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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 May 2019 (16.05.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/094933 A1

(51) International Patent Classification:
G06F 11/36 (2006.01)

(21) International Application Number:
PCT/US2018/060747

(22) International Filing Date:
13 November 2018 (13.11.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/585,416 13 November 2017 (13.11.2017) US

(71) Applicant: **THE CHARLES STARK DRAPER LABORATORY, INC.** [US/US]; 555 Technology Square, Cambridge, MA 02139 (US).

(72) Inventors: **HARER, Jacob**; 555 Technology Square, Cambridge, MA 02139 (US). **LAZOVICH, Tomo**; 555 Technology Square, Cambridge, MA 02139 (US). **RUSSELL, Rebecca**; 555 Technology Square, Cambridge, MA 02139 (US). **OZDEMIR, Onur**; 555 Technology Square, Cambridge, MA 02139 (US). **KIM, Louis**; 555 Technology Square, Cambridge, MA 02139 (US).

(74) Agent: **WAKIMURA, Mary, Lou** et al.; Hamilton, Brook, Smith & Reynolds, P.C., 530 Virginia Rd, P.O. Box 9133, Concord, MA 01742-9133 (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, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME,

(54) Title: AUTOMATED REPAIR OF BUGS AND SECURITY VULNERABILITIES IN SOFTWARE

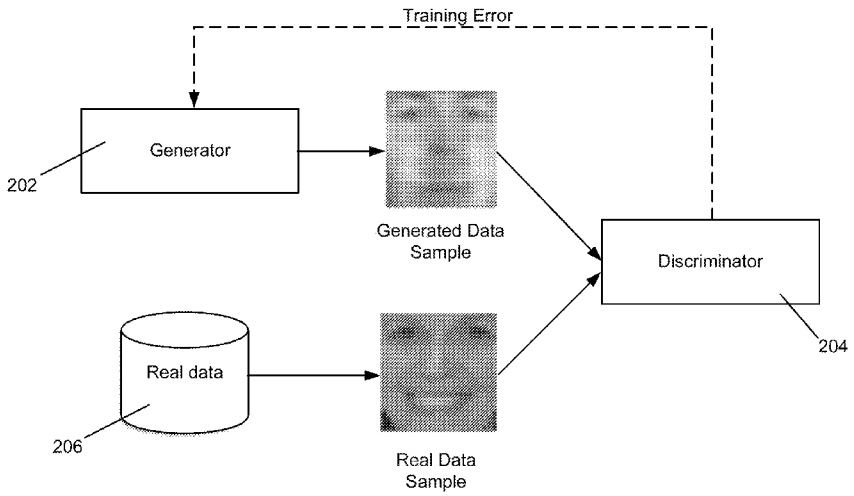

FIG. 2

(57) Abstract: A software instruction code repair system comprising an instruction code example pool. The example pool comprises a set of good instruction code examples and a set of bad instruction code examples. The software instruction code repair system further comprises a sequence-to-sequence (seq2seq) network that is configured to generate a corrected instruction code example, based on one example of the set of bad instruction code examples. The software instruction code repair system further comprises a discriminator configured to randomly select one of the corrected instruction code example and one instruction code example of the set of good instruction code examples to produce a selected instruction code example. The discriminator is further configured to make a determination that the selected instruction code example is most likely taken either the instruction code example pool or the seq2seq network.

WO 2019/094933 A1

[Continued on next page]

MG, MK, MN, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, 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, 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).

Published:

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

AUTOMATED REPAIR OF BUGS AND SECURITY VULNERABILITIES IN SOFTWARE

RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 62/585,416, filed on November 13, 2017. The entire teachings of the above application are incorporated herein by reference.

GOVERNMENT SUPPORT

[0002] This invention was made with government support under FA8750-15-C-0242 from U.S. Department of Defense. The government has certain rights in the invention.

BACKGROUND

[0003] Writing and subsequent validating (e.g., debugging or repairing) of software instruction code is often a time-consuming process. Reducing or eliminating human involvement, particularly in the validation aspect, is desirable, because doing so increases efficiency and reduces costs.

[0004] Existing efforts to automate instruction code debug and/or repair have various associated drawbacks. For example, sequence to sequence (seq2seq) techniques (described in more detail herein) have been shown to perform well on datasets such as language translation and language correction, and efforts are ongoing to extend seq2seq techniques to correction of software instruction codes. One of the major downsides to seq2seq, and indeed most of the previous work on sequence correction, is that it requires the data be in pairs of incorrect to correct code. The models are then trained to perform the correction provided given the incorrect code. This type of data can be difficult to obtain, and is most likely required to be hand annotated.

SUMMARY

[0005] The described embodiments are directed to automatically correcting flaws in software source code without humans in the loop. Many software bugs (i.e., flaws in source code) go undetected for long periods of time. The described embodiments employ novel

machine learning techniques that will take in a flawed or buggy piece of source code and automatically output a corrected version of the source code. While the described embodiments are described with respect to C/C++ source code, the underlying concepts of the described embodiments may be generalized to accommodate any software language. Further, the described embodiments may be used to in other applications, for example natural language correction. In fact, the underlying concepts of the described embodiments may be generally applied to any application characterized by a sequence that may have an improved "good" state associated with an initial "bad" state.

[0006] Novel aspects of the described embodiments include, but are not limited to, (i) the use of a sequence to sequence (seq2seq) network model as a generator in a Generative Adversarial Network (GAN) architecture, and (ii) the application of such a GAN architecture to repair or otherwise correct flaws in software instruction code (i.e., producing "good," corrected instruction code from "bad," buggy instruction code, based on examples of real, confirmed good software instruction code).

