(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 December 2024 (19.12.2024)

WIPO | PCT

(10) International Publication Number
WO 2024/258676 A1

(51) International Patent Classification:
G06F 21/56 (2013.01) *G06F 8/33* (2018.01)
G06F 21/62 (2013.01) *G06N 20/00* (2019.01)

(21) International Application Number:
PCT/US2024/032377

(22) International Filing Date:
04 June 2024 (04.06.2024)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/521,191 15 June 2023 (15.06.2023) US
18/731,845 03 June 2024 (03.06.2024) US

(71) Applicant: **NEC LABORATORIES AMERICA, INC.** [US/US]; 4 Independence Way, Suite 200, Princeton, New Jersey 08540 (US).

(72) Inventors: **CHENG, Wei**; 5 Arnold Drive, Princeton Junction, New Jersey 08550 (US). **YANG, Xianjun**; 4076 Foothill Road, Santa Barbara, California 93110 (US). **CHEN, Haifeng**; 11 Blackhawk Court, West Windsor, New Jersey 08550 (US).

(74) Agent: **BITETTO, James J.**; 401 Broadhollow Road, Suite 402, Melville, New York 11747 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

(54) Title: DETECTING ARTIFICIAL INTELLIGENCE GENERATED COMPUTER CODE

(57) Abstract: Systems and methods for detecting artificial intelligence (AI) generated computer code. Lines of code can be masked (110) from a candidate code to obtain perturbed codes. Missing code can be generated (120) from the perturbed codes by employing an AI code generator model to obtain machine-filled codes. Probabilities of the candidate code probability and the machine-filled codes as AI-generated can be predicted (130) by employing a surrogate model. The candidate code can be distinguished (140) as AI-generated by comparing the probabilities against a detection threshold to obtain detection results.

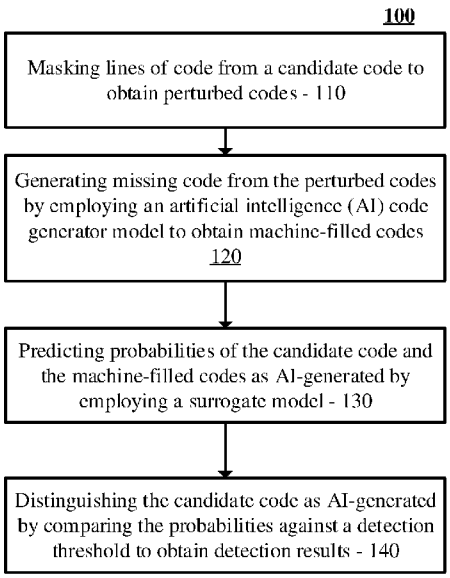

FIG. 1

WO 2024/258676 A1

[Continued on next page]

Published:
— *with international search report (Art. 21(3))*

DETECTING ARTIFICIAL INTELLIGENCE GENERATED COMPUTER CODE

RELATED APPLICATION INFORMATION

[0001] This application claims priority to U.S. Provisional App. No. 63/521,191, filed on June 15, 2023, and U.S. Patent App. No. 18/731,845 filed on June 3, 2024, incorporated herein by reference in its entirety.

BACKGROUND

Technical Field

[0002] The present invention relates to computer code analysis and more particularly to detecting artificial intelligence generated computer code.

Description of the Related Art

[0003] The remarkable progress in large pre-trained large language models (LLMs) has brought machine-generated text closer to human-written text in both fluency and diversity. In addition to generating text, computer code has also been generated using LLMs. As such, this poses a difficult question for distinguishing whether computer code has been machine-generated or human created.

SUMMARY

[0004] According to an aspect of the present invention, a computer-implemented method for detecting artificial intelligence (AI) generated computer code is provided, including masking lines of code from a candidate code to obtain perturbed codes, generating missing code from the perturbed codes by employing an AI code generator model to obtain machine-filled codes, predicting probabilities of the candidate code and the machine-filled codes as AI-generated by employing a surrogate model, and

distinguishing the candidate code as AI-generated by comparing the probabilities against a detection threshold to obtain detection results.

[0005] According to another aspect of the present invention, a system for detecting artificial intelligence (AI) generated computer code is provided, including a memory, and one or more processor devices in communication with the memory configured to mask lines of code from a candidate code to obtain perturbed codes, generate missing code from the perturbed codes by employing an AI code generator model to obtain machine-filled codes, predict probabilities of the candidate code and the machine-filled codes as AI-generated by employing a surrogate model, and distinguish the candidate code as AI-generated by comparing the probabilities against a detection threshold to obtain detection results.

[0006] According to yet another aspect of the present invention, a non-transitory computer program product is provided including a computer-readable storage medium including program code for detecting artificial intelligence (AI) generated computer code, wherein the program code when executed on a computer causes the computer to perform masking lines of code from a candidate code to obtain perturbed codes, generating missing code from the perturbed codes by employing an AI code generator model to obtain machine-filled codes, predicting probabilities of the candidate code and the machine-filled codes as AI-generated by employing a surrogate model, and distinguishing the candidate code as AI-generated by comparing the probabilities against a detection threshold to obtain detection results.

[0007] These and other features and advantages will become apparent from the following detailed description of illustrative embodiments thereof, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF DRAWINGS

[0008] The disclosure will provide details in the following description of preferred embodiments with reference to the following figures wherein:

[0009] FIG. 1 is a flow diagram illustrating a high-level overview of method for detecting AI-generated computer code, in accordance with an embodiment of the present invention;

[0010] FIG. 2 is a block diagram illustrating a computing system for detecting AI-generated computer code, in accordance with an embodiment of the present invention;

[0011] FIG. 3 is a block diagram illustrating a software program for detecting AI-generated computer code, in accordance with an embodiment of the present invention;

[0012] FIG. 4 is a block diagram illustrating a system integrating a practical application for detecting AI-generated computer code, in accordance with an embodiment of the present invention; and

[0013] FIG. 5 is a block diagram illustrating an overview of deep learning neural networks, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0014] In accordance with embodiments of the present invention, systems and methods are provided for detecting artificial intelligence (AI) generated computer code.

[0015] In an embodiment, a candidate code can be distinguished as AI generated by comparing predicted probabilities of the candidate code and machine-filled codes against a detection threshold. Probabilities of the candidate code and machine-filled codes as AI-generated can be predicted by employing a surrogate model. Machine-filled codes can be obtained by generating missing code from perturbed codes by

employing an AI code generator model. Perturbed codes can be obtained by masking lines of code from the candidate code.

[0016] In an embodiment, the distinguished candidate code can be flagged as AI-generated to provide transparency to the computer code generation process for a decision-making entity to perform an action. In an embodiment, the action can be securing a healthcare management system handling patient vital data by patching the flagged candidate code for potential security risks.

[0017] The remarkable progress in large pre-trained language models (LLMs) has brought machine-generated computer codes closer to human-written code. The proliferation of large language models has revolutionized natural language processing (NLP) tasks, but it has also raised concerns regarding their potential misuse for generating malicious or unethical code. This poses a pressing challenge for distinguishing between the origin of the codes. Meanwhile, research on detection of AI-generated code still lags behind the development of LLMs. Few research efforts have been devoted to this pressing demand in the literature. As AI-generated codes gradually approaches human level, there is some fundamental difficulty in effectively detecting AI-generated codes. This leads to a recent debate of whether AI-generated codes can be detected or not. However, there is still a lack of practical tests on AI-generated code detection, especially in the era of ChatGPT™.

