GE ASKE TA1 ANSWER M11 Report Part 1: Progress

Andrew Crapo, Varish Mulwad, Nurali Virani, Narendra Joshi
GE Research
October 18, 2019

This work is supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00111990006. The latest version of this report is available at https://github.com/GEGlobalResearch/DARPA-ASKE-TA1/blob/master/StatusReports/Phase2/M11/M11_ANSWER_Report_Part1.pdf.

Contents

Introduction

This report provides an overview of the work done by GE Research in the DARPA ASKE TA1 ANSWER project from the beginning of month ten to the publication date of this report. The report is divided into five sections, excluding the Introduction and Conclusions and Next Steps sections. The first four cover the main four thrusts of our research, namely, 1) extraction of scientific concepts from code, 2) extraction of scientific concepts from text, 3) capturing scientific knowledge in a K-CHAIN computational graph, and 4) developing a collaborative environment that facilitates keeping the human in the loop. The fifth section addresses the relevance of the research to GE and to DARPA. In some cases, greater detail is provided in appendices or in linked documents.

Extraction of Scientific-Concepts from Code

As described in detail in the M9 report, it is our considered position that scientific knowledge in the form of computational models will only be usefully applied by an artificial intelligence in support of human scientific endeavor if the computational knowledge is well-connected to the semantics of the domain. We have proposed that, in the case of equations capturing relationships either in the form of physics-based models or in the form of data-driven models, the requisite semantic information can be captured as augmented types. Augmented types identify the meaning, in domain terms, of the inputs and outputs of a model as well as how the inputs and outputs are related to each other, how they are constrained, and what assumptions must be met to legitimately apply the model.

We have made good progress in the syntax of Augmented Types and in their implementation. The meta-model, in the SadlImplicitModel, is now stable and the code to utilize Augmented Types is largely in-place and well-exercised. The challenge ahead is that of extracting the information to populate Augmented Types from code and associated text. At least in our selected source of code and text, the NASA Glenn Hypersonic Web site, there are few indications in the Applets' Java code of augmented type information. Even a human knowledgeable in the domain must take considerable time and care to figure out what each input variable and the method return value represent in the domain. The difficulty of the task does not diminish its critical importance to the end goal of allowing an artificial intelligence to reliably use the extracted knowledge to inform and support human scientific endeavor.

The Dialog mixed-initiative interaction mechanism provides the means for a human to review an extracted equation and add necessary augmented type information not supplied by extraction before persisting the equation to the curated scientific knowledge and to the computational graph. Obviously, this is not ideal as it would be preferable to extract all the relations of inputs and outputs to each other and to domain concepts, and any applicable constraints and assumptions, and show the fully qualified equation to the user for approval.

In the NASA Hypersonic case, the text is a potentially valuable source of augmented type information as the text often says what domain concept a variable represents. As we have previously discussed, that is why we are working to integrate extraction from code and text

using “locality search”. Text directly documenting the code, or describing the concepts and relationships captured in the code, is of the greatest potential value, while more general texts are much less likely to provide semantic information about a specific equation.

Extraction of Scientific Concepts and Equations from Text

We have continued our work on improving the extraction of an equation’s context found in the surrounding the text. We are in the process of setting up and evaluating the University of Arizona’s AutoMATES Text Reading pipeline within the GE Research infrastructure. Since our M9 report, the work on extraction of scientific concepts and equations from text (aka text to triples) has focused on the following:

- Updating the main text to triples API to implement the notion of “locality” and facilitate better integration with ANSWER’s curation and dialog managers
- Improving the “locality search” API that allows users to search by variable names appearing in code or in equations extracted from text
- Introduction of an additional “locality search” API that allows users to parse comments appearing in code with a specific focus on units
- Improving the installation process

One of the interesting challenges that we have encountered on the NASA Web site is what to do with equations that have an expression on the left-hand side. The Isentrop.html page provides a good illustration of this kind of equation as a number of equations here have a ratio on the left-hand side. We are investigating the best way to convert these equations into Python compatible with K-CHAIN and DBN.

Main Text to Triples API Update

The main text to triples API was updated to implement the notion of “locality” and facilitate better integration with ANSWER’ user-interface. While we discussed the use of locality of during Phase 1, it was implemented and integrated into text to triples as part of our M11 release. The main service now allows the users to provide a local semantic graph along with a blob of text where the extracted concepts and equations need to be temporarily stored. Once the extraction process is complete, the users are notified with the count of extracted concepts and equations. Any future queries to support the joint extraction of scientific models from code and text are performed against this local graph. The two locality search APIs below illustrate how this could be achieved. The main API is now complimented by two supporting APIs – one for the ANSWER system to retrieve all the relevant RDF triples serialized in N3 for curation purposes and one to discard the extracted triples if the users wishes to. As part of our M11 release, the dependence of text to triples on an external triple store was also removed and was replaced with an in memory semantic graph. This aided us to improve the installation process for others to use our text to triples services.