[0007] In one aspect, the invention may be a sequential data repair system comprising an example pool comprising a set of good examples and a set of bad examples, a sequence-to-sequence (seq2seq) network configured to generate a corrected example based on an example of the set of bad examples, and a discriminator. The discriminator may be configured to randomly select either the corrected example or an example of the set of good examples, so as to produce a selected example, and make a determination from which source the selected example was most likely selected -- the example pool or the seq2seq network.

[0008] The discriminator and seq2seq network may be configured as a generative adversarial network (GAN). The seq2seq network may comprise an encoder configured to produce an intermediate representation of an input data sequence, and a decoder configured to produce and output data sequence based on the intermediate representation of the input data sequence.

[0009] In an embodiment, each good example in the set of good examples may represent a sequence of data elements having a quality that equals or exceeds a threshold quality level of sequential data associated with the sequential data repair system. Each bad example in the set of bad examples may represent a sequence of data elements having a quality that is less than or equal to a minimum threshold quality level of sequential data associated with the sequential data repair system. Each bad example in the set of bad examples may be labeled

to indicate a bad example, and each good example in the set of good examples is labeled to indicate a good example.

[0010] The discriminator may generate a training error signal based on the determination. The seq2seq network may comprise a deep learning neural network model, and the seq2seq network may use the training error signal to train the deep learning neural network model.

[0011] In another aspect, the invention may be a software instruction code repair system comprising an instruction code example pool comprising a set of good instruction code examples and a set of bad instruction code examples, a sequence-to-sequence (seq2seq) network configured to generate a corrected instruction code example based on one example of the set of bad instruction code examples, and a discriminator. The discriminator may be configured to randomly select either the corrected instruction code example or an instruction code example of the set of good instruction code examples, to produce a selected instruction code example, and make a determination that the selected instruction code example was most likely selected from either the instruction code example pool or the seq2seq network.

[0012] In another aspect, the invention may be a method of repairing a sequence of data elements, comprising providing a set of good instruction code examples and a set of bad instruction code examples, generating, using a sequence-to-sequence (seq2seq) network, a corrected instruction code example based on one example of the set of bad instruction code examples, randomly selecting, using a discriminator, either the corrected instruction code example or an instruction code example of the set of good instruction code examples, to produce a selected instruction code example, and making a determination that the selected instruction code example was most likely selected from a particular source – either the instruction code example pool or the seq2seq network.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The foregoing will be apparent from the following more particular description of example embodiments, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating embodiments.

[0014] FIG. 1 shows the basic operation of the seq2seq network model.

[0015] FIG. 2 shows an example of a generative adversarial network (GAN).

[0016] FIG. 3 shows an example embodiment of a code repair system according to the invention.

[0017] FIG. 4 shows an example embodiment of an internal structure of a processing system that may be used to implement one or more of the embodiments herein.

[0018] FIG. 5 illustrates an example embodiment of a method of repairing a sequence of data elements.

DETAILED DESCRIPTION

[0019] A description of example embodiments follows.

[0020] The teachings of all patents, published applications and references cited herein are incorporated by reference in their entirety.

[0021] The described embodiments are directed to methods of and systems for automated processing of software instruction code. Related information in this general technology field may also be found in U.S. Patent Application No. 14/735639, entitled "Systems And Methods For Software Analysis," filed on June 10, 2015, the entire contents of which are hereby incorporated by reference.

[0022] The described embodiments are directed to a technique for training a neural network model to perform repairs on source code, without the need for good/bad examples of the same piece of code to use for training. This technique is referred to herein as "RepairGAN."

[0023] In general, RepairGAN is a deep learning neural network model configured to learn how to repair sequential data without the need for paired training examples to demonstrate fixes. Instead, the network trains purely on labels associated with individual examples. In short, the network can learn how to translate bad or non-compliant designs into good designs by observing many individual examples of both bad and good cases.

[0024] RepairGAN is a combination of two existing, commonly used techniques in the machine learning community: (i) sequence-to-sequence (seq2seq) networks and (ii) generative adversarial networks (GANs). The seq2seq network learns how to take in a sequence of data and output a transformed sequence. The GAN training setup is used to "adversarially train" the seq2seq network so that it learns how to transform bad sequences into good sequences, without needing examples of good/bad pairs.

[0025] One example use of seq2seq networks is machine translation. Given an input sequence of tokens from a sentence in one language, the seq2seq network may be trained to output the same sentence in a different language. Seq2seq networks may also be effective for grammar correction.

[0026] FIG. 1 illustrates the basic operation of the seq2seq network model 100. The seq2seq network model 100 comprises an encoder network 102 and a decoder network 104. The encoder network 102 takes in a first sequence of tokens 106, and maps the first sequence of tokens 106 into an encoded vector representation 108 (also referred to herein as an intermediate representation). The decoder network 104 takes in the intermediate representation 108 and outputs a second sequence of tokens 110. For example, the encoder network 102 may take in an English sentence, and map the English sentence to the intermediate representation 108. The decoder network 104 may then read in this intermediate representation 108 and output a corresponding French sentence.