[0018] Prior art has failed to explore detection on codes generated from Large Language Models like GPT™-4. Additionally, prior art directed to training-based text detectors fail for code detection, possibly due to the distinctive statistical characteristics inherent in code structures. To solve these shortcomings, the present embodiments propose training-free detection of AI-generated codes to mitigate the risks associated with their indiscriminate usage.

[0019] Detecting AI-generated computer code can be useful in various scenarios to ensure code quality, security, and maintainability. Here are some potential applications:

[0020] Code Review: AI-generated code might not adhere to best practices, coding conventions, or quality standards. By detecting such code, developers can ensure that human-written code meets the required standards before merging it into the main codebase.

[0021] Security Analysis: AI-generated code might contain vulnerabilities, intentional or unintentional. Detecting it can help identify potential security risks, such as injection attacks, buffer overflows, or other vulnerabilities that could be exploited.

[0022] Plagiarism Detection: In academic or professional settings, it's important to identify instances where code has been plagiarized or copied from existing sources. By distinguishing between human-written and AI-generated code, educators and organizations can detect instances of plagiarism and ensure intellectual property rights are respected.

[0023] Debugging and Maintenance: AI-generated code might be harder to debug or maintain since it lacks the logical consistency and intent of human-written code. By identifying AI-generated sections, developers can focus on those areas during debugging and maintenance to ensure they are adequately addressed.

[0024] Automation Testing: AI-generated code might introduce errors or unpredictable behavior that could go undetected in traditional testing. Identifying such code allows for targeted testing strategies to ensure the correctness and reliability of the software being developed.

[0025] Code Refactoring and Optimization: AI-generated code might contain redundancies, unnecessary complexity, or inefficient constructs. Detecting these

sections can guide developers in refactoring and optimizing the code for improved performance and maintainability.

[0026] Intellectual Property Protection: In scenarios where proprietary or sensitive code needs to be safeguarded, detecting AI-generated code can help identify potential leaks or unauthorized use of such code.

[0027] Referring now in detail to the figures in which like numerals represent the same or similar elements and initially to FIG. 1, a flow diagram showing a high-level overview of method for detecting AI-generated computer code, in accordance with an embodiment of the present invention.

[0028] In an embodiment, a candidate code 501 (shown in FIG. 2) can be distinguished as AI generated by comparing predicted candidate code probability 514 (shown in FIG. 3) and machine-filled codes probabilities 515 (shown in FIG. 3) against a detection threshold 517 (shown in FIG. 3) to obtain a detection result 518 (shown in FIG. 3). Probabilities that the candidate code 514 and machine-filled codes 515 are AI-generated can be predicted by employing a surrogate model 511 (shown in FIG. 3). Machine-filled codes 505 (shown in FIG. 3) can be obtained by generating missing code 520 (shown in FIG. 3) from perturbed codes 503 (shown in FIG. 3) by employing an AI code generator model 504 (shown in FIG. 4). Perturbed codes 503 can be obtained by masking lines of code from the candidate code 501.

[0029] In block 110, lines of code from a candidate code 501 can be masked to obtain perturbed codes 503.

[0030] In an embodiment, the candidate code 501 can be a stream of text that is written in a programming language to solve a given problem. The candidate code can be written in various programming languages such as Python™, Java™, C, Swift™, etc. For example, a candidate code 501 can be a computer code written in the Python™

programming language to solve the following problem: “Mr. Chanek gives you a sequence a indexed from 1 to n. Define f(a) as the number of indices where a_i = i. You can pick an element from the current sequence and remove it, then concatenate the remaining elements together. For example, if you remove the 3-rd element from the sequence [4, 2, 3, 1].” The candidate code 501 can be:

```
def main():
n = int(input())
a = list(map(int, input().split()))
d = [0] * (n + 1)
for i in range(n):
if a[i] - i - 1 >= 0:
d[a[i] - i - 1] += 1
print(max(d))
if __name__ == '__main__’:
main()
```

[0031] In an embodiment, the candidate code 501 can be entered by an entity through peripheral devices 595 (shown in FIG. 2). In another embodiment, the candidate code 501 can be saved and accessed through memory 591 (shown in FIG. 2). In another embodiment, the candidate code 501 can be obtained through a network.

[0032] In an embodiment, lines of code can be masked by replacing the lines of code with mask texts 521. For example, the mask text 521 (shown in FIG. 3) can be “<MASK: 1 (e.g., number of mask)>.” In an embodiment, lines of code to be masked can be selected randomly. In an embodiment, the number of lines of code to be masked can be an element of a predefined set of numbers *m*. For example, $m \in \{1, 2, 4, 8, 16, 32, 64, 80, 100\}$. In another embodiment, the number of lines of code to be masked

can be determined by a masking ratio. For example, masking ratio *r* can be 15% of the total number of lines of code of the candidate code 501.

[0033] In an embodiment, the perturbed codes 503 can be the revised version of the candidate code 501 with the mask texts 521. For example, using the example above, the perturbed code 503 can be:

```
def main():
n = int(input())
<MASK: 0>
d = [0] * (n + 1)
for i in range(n):
if a[i] - i - 1 >= 0:
<MASK: 1>
print(max(d))
if __name__ == '__main__':
main()
```

[0034] In block 120, missing code from the perturbed codes 503 can be generated by employing an AI code generator model 504 to obtain machine-filled codes 505.

[0035] In an embodiment, missing code 520 from the perturbed codes 503 can be generated by removing the mask texts 521 in the perturbed codes 503 and employing an AI code generator model 504 to fill-in-the-middle (FIM) and obtain machine-filled codes 505. In another embodiment, missing code 520 from the perturbed codes 503 can be generated by directly inputting the perturbed codes 503 to the AI code generator model 504 to perform FIM and obtain machine-filled codes 505. FIM can be a code generation task that can be learned by the AI code generator model 504 to fill the missing code 520 in the perturbed codes 503.

[0036] In an embodiment, the AI code generator model 504 can be an autoregressive model such as Incoder-6B. An autoregressive model can be used as it can be pretrained with the FIM task and perform code infilling without reducing its left-to-right generative capabilities. In another embodiment, the AI code generator model 504 can be a generative pre-trained transformer (GPT™-4, GPT™-35-turbo), Large Language Model Association (LLaMa-13B), or text-davinci-edit-001. Other AI code generator models can be employed.

[0037] In an embodiment, the AI code generator model 504 can be pre-trained with hand-written codes evaluation dataset such as human evaluated generation model dataset (HumanEval), (HumanEval-X), or CodeContests dataset.

[0038] In block 130, the candidate code probability 514 and the machine-filled codes probabilities 515 can be predicted by employing a surrogate model 511.

[0039] In an embodiment, a surrogate model 511 can be used to predict whether the candidate code probability 514 and the machine-filled codes probabilities 515 as AI generated.

[0040] In an embodiment, the surrogate model 511 can be autoregressive models trained with the FIM task such as Python Code Generator model (PyCodeGPT-110M), "PolyCoder-160M," "CodeParrot-1.5B," and "LLaMa-13B."

[0041] In an embodiment, the surrogate model 511 can be pre-trained with hand-written codes evaluation dataset such as a human evaluated generation model dataset (HumanEval), (HumanEval-X), or a CodeContests dataset.

[0042] In an embodiment, the candidate code probability 514 can be predicted by the surrogate model by predicting the probability of generating a remainder code 508 given a prefix code 507 obtained from a candidate code 501. In an embodiment, the prefix code 507 and remainder code 508 can be obtained by splitting the candidate code 501

by a split ratio 522 (shown in FIG. 3). In an embodiment, the split ratio 522 can be 90%. In another embodiment, the split ratio 522 can be 50%.

