Temple AGI Team Technical Report

A NARS-Based Diagnostic Model

Pei Wang, Bill Power, Xiang Li
Department of Computer and Information Sciences, Temple University
{pei.wang, bill.power, xiang.li}@temple.edu

1. Introduction

This report summarizes the current status of a long-term effort; applying the techniques developed in the NARS project [1,2] to practical applications [3,4].

In this project, "diagnosis" roughly corresponds to the process of deriving unknown properties of an instance, or explaining its abnormal features, according to its known properties and features, as well as relevant background knowledge.

The proposed model, GDM (general diagnostic model), has the following characteristics that distinguish it from other approaches:

- It is designed as a reasoning system, in which every step follows the Non-Axiomatic Logic. Therefor, each step can be explained and justified. The quality of each conclusion is also indicated quantitatively.
- It does not require large amount of data like most machine learning techniques does, and can work properly with any amount of domain data, though cannot promise perfect solutions.
- It uniformly represents multiple types of uncertainty; randomness in unpredictable outcomes, fuzziness in ill-defined categories, ignorance due to incomplete knowledge and changing environment, inconsistency among distinct information sources, etc.
- It includes multiple rules of reasoning and learning; deduction, induction, abduction, revision, choice, comparison, analogy, and compound composition. In this way it unifies reasoning and learning, enabling it to learn from small data and incrementally revise its knowledge.
- When working in real time with restricted memory space, it can prioritize its tasks to achieve overall efficiency.
- The model is designed as domain-independent, and can be applied in many fields.

GDM is based on NARS, so also obeys the Assumption of Insufficient Knowledge and Resources (AIKR), and it uses the following components in NARS:

- Its representation language and inference rules are compatible with Non-Axiomatic Logic (NAL) layers 1-6.
- Its memory structure and control mechanism are designed according to the same principles as in NARS, especially the assumption of insufficient knowledge and resources.

GDM differs from NARS in the following major aspects:

- GDM is not an AGI (artificial general intelligence) in the spirit of NARS. It only accepts a predetermined set of tasks for which task-specific algorithms are followed to process them.

Compared to NARS, in which dynamic resource allocation is use to decide the processing path for a task.

- GDM is a question-answering system and does not directly interact with its environment through a sensorimotor interface as NARS. As such, it does not implement NAL layers 7-9.
- For each application domain, customization and configuration are necessary for GDM to solve domain problems, while NARS only requires proper education via the user-interface.

Due to these differences, GDM will not use the OpenNARS code, and will be implemented separately.

2. Data Structures

GDM is built around three major data structures: a case base, a knowledge base, and a task queue.

2.1 Case Base

GDM is built around three major data structures: a case base, a knowledge base, and a task queue.

Each application of GDM will be built around a set of instances of a certain type. Each of these instances will be referred to as a "case" in the following descriptions. Intuitively, it is an object, entity, or event in the application domain. In GDM, each case is given a unique ID. In the case base, all cases are described by a set of attributes. Therefore, conceptually speaking, the case base can be represented by the following table:

	A1	A2	A3	...	Am
C1	V11	V12	V13		V1m
C2	V21	V22	V32		V2m
C3	V31	V32	V33		V3m
...					
Cn	Vn1	Vn2	Vn3		Vnm

In the table, C1 ... Cn are instances, A1 ... Am are attributes, and Vij is the value of Ci at Aj. Beside directly using the tables in a relational database, it is also possible to use an object-oriented database, or even a spreadsheet, to implement this case base, though the table form will be used in this design document.

In NARS, the knowledge "Ci has value Vij on attribute Aj" is interpreted in NAL as the inheritance statement "(*, {Ci}, {Vij}) → [Aj]", which can be simplified as "(Ci * Vij) → Aj" with the assumption that Ci and Vij are extensional sets with a sole instance, and Aj is an intensional set with a sole property. The statement can also be represented as statement "Ci → (Aj / Vij)" with the assumption that the predicate is an alternative format of (/, Aj, _, Vij).

For example, "The type of chest pain of John is considered as angina pectoris" can be represented as

(*, {John}, {angina-pectoris}) → [pain]

The truth-value of a statement in NARS is a pair of values <f, c>, where the first component is "frequency", defined as the proportion of positive evidence among all available evidence, while the

second is “confidence”, defined as the proportion of the current evidence among the total evidence after a constant amount of new evidence is obtained. In this quantification “evidence” is interpreted in a broader sense than in statistics, as it includes not only the direct and extensional evidence coming from the instances of the concepts in the statement, but also the intensional evidence coming from the properties of the concepts, and the indirect evidence derived through various types of inference.

