Devising a Respondent Network Bot

Richard Pham

August 2022

1 Introduction

This paper provides a generic description to a network bot called Respondent Network Bot (RNB), inspired by question-and-answer dynamical systems and truth logic. Question-and-answer dynamical systems require two classes of entities, questioners and answerers in which the questioning object queries the answering object for an answer. In more complex situations, these two classes of entities can continually swap their roles so that each of them does not have to remain as an entity that questions the other, an active role, or the passive role of an entity that answers to the other. The program, Respondent Network Bot, does not consider the complexities that follow from such role-swapping and does not grant any entity the capability to switch roles. Respondent Network Bot is given its name due to the program's central focus of the answering object's response.

The components of this program were designed with simplicity in mind. There is only one questioning entity called Q and Q contains a set of questions q'. For each question in q', Q may know the answer before it conducts questioning so that any answer it receives can be cross-referenced with its known answer. The body of agents that Q is to question is a network of nodes, each node is dubbed a RNBNode (respondent node bot node). Described is a brief overview of the forces involved in Respondent Network Bot.

2 Answers and Truth in RNB

There are some premises regarding answers and truth that this program satisfies.

Lemma 2.1. *An answer by any entity to a question is not necessarily the truth.*

Lemma 2.2. *A known answer by an entity to a question is a special class of an answer such that the entity believes with confidence of 100 percent that the answer is a factual response to that question. Such answers may be considered wrong.*

Lemma 2.3. *Each known answer by any entity in this program is static. These statements are immutable and cannot be modified by any agent.*

Lemma 2.4. *Each known answer is present in the Respondent Network Bot system before the program commences its question-and-answer event chain.*

Lemma 2.5. *In the event that the questioner Q does not possess a known answer to a question q, the only answer/s that Q can gather is through its questioning of nodes from the network N.*

Lemma 2.6. *The answer that an answering object will provide to the questioning object is its objective answer.*

One prevailing principle behind the premises stated above is that the questioner Q and the questioned nodes are not pre-ordained to the status of being "right" or "wrong" based on any of their possessions including their answers to their questions. A third-party will not know with certainty what the factual answer to any question q in q' is by observing the event chain of the entities in an instance of Respondent Network Bot, but will be able to gather all pertinent information about how the nodes in the network and Q act in accordance with the set of questions.

3 The RNB Node

The RNB Node is a data structure that depends on another data structure called the Respondent Node Database (RNDB) for data-processing capabilities.

Definition 3.1. RNDB. A database structure owned by any RNB Node r with the following attributes:
1. An answers map A_m:

$$\texttt{question identifier} \rightarrow \texttt{possible known answer of node}$$

$$n_0 \in \mathbb{N} \rightarrow (n_1 \in \mathbb{Z})|?.$$

2. An objective map O_m:

$$\texttt{question identifier} \rightarrow \texttt{objective}$$

$$n_0 \in \mathbb{N} \rightarrow n_1 \in \{0, 1, 2\}.$$

3. A satisfaction rate map S_m that records how well every other RNB Node known to r answers according to the objective of r to every $q \in Q$:

$$\texttt{node identifier} \rightarrow \{\texttt{question identifier} \rightarrow \texttt{satisfaction rate}\}$$
$$n_0 \in \mathbb{N} \rightarrow \{n_1 \in \mathbb{N} \rightarrow s \in [0, 1]\}.$$

Attributes 1 and 2 are immutable values.

Respondent Node Database is the generic term given to the information that can be obtained by an RNB Node in the sequence of events that take place during the run of a Respondent Network Bot.