[0027] The seq2seq network model may be used to "translate" buggy code into correct code. The best translation networks may be trained on hundreds of millions of sentence pair examples, where the input sentence and corresponding output sentence are known. Obtaining this many examples in the software domain is virtually impossible. The described embodiments of the RepairGAN approach, described herein, may be used to avoid a need for such large numbers of input/output training pairs. Thus, while seq2seq networks are powerful, it is difficult to find enough data in most domains on which to train them. Therefore, the described embodiments combine a seq2seq networks with another network, the generative adversarial network, to make a robust, trainable repair system.

[0028] Generative adversarial networks (GAN) utilize a model that is trained to generate data associated with a particular training set. In the example embodiment shown in FIG. 2, the generated data is a set of celebrity head-shots. The GAN comprises a generator network 202 and a discriminator network 204, which compete in a minimax game. The discriminator network 204 randomly (e.g., blindly or arbitrarily) takes in a data point with a goal of determining whether the selected data point was sampled from the true data distribution (i.e., the training set 206 – stored in, e.g., a database or other data storage facility), or from the generator network 202 directly. The goal of the generator network 202 is produce a generated image, of a quality level that is so close to the real data distribution, such that the discriminator network 204 will be “fooled” into making a determination that the generated

image was sampled from the real data distribution. Through feedback 208, from the discriminator network 204, the generator network 202 thus learns how to generate very realistic looking data samples.

[0029] The RepairGAN approach of the described embodiments employs a novel use of the GAN discriminator network 204 described herein. FIG. 3 is an example embodiment of a code repair system according to the invention. Rather than simply discriminating between generated and real examples, the RepairGAN discriminator network 304 is also trained to distinguish between (i) good coding examples from the training set, and (ii) good coding examples that have been generated by the generator network based on bad coding instances.

[0030] The seq2seq network 100, described herein, is employed as the RepairGAN generator network 302. A training set of real coding examples 306, comprising both good examples 308 and bad examples 310, is stored in a coding pool (e.g., database or other storage facility). The bad examples 310 are provided to the seq2seq network 302. Samples 312 from the "generated" distribution are made by feeding known bad examples 310 into the seq2seq network 302 and obtaining a corrected, “good” example 312 from the output of the decoder. The seq2seq network 302 uses feedback 316 from the discriminator 304 to improve its sequence conversion ability, with the goal of generating a good coding example 312 from a bad coding example 310, such that the generated coding example approaches or matches the quality of the good coding example 308. Thus, the seq2seq network 302 learns how to transform know bad examples into examples that “look like” good examples.

[0031] The “good” coding examples 314 are sampled from a known good coding example distribution 308, which are stored in the real coding pool 306. Each of the coding examples 312, 314, provided to the discriminator 304, includes a label (e.g., “good _real,” “good_generated”) that characterizes the coding example. The discriminator network 304 uses the labels to provide training error feedback 316 to the seq2seq network 302. The training error feedback may reflect the quality of the generated coding sample 312 relative to the real coding sample 314. Using this feedback 316, the seq2seq network learns how to transform known bad examples into generated examples 312 that look like the real good examples 314. The labels may be used to update the discriminator as well. The label may be used to both train the discriminator to be better at differentiating real examples from generated examples, and to update the generator to better fool the discriminator.

[0032] A significant advantage of the described embodiments is that individual labeled examples may be used, rather than fully paired before-and-after examples of the same piece of code. The use of labeled examples thus significantly reduces difficulty of needing to obtain data required to train the seq2seq network 302, as compared to other such network architectures.

[0033] When benchmarked on a synthetic grammar repair problem, an example code repair system, constructed according to the described embodiments, was able to repair 64 percent of broken sentences on a single pass. Similarly, the example code repair system repaired about 20 percent of broken code examples on its first attempt.

[0034] FIG. 4 is a diagram of an example internal structure of a processing system 400 that may be used to implement one or more of the embodiments herein. Each processing system 400 contains a system bus 402, where a bus is a set of hardware lines used for data transfer among the components of a computer or processing system. The system bus 402 is essentially a shared conduit that connects different components of a processing system (e.g., processor, disk storage, memory, input/output ports, network ports, etc.) that enables the transfer of information between the components.

[0035] Attached to the system bus 402 is a user I/O device interface 404 for connecting various input and output devices (e.g., keyboard, mouse, displays, printers, speakers, etc.) to the processing system 400. A network interface 406 allows the computer to connect to various other devices attached to a network 408. Memory 410 provides volatile and non-volatile storage for information such as computer software instructions used to implement one or more of the embodiments of the present invention described herein, for data generated internally and for data received from sources external to the processing system 400.

[0036] A central processor unit 412 is also attached to the system bus 402 and provides for the execution of computer instructions stored in memory 410. The system may also include support electronics/logic 414, and a communications interface 416. The communications interface may, for example, receive good and bad code examples from a data storage facility as described herein.

[0037] In one embodiment, the information stored in memory 410 may comprise a computer program product, such that the memory 410 may comprise a non-transitory computer-readable medium (e.g., a removable storage medium such as one or more DVD-ROM's, CD-ROM's, diskettes, tapes, etc.) that provides at least a portion of the software

instructions for the invention system. The computer program product can be installed by any suitable software installation procedure, as is well known in the art. In another embodiment, at least a portion of the software instructions may also be downloaded over a cable communication and/or wireless connection.

[0038] It will be apparent that one or more embodiments described herein may be implemented in many different forms of software and hardware. Software code and/or specialized hardware used to implement embodiments described herein is not limiting of the embodiments of the invention described herein. Thus, the operation and behavior of embodiments are described without reference to specific software code and/or specialized hardware – it being understood that one would be able to design software and/or hardware to implement the embodiments based on the description herein.

