HyperThesis: Topological Hypothesis Management in a Hypergraph Knowledgebase

Cliff A Joslyn[1], Michael Robinson[2], J Smart[3], Khushbu Agarwal[4], David Bridgeland[3], Adam Brown[5], Sutanay Choudhury[4], Brett Jefferson[1], Brenda Praggastis[1], Emilie Purvine[1], William P Smith[1], Dimitri Zarzhitsky[1]

Introduction

HyperThesis is a system built in response to the needs of DARPA's Active Interpretation of Disparate Alternatives (AIDA) program for the ingest and digest of massive multi-source corpora, and management of hypotheses against them. HyperThesis specifically aims at TA2 tasks for knowledgebase population, and TA3 for hypothesis formation and reasoning. Our approach is based on 1) knowledge representations rooted in hypergraph reasoning; 2) a topological representation of hypotheses for generation and management. This paper reports on progress from January 2018 through the September/October trial evaluation period.

System Overview

HyperThesis is a hybrid system built for the DARPA AIDA program's TA-2 (knowledge base population (KBP)) and TA-3 (hypothesis management) technical areas. Our planned system architecture is shown in Figure 1, . Components include

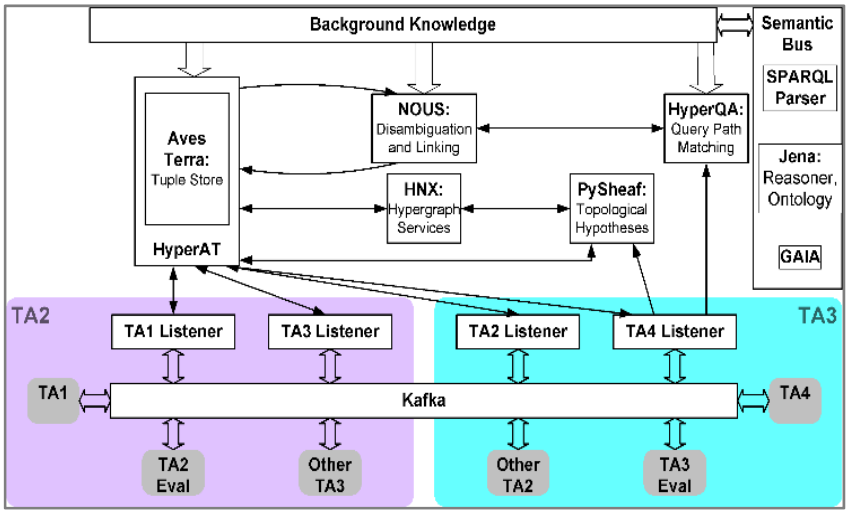


Figure 1: HyperThesis architecture

- **AvesTerra and HyperAT (HAT)** to support hypergraph storage, traversal, and query;
- **HyperNetX (HNX)** to build hypergraph representations of, and methods on, semantic graphs;
- **NOUS** which provides disambiguation and linking of the TA1-provided mini-KBs;
- **PySheaf** which implements a hypothesis scoring method built from consistency measures of shared, typed entities and relations cast as topological sheaves;
- **HyperQA,** a hypothesis generation and ranking module using graph embedding; and
- **Semantic Infrastructure** to supporting the transfer, ingestion, and manipulation of semantic data within the AIDA Interchange Format (AIF) and Seedling ontologies. TA2 queries were completed with SPARQL and exported to AIF.

[1] National Security Directorate, Pacific Northwest National Laboratory, Seattle, WA
[2] Mathematics and Statistics Department, American University, Washington, DC
[3] Georgetown University, Washington, DC
[4] Physical Sciences and Computation Directorate, Pacific Northwest National Laboratory, Richland, WA
[5] Mathematics Department, University of Utah, Salt Lake City, UT

M9 Evaluation Deployment

For the Month 9 AIDA evaluation, the future deployed capability of HAT was provided by reification of hypergraphs in semantic graph database tools interpreting received data and rules, which we called the "Semantic Bus"; and the future deployed capability of PySheaf was supported by custom Sheafbox (SBX) module for sheaf-based combinatorial optimization.