Locality Search – Variable Name Search

The implementation of locality as part of the main service allows us to integrate the variable name search API as part of the text to triples services. While we demonstrated this capability as part of our Phase 1 release, with this integration, users can perform a local variable name search

on any text processed by the main text to triples service, by simply providing the local graph URI in which the extracted concepts and equations were added. The variable name search API leverages SPARQL queries to perform matching at two levels – first it performs an exact match between the variable name provided by the user and the ones that appear in the local graph. If that produces no results, it performs a fuzzy match between the two. The API returns the equation in which the variable name appears, its associated scientific concept and the Wikidata grounding of the concept. We are in the process of integrating this as part of ANSWER's user-interface.

Locality Search – Code Comment Unit's Search

We also introduce a new locality search API as part of our M11 release. This API focuses on extracting relevant information from code comments to guide the extraction of scientific models from code. More specifically, this API detects references to units that appear in code comments. The units are extracted using an UIMA ConceptMapper based ontology-driven lookup. The API currently uses the "Units of Measurement" ontology as part of its lookup process. This ontology lists several units (and their synonyms) along with the scientific concepts measured by these units. This ontology can be swapped out for a different one via the text to triple configuration file. If units are detected, it identifies the associated the scientific concept measured with the unit and then searches for its reference in a user-specified local semantic graph. If the concept has been extracted in the local graph, the API returns the equations in which it appears, its associated variable name and the Wikidata grounding.

Easy Install for Text to Triples

We have vastly improved the ease of installation and deployment of the text to triples services as part of our M11 release. Instructions can be found in our GitHub repo here: https://github.com/GEGlobalResearch/DARPA-ASKE-TA1/tree/master/ModelsFromText. The text to triples service depends on loading and search through Wikidata terms in Elasticsearch. This search service is now provided via a Docker container, with the users no longer needing to download and install Elasticsearch and setup a Wikidata index by loading in the terms. The docker process automatically downloads Elasticsearch, loads in the terms, creates the index and starts the relevant service. Other dependencies such as pre-trained models for extraction of scientific concepts and equations as well as the concept mapper service are provided via the "releases" tab on our GitHub repo. We provide additional scripts (`getTextExtractionModels.sh` and `getAndRunConceptMapperService.sh`) to download and integrate these automatically. A `startServicesModelsFromText.sh` script launces all the relevant services. A jupyter notebook provides high level overview of the services discussed above as well as acts as live documentation to use and run them. This notebook is available in our GitHub repo under `DARPA-ASKE-TA1/ModelsFromText/api-example-code/text_to_triples_client.ipynb`.

Computational Modeling with K-CHAIN

Overview

We have created computational modeling capability that creates computational graph in TensorFlow as an execution layer of the knowledge captured in the semantic layer. This modeling

approach is “anytime”, which means that at any step of curation, we have access to executable instantiation of the currently curated knowledge. As the agent curates more knowledge, the computational model evolves with it. The capabilities of Knowledge-consistent hybrid AI networks (K-CHAIN) includes:

- Building models:
    - with physics models with simple equations provided as strings
    - with physics models captured in TF-compatible python code, where the code itself was derived from extracted text or code
    - with experimental data
    - with default values of certain inputs
- Evaluating models during inference:
    - agnostic to whether model was built as data-driven or physics-based
    - with ability to use default values (if not provided at inference time) and inform user about all default values used in the computation
    - with ability to inform user that key variables are missing to conduct inference
- Appending existing models with new model fragments, where:
    - new model fragment uses outputs from existing model
    - new model fragment is input into existing model
    - new model fragment needs gradient information of nodes in existing model

These capabilities are systematically demonstrated in a Jupyter Notebook (https://github.com/GEGlobalResearch/DARPA-ASKE-TA1/blob/master/Notebooks/Demo%20for%20KCHAIN%20Services.ipynb) and more capabilities which are not completely integrated in ANSWER agent is available here: https://github.com/GEGlobalResearch/DARPA-ASKE-TA1/blob/development_kchain/Notebooks/Demo%20for%20KCHAIN%20Services-Beta%20Release.ipynb. These Notebooks also serve as usage documentation of K-CHAIN services.