Definition 3.2. RNB Node. An object r that can connect to other objects of its same class. This object is a unit in the Respondent Network Bot with the following attributes:
1. An identifier,i, in the form of an integer.
2. A set of integers,S, that identifies its neighbor nodes.
3. A non-increasing float,t,that enumerates its resistance to failing its objective with respect to any question.
4. An RNDB, d.
5. A feedback map F that determines how well the the questioner Q responds to the answer provided by r for question q:

`question identifier` $\rightarrow$ `sequence of deltas of resistance by Q's response to r's answer.`

$$n \in \mathbb{N} \rightarrow < q | q \in \mathbb{R} > .$$

The implementation of Respondent Network Bot restricts Definition 3.2.2, the neighbor nodes, of RNB node to an immutable set so that the network of RNB nodes will have identical edges at the end of the Respondent Network Bot's run.

3.1 To Answer or Let Others Answer?

The satisfaction rate map discussed in Definition 3.1.3 pertains to the action of question delegation. Delegation is one of two primary actions of each RNB node n_r in response to a question q posed by the questioner Q. *Instead of directly answering q to Q, n_r can answer q by "delegating" to other nodes*, in other words, distributing the identifier of q to other nodes and "deriving" a "collective" answer from all the answers it gathered. The node n_r will use its knowledge obtained throughout the running instance of Respondent Network Bot to determine those nodes best fit to answer according to its objective answer. This knowledge is represented by S_m provided in Definition 3.1.3. The map S_m is a mutable structure that when accessed by keys i, j outputs the satisfaction rate of node i in answering question j if n_r chooses to delegate question j to node i. The satisfaction rate of a node i to answer question j can be determined by measuring the degree of difference between the objective answer of delegate node i in response to question j for source node n_r. For practical purposes, this degree of difference is in the range $[0, 1]$ and is scaled according to the number of times node i serves as one delegate for question j to source node n_r. Please read the computer code for the specific formulas used to calculate satisfaction rates involved in this implementation of Respondent Network Bot.

The task of n_r finding the appropriate delegate nodes for a question q involves n_r distributing the identifier of q to other nodes in a breadth-first search

manner. For every one of these nodes n_{r_x} that receives the question identifier message from n_r, n_r determines if n_{r_x} is suitable based on the value $S_m[n_{r_x}.i][q.i]$. If a specific node n_{r_x} is determined suitable, then n_{r_x} distributes the identifier of q to its neighboring nodes not yet considered by n_r. Otherwise, it will not distribute the identifier of q to any other node, and the number of nodes calculated to be suitable as delegates to n_r will not change through the medium n_{r_x}. There is one big implication for this design for question delegation.

- The delegation process is edge-distance pertinent. Each node n_{r_x} considered by n_r to be a delegate for question q is required to either have a neighbor that is a suitable candidate for delegation by n_r or be a neighbor to n_r.

Another logistic of question delegation is that it first simulates delegating to suitable nodes. Every time n_r is questioned with question q, it performs a simulatory delegation that determines the feasibility of delegating. If the contradiction from some cost function D_f is less than that if the node n_r were to directly answer, then n_r will choose to delegate using the candidate nodes for delegation in that simulation. Read Section 5.1 in the proceeding pages for elaboration on what contradiction is in the dynamic between Q and the questioned node n_r.

For a network R of RNB Nodes, each node $n_r \in R$ will not initially know if any other node will satisfy the objective answer of n_r. So at the start of Respondent Network Bot, the satisfaction map S_m of node n_r's RNDB will have for each possible (i, j) pair a value of 1.0, full satisfaction. And every time n_r actually delegates, it updates the map S_m with revised values, typically according to the frequency of the considered node n_{r_x}.

Remark. The description above of how an RNB node decides to delegate is general and does not use specific formulae or numbers.

4 A Brief Description of the Q Structure

The Q structure is the data structure responsible for questioning the network of RNB nodes. The following variables are of principal importance for Respondent Network Bot's question-and-answer feedback loop.

Definition 4.1. Question. A structure possessed by both the questioning object and the answering objects with three components:
1. A unique identifier that differentiates the question from another question.
2. A range of possible answers, represented as a range of values $v_r = [v_{r_0}, v_{r_1}] \in \mathbb{Z}$, $v_{r_0} \leq v_{r_1}$.
3. Either a value $v \in v_r$ or an unknown.