Hypergraph Knowledge Representation Methodology

Conflict/Attack = CA	Roles	ATTACKER	TARGET	CA.INST	PLC
	Types	PER|ORG|GPE	PER|PLC|Thing	WEA|VEH|Thing	PLC
	Instance	Pro-Russian Separatists	MH-17	Buk-332	Donbass
Move Transport Artifact = MTA	Roles	AGENT	ARTIFACT	MTA.INST	DEST
	Types	PER|ORG|GPE	Thing	VEH|WEA	PLC
	Instance	Ukraine	MH-17		Donbass

Figure 2 Examples of AIDA event architecture

Hyperthesis approaches AIDA needs from the perspective of mathematical modeling and consistent mathematical representations. Figure 2 shows examples of the event structure drawn from the AIDA evaluation ontology, specifically around the two event types Conflict/Attack (CA) and Move/Transport Artifact (MTA). Our representational approach is grounded on two planks, which work in conjunction and will be illustrated below:

Figure 3 Reified graph representation of MTA1 event

- **Hypergraph Representations of n-ary Edges:** Since AIDA events can have *n* slots (and relations can have two), we represent them as whole *n*-ary hyperedges as part of a hypergraph knowledgebase, rather than as reified graphs linking to *n* arguments. The left side of Figure 3 shows an instance of an MTA event as a 3-hyperedge. While preferable for internal representation, hyperedges correspond to unique reified graphs as shown on the right. HyperThesis uses reification strategies to communicate its hypergraph KB to other AIDA components.
- **Semantic Role Mappings:** Note that the slots of each event type are characterized by information of different kinds:
  - **Argument Roles:** The semantic role played by a particular slot in an event;
  - **Entity Types:** The types syntactically legal to fill a slot; and finally
  - **Entity Instances and Fillers:** The actual value of an entity instance or filler participating in a slot.

  We recognize these three forms of information existing in mutual many-many relations: each argument role can have many entity types, and many instances; and *vice versa*.

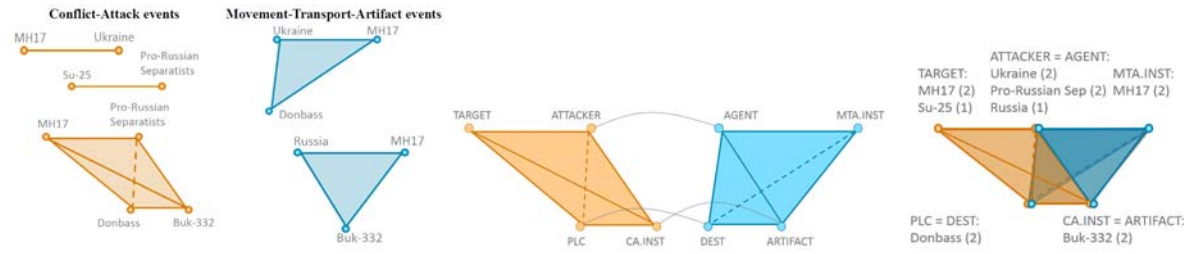


Figure 4 (Left) Example event instances (Center) Event gluing on roles (Right) Glued events instances with slot counts.

The left side of Figure 4 shows example instances of these two event types, now using the illustrative style of topological "*n*-simplices", or n-dimensional hypertetrahedrons. Note that most of these events are partially filled, except the CA in the lower left. The center shows the event types now with vertices adorned with their roles, and a role gluing mapping. The right side shows the results of the gluing over all the instances, representing the combined information about these two event types. Note the counts of all the slot instances, for example two

(consistent) claims that MH17 is the instrument of the MTA event, and five total inconsistent claims about the CA Attacker, equivalent to the MTA Agent, being either Ukraine (2 counts), Pro-Russian Separatists (2 counts), or Russia (1 count).