Key progress

The key progress in Phase 2 has been:

- to integrate build and evaluate services with the ANSWER agent, so that models can directly be built and queried from Dialog interface, and
- to create a new “append” service, that allows to append computational models to existing models graphs.

The outcomes of integration effort are shown in our demo. The video link to the demo is shown here: https://github.com/GEGlobalResearch/DARPA-ASKE-TA1/raw/master/StatusReports/Phase2/M11/GE-ASKE-TA1-October2019-Demo_Part1.mp4
In this first part of the demo the integrated capabilities are demonstrated where K-CHAIN services are used in background. In later part of the video, we demonstrate some the capabilities which are available in the Jupyter Notebooks. The second key progress item was to successfully implement “append” service where an existing computational graph can evolve as more computational models are extracted from code or text (see Figure 1 below for an example). The

challenges associated with this step included that TensorFlow naively would just create a new graph node even if the same variable name is used in new model fragment. Thus, we implemented a node dictionary to that can persist URIs of each graph node and reuse existing node when name and other contexts would match with nodes in new model fragment. This approach did not work in Option 2 of append (See Figure 2 below), where new model output was an input node of existing model. We detect such instances and rebuild graph to avoid such conflicts using a list data structure for model fragments that have been appended. This data structure can also hold provenance information of the model fragment. This issue too some effort to resolve, but it is now available on development_kchain branch of the repository and it will be integrated with Dialog interface next.

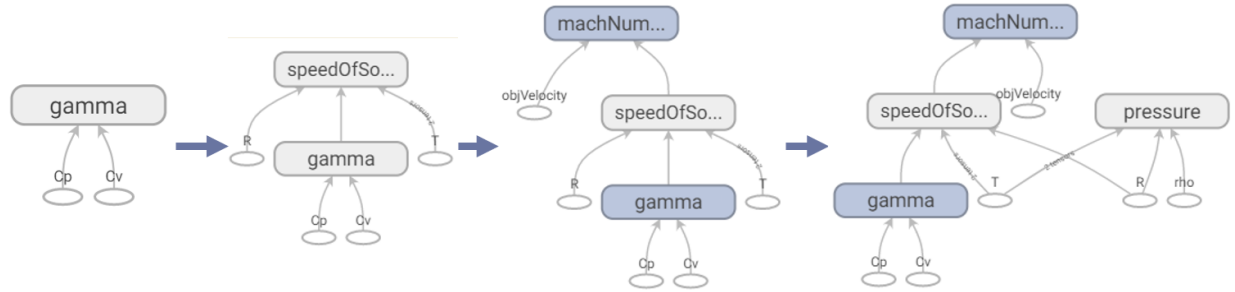

Figure 1: Example of computational graph evolution

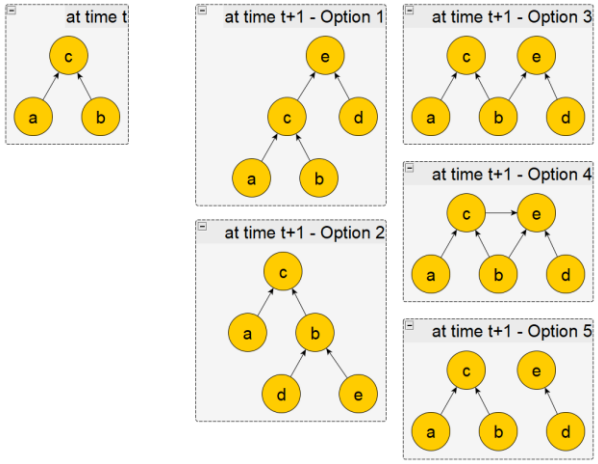

Figure 2: Non-exhaustive examples of types of computational graph evolution

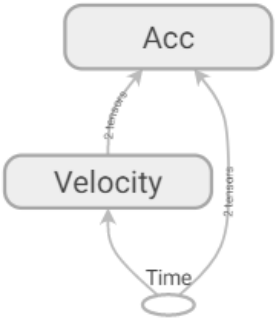

Figure 3: Example of a simple gradient operation appended to a computational graph a = dv/dt, where v as a function of t is known.

Next steps

- Minor necessary steps:
    a. Maintaining default values after append (of type Option 2): In specific type of append operation, we realized that the default values associated with existing nodes might be erased due to rebuilding of computational graph. This issue will be fixed next.
    b. Overwriting models with new equations: If a new model equation is provided for an existing subgraph either due to incorrect extraction or modification suggested by user,

then we would want to overwrite that existing subgraph. This capability is being implemented currently.

c. Data-driven modeling in append: Although data-driven modeling is stable and works on master branch, the new edits in development branch with "append" service has created a few challenges of building and appending these models. Once this issue is resolved, we can append and train these subgraphs which are data-driven models. Moreover, as the physics relation is discovered or extracted, it can overwrite the existing data-driven model after performance validation.