[0039] Further, certain embodiments of the example embodiments described herein may be implemented as logic that performs one or more functions. This logic may be hardware-based, software-based, or a combination of hardware-based and software-based. Some or all of the logic may be stored on one or more tangible, non-transitory, computer-readable storage media and may include computer-executable instructions that may be executed by a controller or processor. The computer-executable instructions may include instructions that implement one or more embodiments of the invention. The tangible, non-transitory, computer-readable storage media may be volatile or non-volatile and may include, for example, flash memories, dynamic memories, removable disks, and non-removable disks.

[0040] FIG. 5 illustrates an example embodiment of a method of repairing a sequence of data elements, comprising providing 502 a set of good instruction code examples and a set of bad instruction code examples, and generating 504, using a sequence-to-sequence (seq2seq) network, a corrected instruction code example based on one example of the set of bad instruction code examples. The method may further comprise randomly selecting 506, using a discriminator, either the corrected instruction code example or an instruction code example of the set of good instruction code examples, to produce a selected instruction code example, and making a determination 508 that the selected instruction code example was most likely selected from either the instruction code example pool or the seq2seq network.

[0041] While example embodiments have been particularly shown and described, it will be understood by those skilled in the art that various changes in form and details may be

made therein without departing from the scope of the embodiments encompassed by the appended claims.

CLAIMS

What is claimed is:

1. A sequential data repair system, comprising:

an example pool comprising a set of good examples and a set of bad examples;

a sequence-to-sequence (seq2seq) network configured to generate a corrected example based on an example of the set of bad examples;

a discriminator configured to:

(a) randomly select either the corrected example or an example of the set of good examples, to produce a selected example, and

(b) make a determination that the selected example was most likely selected from the example pool or the seq2seq network.

2. The sequential data repair system of claim 1, wherein the discriminator and seq2seq network are configured as a generative adversarial network (GAN).

3. The sequential data repair system of claim 1, wherein the seq2seq network comprises an encoder configured to produce an intermediate representation of an input data sequence, and a decoder configured to produce and output data sequence based on the intermediate representation of the input data sequence.

4. The sequential data repair system of claim 1, wherein each good example in the set of good examples represents a sequence of data elements having a quality that equals or exceeds a threshold quality level of sequential data associated with the sequential data repair system.

5. The sequential data repair system of claim 1, wherein each bad example in the set of bad examples represents a sequence of data elements having a quality that is less than or equal to a minimum threshold quality level of sequential data associated with the sequential data repair system.

6. The sequential data repair system of claim 1, wherein each bad example in the set of bad examples is labeled to indicate a bad example, and each good example in the set of good examples is labeled to indicate a good example.

7. The sequential data repair system of claim 1, wherein the discriminator generates a training error signal based on the determination.

8. The sequential data repair system of claim 7, wherein the seq2seq network comprises a deep learning neural network model, and the seq2seq network uses the training error signal to train the deep learning neural network model.

9. A software instruction code repair system, comprising:

an instruction code example pool comprising a set of good instruction code examples and a set of bad instruction code examples;

a sequence-to-sequence (seq2seq) network configured to generate a corrected instruction code example based on one example of the set of bad instruction code examples;

a discriminator configured to:

(a) randomly select either the corrected instruction code example or an instruction code example of the set of good instruction code examples, to produce a selected instruction code example, and

(b) make a determination that the selected instruction code example was most likely selected from either the instruction code example pool or the seq2seq network.

10. The software instruction code repair system of claim 9, wherein the discriminator and seq2seq network are configured as a generative adversarial network (GAN).

11. The software instruction code repair system of claim 9, wherein the seq2seq network comprises an encoder configured to produce an intermediate representation of an input software instruction code, and a decoder configured to produce and output code sequence based on the intermediate representation of the input software instruction code.

12. The software instruction code repair system of claim 9, wherein each good example in the set of good examples represents a software instruction code example having a quality that equals or exceeds a threshold quality level of software instruction code associated with the software instruction code repair system.

13. The software instruction code repair system of claim 9, wherein each bad example in the set of bad examples represents a software instruction code example having a quality that is less than or equal to a minimum threshold quality level of software instruction code associated with the software instruction code repair system.

14. The software instruction code repair system of claim 9, wherein each bad example in the set of bad examples is labeled to indicate a bad software instruction code example, and each good example in the set of good examples is labeled to indicate a good software instruction code example.

15. The software instruction code repair system of claim 9, wherein the discriminator generates a training error signal based on the determination.

16. The sequential data repair system of claim 15, wherein the seq2seq network comprises a deep learning neural network model, and the seq2seq network uses the training error signal to train the deep learning neural network model.

17. A method of repairing a sequence of data elements, comprising:

providing a set of good instruction code examples and a set of bad instruction code examples;

generating, using a sequence-to-sequence (seq2seq) network, a corrected instruction code example based on one example of the set of bad instruction code examples;

randomly selecting, using a discriminator, either the corrected instruction code example or an instruction code example of the set of good instruction code examples, to produce a selected instruction code example, and

making a determination that the selected instruction code example was most likely selected from either the instruction code example pool or the seq2seq network.

18. The method of claim 17, further comprising configuring the discriminator and seq2seq network as a generative adversarial network (GAN).

19. The method of claim 17, further comprising generating, by the discriminator, a training error based on the determination.