AvesTerra and HyperAT

There was substantial advancement of the AvesTerra knowledgebase component in support of the M9 evaluation. We ingested 877,170 TA1 knowledge elements into AvesTerra from BBN's TA output, including 555,725 entities, 291,264 events, and 30,181 relations. The knowledge elements are represented as hypergraph edges instead of semantic triples. For example, the MTA event shown in Figure 3 is represented in AvesTerra as a single AvesTerra (hyperedge) event entity, shown in Figure 5.

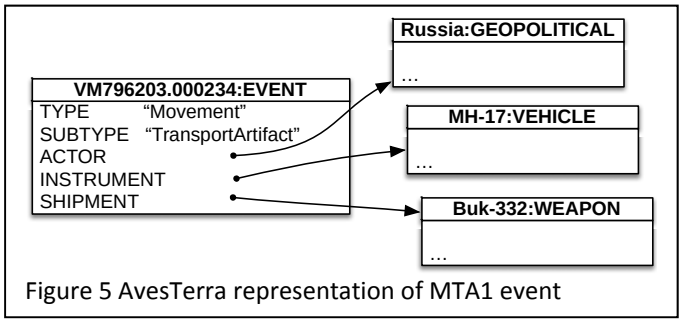


Figure 5 AvesTerra representation of MTA1 event

As a first step toward the HyperAT extension of AvesTerra for Hyperthesis, we implemented a query API for AvesTerra. The query API includes functionality to find *node neighbors* and *edge neighbors*. Node n_2 is a node neighbor of some node n_1 if and only if there is some hyperedge including n_1 and n_2. And hyperedge e_2 is an edge neighbor of hyperedge e_1 if and only if there is some node n included in both e_1 and e_2 (Figure 6).

When

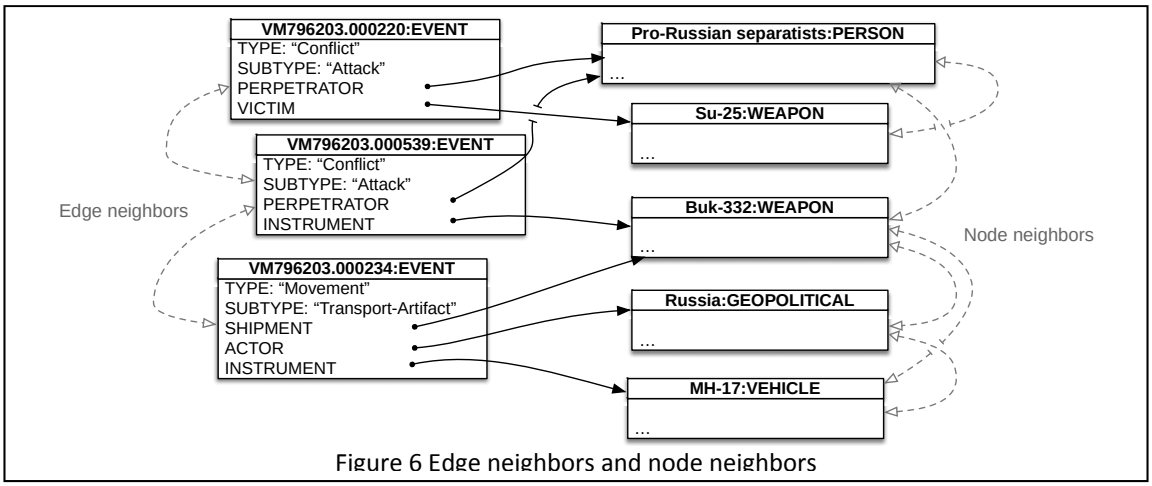


Figure 6 Edge neighbors and node neighbors

get_node_neighbors() is invoked on the node **Buk-332**, the three nodes **Pro-Russian separatists**, **Russia**, and **MH-17** are found. **Pro-Russian separatists** is a node neighbor of **Buk-332** because the two nodes are connected via the Conflict/Attack hyperedge **VM796203.000539**. **Russia** and **MH-17** are node neighbors of **Buk-332** because all three are connected via the Movement/Transport-Artifact hyperedge **VM796203.000234**.