- Big steps:
    a. KCHAIN integration with TA2: Currently TA2 framework constructs a json specification of DBN model for evaluations, if we can similarly construct json specifications for K-CHAIN, we can interchangeably use DBN and K-CHAIN for simple or composite models.
    b. Using and solving ODEs: Currently, we demonstrated that ODEs can be added to computational graph and be used to compute those derivatives or gradients using automatic differentiation (See Figure 3 above given v(t) find acceleration). However, we have not yet implemented capability to solve ODEs and use them to answer queries that need to perform integration (i.e. given acceleration(t) find velocity and position). This work will be taken up and demonstrated using ODE models from Hypersonics domain.
    c. Increase adoption of units: The units are being expected to be managed and captured by semantic layer. However, when we perform graph append operation, the computational model must at least have consistent units. We plan to check consistency of scientific units during graph building and append process and then, automatically append suitable unit conversion subgraph if units are not consistent. This work has been accomplished in semantic layer by TA2 and insights will be use to solve this in TA1 computational layer.
    d. Context-dependency of models: Capturing context under which a model is to be used is being captured under semantic constraints of model. However, in computational layer, we have not yet answered that when a new model with similar outputs and (potentially) similar inputs compared to an existing subgraph is obtained, then should we persist the older model with different context or inform the user before overwriting? Append operation where an existing output node is output of a new subgraph has not been addressed yet, we will need user engagement to understand if model needs to be overwritten or persisted with a separate context and how to represent that new context computationally (as a boolean flag, perhaps).

Unanswered issues:

1. Substitution: Although we might not have seen exact usage and instance of an equation as humans, we can rearrange equation terms and substitute them in other equations to answer queries that need composite models. We can handle composite models in K-CHAIN, but, in the model composition process equation inputs and outputs are specified during build and append process and suitable subgraph can be identified during inference. However, state equations, such as ideal gas law PV = nRT, do not have clear demarcation of inputs and outputs and can be used based on what values are available to substitute for other unknowns. This capability is still not possible and needs more thought.
2. Representing cyclic set of equations: On a related note, certain set of equations can create a cyclic graph which cannot be handled in various computational modeling frameworks.

This problem is more serious during curation as we might encounter various versions of a relation (F = ma and a = F/m), which together can introduce a cycle. The ambiguity can be resolved during inference based on which inputs are available, but this representation is currently an unsolved issue.

Keeping the Human in the Loop through Collaboration

The Phase 1 ANSWER Dialog collaboration window was a fragile, proof-of-concept implementation. In Phase 2 we have significantly improved the Dialog interface both in terms of functionality and especially in terms of robustness. Conversations are now represented in a more wholistic manner with questions and answers identified with each other so that a question, once answered, is not answered again, even if the dialog content is saved and reloaded into a Dialog window. More of the content of the window is expressed in controlled-English rather than quoted text, allowing more complete semantic highlighting and linking of concepts to their definitions and to other uses.

The Dialog vocabulary has been expanded to support saving of extracted models as curated knowledge and computational graph content. The ultimate objective is to allow a user to connect either to GE TA2's Dynamic Belief Network (DBN) computational graph or to the TA1 K-CHAIN computational graph (or to other computational graphs in the future). Preferences have been implemented to allow query answering to use either computational mechanism, or to use them both in parallel, but additional work remains to achieve complete functionality. Lacking query-answering capability using K-CHAIN, and also as a useful testing mechanism for curated equations, the "evaluate" command has been added to the vocabulary, allowing a user to invoke a particular model in the computational graph, provided inputs, and get back the computed results.