[0043] In an embodiment, the surrogate model 511 can predict the candidate code probability 514 by predicting the probability of generating a remainder code 508 given a prefix code obtained from a candidate code 501 by computing the rightmost token logits.

[0044] In an embodiment, the machine-filled codes probabilities 515 can be predicted by the surrogate model 511 by predicting the probability of generating a remainder filled code 510 given a prefix filled code 509 obtained from a machine-filled code 505. In an embodiment, the prefix filled code 509 and remainder filled code 510 can be obtained by splitting the machine-filled code by a split ratio 522. In an embodiment, the split ratio 522 can be 90%. In another embodiment, the split ratio can be 50%.

[0045] In an embodiment, the surrogate model 511 can predict the machine-filled codes probabilities 515 by predicting the probability of generating a remainder filled code 510 given a prefix filled code 509 obtained from a machine-filled code 505 by computing the rightmost token logits.

[0046] In another embodiment, n-grams 519 of the candidate code 501 and the machine-filled codes can be obtained. An n-gram can be a sequence of n adjacent symbols or words in a particular order.

[0047] In block 140, the candidate code 501 can be distinguished as AI-generated by comparing the probabilities against a detection threshold 517 to obtain detection result 518.

[0048] In an embodiment, the detection threshold 517 can be a predetermined ratio that can be an element of zero to one. In an embodiment, the detection threshold 517

can be 0.9. In another embodiment, the detection threshold 517 can be 0.95. The detection threshold 517 can be obtained by maintaining a True Positive Rate (TPR) while minimizing the False Positive Rate (FPR).

[0049] In an embodiment, the probabilities of the machine-filled codes 515 and the candidate code 514 can be compared against detection threshold 517 to obtain detection result 518 with the following:

[0050] $p(Y_0 \mid X) - \frac{1}{N}\sum_{n=1}^{N} p(Y_n \mid X_n) > T$

[0051] where $p(Y_0 \mid X)$ can be the candidate code probability 514 which can be the probability of generating remainder code 508 (Y_0) based on prefix code 507 (X); $p(Y_k \mid X)$ is the machine filed codes probabilities 515 which can be the probability of obtaining remainder filled code 510 (Y_n) based on corresponding prefix filled code 509 (Xn); N can be a number of machine-filled codes, n can be an element of N, and T can be the detection threshold 517.

[0052] In another embodiment, a candidate code 501 can be distinguished as AI generated by comparing n-gram divergences 519 of the candidate code 501 and machine-filled codes 505 against a detection threshold 517 to obtain detection result 518. To compute the n-gram divergences 519 of the candidate code 501 and the machine-filled codes 505 can be obtained by the following:

[0053] $\frac{1}{K}\sum_{k=1}^{K}\sum_{n=n_0}^{N} f(n)\frac{|grams\ (Y_k,n) \cap grams\ (Y_0,n)|}{|Y_k||grams\ (Y_0,n)|} > T;$

[0054] where *grams(Y, n)* can denote a set of all sequence n-grams 504 in sequence Y, Y can include sequences of remainder code Y_0 from the candidate code 501 for the candidate code 501 n-gram, and sequences of remainder filled code 510 Y_k for machine-filled codes 505 n-gram, k can be an element of sample size K, N can be a number of sequences, *f(n)* can be an empirically chosen weight function for different lengths n,

$|Y_k|$ can be a normalized length of sequence Yk used to normalize $grams\ (Y_0, n)$, and T can be the detection threshold 517. In an embodiment, $f(n)$ can be $nlog(n)$, $n_0 = 4$, and $N = 25$.

[0055] In another embodiment, the model output probabilities 513 of the candidate code 501 and machine-filled codes 505 can be obtained from the surrogate model 511 and compared against detection threshold 517 to obtain detection result 518.

[0056] $\frac{1}{N}\sum_{k=1}^{K}\log\frac{p(Y_0\,|X)}{p(Y_k\,|X)} > T;$

[0057] where $p(Y_0\,|X)$ can be a model output probability of remainder code sequence Y_0 and prefix code X; $p(Y_k\,|X)$ can be a model output probability of remainder filled code sequence Y_k and prefix filled code X; k can be a number within sample size K; N can be a number of sequences, and T can be the detection threshold 517.

[0058] Referring now to FIG. 2, a block diagram showing a computing system for detecting AI generated computer code 500, in accordance with an embodiment of the present invention.

[0059] The computing device 500 illustratively includes the processor device 594, an input/output (I/O) subsystem 590, a memory 591, a data storage device 592, and a communication subsystem 593, and/or other components and devices commonly found in a server or similar computing device. The computing device 500 may include other or additional components, such as those commonly found in a server computer (e.g., various input/output devices), in other embodiments. Additionally, in some embodiments, one or more of the illustrative components may be incorporated in, or otherwise form a portion of, another component. For example, the memory 591, or portions thereof, may be incorporated in the processor device 594 in some embodiments.

[0060] The processor device 594 may be embodied as any type of processor capable of performing the functions described herein. The processor device 594 may be embodied as a single processor, multiple processors, a Central Processing Unit(s) (CPU(s)), a Graphics Processing Unit(s) (GPU(s)), a single or multi-core processor(s), a digital signal processor(s), a microcontroller(s), or other processor(s) or processing/controlling circuit(s).

[0061] The memory 591 may be embodied as any type of volatile or non-volatile memory or data storage capable of performing the functions described herein. In operation, the memory 591 may store various data and software employed during operation of the computing device 500, such as operating systems, applications, programs, libraries, and drivers. The memory 591 is communicatively coupled to the processor device 594 via the I/O subsystem 590, which may be embodied as circuitry and/or components to facilitate input/output operations with the processor device 594, the memory 591, and other components of the computing device 500. For example, the I/O subsystem 590 may be embodied as, or otherwise include, memory controller hubs, input/output control hubs, platform controller hubs, integrated control circuitry, firmware devices, communication links (e.g., point-to-point links, bus links, wires, cables, light guides, printed circuit board traces, etc.), and/or other components and subsystems to facilitate the input/output operations. In some embodiments, the I/O subsystem 590 may form a portion of a system-on-a-chip (SOC) and be incorporated, along with the processor device 594, the memory 591, and other components of the computing device 500, on a single integrated circuit chip.

[0062] The data storage device 592 may be embodied as any type of device or devices configured for short-term or long-term storage of data such as, for example, memory devices and circuits, memory cards, hard disk drives, solid state drives, or other

data storage devices. The data storage device 592 can store program code for detecting AI generated computer code 100. Any or all of these program code blocks may be included in a given computing system.

[0063] The communication subsystem 593 of the computing device 500 may be embodied as any network interface controller or other communication circuit, device, or collection thereof, capable of enabling communications between the computing device 500 and other remote devices over a network. The communication subsystem 593 may be configured to employ any one or more communication technology (e.g., wired or wireless communications) and associated protocols (e.g., Ethernet, InfiniBand®, Bluetooth®, Wi-Fi®, WiMAX, etc.) to effect such communication.

[0064] As shown, the computing device 500 may also include one or more peripheral devices 592. The peripheral devices 592 may include any number of additional input/output devices, interface devices, and/or other peripheral devices. For example, in some embodiments, the peripheral devices 592 may include a display, touch screen, graphics circuitry, keyboard, mouse, speaker system, microphone, network interface, and/or other input/output devices, interface devices, GPS, camera, and/or other peripheral devices.