When **get_edge_neighbors()** is invoked on the hyperedge **VM796203.000539**, the two hyperedges **VM796203.000220** and **VM796203.000234** are found. **VM796203.000220** is an edge neighbor of hyperedge **VM796203.000539** because both connect node **Pro-Russian separatists**. **VM796203.000234** is an edge neighbor of hyperedge **VM796203.000539** because both connect node **Buk-332**.

Semantic Bus Infrastructure

The AIDA program requires use of the Semantic Web technology software stack to natively manipulate RDF, be cognizant of clear text constructs and blank nodes, query in memory and triple store RDF graphs, interpret and generate SPARQL, and do so at the scale of an annotation graph containing at least 85 million triples. Many existing projects and toolsets have been deprecated as community interests diverge, but there are several active and well-maintained projects. To address AIDA technical requirements, we selected the tools detailed below.

Several of these tools support community guidelines sufficient for completing projects with AIDA's unique infrastructure. While commercial enterprise products like StarDog, AllegroGraph and Top Braid built for scale and high-speed processing of billions of triples, there are financial and logistical barriers to their use in research.

Two core ontologies, and a small bridge vocabulary of mappings between the two, encode the result graphs shared across Topic Areas. The first ontology named the AIDA Integration Framework (AIF) serves as a semantic wrapper encoding triples, annotation metadata, and hypotheses for distribution across teams. The triples encoded by AIF contain the second ontology known as the LDC Seedling, and this ontology is this knowledge framework that defines entities, events, and relationships observed in the core article dataset.

The LDC Seedling Ontology (v7) was composed as a Semantic Graph and populated with annotations provided by NIST. The current annotation import graph contains 130,090 triples, including 23,553 entities, and provides formatted data structures to TA2/TA3 performers for data disambiguation (TA2) and hypotheses generation (TA3). To access this information an inference engine, or corresponding algorithm, must first "unwrap" the LDC Seedling information from corresponding AIF S-P-O encoding, and then treat this new graph as a unique knowledge base for querying the document annotation / clustering graph.

Result graphs from TA2/TA3 contain new clustering and hypotheses nodes which must be "re-wrapped" into AIF and associated to the corresponding AIF node containing the LDC Seedling entities, relationships, or events. This continuous process of creating new knowledge graphs and updating existing annotation with clustering and hypothesis data is accomplished through the Semantic Bus, a global wrapper containing specific RDF graph manipulation endpoints depending on Topic Area team specifications. Because teams and algorithms often like to communicate and process graph data in native formats the Semantic Bus creates a bridge between the Semantic Technology stack, RDF graph manipulation, and Topic Area data requests via a push/pull data retrieval model.

- **Apache Jena:** Apache Jena (Jena) is a free and open source Java framework for building Semantic Web and Linked Data applications. The framework is composed of different APIs interacting together to process RDF data, and Jena is the most common framework for interacting with RDF encoded data and graphs programmatically (https://jena.apache.org/).
- **Redland RDF:** Free software C libraries that provide support for the Resource Description Framework (RDF). These libraries include object-based functionality and APIs for manipulating the RDF graph, triples, URIs and Literals. Storage for graphs is supported in memory and persistently with Oracle Berkeley DB, MySQL 3-5, and querying with SPARQL and RDQL is supported (http://librdf.org/).
- **OpenLink Virtuoso:** Virtuoso Server is a middleware and Triple Store engine hybrid that supports disk-based graph storage and SPARQL querying of RDF graphs. Virtuoso contains a SPARQL 1.1 compliant internal inference engine and supports a small subset of OWL 2 DL reasoning capabilities (https://virtuoso.openlinksw.com).
- **StarDog:** Enterprise Triple Store that supports disk-based graph storage and SPARQL querying of RDF graphs. API libraries are available in several languages, and support is provided for Apache Jena integration for direct communication with the Triple Store. StarDog contains a SPARQL 1.1 compliant internal inference engine and supports a large subset of OWL 2 DL reasoning capabilities (https://www.stardog.com/).
- **Apache Jena TDB2:** Light-weight Triple Store that supports disk-base d graph storage and SPARQL querying of RDF graphs. While TDB is SPARQL 1.1 compliant, optimization of query statement execution is top down creating vastly different runtimes depending on query construction. Inference engines must be provided as sub-packages or separate product installations (https://jena.apache.org/documentation/tdb).