20. The method of claim 17, further comprising using, by the seq2seq network, the training error signal to train a deep learning neural network model, which is a constituent component of the seq2seq network.

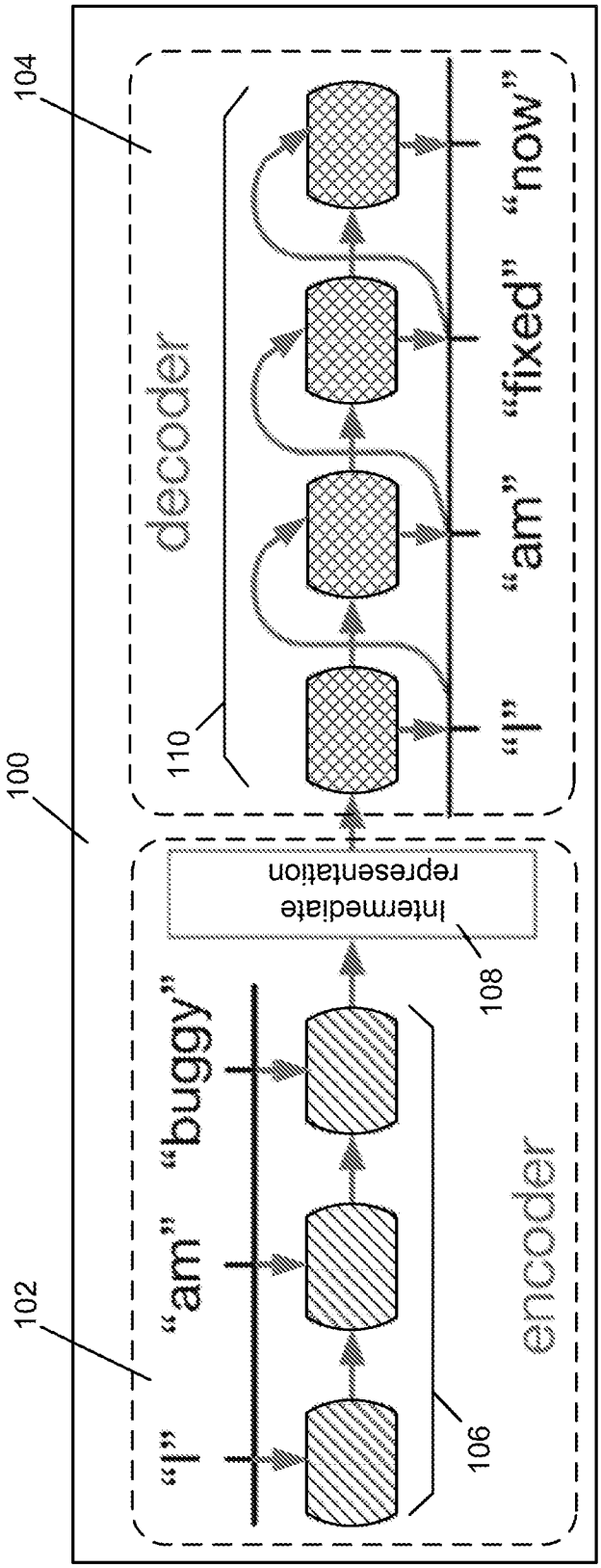
100
102
104
106
108
110
"I"
"am"
"buggy"
encoder
intermediate representation
decoder
"I"
"am"
"fixed"
"now"

FIG. 1

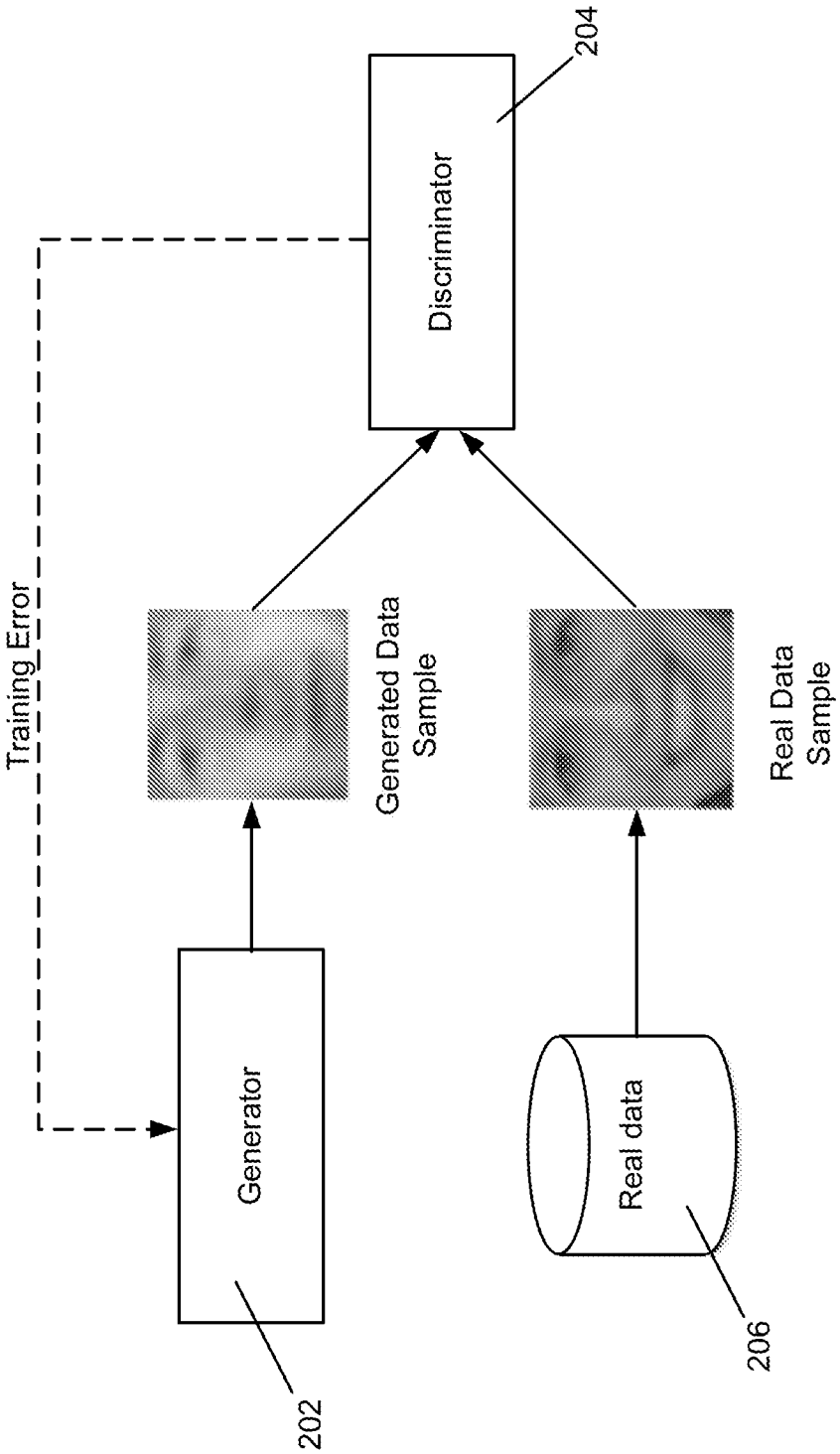
Discriminator
204
Training Error
Generated Data Sample
Real Data Sample
Generator
Real data
202
206