Components 1 and 2 are immutable by the specific entity that holds them, and every instance of this structure held by two different entities will have

components 1 and 2 that are identical. The only difference could be component 3, their known or unknown answer.

The questioning object Q has read-write access to a database structured according to Definition 4.2.

Definition 4.2. D_Q. A database used by Q during its activity in a running instance of Respondent Network Bot. The database has four primary variables, each represented as an n x m two-dimensional matrix, n is the number of nodes in the Respondent Network Bot and m is the number of questions that each RNB node and Q focuses its activity on. An element at index i, j of one of these variables corresponds to node i and question j. The four variables are then:
1. The rate that node i is a delegate to question j , W.
2. The rate of contradiction, X.
3. The duplicates of a question j asked to node i, Y.
4. The mean of the answers of a node i to a question j, Z.

Definition 4.3. Q Structure. The one questioning object in Respondent Network Bot with the attributes:
1. A vector $V_q =< \texttt{q}|\texttt{q a question} >$.
2. A database instance of D_Q.
3. A set of identifiers for the subset of nodes with the F_2 label in Respondent Network Bot.
4. A finite positive non-increasing integer,f, that denotes the amount of finite energy Q has for acting against the RNB nodes in a network.

More on Definition 4.3.3 and 4.4.4 will be provided later. *The general objective of a Q structure in a running instance of Respondent Network Bot is to obtain the least amount of contradiction in the answers it gathers from the RNB nodes in the network with the most remaining fuel f left.*

Further description of this objective will be provided in Section 5.1 because the cost functions for a particular action of Q has not yet been touched on.

5 The Simplicity of Answers in RNB

Real-life answers are complex and there are certainly instances where those that are questioned give answers that fall out of the bounds of the expectations of the questioner, and those answers may take various forms outside of the scope of immediate numerical representation by an intelligent entity. Respondent Network Bot ignores the complexities posed by this possibility, that of finding an accurate mapping between an input space (such as words or sentences or pictures) and the output space (a numerical structure). The answers present in Respondent Network Bot is the output space, consisting of values in the field $\mathbb{Z}$.

The possible answers for each question is a continuous range of integers, and the question structure in Respondent Network Bot is identified by a number $z \in \mathbb{Z}$.

The similarity between each integer in the range of integers is distance-proportional. For example, given a range $v_r = (-23, 100)$, the similarity function S_f between two answers will satisfy the relation

$$S_f(24, 25) > S_f(24, 26) > \cdots > S_f(24, 100).$$

Each node (an answerer) will output an answer $a \in \mathbb{Z}$ to a question in the form of an integer from the permitted range of integers $v_r \in \mathbb{Z}$ the question is designated with. If a node has a known answer to a question, it will output an answer based on its objective regarding the questioner by the general guidelines:

- If objective is true (no deception), output the known answer.
- If objective is false (deception), output the farthest integer value in the possible range of answers.
- If objective is unknown, output a random value in the range v_r.

Corollary 5.0.1. *Initial Node Answer Assumption. For a given question $q \in Q$, an RNB node n_r will not know if Q knows that it is attempting to deceive it at the initial timestamp $t = 0$. This same node will not know the known answer of Q at the starting point, so it will readily provide an answer to Q for q that satisfies the known answer of n to q with respect to the objective of n for q. By doing so, this node ignores the possible consequences posed by Q when it answers q.*

Corollary 5.0.1 falls under the rationale of the abstract notion called Occam's Razor in that the initial answer from an RNB Node n_r for a question q to Q will most reveal to Q the node's objective with q. After n_r gives its initial answer to a question q, it might use other actions besides direct answers by delegating the question to the neighboring nodes to incur less resistance cost to Q while satisfying its objective. The term "contradict" is used frequently in this paper to describe nodes and their answers, so deserves a little definition for reading clarity.

Definition 5.1. Contradicting. A descriptor for an answer a to a question q by an entity A that does not equal the known answer of another entity B to that same question. An entity with a contradicting answer to q is a contradicting entity.

