decision process may be steered preferentially along a specific path of traversing the decision tree, wherein each step of the decision process is determined by the outcome of a previous step. However, in conventional prompt chaining, each successive prompt is treated as a separate new submission, ending up incurring substantial costs when the respective LLM charges per prompt.

[0079] In contrast to such conventional solutions, some embodiments enable a dynamic inline updating of the input to the respective language model. This allows submitting a single prompt to the LLM and subsequently performing gradual adjustments to the respective prompt according to LLM outputs, without re-initializing the entire language model. In one such example, an initial formulation of the prompt may instruct the LLM to carry out a first text analysis task. The prompt may be then updated inline, to instruct the LLM to carry out another text analysis task selected according to the output of the first task. Conventional LLMs already perform an inline updating of the input prompt, typically by appending the latest output of the LLM to the existing input prompt. In contrast to such

conventional prompt updating, some embodiments update the input prompt more substantially and at positions within the respective prompt other than the end. Some parts of the prompt may be removed, while new content may be added. Also, in embodiments, the content of such prompt modifications is determined according to a result of computations executed outside of the LLM itself.

5 [0080] Some embodiments further submit a set of prompt-editing instructions in the form of code snippets embedded directly in the chatbot prompt. While some conventional LLMs are capable of receiving, executing, and otherwise manipulating computer code, in embodiments of the present invention the code snippets embedded in the input prompt are not meant for execution by the LLM itself, but instead are used for dynamically updating the LLM prompt. In exemplary embodiments, a prompt manager distinct from the computer module implementing the LLM is charged with executing the prompt-editing instructions. The prompt manager may remove such code snippets from the input prompt before submitting the respective prompt to the LLM itself. Bundling up an LLM prompt together with prompt-editing instructions into one logical prompt as described herein has multiple advantages over conventional prompt management. For instance, logical prompts improve portability and facilitate readability and code maintenance by keeping all resources in one place.

10 15 [0081] One substantial advantage of systems and methods presented herein is that they enable a more efficient solving of problems involving complex computations and/or complex logical operations, compared to conventional approaches using multiple individual LLM prompts. Mainly, they allow solving such problems using one single and relatively short LLM prompt, which may substantially reduce financial and computational costs associated with using the respective LLM. The proposed approach allows, for instance, using an out-of-the-box generic pre-trained LLM to perform computations of arbitrary scope and complexity, wherein the linguistic and generalization powers of the LLM can be combined with additional calculations performed by the prompt manager and/or any other computation module, software, or service. In one such example directed at prevention of online fraud, the LLM may be used to extract a set of features or preliminary verdicts about a target text, which may then be passed on to dedicated anti-fraud software, which produces a definitive verdict.

[0082] Substantial reductions in computational costs enabled by embodiments of the present invention come from avoiding having to recompute internal representations of individual tokens of the LLM prompt, relying on the observation that the inline updating described herein preserves the majority of existing prompt tokens (and thus most of the internal state of the LLM). Typical prompts consist of hundreds or thousands of individual tokens. An inline update of the prompt as described herein may only change a few tokens at a time, realistically amounting to a few percent of the entire LLM prompt. Therefore, once the LLM computes internal representations/embeddings of tokens belonging to the initial LLM prompt, ulterior changes to the respective prompt may only require relatively minor updates to the internal LLM state. In contrast, submitting an entirely new LLM prompt typically resets the entire internal state of the LLM, which may require re-computing hundreds or thousands of token embeddings. Other savings in time and costs come from avoiding redundant transmission of data between the LLM and other components of the system. When the LLM is accessed remotely, for instance as a web service, such savings may be substantial.

[0083] Although most of the disclosure above was directed at detecting online fraud, an artisan will know that the described systems and methods can be adapted to other applications that extend beyond the traditional scope of computer security, such as detection of bullying, hate speech, fake news, and AI-generated content, among others. Such embodiments may use a single, dynamically evolving LM prompt to instruct GLM 40 to perform a sequence of text analysis tasks specific to the respective application. The output of each task may be signaled via a task-specific flag token. Each flag token may in turn trigger the execution of a token-specific code snippet that causes an inline update of the LM prompt, as described in detail above.