[0065] Of course, the computing device 500 may also include other elements (not shown), as readily contemplated by one of skill in the art, as well as omit certain elements. For example, various other sensors, input devices, and/or output devices can be included in computing device 500, depending upon the particular implementation of the same, as readily understood by one of ordinary skill in the art. For example, various types of wireless and/or wired input and/or output devices can be employed. Moreover, additional processors, controllers, memories, and so forth, in various configurations can also be utilized. These and other variations of the computing system 500 are readily

contemplated by one of ordinary skill in the art given the teachings of the present invention provided herein.

[0066] Referring now to FIG. 3, a block diagram showing a software program for detecting AI generated computer code 300, in accordance with an embodiment of the present invention.

[0067] In an embodiment, a candidate code 501 can be masked by employing a masking module 502 to obtain perturbed codes 503. The perturbed codes 503 having missing codes can be generated by employing an AI code generator model 504 to obtain machine-filled codes 505. The candidate code 501 can be split into a prefix code 507 and a remainder code 508 by employing a splitting module 506. The machine-filled codes 505 can be split into prefix filled codes 509 and remainder filled codes 510. A candidate code probability 514 can be predicted by utilizing the prefix code 507 and the remainder code 508 and by employing the prediction module 512. Machine-filled codes probabilities 515 can be predicted by utilizing prefix filled codes 509 and remainder filled codes 510 and by employing the prediction module 512. The prediction module can utilize the surrogate model 511. The surrogate model 511 can be employed to obtain model output probabilities 513. The detection result 518 can be obtained by comparing candidate code probability 514 and machine-filled codes probabilities 515 against a detection threshold 517.

[0068] In another embodiment, detection result 518 can be obtained by comparing N-gram divergences 519 of remainder code 508 and remainder filled codes 510 against a detection threshold 517. In another embodiment, detection result 518 can be obtained by comparing model output probabilities 513 of remainder code 508 over prefix code 507 and remainder filled codes 510 over corresponding prefix filled codes 509 against a detection threshold 517.

[0069] Referring now to FIG. 4, a block diagram showing a system integrating a practical application for detecting AI generated computer code 600, in accordance with an embodiment of the present invention.

[0070] In an embodiment, an entity 601 can obtain candidate code 501 from healthcare management system 602 that can contain patient vital data 603. AI generated malicious code 611 can be introduced to healthcare management system 602 that can induce potential security risks such as a data breach and expose patient vital data 603 to malicious actors. The candidate code 501 can be inputted into computer system 605 that implements detecting AI generated computer code 100 to obtain flagged candidate code 608. The flagged candidate code 608 can be presented to decision-making entity 609 to perform an action 610. The action 610 can be patching the healthcare management system 602 to secure the data breach and remove the flagged candidate code 608 as malicious code. In another embodiment, the decision-making entity 609 can update the configuration of the healthcare management system 602 to autonomously perform the action 610.

[0071] In another embodiment, AI generated malicious code 611 can be introduced to healthcare management system 602 that can generate false patient vital data. The action 610 can be patching the healthcare management system 602 to remove the false patient vital data and remove the flagged candidate code 608 as malicious code. In another embodiment, the decision-making entity 609 can update the configuration of the healthcare management system 602 to autonomously perform the action 610.

[0072] In another embodiment, AI generated malicious code 611 can be introduced to healthcare management system 602 that can generate false or incorrect medical diagnosis based on patient vital data 603. The action 610 can be patching the healthcare management system 602 to remove the false or incorrect medical diagnosis and remove

the flagged candidate code 608 as malicious code. In another embodiment, the decision-making entity 609 can update the configuration of the healthcare management system 602 to autonomously perform the action 610.

[0073] In another embodiment, AI generated malicious code 611 can be introduced to healthcare management system 602 that can generate inappropriate information such as hate speech, obscenities, defamatory language, threats, blackmail, etc., based on patient vital data 603. The action 610 can be patching the healthcare management system 602 to remove the inappropriate information such as hate speech, obscenities, defamatory language, threats, blackmail, etc., and remove the flagged candidate code 608 as malicious code. In another embodiment, the decision-making entity 609 can update the configuration of the healthcare management system 602 to autonomously perform the action 610.

[0074] In another embodiment, the healthcare management system 602 can be a different computer system not limited to healthcare, such as enterprise systems, public systems, educational institution systems, etc. Other computer systems are contemplated.

[0075] For the following embodiments, the system 602 can be an enterprise system, public data system, educational institution system, etc.

[0076] In another embodiment, the action 610 can be a decision-making entity 609 approving the flagged candidate code 608 as adhering to best practices, coding conventions, or quality standards. The system 602 can then merge the approved candidate code with the main codebase.

[0077] In another embodiment, the action 610 can be a decision-making entity 609 labeling the flagged candidate code 608 as plagiarized. The system 602 can then

provide evidence of plagiarism to the decision-making entity and the author of the candidate code 501.

[0078] In another embodiment, the action 610 can be a decision-making entity 609 checking the flagged candidate code 608 for debugging and maintenance. The system 602 can then create code flags that can include flagged candidate code 608 for debugging.

[0079] In another embodiment, the action 610 can be creating code hooks including the flagged candidate code 608 that would flag code behavior and target testing strategies to ensure the correctness and reliability of the software being developed.

[0080] In another embodiment, the action 610 can be labelling the flagged candidate code 608 as containing redundancies, unnecessary complexity, or inefficient constructs for refactoring and optimizing the code for improved performance and maintainability.

[0081] In another embodiment, the action 610 can be alerting the original author of a flagged candidate code 608 as proprietary code that has been detected in an unauthorized use of such code.

[0082] Referring now to FIG. 5, a block diagram showing an overview of deep learning neural networks, in accordance with an embodiment of the present invention.

[0083] A neural network is a generalized system that improves its functioning and accuracy through exposure to additional empirical data. The neural network becomes trained by exposure to the empirical data. During training, the neural network stores and adjusts a plurality of weights that are applied to the incoming empirical data. By applying the adjusted weights to the data, the data can be identified as belonging to a particular predefined class from a set of classes or a probability that the inputted data belongs to each of the classes can be output.

[0084] The empirical data, also known as training data, from a set of examples can be formatted as a string of values and fed into the input of the neural network. Each example may be associated with a known result or output. Each example can be represented as a pair, (x, y), where x represents the input data and y represents the known output. The input data may include a variety of different data types and may include multiple distinct values. The network can have one input node for each value making up the example's input data, and a separate weight can be applied to each input value. The input data can, for example, be formatted as a vector, an array, or a string depending on the architecture of the neural network being constructed and trained.

[0085] The neural network "learns" by comparing the neural network output generated from the input data to the known values of the examples and adjusting the stored weights to minimize the differences between the output values and the known values. The adjustments may be made to the stored weights through back propagation, where the effect of the weights on the output values may be determined by calculating the mathematical gradient and adjusting the weights in a manner that shifts the output towards a minimum difference. This optimization, referred to as a gradient descent approach, is a non-limiting example of how training may be performed. A subset of examples with known values that were not used for training can be used to test and validate the accuracy of the neural network.

[0086] During operation, the trained neural network can be used on new data that was not previously used in training or validation through generalization. The adjusted weights of the neural network can be applied to the new data, where the weights estimate a function developed from the training examples. The parameters of the estimated function which are captured by the weights are based on statistical inference.