Contradiction can be extended outside of the boolean context. For an integer range v_r that represents a range of answers, an answer $b \in v_r$ such that $b \neq a$ contradicts $a \in v_r$, and for another answer $c \in v_r$, $|a - b| < |a - c|$ means that c contradicts a to a greater extent than b's contradiction to a. This next formula, called degree of contradiction, will be useful for calculating the degree $g \in [0, 1]$

of an answer b's contradiction given a known answer a to a question q with answer range v_r.

$$d(v_r, a, b) = \frac{|a - b|}{max(< |a - v_r.0|, |a - v_r.1| >)}.$$

5.1 The Answers of Q and an RNB Node

The Q structure is responsible for gathering answers from each node $n_i \in R$, the network, to the set of questions S_q. By Definition 4.1, Q may have a known answer to the question or it may not. In the first case, that Q does have a known answer v to question q, and asks node n_r for the answer so that n_r provides the *direct answer* v_1, the resistance $n_r.r$ will incur a cost by a function

$$\delta(n_r.r) = C(v, v_1).$$

A basic example could be

$$C(v, v_1) = -1 \times |v - v_1|.$$

In the second case in which Q does not have a known answer to q, it will use the variables X,Y, and Z from its database D_Q as a substitute for its known answer. This substitute answer is for the intents and purposes of Q its known answer because in the Respondent Network Bot environment, Q has no other route of determining the answer to q if it does not have a known answer to it before the run. In the expression $C(v, v_1)$, this value would be assigned as the first variable. An example formula for this substitute value is

$$(\sum X.col[q.i] \times Y.col[q.i])/\sum Y.col[q.i].$$

The two cases above consider an RNB node that directly answers to Q with answer v_r. In these two cases, Q will measure the contradiction c between v_r and either a known answer or a substitute answer v_s of Q using a formula proportional to the distance between v_r and v_s. This contradiction value should stay in the range $[0, 1]$, in which 1.0 signifies complete contradiction. Q updates the value at $(n_r.i, q.i)$ in the contradiction matrix X of its database D_Q with a value of $(c + X[n_r.i, q.i] * Y[n_r.i, q.i])/(Y[n_r.i, q.i] + 1)$. Then Q increments $Y[n_r.i, q.i] = Y[n_r.i, q.i] + 1$ and modifies $Z[n_r.i, q.i]$ with the revised mean answer of node n_r to question q.

Any RNB node n_r will start off in Respondent Network Bot's run with the capability to directly answer or to delegate. If n_r directly answers a question q to Q with answer v_n, then its resistance will fall by a positive value proportional to the distance between v_n and Q's known value of q or the substitute value derived from its database D_Q variables X, Y, and Z.

However, *if this same node n_r chooses to delegate to set of nodes $\{S_d\}$, then its resistance will not incur any cost.* A base approach to calculating the answer of n_r to question q based on its delegates $\{S_d\}$ is

$$d_l(n_r) = \sum_{s \in S_d} (\texttt{direct answer of s to q})/|S_d|.$$

The measured contradiction c between this answer $d_l(n_r)$ and the known or substitute answer of Q will rely on the same contradiction measure as in the case of a direct answer from n_r. But the contradiction c will be distributed equally among all involved nodes, $\{n_r\} + \{S_d\}$, for a value of $c_0 = c/|\{n_r\} + \{S_d\}|$. This entails modifying contradiction matrix X with the value c_0 and mean answer matrix Z at index $(n_r.i, q.i)$ by the identical formulae of the case in which n_r provides the direct answer (see the paragraph three paragraphs above). Additionally, each delegate node of the set $\{S_d\}$ will result in the same modification at their respective indices in matrix X. But those same indices in the mean answer matrix Z will remain identical before and after this modification procedure for this set of delegate nodes. The reason is due to the program's specification below.

- The questioner Q will regard each answer for question q from node n_r as an answer from n_r alone, regardless if the node delegates or not.