- **Stanford Protégé:** Integrated Development Environment (IDE) specialized for creating and maintaining ontologies. Capabilities include loading and parsing RDF in a range of provided formats, visualization of semantic graphs, plug-in inference engine capabilities, graph validation depending on selected inference library, and exporting the final ontology and entities. A web version of the software exists for team collaboration, but functionality and stability are significantly worse than the single user desktop application (https://protege.stanford.edu/)

Component	Version	Description
AWS-EC2	m5.xlarge	16G Ram
Apache Jena	2.10.0	Java 1.8
Redland RDF	2.0.15	LIBRDF / Raptor
Testing Triple Store		Virtuoso / StarDog
Evaluation Triple Store	3.5.0	TDB / TDB2

Table 1:

TA2 Knowledge Base (KB) Completion

We are developing a hypergraph-based extension of NOUS[6] (Choudhury, 2016) to provide algorithms for entity and event coreference resolution in HyperThesis.

Entity Disambiguation

We extended the collective entity disambiguation method developed by Han et al. (Han, 11) for hypergraphs. The input to the algorithm is a set of associated entity mentions in document or event context, and a background KB. The algorithm maps the entity mentions in each event slot to an entity in the background KB. For entity mentions that are not associated with any event, all such mentions in the same document are collectively disambiguated. When disambiguation is successful, the mention's "disambiguated entity id" is set to the id of the appropriate node in the background KB. When a successful match was not available it remains NIL.

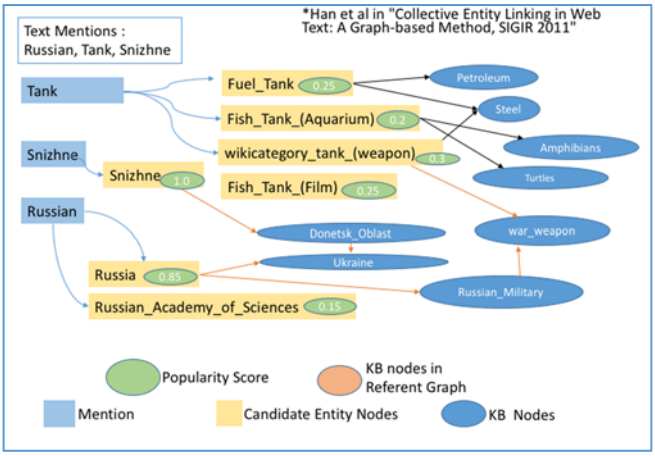


Figure 5 Referent graph for Collective Entity Disambiguation

This algorithm identifies a set of candidate entities for each mention. We perform candidate generation by building a Lucene index for all node labels in the background KB and performing approximate string matching to generate candidates. The candidate entities are given an initial "popularity score" based on their node degree in the KB. Next, we build a "referent" graph where each node either represents a mention in the input event, or their candidate nodes from the background KB. Each edge between a mention to candidate entity is weighted based on their contextual similarity computed as the Jaccard similarity of their neighbor sets, and each edge between two entities are ranked based on their semantic relatedness.

$$SR(a,b) = 1 - \frac{\log(\max(|A|,|B|)) - \log(|A \cap B|)}{\log(|W|) - \log(\min(|A|,|B|))}$$

Finally, we perform belief propagation on the referent graph and select the combination of candidate nodes leading to a maximally weighted graph.