[0087] The deep neural network 1000, such as a multilayer perceptron, can have an input layer 911 of source nodes 912, one or more computation layer(s) 926 having one or more computation nodes 932, and an output layer 940, where there is a single output node 942 for each possible category into which the input example could be classified. An input layer 911 can have a number of source nodes 912 equal to the number of data values 912 in the input data 911. The computation nodes 932 in the computation layer(s) 926 can also be referred to as hidden layers, because they are between the source nodes 912 and output node(s) 942 and are not directly observed. Each node 932, 942 in a computation layer generates a linear combination of weighted values from the values output from the nodes in a previous layer, and applies a non-linear activation function that is differentiable over the range of the linear combination. The weights applied to the value from each previous node can be denoted, for example, by w_1, w_2, ... w_{n-1}, w_n. The output layer provides the overall response of the network to the inputted data. A deep neural network can be fully connected, where each node in a computational layer is connected to all other nodes in the previous layer, or may have other configurations of connections between layers. If links between nodes are missing, the network is referred to as partially connected.

[0088] In an embodiment, the computation layers 926 of the AI code generator model 504 can generate series of sequences of code based on a candidate code 501, and the context and syntax of the candidate code 501. The output layer 940 of the AI code generator model 504 can then provide the overall response of the network to the candidate text 501 as a generated missing code 520. In another embodiment, the computation layers 926 of the surrogate model 511 can generate probability weights of the generated missing code 520 sequences based on the candidate code 501. The output layer 940 of the surrogate model 511 can then provide the overall response of the

network to the candidate code 501 as a generated missing code 502 and a model output probability 513.

[0089] Training a deep neural network can involve two phases, a forward phase where the weights of each node are fixed and the input propagates through the network, and a backwards phase where an error value is propagated backwards through the network and weight values are updated.

[0090] The computation nodes 932 in the one or more computation (hidden) layer(s) 926 perform a nonlinear transformation on the input data 912 that generates a feature space. The classes or categories may be more easily separated in the feature space than in the original data space.

[0091] Embodiments described herein may be entirely hardware, entirely software or including both hardware and software elements. In a preferred embodiment, the present invention is implemented in software, which includes but is not limited to firmware, resident software, microcode, etc.

[0092] Embodiments may include a computer program product accessible from a computer-usable or computer-readable medium providing program code for use by or in connection with a computer or any instruction execution system. A computer-usable or computer readable medium may include any apparatus that stores, communicates, propagates, or transports the program for use by or in connection with the instruction execution system, apparatus, or device. The medium can be magnetic, optical, electronic, electromagnetic, infrared, or semiconductor system (or apparatus or device) or a propagation medium. The medium may include a computer-readable storage medium such as a semiconductor or solid state memory, magnetic tape, a removable computer diskette, a random access memory (RAM), a read-only memory (ROM), a rigid magnetic disk and an optical disk, etc.

[0093] Each computer program may be tangibly stored in a machine-readable storage media or device (e.g., program memory or magnetic disk) readable by a general or special purpose programmable computer, for configuring and controlling operation of a computer when the storage media or device is read by the computer to perform the procedures described herein. The inventive system may also be considered to be embodied in a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner to perform the functions described herein.

[0094] A data processing system suitable for storing and/or executing program code may include at least one processor coupled directly or indirectly to memory elements through a system bus. The memory elements can include local memory employed during actual execution of the program code, bulk storage, and cache memories which provide temporary storage of at least some program code to reduce the number of times code is retrieved from bulk storage during execution. Input/output or I/O devices (including but not limited to keyboards, displays, pointing devices, etc.) may be coupled to the system either directly or through intervening I/O controllers.

[0095] Network adapters may also be coupled to the system to enable the data processing system to become coupled to other data processing systems or remote printers or storage devices through intervening private or public networks. Modems, cable modem and Ethernet cards are just a few of the currently available types of network adapters.

[0096] As employed herein, the term "hardware processor subsystem" or "hardware processor" can refer to a processor, memory, software or combinations thereof that cooperate to perform one or more specific tasks. In useful embodiments, the hardware processor subsystem can include one or more data processing elements (e.g., logic

circuits, processing circuits, instruction execution devices, etc.). The one or more data processing elements can be included in a central processing unit, a graphics processing unit, and/or a separate processor- or computing element-based controller (e.g., logic gates, etc.). The hardware processor subsystem can include one or more on-board memories (e.g., caches, dedicated memory arrays, read only memory, etc.). In some embodiments, the hardware processor subsystem can include one or more memories that can be on or off board or that can be dedicated for use by the hardware processor subsystem (e.g., ROM, RAM, basic input/output system (BIOS), etc.).

[0097] In some embodiments, the hardware processor subsystem can include and execute one or more software elements. The one or more software elements can include an operating system and/or one or more applications and/or specific code to achieve a specified result.

[0098] In other embodiments, the hardware processor subsystem can include dedicated, specialized circuitry that performs one or more electronic processing functions to achieve a specified result. Such circuitry can include one or more application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), and/or programmable logic arrays (PLAs).

[0099] These and other variations of a hardware processor subsystem are also contemplated in accordance with embodiments of the present invention.

[0100] Reference in the specification to "one embodiment" or "an embodiment" of the present invention, as well as other variations thereof, means that a particular feature, structure, characteristic, and so forth described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, the appearances of the phrase "in one embodiment" or "in an embodiment", as well any other variations, appearing in various places throughout the specification are not necessarily all referring

to the same embodiment. However, it is to be appreciated that features of one or more embodiments can be combined given the teachings of the present invention provided herein.

[0101] It is to be appreciated that the use of any of the following "/", "and/or", and "at least one of", for example, in the cases of "A/B", "A and/or B" and "at least one of A and B", is intended to encompass the selection of the first listed option (A) only, or the selection of the second listed option (B) only, or the selection of both options (A and B). As a further example, in the cases of "A, B, and/or C" and "at least one of A, B, and C", such phrasing is intended to encompass the selection of the first listed option (A) only, or the selection of the second listed option (B) only, or the selection of the third listed option (C) only, or the selection of the first and the second listed options (A and B) only, or the selection of the first and third listed options (A and C) only, or the selection of the second and third listed options (B and C) only, or the selection of all three options (A and B and C). This may be extended for as many items listed.

[0102] The foregoing is to be understood as being in every respect illustrative and exemplary, but not restrictive, and the scope of the invention disclosed herein is not to be determined from the Detailed Description, but rather from the claims as interpreted according to the full breadth permitted by the patent laws. It is to be understood that the embodiments shown and described herein are only illustrative of the present invention and that those skilled in the art may implement various modifications without departing from the scope and spirit of the invention. Those skilled in the art could implement various other feature combinations without departing from the scope and spirit of the invention. Having thus described aspects of the invention, with the details and particularity required by the patent laws, what is claimed and desired protected by Letters Patent is set forth in the appended claims.

WHAT IS CLAIMED IS:

1. A computer-implemented method for detecting artificial intelligence (AI) generated computer code, comprising:

masking (110) lines of code from a candidate code to obtain perturbed codes;

generating (120) missing code from the perturbed codes by employing an AI code generator model to obtain machine-filled codes;

predicting (130) probabilities of the candidate code and the machine-filled codes as AI-generated by employing a surrogate model; and

distinguishing (140) the candidate code as AI-generated by comparing the probabilities against a detection threshold to obtain detection results.

2. The computer-implemented method of claim 1, further comprising flagging the candidate code as AI-generated to detect malicious code for a decision-making entity to perform an action.

3. The computer-implemented method of claim 2, wherein the action is securing a healthcare management system handling patient vital data by patching the flagged candidate code for potential security risks.