Technology Relevance to GE and Value to DARPA

The NASA Hypersonic Web site contains all the equations necessary to support a thorough characterization of the design space for hypersonic flight. However, considerable expertise is needed to correctly apply these equations to any actual design problem. We have exercised our code and text extraction capabilities to see how completely we can find all the relevant equations and add them to a computational graph. (As noted above, the presence of ratios rather than a single output variable on the left-hand side of the equation in text poses an issue that is under investigation.) The exercise of extraction of the equations, the written description and creation of the K-CHAIN or DBN computational graph is equivalent to a human's assimilation of new knowledge from literature and its application to solve a useful problem. This process is also similar to the learning process for a new engineer reading existing literature to come up to speed in a new technical area. The value of this machine learning exercise lies in the application of the process to reading code and text and subsequently providing a designer with all prior experiences, (including setting of the requirements of a design problem, performance calculations, manufacturing data, design trade-offs and field durability and utilization), in the use of features of a design and in aiding the design with appropriate design equations and solutions. The application of a tool enabled by ASKE/ANSWER capabilities has

the potential of a) delivering an electronic assistant to the designer, one which has prior knowledge of how each feature is applied and used, resulting in knowledge retention and transfer, b) reducing design errors which otherwise would have caused one or more design iterations to correct ,and c) as this methodology is applied across complex systems, reduction in time and cost in deployment of new capabilities to the warfighter.

Metrics

Extraction Metrics

Extraction metrics should measure what was extracted from a given source and compare that with what should have been extracted as ground truth. What should have been extracted is not well-defined. For the extraction of knowledge from scientific code, some portions of the code may be considered irrelevant, and we do not want to extract this. Of what is potentially relevant, some code segments will be more easily extracted than others. Therefore, we must set a target level of difficulty, assess what elements of the code fall within this level of extraction difficulty, and compare what is extracted with this set of expected extractions.

Another relevant metric is the quality of the extraction. For the extraction of equations from code, we can quite deterministically determine whether the extracted equation is correct by comparing its computed output with what would have been computed by the original code given the same inputs.

Extraction of code from text is similar but not identical in terms of desirable metrics. For one thing, equations in text are not generally computable so we cannot exercise the original equation and the extracted equation to make sure they give the same outputs for the same inputs. Evaluation of the quality of extraction in this case may require a human to compare the extracted equation with the original equation in the text and grade it as the same (pass) or different (fail).

It will sometimes be the case that the equation in text may not be complete. For example, it might have an input which is not defined, or it may reference another equation (call a method) which is not defined in the text. In this case the equation should be excluded from the set of anticipated extractions. However, this is again most likely a labor-intensive human activity to determine the set of expected extractions.

To be more specific, we plan to identify a set of approximately six pages from the NASA Glenn Hypersonics Web site that have Java applets associated with them. For each of the Java applets and the associated text page, we will analyze a set of desired extractions. These will primarily be equations in the text and methods in the Java code, but can also include constants, variable semantic types (e.g., temperature, mass, etc.), and applicable constraints (e.g., this equation applies only for altitudes between 36,152 ft and 82,345 ft).

We will then run our extraction processes on code and text and compare our extractions with the expected ones by category. Categories are, in approximate order of importance:

1. Equations and methods
2. Constants
3. Scientific concepts associated with equation or method variables
4. Units for equation/method inputs and outputs
5. Constraints or assumptions which must be met to use the equation or method
6. Scientific concepts relevant to the domain but missing from the domain model.

We will also measure how many of the equations that we do successfully extract are added to the computational graph, can be executed, and provide correct answer.

Curation Metrics

One aspect of curation metrics measures how well the curated knowledge may be used to compose models to answer questions. This is primarily an ASKE TA2 objective, so we will not include these in our TA1 metrics.

More relevant to TA1 is the measure of the curation process itself. Metrics in this category include the following.

1. How well can the user collaboratively doing curation of extracted knowledge understand the extracted equations and related material? This involves human subjects and we will only do informal evaluation with our subject matter expert. Our results will be qualitative.
2. How well can the user provide missing information necessary to supply sufficient information to allow the computational graph to compose models as needed to answer users' questions? This also involves human subjects and we will only do informal evaluation with our subject matter expert. Results will be qualitative.
3. Reliability of the curation process can be assessed by curating knowledge, restarting the system, and verifying that the knowledge is still available and usable. This measure will be quantitative and should be 100%.
4. Scalability of the curation process will consider the effect of extraction from many different sources and measure the quality of query answering as the size of the knowledge graph and the computational graph grow. This will be reported as the trend of TA2 metrics for query answering as a function of number of sources extracted.

Conclusions and Next Steps

The ANSWER ASKE TA1 team has made steady progress towards our project goals. The most important challenges that we will be addressing in the next few months are the following.

1. Extract augmented type information (domain meaning of inputs, outputs, relationship of inputs and outputs to each other) from code and text and insertion into the semantic knowledge graph
2. Improved integration of extraction from code and text using locality search
3. Integration of K-CHAIN into the TA2 query-answering framework
4. Handling of ratio equations where the left-hand side is a ratio, e.g., temperature over total temperature. These equations are very important in the hypersonics domain.
5. Enhanced extraction from code including detection of constants.