Event Coreference Resolution

Entity and event coreference are performed by two different algorithms following same design principles. The key idea is to cluster events (or entities) in the disambiguated graph (H_{event}) as a pre-processing phase. Next, each pair of elements of a cluster (which can be either an entity or an event) are examined and merged. The merging is performed using a combination of rules in the current implementation, which considers entities "diambiguated_node_id", entity type and entity role in the event.

As with any clustering algorithm, clustering entities or events begins with computing a similarity matrix. For both entity and event clustering, we compute Jaccard similarity measure between the hypergraph neighborhood of any entity or event. We provide an efficient implementation of the similarity matrix by avoiding any all-pair computation. Any hypergraph can be represented by an equivalent bipartite graph. In this case, we construct a bipartite representation of H_{event} where two partite sets represent events and entities. We build the similarity matrix by only iterating over entity (or event) pairs that share a common event (or entity).

We use the Label Propagation Algorithm (Cordasco12) for clustering. Our preference for LPA stems from its speed over competing methods. It detects clusters (or communities) in a graph based on the network structure and does not require any pre-defined objective function. The intuition behind the algorithm is that a single label can quickly become dominant in a densely connected group of nodes, but will have trouble crossing a sparsely connected region. Labels will get trapped inside a densely connected group of nodes, and those nodes that end up with the same label when the algorithms finish can be considered part of the same community.

TA3: Hypothesis Generation

Hyperthesis is fielding two distinct TA3 hypothesis generation methods, one based on mathematical sheaves, and HyperQA, a hypothesis generation and ranking module using graph embedding.

Sheaf-Based Hypothesis Generation

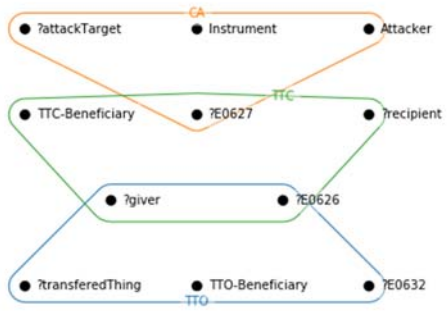
SheafBox (SBX) is a custom module based on the HyperNetX(HNX)[7] and PySheaf[8] Python libraries and created initially specifically for AIDA's M9 TA3 task evaluation. SBX models a collection of AIF-defined events and relations as a hypergraph, and then observations ("assignments") to the facets of these objects as "sections" in a mathematical object called a "sheaf of sets". Mathematically, a sheaf is a data bundle attached to a topological space. In our context the sheaf attaches a set of all possible assignments to each of the roles and facets described in the hypergraph, and as constrained by the AIF ontology types. A "section" of a sheaf is a single consistent assignment to a subset of the roles contained within a collection of events and relations. Each event drawn from the TA2 graph is represented as a section. Two sections may be combined into a single section if they satisfy certain consistency requirements. SBX computes the "maximal consistent sections" of the sheaf using an algorithm suggested in (Praggastis, 2016) and optimized for AIDA. The returned sections represent the maximal consistent hypotheses extracted from the original semantic graph. Hypotheses are scored and ranked using coherence measures developed by our team, and described below.

To minimize query latency, SBX first extracts a subgraph of the TA2 graph by restricting to justifications sourced in one of the documents referenced in the entry points specified by the information request. Using this subgraph, SBX queries for each entry point, filtering on offsets and roles. A query result represents a single solution to one of the entry point queries and has an id referencing the event or relation mention to which it belongs. We enrich the results by querying for all facets associated with each referencing id. This is necessary to assemble complete solutions consistent with all parts of the events and relations.