But there is still another modification to the database D_Q that needs to be performed for each of these delegate nodes. For each delegate node $n_d \in \{S_d\}$,

$$W[n_d.i, q.i] = W[n_d.i, q.i] + |\{S_d\}|.$$

The design of these modification rules to D_Q, although somewhat arbitrary, ensures that the contradiction value c is distributed by the value c_0 so that the cumulative changes due to a contradicting answer shared between n_r and any delegate node $n_d \in \{S_d\}$ is equivalent to the original contradiction value c. *Through this design, the summation of the effects of a contradicting answer over all involved nodes is magnified by a scalar proportional to the number of delegate nodes, but the effects of the contradiction measure of node n_r at its position in matrix X is divided by that same scalar plus one.*

6 The Actions of Each Class of Entity

6.1 Termination

If the entity is a questioner Q, it will "terminate" or "die" when its fuel level hits a value at or below 0. Likewise, if the entity is an answering RNB node, it will "terminate" or "die" when its resistance hits a value at below 0 .

The fuel level of Q is a finite non-increasing value, and every time Q asks a node n_r any question q, its fuel decreases by a specified function A,

$$\delta(f) = A(Q, n_r, q).$$

A simple function could be

$$\delta(f) = -1.$$

6.2 Problems for Q in Extreme Cases

The logistical design of Respondent Network Bot means that an answering RNB node n_r for question q under feasible conditions for delegation can simply delegate the question that it would otherwise directly answer with contradiction. The delegation to suitable delegate nodes will still yield an answer that contradicts the answer (known or substitute) of the questioner Q to a similar extent, but n_r will have a resistance identical to its resistance before answering Q through delegation. The lemma below provides a principle on this matter.

Lemma 6.1. *For a particular question q, if all RNB nodes in a network uniformly contradict so that all their answers are equally different to that of the known or substitute answer of Q, then Q will not be able to differentiate between any RNB node in the network based on the contradiction scores it has gathered in its database* D_Q.

The effect of this logistic is that Q will expend a disproportionate amount of fuel asking n_r for an answer to q only for n_r to delegate at no cost to its resistance. However, Q is equipped with another tool called the F2-fix that remedies such instances of contradicting nodes

Definition 6.1. F2-Fix. An action that can be taken by a questioner Q onto an RNB node n_d in response to any set of RNB nodes $\{n_r\}$ that continually delegates to n_d for a particular subset of questions $q' \subseteq Q.V_q$. This action "disables" the capability of n_d to be a delegate to any node in the network. When an node n_{r_x} decides to perform a simulatory delegation to determine its answers, the answer of a delegate n_d will not be considered as a value in the "collective" answer produced by n_{r_x} for question q through delegation.

There is a cost to the fuel of Q if Q decides to perform a F2-fix.

Definition 6.2. F2-Fix Fuel Cost. The fuel cost for Q to perform an F2-fix on n_d is equal to the cumulative contradiction of that node n_d to Q. Suppose for a vectorized set of questions $S_q =< q_0, \ldots, q_{n-1} >$, Q has the answer vector (either known and substitute) A_v. Then for RNDB database V_q of Q, first calculate the predicted contradicting pre-answer vector A_w of n_d:

$$A_w = (X \times W).row(n_d.i)/W.row(n_d.i).$$

The inverse function of the degree of contradiction formula d in Section 5.1 is $d^{-1}(v_r, a, g)$, which outputs a value $b \in v_r$ that contradicts a by degree $g \in [0, 1]$. Then the predicted answer vector for n_d calculated by Q is A_x such that for an element at index i is

$$d^{-1}(S_q[i].v_r, A_v[i], A_w[i]).$$

Remark. The above formula used to calculate the fuel cost is barebones and intended for bot use.

When a questioner Q decides to F2-fix a node n_d, another node n_r that determines n_d to be a suitable delegate will calculate a "collective" answer to q that contradicts the objective answer of n_r to a greater extent. The greater contradiction is due to n_d staying "silent" because it has been F2-fixed by Q. This will result in n_r recording a lesser satisfaction value for the delegate n_d with question q in its $RNDB$ database, which in turn eventually leads to n_r excluding n_d as a suitable delegate for the question q.