FIG. 2

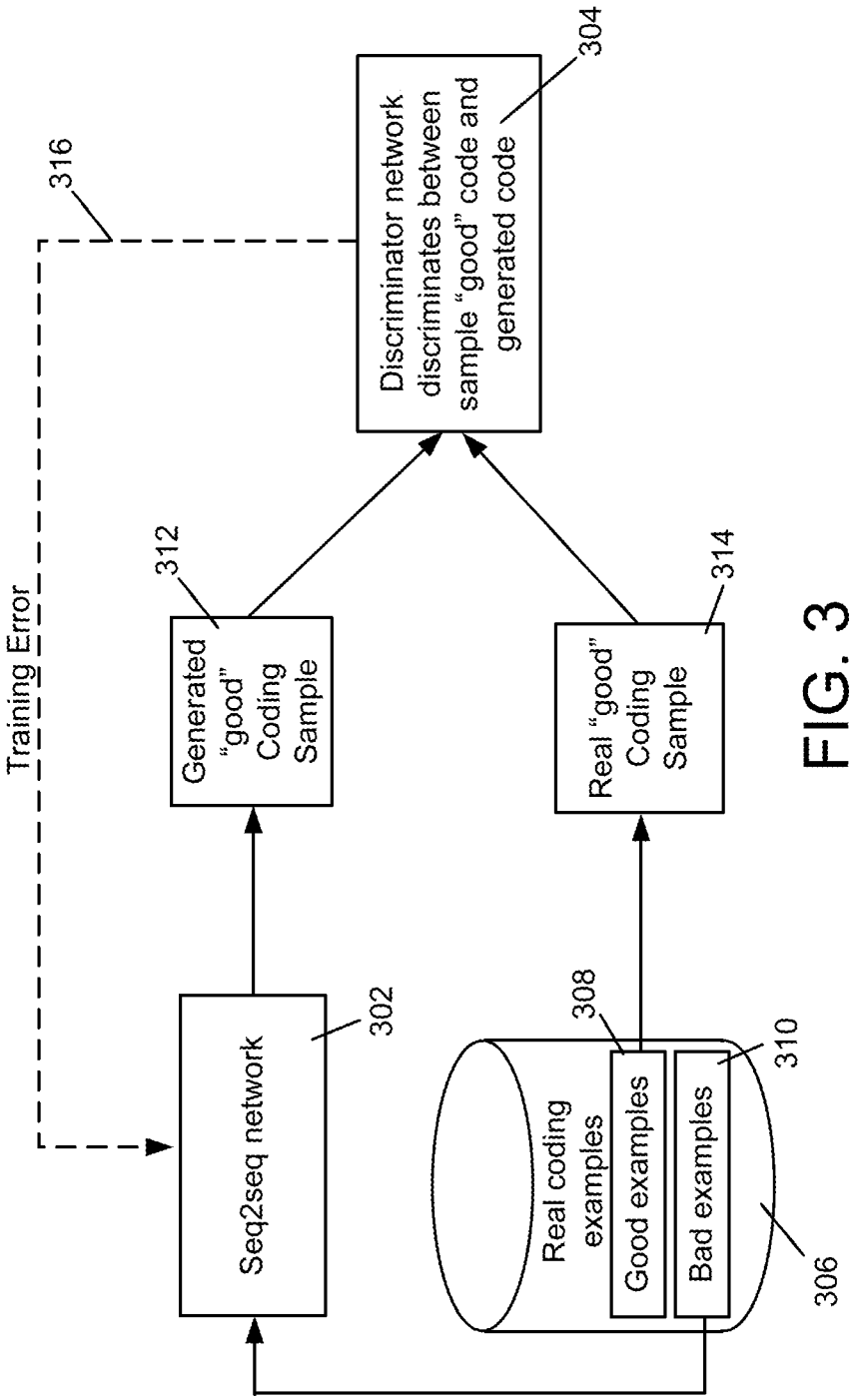
Discriminator network discriminates between sample "good" code and generated code
304
316
Training Error
Generated "good" Coding Sample
312
Real "good" Coding Sample
314
Seq2seq network
302
Real coding examples
Good examples
Bad examples
308
310
306