---

[7] https://github.com/pnnl/HyperNetX
[8] https://github.com/kb1dds/pysheaf

We model each information need frame as a hypergraph *H*. The hyperedges are the event or relation types referenced in the frame, and the nodes are the AIF defined facets (roles) associated to each of these types. Constrained facets assigned to the same variable by frame edges are identified.[9]




```
{'Transaction.TransferOwnership': {
  'Giver': '?giver',
  'Recipient': '?E0632',
  'Thing': '?transferedThing',
  'Place': '?E0626',
  'Beneficiary': 'TTO-Beneficiary'},
 'Conflict.Attack': {
  'Place': '?E0627',
  'Target': '?attackTarget',
  'Attacker': 'Attacker',
  'Instrument': 'Instrument'},
 'Transaction.TransferControl': {
  'Territory': '?E0627',
  'Place': '?E0626',
  'Giver': '?giver',
  'Recipient': '?recipient',
  'Beneficiary': 'TTC-Beneficiary'}}
```

Figure 6 Hypergraph for "AIDA_M09_TA3_P103_Q002Q004Q005_F3" mapping facets to node names

Figure 6 shows an example for the frame "AIDA_M09_TA3_P103_Q002Q004Q005_F3", defining *H* to be the hypergraph with hyperedges given by the types in {Transaction.TransferOwnership(TTO), Conflict.Attack(CA), Transaction.TransferControl(TTC)} and nodes corresponding to their facets. Facets assigned to the same frame variables by a frame edge are identified as shown in the mapping below.[10]

The hyperedges form a cover of the set of facets. The corresponding *nerve* of the cover generates a combinatorial object called an abstract simplicial complex (ASC) *X*. (Hatcher, 2001) To construct the nerve, let *S* be the set of hyperedges, then a cell in *X* is a subset of *S* such that the intersection of all of the hyperedges in the subset is nonempty. Associate to each cell in *X* the set of facets (nodes) in the intersection of the hyperedges in the cell.

Continuing our example, let *H* be as above, then it contains three hyperedges, { TTO, CA, and TTC } . The ASC *X* then contains five cells: *X* = {{CA},{TTO},{TTC},{CA,TTO}, {TTO,TTC}}. The correspondence between cells in *X* and nodes in *H* is then shown to the right.

{TTO}: {?giver, ?E0632, ?transferedThing, ?E0626, TTO-Beneficiary}

{CA}: {?E0627, ?attackTarget, Attacker, Instrument}

{TTC}: {?E0627, ?E0626, ?giver, ?recipient, TTC-Beneficiary}

{CA,TTC}: {?E0627}

The collection of all possible consistent assignments of values to the facets in the hypergraph is modeled as a sheaf of sets over the associated ASC.[11] The sheaf maps a set of assignments to each open set in the topology of the ASC. A section in the sheaf is one of these assignments and is defined on a specific open set. Each distinct entity and relation id found in the records defines a section. All consistent hypotheses for the information need are uniquely described as a section in the sheaf over the ASC.

Still continuing our example, let *X* be as before. We define a topology *T* for *X* as follows. For each $a,b \in X$, and open set $U \in T$, if $a \in U$ and $a \subseteq b$ then $b \in U$. For example, any open set in *T* containing {CA} $\in X$ must also contain {CA,TTC} since {CA} $\subseteq$ {CA,TTC}. Similarly any open set in *T* containing {TTC} must also contain {CA,TTC} and {TTC,TTO}.[12] Let E1 be an event assigning values to the facets in CA given by: E1= {?E0627: Donetsk, ?attackTarget: Russian Ministry of Foreign Affairs, Attacker: Russia, Instrument: null}. Let s1 be the corresponding section in the sheaf defined on the open set U1 = {{CA},{CA,TTC}}. Let E2 be an event assigning

[9] Additional facet correspondence may be developed for future evaluations.
[10] SBX drops the Time facet from all types due to its inconsistent appearance in the TA2 graph.
[11] For a complete explanation of sheaf theory for sensor integration see (Robinson, 2016).
[12] For a discussion of finite topologies see (Barmak, 2011). In this example the topology is:
T = { {{CA,TTC}}, {TTC,TTO}}, {{CA}{CA,TTC}}, {{TTO},{TTC,TTO}}, {{TTC},{TTC,TTO},{CA,TTC}}, {{CA},{TTC},{TTC,TTO},{CA,TTC}}, {{TTO},{TTC},{TTC,TTO},{CA,TTC}}, X, ∅ }.