4. The computer-implemented method of claim 1, wherein predicting probabilities further comprises predicting the candidate code probability by predicting the probability of generating a remainder code given a prefix code.

5. The computer-implemented method of claim 4, wherein distinguishing the candidate code further comprises comparing n-gram divergence of a prefix code and a remainder code and n-gram divergence of prefix filled codes and remainder filled codes.

6. The computer-implemented method of claim 1, wherein predicting probabilities further comprises predicting machine-filled codes probabilities by predicting the probability of generating a remainder filled code given a prefix filled code.

7. The computer-implemented method of claim 1, wherein distinguishing the candidate code further comprises computing a difference between the candidate code probability and the machine-filled codes probabilities.

8. A system for detecting artificial intelligence (AI) generated computer code, comprising:

a memory (592); and

one or more processor devices (594) in communication with the memory (592) configured to:

mask (110) lines of code from a candidate code to obtain perturbed codes;

generate (120) missing code from the perturbed codes by employing an AI code generator model to obtain machine-filled codes;

predict (130) probabilities of the candidate code and the machine-filled codes as AI-generated by employing a surrogate model; and

distinguish (140) the candidate code as AI-generated by comparing the probabilities against a detection threshold to obtain detection results.

9. The system of claim 8, further comprising the processor device flagging the candidate code as AI-generated to detect malicious code for a decision-making entity to perform an action.

10. The system of claim 9, wherein the processor device performs the action securing a healthcare management system handling patient vital data by patching the flagged candidate code for potential security risks.

11. The system of claim 8, wherein predicting probabilities by the processor device further comprises predicting the candidate code probability by predicting the probability of generating a remainder code given a prefix code.

12. The system of claim 11, wherein distinguishing the candidate code by the processor device further comprises comparing n-gram divergence of a prefix code and a remainder code and n-gram divergence of prefix filled codes and remainder filled codes.

13. The system of claim 8, wherein predicting probabilities by the processor device further comprises predicting machine-filled codes probabilities by predicting the probability of generating a remainder filled code given a prefix filled code.

14. The system of claim 8, wherein distinguishing the candidate code by the processor device further comprises computing a difference between the candidate code probability and the machine-filled codes probabilities.

15. A non-transitory computer program product comprising a computer-readable storage medium including program code for detecting artificial intelligence (AI) generated computer code, wherein the program code when executed on a computer causes the computer to perform:

masking (110) lines of code from a candidate code to obtain perturbed codes;

generating (120) missing code from the perturbed codes by employing an AI code generator model to obtain machine-filled codes;

predicting (130) probabilities of the candidate code and the machine-filled codes as AI-generated by employing a surrogate model; and

distinguishing (140) the candidate code as AI-generated by comparing the probabilities against a detection threshold to obtain detection results.

16. The non-transitory computer program product of claim 15, further comprising flagging the candidate code as AI-generated to detect malicious code for a decision-making entity to perform an action.

17. The non-transitory computer program product of claim 16, wherein the action is securing a healthcare management system handling patient vital data by patching the flagged candidate code for potential security risks.

18. The non-transitory computer program product of claim 15, wherein predicting probabilities further comprises predicting the candidate code probability by predicting the probability of generating a remainder code given a prefix code.

19. The non-transitory computer program product of claim 15, wherein predicting probabilities further comprises predicting machine-filled codes probabilities by predicting the probability of generating a remainder filled code given a prefix filled code.

20. The non-transitory computer program product of claim 18, wherein distinguishing the candidate code further comprises computing a difference between the candidate code probability and the machine-filled codes probabilities.

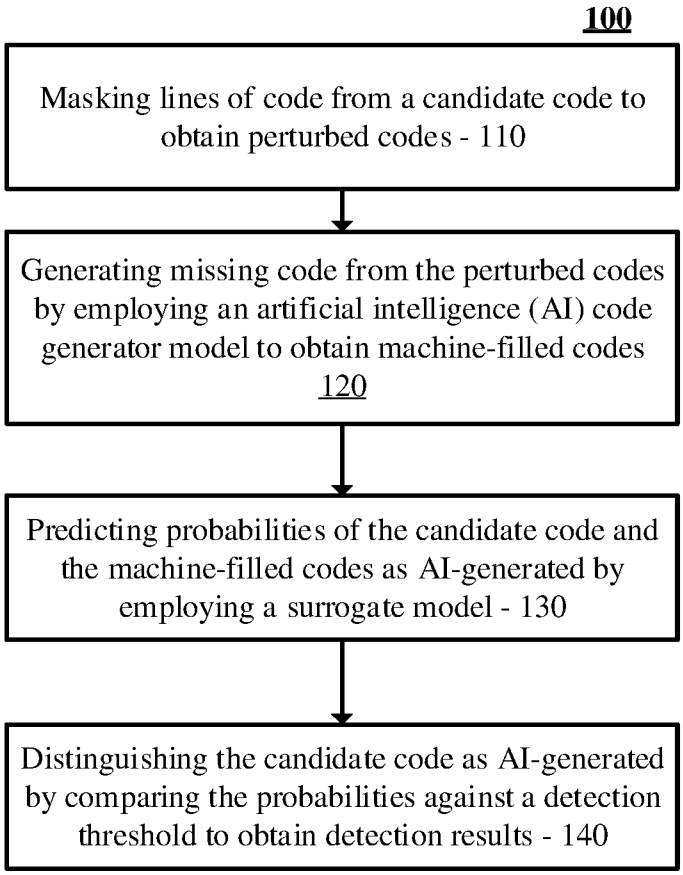
100
Masking lines of code from a candidate code to obtain perturbed codes - 110
Generating missing code from the perturbed codes by employing an artificial intelligence (AI) code generator model to obtain machine-filled codes 120
Predicting probabilities of the candidate code and the machine-filled codes as AI-generated by employing a surrogate model - 130
Distinguishing the candidate code as AI-generated by comparing the probabilities against a detection threshold to obtain detection results - 140

FIG. 1

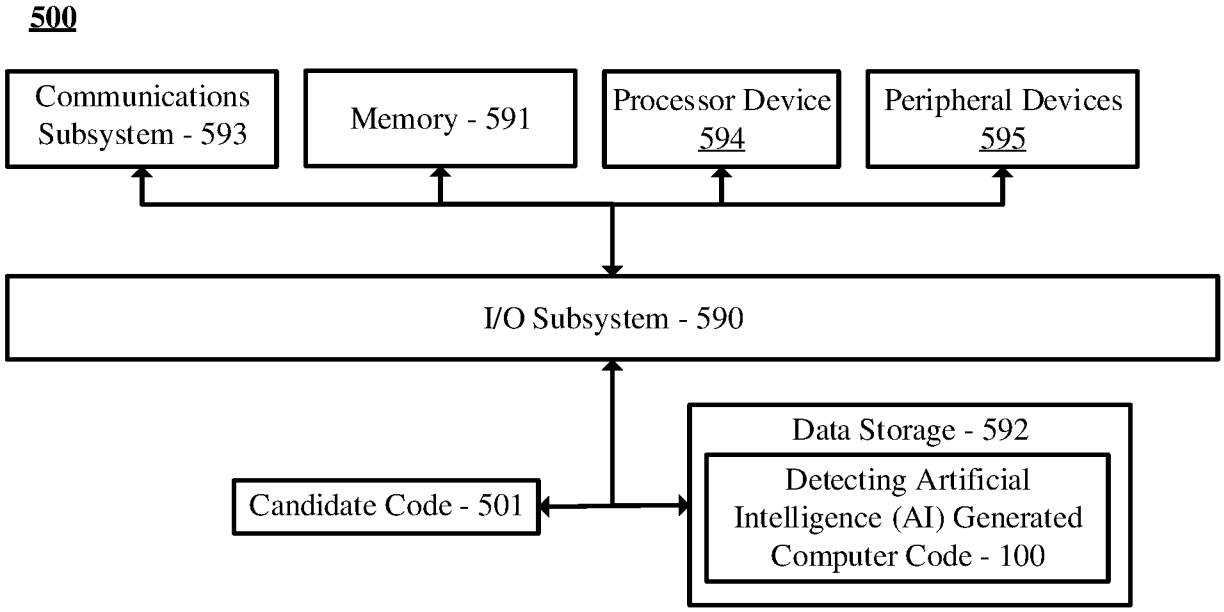
500
Communications Subsystem - 593
Memory - 591
Processor Device 594
Peripheral Devices 595
I/O Subsystem - 590
Data Storage - 592
Detecting Artificial Intelligence (AI) Generated Computer Code - 100
Candidate Code - 501