FIG. 3

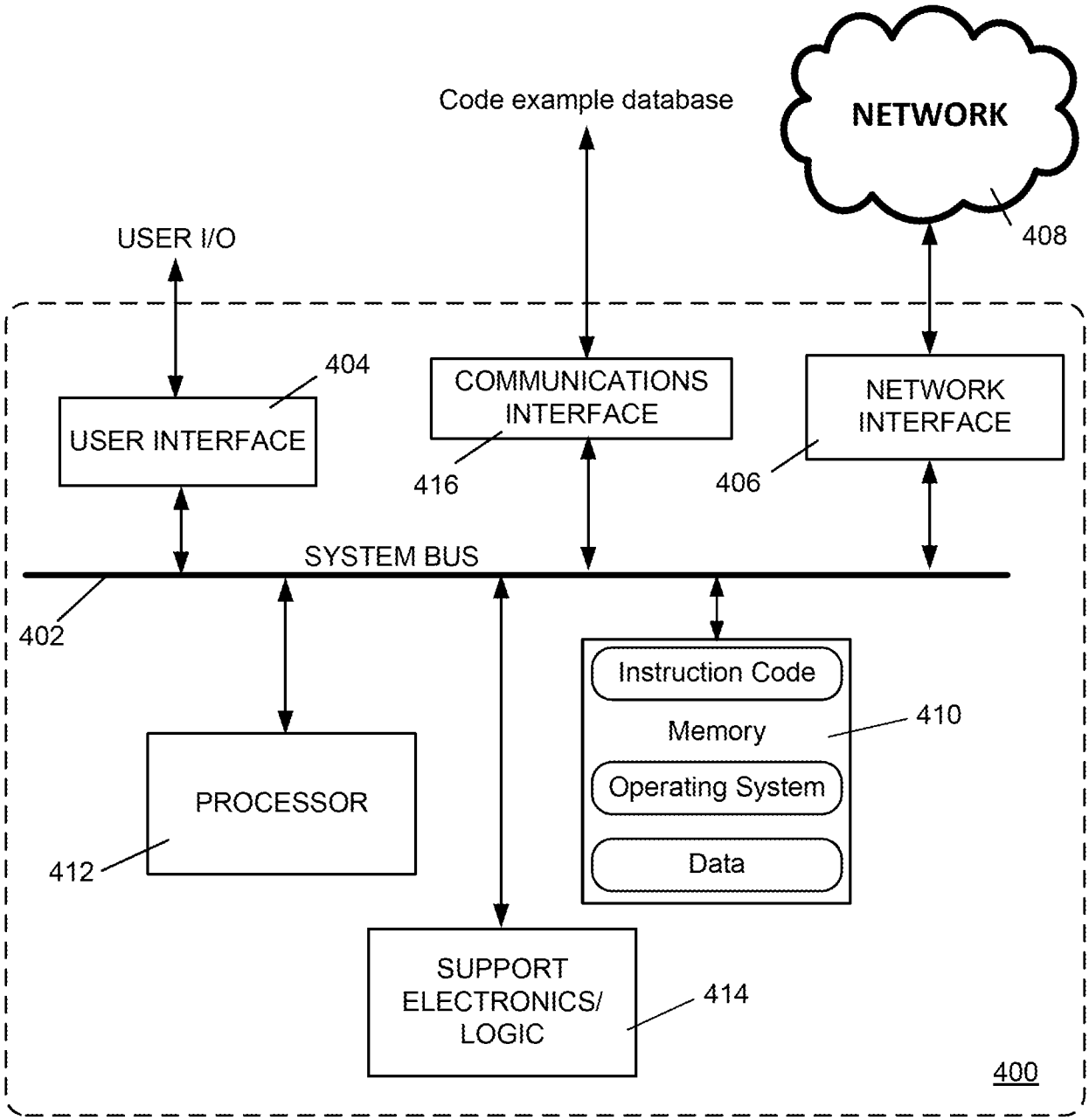
Code example database
NETWORK
408
USER I/O
404
USER INTERFACE
COMMUNICATIONS INTERFACE
416
NETWORK INTERFACE
406
SYSTEM BUS
402
Instruction Code
Memory
410
Operating System
Data
PROCESSOR
412
SUPPORT ELECTRONICS/ LOGIC
414
400