[0084] Figs. 15 and 16 illustrate an exemplary embodiment directed at determining whether a target text comprises fake news. Exemplary logical prompt 20b includes a code snippet between matching @dprompt tags. In response to receiving prompt 20b, prompt manager 30 may formulate an initial LM prompt 22k causing GLM 40 to carry out a first fake news detection task (determine an initial verdict indicating whether the respective target text is likely to be fake news). Prompt manager 30 then scans the output of GLM 40 (predicted token sequence 27a) for the flag token 'initial_verdict'. When GLM 40

evaluates `initial_verdict` to 1, manager **30** updates the LM prompt from version **22k** to version **22m** where some of the contents of LM prompt **22k** was removed and a new line was inserted instructing GLM **40** to carry out another fake news detection task (determine whether the target text is likely to be a pamphlet or similar). The code snippet included in logical prompt **20** further replaces the tokens `'initial_verdict: 1'` with `'verdict: 1'`. Manager **30** then triggers GLM **40** to execute a new inference step and scans for the new flag token `'fp_pamphlet'` within the output of GLM **40**. Item **22n** illustrates an exemplary evolution of the LM prompt when GLM **40** evaluates `'fp_pamphlet'` to 0.

[0085] Figs. **17-18** illustrate another exemplary application of described systems and methods to detection of bullying and/or hate speech. In such embodiments, security chatbot **10** may be used as an online moderator analyzing the content of an online conversation, of a content of a social media wall, etc. In another exemplary embodiment, messaging system **14** may serendipitously extract a content of an online conversation carried out by a vulnerable party (e.g., a child, a member of a minority group, etc.), and transmit the respective content to chatbot **10** for analysis. A determination that the respective conversation is indicative of bullying may then trigger an alert, for instance transmitting a warning to a parent, teacher, etc.

[0086] Exemplary logical prompt **20c** in Fig. **17** includes a code snippet in between matching `@dprompt` tokens, the illustrated code snippet instructing GLM **40** to selectively output exemplary flag tokens **29e-g** according to a result of various respective tasks. Fig. **18** shows an exemplary evolution of the LM prompt caused by processing logical prompt **20c** in Fig. **17** according to some embodiments of the present invention. In response to receiving prompt **20c**, prompt manager **30** may formulate an initial LM prompt **22p** causing GLM **40** to carry out a first hate speech detection task (determine an initial verdict indicating whether the respective target text is obscene and harmful). Prompt manager **30** then scans the output of GLM **40** (predicted token sequence **27d**) for flag token **29e**. When GLM **40** evaluates `assessment_obscene_harmful` to 1, manager **30** updates the LM prompt from version **22p** to version **22q** where some of the contents of LM prompt **22p** was removed and a new line was inserted instructing GLM **40** to carry out another hate speech detection detection task (determine whether the target text

is decent). Manager **30** then triggers GLM **40** to execute a new inference step and scans for new flag token **29g** within the output of GLM **40**. Item **22r** illustrates an exemplary version of the LM prompt when GLM **40** evaluates 'fp_decent' to 0.

5 [0087] It will be clear to one skilled in the art that the above embodiments may be altered in many ways without departing from the scope of the invention. Accordingly, the scope of the invention should be determined by the following claims and their legal equivalents.

Claims

1. A computer system comprising at least one hardware processor configured to analyze a target text formulated in a natural language, wherein analyzing the target text comprises executing a prompt manager communicatively coupled to a generative language module (GLM), wherein:
- 5 the GLM is configured to:
- receive from the prompt manager a language model (LM) prompt formulated in the natural language, and
- in response, output to the prompt manager a predicted token comprising a likely
- 10 continuation of the received LM prompt; and
- the prompt manager is configured to:
- receive a logical prompt, the logical prompt including the target text and a code snippet comprising computer code for updating the LM prompt,
- formulate the LM prompt according to the logical prompt, the LM prompt
- 15 instructing the GLM to perform a text analysis task according to the target text and in response, to output a flag token,
- in response to receiving the predicted token, initiate an execution of the code snippet to produce an updated LM prompt,
- transmit the updated LM prompt to the GLM, and
- 20 formulate a reply message comprising a result of analyzing the target text, the reply message formulated according to an output of the GLM in response to the updated LM prompt.
2. The computer system of claim 1, wherein the execution of the code snippet causes the prompt manager to:
- 25 determine whether the predicted token includes the flag token; and
- in response, if the predicted token includes the flag token, insert an LM instruction into the LM prompt, the LM instruction instructing the GLM to perform another text analysis task according to the target text.