FIG. 2

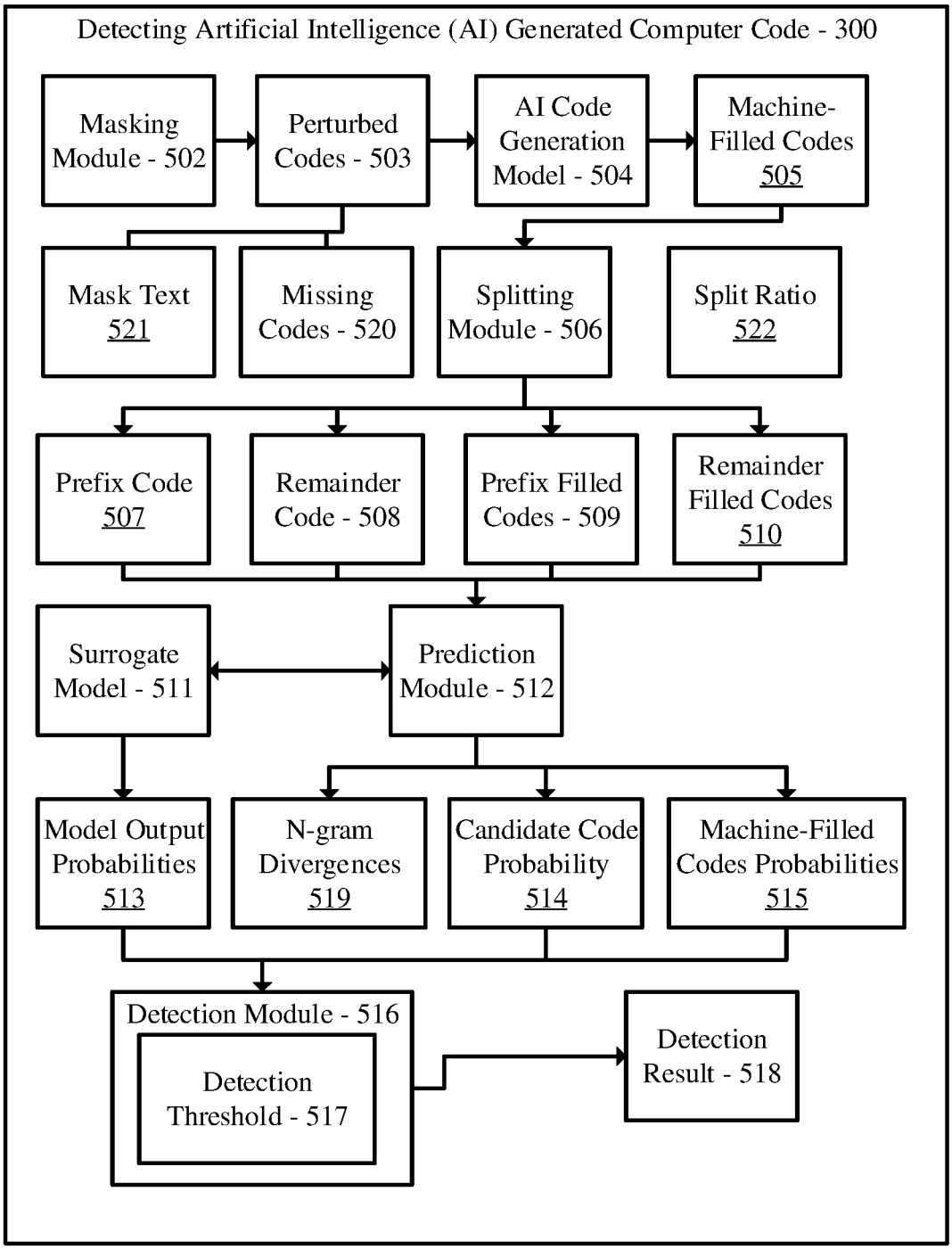
Detecting Artificial Intelligence (AI) Generated Computer Code - 300
Masking Module - 502
Perturbed Codes - 503
AI Code Generation Model - 504
Machine-Filled Codes 505
Mask Text 521
Missing Codes - 520
Splitting Module - 506
Split Ratio 522
Prefix Code 507
Remainder Code - 508
Prefix Filled Codes - 509
Remainder Filled Codes 510
Surrogate Model - 511
Prediction Module - 512
Model Output Probabilities 513
N-gram Divergences 519
Candidate Code Probability 514
Machine-Filled Codes Probabilities 515
Detection Module - 516
Detection Threshold - 517
Detection Result - 518

FIG. 3

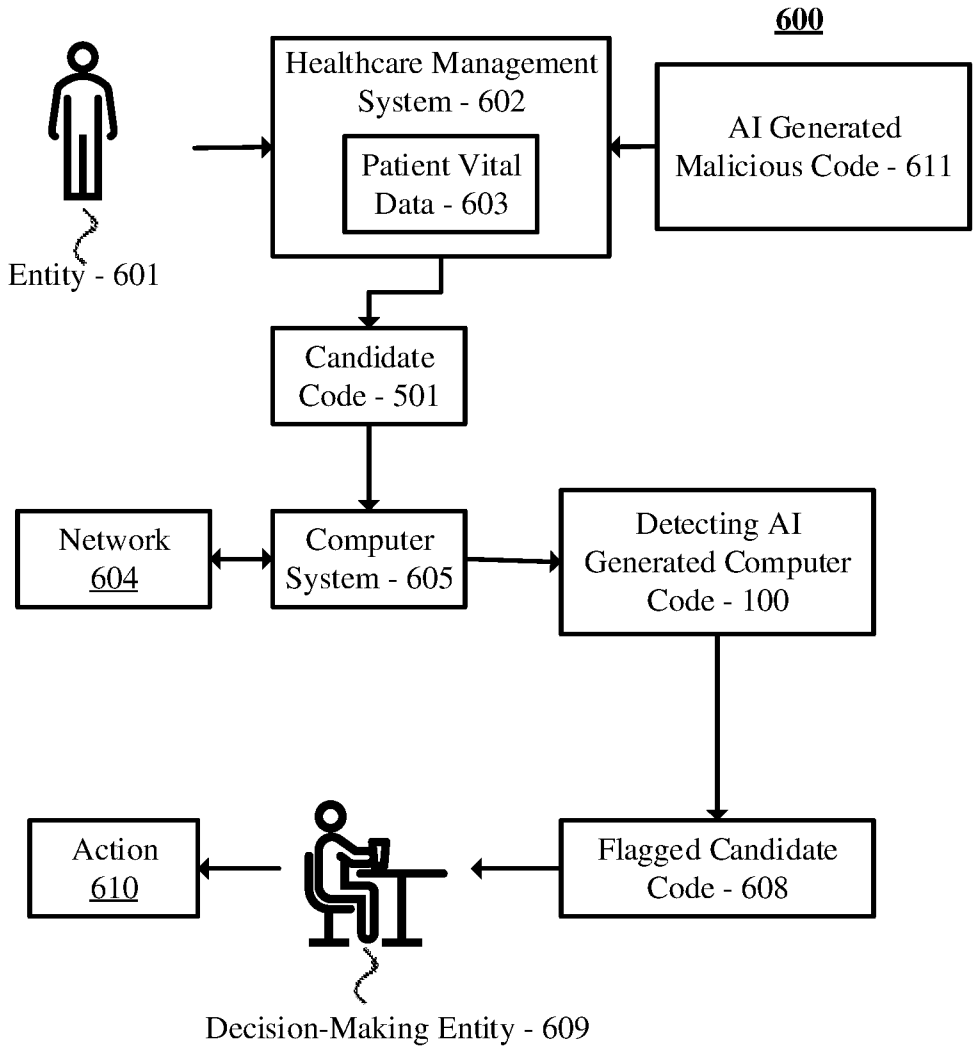
600
Healthcare Management System - 602
Patient Vital Data - 603
AI Generated Malicious Code - 611
Entity - 601
Candidate Code - 501
Network 604
Computer System - 605
Detecting AI Generated Computer Code - 100
Action 610
Flagged Candidate Code - 608
Decision-Making Entity - 609