FIG. 4

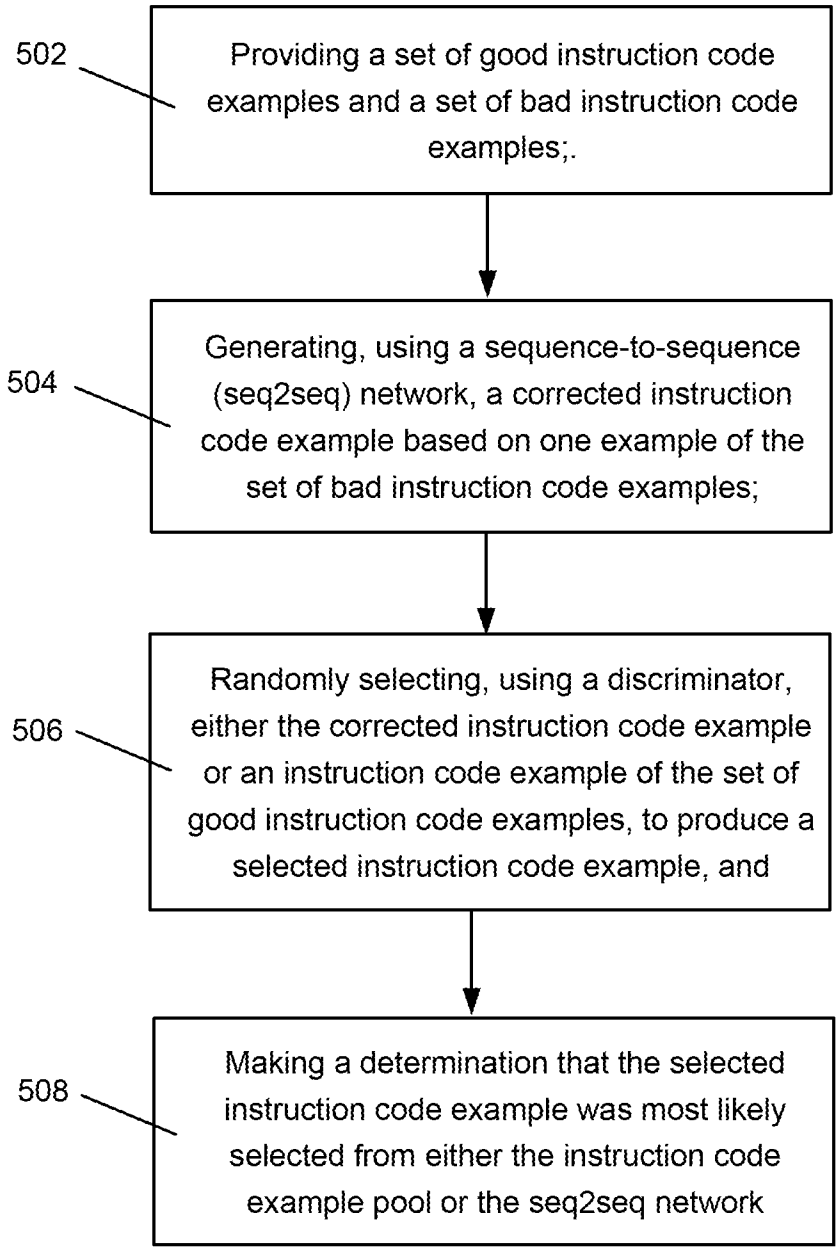
502
Providing a set of good instruction code examples and a set of bad instruction code examples;.
504
Generating, using a sequence-to-sequence (seq2seq) network, a corrected instruction code example based on one example of the set of bad instruction code examples;
506
Randomly selecting, using a discriminator, either the corrected instruction code example or an instruction code example of the set of good instruction code examples, to produce a selected instruction code example, and
508
Making a determination that the selected instruction code example was most likely selected from either the instruction code example pool or the seq2seq network

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2018/060747

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F11/36
ADD.

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

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Jacob Devlin ET AL: "Semantic Code Repair using Neuro-Symbolic Transformation Networks", , 30 October 2017 (2017-10-30), XP055552310, Retrieved from the Internet: URL:https://arxiv.org/pdf/1710.11054.pdf [retrieved on 2019-02-06] the whole document ----- -/--	1-20

[X] Further documents are listed in the continuation of Box C.

[X] 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

"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
6 February 2019	20/02/2019

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Renault, Sophie

8

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CLAIRE LE GOUES ET AL: "GenProg: A Generic Method for Automatic Software Repair", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING., vol. 38, no. 1, 1 January 2012 (2012-01-01), pages 54-72, XP055552517, US ISSN: 0098-5589, DOI: 10.1109/TSE.2011.104 the whole document	1-20
A	WO 2015/191731 A1 (DRAPER LAB CHARLES S [US]) 17 December 2015 (2015-12-17) abstract	1-20
A	GUPTA AND AL.: "DeepFix : Fixing Common C Language Errors by Deep Learnibg", PROCEEDINGS OF THE THIRTY-FIRST AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 4 February 2017 (2017-02-04), pages 1345-1351, XP002788667, the whole document	1-20

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2018/060747

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2015191731 A1	17-12-2015	CA 2949244 A1	17-12-2015
		CA 2949248 A1	17-12-2015
		CA 2949251 A1	17-12-2015
		CN 106537332 A	22-03-2017
		CN 106537333 A	22-03-2017
		CN 106663003 A	10-05-2017
		EP 3155512 A1	19-04-2017
		EP 3155513 A1	19-04-2017
		EP 3155514 A1	19-04-2017
		JP 2017517821 A	29-06-2017
		JP 2017519300 A	13-07-2017
		JP 2017520842 A	27-07-2017
		US 2015363196 A1	17-12-2015
		US 2015363197 A1	17-12-2015
		US 2015363294 A1	17-12-2015
		WO 2015191731 A1	17-12-2015
		WO 2015191737 A1	17-12-2015
		WO 2015191746 A1	17-12-2015