If the values of attribute Aj are numerical, or symbolic and form a total order (like “very low, low, medium low, medium, medium high, high, very high”), they can be treated either as above, or simplified by treating the knowledge as “Ci → Aj” and turn Vij into the frequency value corresponding to the quantile (relative rank) of Vij among all values in the dataset, or an approximation of it. For the other values, if there are associated statistical information on “(Ci * Vij) → Aj”, the truth-value can be determined according to the definition of <f, c>, otherwise a default <1, cj> is used, where cj is the reliability of the data source, represented as a certain amount of evidence.

Domain-specific preprocessing may be necessary to convert the raw data into the above format. Missing values correspond to truth-value <0.5, 0>, which is also used for attributes undefined on a case. If necessary, cases of different types can be mixed in the case base, and consequently each of them may only have values at some attributes. For each application, the primary attributes should be decided at the design time, though secondary attributes can be introduced at the run time.

2.2 Knowledge base

The knowledge base is a conceptual network, where each node may be an attribute (the most common situation), a case (usually only the important ones), or a compound of attributes or cases composed using the term connectors in NARS (such as “Ai & Aj”). There are only four types of link, corresponding to the copulas in NARS (inheritance, similarity, implication, and equivalence). Each link has a truth-value <f, c>, like the statements in the case base.

The knowledge base can be implemented either as a directed and weighted graph with labeled edges, or as a collection of terms, each with a few tables containing links of each type (i.e., extension and intension). The content of the knowledge base can be either imported from an external source (such as ConceptNet) or some machine learning algorithms, or derived by GDM itself.

2.3 Task Queue

The task queue contains all the active tasks under processing, either input or derived. Each task has a priority value indicating its relative rank in resource allocation. Depending on the nature of the application, this queue can be implemented either as a priority queue, or a “bag” of NARS (i.e., a probabilistic priority queue). For each query task that demands a return value, the best-so-far answer is maintained.

3. Inference Steps

As a reasoning system, GDM has an inference step as the basic unit of processing, and each step is based on an inference rule of NAL, and generates one or more conclusions according to the combination of the two premises, and also calculates the truth-value of the conclusion according to those of the premises.

The major types of inference include

- **[DEDUCTION]** For example, from "Ci → Aj" (in case base) and "Aj → Ak" (in knowledge base) to derive "Ci → Ak" (to be added to case base). This is like statistical inference, where the first premise is about the membership of the case in a category, and the second about the correlation between two categories. The truth-value function of this rule extends the transitivity of the inheritance copula (→) from binary to <f, c>.
- **[INDUCTION]** For example, from "Ci → Aj" and "Ci → Ak" (both in case base) to derive "Aj → Ak" (to be added to knowledge base). This is like statistical learning, where the case provides a piece of evidence to the substitutability between the two categories. Compared to deduction, this inference produces conclusions with low confidence values, as each is at most supported by one case only.
- **[ABDUCTION]** For example, from "Ci → Aj" (in case base) and "Ak → Aj" (in knowledge base) to derive "Ci → Ak" (to be added to case base). This is like category membership evaluation, where an attribute of a case is compared to the attribute of a category, so as to assess the membership of the case to the category. Like induction, abduction also only produces low-confidence conclusions.
- **[COMPARISON]** For example, from "Ci → Aj" and "Ci → Ak" to derive "Aj ↔ Ak", or from "Ci → Aj" and "Ck → Aj" to derive "Ci ↔ Ck". This rule is like induction and abduction, except that it produces similarity statements.
- **[ANALOGY]** For example, from "Ci → Aj" and "Aj ↔ Ak" to derive "Ci → Ak", or from "Ci → Aj" and "Ci ↔ Ck" to derive "Ck → Aj". This rule is like deduction, except that it uses a similarity premise.
- **[REVISION]** If the same statement can be derived from different bodies of evidence, the conclusions are merged into a summary, whole truth-value is based on the accumulated evidence. This is how non-deductive conclusions gradually obtain high confidence, as will as a simple way (though not the only way) to resolve inconsistency among data.
- **[CHOICE]** If there are competing conclusions, such as both "(Ci * V1) → Aj" and "(Ci * V2) → Aj" are derived, but V1 and V2 are incompatible, then the one with a higher expectation value is chosen, as expectation combines frequency and confidence into a single estimation of future frequency.
- **[COMPOSITION]** For example, from "Ci → Aj" and "Ci → Ak" to derive "Ci → (Aj & Ak)", where (Aj & Ak) is a compound with internal structure. If there are many cases that have both attributes, the compound will be kept and treated as a single attribute, so as to recognize stable patterns in data, as well as to improve the efficiency of representation and processing. Composition is carried out incrementally and recursively.