FIG. 4

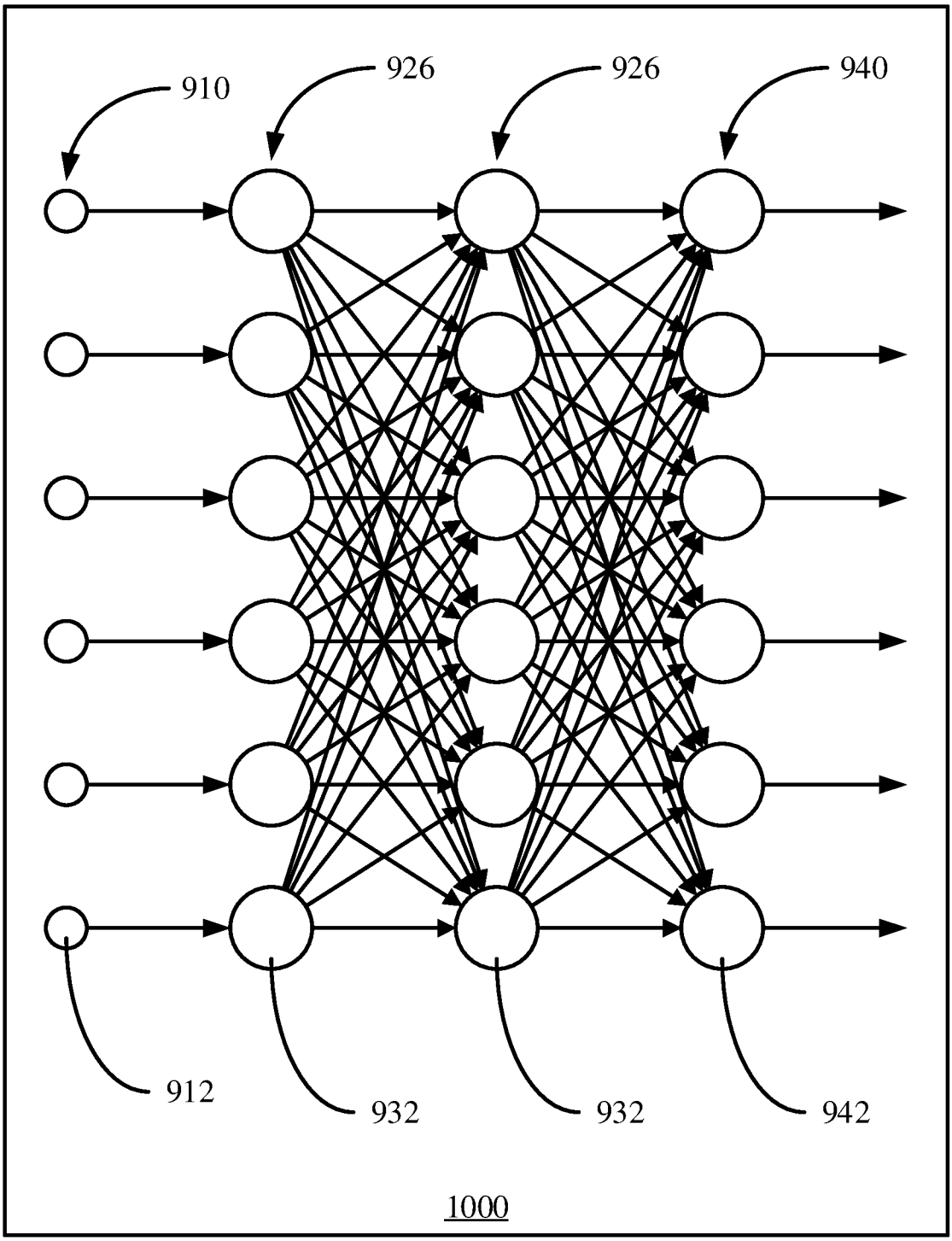

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2024/032377

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/56(2013.01)i; **G06F 21/62**(2013.01)i; **G06F 8/33**(2018.01)i; **G06N 20/00**(2019.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 21/56(2013.01); G06F 21/36(2013.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: detect, artificial intelligence, generate, computer code, masking, line, candidate, perturb, missing, machine-filled, probability, surrogate model, distinguish, compare, threshold

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	ERIC MITCHELL et al., 'DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature', arXiv:2301.11305v1, January 2023 [retrieved on 2024-09-04]. Retrieved from: <URL: https://arxiv.org/abs/2301.11305v1> pages 1-8	1-20
A	ZHIJIE DENG et al., 'Efficient Detection of LLM-generated Texts with a Bayesian Surrogate Model', arXiv:2305.16617v1, May 2023 [retrieved on 2024-09-04]. Retrieved from: <URL: https://arxiv.org/abs/2305.16617v1> pages 1-10	1-20
A	JIAN WANG et al., 'Evaluating AIGC Detectors on Code Content', arXiv:2304.05193v1, April 2023 [retrieved on 2024-09-04]. Retrieved from: <URL: https://arxiv.org/abs/2304.05193v1> pages 1-11	1-20
A	ZHANGYIN FENG et al., 'CodeBERT: A Pre-Trained Model for Programming and Natural Languages', arXiv:2002.08155v1, February 2020 [retrieved on 2024-09-04]. Retrieved from: <URL: https://arxiv.org/abs/2002.08155v1> pages 1-8	1-20

☑ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 September 2024	**23 September 2024**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office **189 Cheongsa-ro, Seo-gu, Daejeon** **35208, Republic of Korea**	**YANG, Jeong Rok**
Facsimile No. **+82-42-481-8578**	Telephone No. **+82-42-481-5709**

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2024/032377

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2017-0109515 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 20 April 2017 (2017-04-20) paragraphs [0008]-[0066]; claims 1-3; and figures 1-2	1-20

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2024/032377

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2017-0109515 A1	20 April 2017	US 10311218 B2	04 June 2019