3. The computer system of claim 2, wherein the LM instruction further instructs the GLM to output another flag token in response to performing the other text analysis task.
4. The computer system of claim 1, wherein:
 - 5 the prompt manager is configured to formulate the LM prompt to further include a placeholder token; and
 - the execution of the code snippet causes the prompt manager to insert a set of supplemental tokens into the LM prompt at a position of the placeholder token.
- 10 5. The computer system of claim 1, wherein the execution of the code snippet causes the prompt manager to perform an action selected from a group consisting of inserting a plurality of supplemental tokens into the LM prompt and deleting a set of tokens from the LM prompt.
6. The computer system of claim 1, wherein the execution of the code snippet causes the
15 prompt manager to delete the flag token from the LM prompt.
7. The computer system of claim 1, wherein:
 - the logical prompt comprises a plurality of code snippets for updating the LM prompt, the plurality of code snippets including the code snippet;
 - the prompt manager is configured to associate each code snippet of the plurality of
20 code snippets with a distinct flag token of a plurality of flag tokens; and
 - producing the updated LM prompt comprises executing a selected code snippet of the plurality of code snippets according to whether the predicted token includes a selected flag token of the plurality of flag tokens.
8. The computer system of claim 1, wherein the flag token comprises an attribute-value
25 pair.
9. The computer system of claim 1, wherein the text analysis task comprises an item selected from a group consisting of determining whether the target text is indicative of online fraud, determining whether the target text comprises fake news, and determining whether the target text is indicative of hate speech.

10. A computer-implemented method of analyzing a target text formulated in a natural language, the method comprising employing at least one hardware processor of a computer system to execute a prompt manager communicatively coupled to a generative language module (GLM), wherein:

5 the GLM is configured to:

receive from the prompt manager a language model (LM) prompt formulated in the natural language, and

in response, output to the prompt manager a predicted token comprising a likely continuation of the received LM prompt; and

10 executing the prompt manager comprises:

receiving a logical prompt, the logical prompt including the target text and a code snippet comprising computer code for updating the LM prompt,

formulating the LM prompt according to the logical prompt, the LM prompt instructing the GLM to perform a text analysis task according to the target text and in response, to output a flag token,

15 in response to receiving the predicted token, initiating an execution of the code snippet to produce an updated LM prompt,

transmitting the updated LM prompt to the GLM, and

20 formulating a reply message comprising a result of analyzing the target text, the reply message formulated according to an output of the GLM in response to the updated LM prompt.

11. The method of claim 10, wherein the execution of the code snippet causes the prompt manager to:

determine whether the predicted token includes the flag token; and

25 in response, if the predicted token includes the flag token, insert an LM instruction into the LM prompt, the LM instruction instructing the GLM to perform another text analysis task according to the target text.

12. The method of claim 11, wherein the LM instruction further instructs the GLM to output another flag token in response to performing the other text analysis task.

30 13. The method of claim 10, wherein:

the prompt manager is configured to formulate the LM prompt to further include a placeholder token; and

the execution of the code snippet causes the prompt manager to insert a set of supplemental tokens into the LM prompt at a position of the placeholder token.

5

14. The method of claim 10, wherein the execution of the code snippet causes the prompt manager to perform an action selected from a group consisting of inserting a plurality of supplemental tokens into the LM prompt and deleting a set of tokens from the LM prompt.

10

15. The method of claim 10, wherein the execution of the code snippet causes the prompt manager to delete the flag token from the LM prompt.

16. The method of claim 10, wherein:

the logical prompt comprises a plurality of code snippets including the code snippet;

15

the prompt manager is configured to associate each code snippet of the plurality of code snippets with a distinct flag token of a plurality of flag tokens; and producing the updated LM prompt comprises executing a selected code snippet of the plurality of code snippets according to whether the predicted token includes a selected flag token of the plurality of flag tokens.

20

17. The method of claim 10, wherein the flag token comprises an attribute-value pair.

18. The method of claim 10, wherein the text analysis task comprises an item selected from a group consisting of determining whether the target text is indicative of online fraud, determining whether the target text comprises fake news, and determining whether the target text is indicative of hate speech.

25

19. A non-transitory computer-readable medium storing instructions which, when executed by at least one hardware processor of a computer system, cause the computer system to analyze a target text formulated in a natural language, wherein analyzing the target text comprises executing a prompt manager communicatively coupled to a generative language module (GLM), wherein:

the GLM is configured to:

receive from the prompt manager a language model (LM) prompt formulated in a natural language, and

in response, output to the prompt manager a predicted token comprising a likely continuation of the received LM prompt; and

5

the prompt manager is configured to:

receive a logical prompt, the logical prompt including the target text and a code snippet comprising computer code for updating the LM prompt,

formulate the LM prompt according to the logical prompt, the LM prompt instructing the GLM to perform a text analysis task according to the target text and in response, to output a flag token,

10

in response to receiving the predicted token, initiate an execution of the code snippet to produce an updated LM prompt,

transmit the updated LM prompt to the GLM, and

15

formulate a reply message comprising a result of analyzing the target text, the reply message formulated according to an output of the GLM in response to the updated LM prompt.