When the model is implemented, each inference rule is coded as a standalone function or method, with a uniform signature (i.e., input/output arguments). Some rules may have multiple forms handled as overloading. All the truth-value functions are already specified in NAL.

4. Inference Procedures

GDM is built to accept a set of predetermined tasks, and each task is processed by following an algorithm, consisting of a sequence of inference steps. The algorithm usually updates its solution in an

incremental and iterative manner and is interruptible and always provides the best-so-far solution, like an anytime algorithm.

The most typical diagnostic task is to predict the value Vij for a given case Ci and an attribute Aj. As explained previously, such a conclusion can be derived in several ways in GDM, so such a task usually fork several subtasks, each following a different path of inference, and then the conclusions are merged via revision. The system chooses the best answer by choice. In this way, GDM combines the traditional "rule-based" and "case-based" approaches, plus some others, such as probabilistic and fuzzy inference. Different subtasks will be given different priority values, based on the type of inference involved, the availability of relevant data, and other domain-specific considerations.

Beside this **predictive task**, there are also other types of task GDM can carry out.

A **detective task** compares a given value Vij with an anticipated value Vij' that is derived from the other knowledge, including statistics of the other cases and predictions according to the other attributes of the same case. If the two are different enough, Vij will be reported as abnormal.

An **explanatory task** attempts to drive a given value Vij from a predetermined set of attributes, to answer a "why" question. This task can be carried out alone, or as a follow-up step of a predictive or detective task.

There are also **learning tasks** running in the background, which does not need explicit user request, but triggered by the new cases and knowledge, and serves to self-organize the system's knowledge. For example, the system will spontaneously compare attributes and cases to find correlations among them, to gradually enrich its knowledge base, as well as to provide the anticipated values for its case base as defaults.

For each application, the above algorithms need to be specified by the designer, though the model should include several templates to be customized. Some parameters of an algorithm may be adjustable by the system at the run time, according to the feedback collected when tasks are carried out, so overall GDM is adaptive and self-organizing, without necessarily converges to a stable input-output mapping.

5. Interface

GDM can directly interact with the users using a language or format that is suitable in the domain, with domain-specific programs converting the language into knowledge and tasks used within the system.

Similarly, GDM can be integrated into a larger system or exchange data, tasks, and answers with other programs at run time, if there are adaptors that can translate between the internal and external representations.

GDM works in real time, in the sense that new data and tasks can arrive at any moment, and the system's answers and solutions are available after a short time and then being constantly updated at the user interface.

GDM is capable of active learning, meaning that certain derived tasks will be sent to the user or converted to queries to external knowledge sources.

GDM is fully automatic, though (if it is necessary in a domain) user intervention is allowed, such as to adjust the priority values in the task queue or to directly edit the content of the knowledge base when the system is running.

6. Implementation

The initial implementation effort for GDM consists of the following major activities:

1) Though GDM should be applicable in many domains, a few representative domains will be selected for the initial experiments. It may include medical diagnosis, personalized tutoring and recommendation, troubleshooting in mechanical or electrical devices, etc.
2) To identify and collect data to be processed. Given its different assumptions of the working environment and expectation, it is improper to test GDM with the datasets currently used in traditional machine learning projects. At least it will be a combination of multiple data sources with different format and properties.
3) Physical design of the implementation according to this conceptual design of GDM, which should realize the core functionality, while leaving the flexibility for domain-specific customization and configuration.
4) One important aspect of the implementation will be its interface with other software and hardware to preprocess various types of data, so as to convert the data into the form required by the model.
5) In each selected testing domain, test and tune the implementation to find the proper configuration, parameters, and training strategy.
6) Analyze the results, compare it with the other techniques, and accurately identify the suitable application situations of GDM.

At the end of the initial implementation, the results should include a working prototype, a few tested use-cases, and several documents.

References

[1] Pei Wang, Rigid Flexibility: The Logic of Intelligence Springer, Dordrecht, 2006, ISBN: 9781402050442

[2] Pei Wang, Non-Axiomatic Logic: A Model of Intelligent Reasoning, World Scientific Publishing, 2013, ISBN: 9789814440271

[3] Pei Wang and Seemal Awan, Reasoning in non-axiomatic logic: a case study in medical diagnosis, Proceedings of AGI-11, Mountain View, California, August 2011

[4] Bill Power, Xiang Li, and Pei Wang, Generalized Diagnostics with the Non-Axiomatic Reasoning System (NARS), submitted to AGI-19