6.3 The Nodes That Answer

For every question q that a node n_r is asked, n_r can either delegate or directly answer. If n_r chooses to delegate, the "collective" answer a_c it derives from the answers of those delegate nodes satisfies the following expression:

$$n_r.F[q.i][-1] > |a_c - \texttt{Q's substitute or known answer}|.$$

The above expression translates to the degree of contradiction predicted by n_r through delegate nodes to produce a "collective" answer a_c that is less than the recorded feedback value of Q to the question q. Recall the pertinent Corollary 5.0.1 on how n_r will initially answer to Q by a direct answer. To satisfy that corollary, every node in the network R will have an initial feedback map F with each question q having the map output $V =< 0 >$, a 1-vector with 0 as its only element. The feedback map F will be modified by its RNB node owner n_r if n_r decides to directly answer with value a for question q, so

$$F[q.i] = F[q.i].push(|\texttt{substitute or known answer of Q}-\texttt{objective answer of } n_r|).$$

So a direct answer by n_r will add a value to its feedback map F equal to the absolute value of the non-positive value that is the change to its resistance. And if n_r does not find any suitable delegates by its satisfaction map S_m in its database RNDB, it clearly cannot delegate but have to directly answer.

6.4 The Questioner Q

Recall the general description of the questioner Q's objective in Section 4. There are two actions that Q can take every time the network R prompts it.

- A. Determine the (node n_r,question q) pair wih the "highest priority". Ask the node that question.
- B. Determine a node n_d that is the most contradicting delegate node in the network, and perform an $F2-fix$ on it.

"Highest priority" should take into consideration if a node n_r has ever answered a question q. If it has not, then Q should prioritize (n_r, q). A general guideline by this prioritization is that the answers of all nodes are equally important to Q.

Remark. This two-step procedure is described in a generic manner.

The F1-Fix is a tool used by Q and defined below.

Definition 6.3. F1-Fix. An action that questioner Q performs on an RNB node n_r when its resistance r satisfies $r \leq 0$. After Q performs an F1-Fix on n_r, n_r will output answers to a question q based on the question's assigned *converging number generator.* If there are n questions of pertinence in Respondent Network Bot, then this node n_r will have n converging number generators, each to serve as the source of answers to its assigned question.

An implementation of a converging number generator is found in the file "cng.txt" of the `RNB` computer program. Below is my personal definition for a converging number generator.

Definition 6.4. Converging number generator A structure that can output an infinitely-sized sequence of numbers in a field F such that after a certain index i in the output sequence S, the structure will output a machine-recognizable numerical pattern of numbers which is usually a subset of values in the subsequence $S[0..i]$.

The questioner Q will assign n converging number generators for the n pertinent questions to an RNB node that is to be F1-fixed. These number generators are designed to output answers that are the least contradicting to those answers (known or substitute) by Q.

7 A Modifiable Feedback Loop

This section gathers the definitional specifics of the previous sections to present the feedback loop in Respondent Network Bot.

- Q chooses a (`node` n_r, `question q`) pair to take action on.
- Q might perform an F2-fix on an RNB node in the network.
- RNB node n_r responds to Q by choosing to either delegate the question or directly answer.

The Respondent Network Bot will run until one of the two conditions:

- 1. The fuel of Q is no longer positive.
- 2. All RNB nodes are "terminated" and F1-fixed.

Remark. The feedback loop is the unit of activity in the event sequence of Respondent Network Bot. It is machine-learnable by both entity classes Q and the RNB node. The necessary variables for each of theses classes to learn better decisions are present in this program, Respondent Network Bot, a bot that uses pre-defined and immutable formulae to learn. Respondent Network Bot can be modifed to allow for different cost values and be extended with machine-learning capabilities so that any class of entity will fare better under similar Respondent Network Bot circumstances in future running iterations.