values to the facets in TTC and let s2 be its corresponding section defining an assignment on U2 = {{TTC}{,CA,TTC},{TTC,TTO}}. We would like to extend s1 and s2 to a single assignment on U1 ∪ U2 = {{CA},{TTC},{,CA,TTC},{TTC,TTO}}. To do this s1 and s2 must agree on U1 ∩ U2 = {{CA,TTC}}.[13] For this exercise we define agreement to mean TA2 has assigned them the same role_id. Since the only facet contained in {CA,TTC} corresponds to the label ?E0627, this simply means the assignment defined in E2 for ?E0627 was deemed equivalent to Donetsk by TA2. If not, the corresponding events do not belong to the same hypotheses. In general agreement could be more complicated, depending on the datatype of the facet values being compared. [14]

We define a Boolean consistency function on all pairs of sections defined by the event and relation ids, which indicates if the pair agrees on their overlap. We split the set of sections into finer and finer partitions so that no pair of inconsistent sections exists in the same subset of the partition. This process converges to a unique partition in which each subset is a collection of consistent sections. These are then glued into a single section producing a hypothesis both internally consistent and maximal within the set of retrieved records.[15] We score the hypotheses using two coherence measures for the sections. Measure 1 counts the number of times each facet is assigned a value in the hypothesis. High scores will occur when individual facets are mentioned in a large number of events or relations. Measure 2 takes each pairwise intersection of the events and relations in a hypothesis and counts the number of non-null roles described by both and summing over all. High scores will occur when several events agree non-trivially on a large number of facets. To rank the hypotheses the two measures are summed.

HyperQA Hypothesis Generation

HyperQA is an in-memory, hypergraph based inference engine for event-structured data. It provides APIs for performing hypergraph walks, interfaces for selection and join queries, and an initial implementation of multi-relational embedding learning and scoring of subgraphs in a hypergraph using vector space methods.

HyperQA follows a three-step methodology for hypothesis generation. The first step transforms a SPARQL query into an equivalent hypergraph query, the second step executes a subgraph isomorphism on the hypergraph representation of TA-2 graph, finding all structural matches to the query. This is followed by ranking of returned subgraphs based on coherence measures in the third step. In future, we foresee step 2 and 3 being merged together, yielding greater efficiency by pushing a novel ranking measure into the search process itself.

Figure 6 shows an example how a SPARQL query involving a crash event and a transport-artifact event is converted into a hypergraph query. Each hypergraph query is decomposed as a left-deep binary tree where node represents a primitive operation on the hypergraph. Each leaf corresponds to a “constrained hyperedge query”, and each internal node represents a “hyperedge/hypergraph join” operation across subgraphs composed of hyperedges. Thus, the subgraph isomorphism operation on the hypergraph is reduced to executing the operations on the binary tree starting from the left-deep-most child node.

Our “constrained hyperedge selection” is designed to be a SELECT query for hypergraphs, where we select all hyperedges in the event graph whose satisfy a user specified constraint. Its equivalent SPARQL query is a star-shaped graph as shown in Figure 6. A “hypergraph join” is defined to be an operation to merge two different hypergraphs that contains same entities in event-roles specified as JOIN predicate.

The number of subgraphs returned by isomorphism can be numerous. To generate semantically meaningful hypothesis, we seek to partition the matching subgraphs into groups, where each group represents a “set of alternate interpretations”. We further sample top ranking subgraphs from each group.

---

[13] This is the ‘gluing’ property of sheaves.

[14] See (Robinson, 2018) for a detailed discussion of consistency radius to see how this is done.

[15] Note that most of the sections generated by the TA2 generated data assign null to many of the facets. To accommodate for this theoretically, we think of the original hypergraph as actually having many more hyperedges, corresponding to the possible subsets of each event and relation type. Practically, it means multiple sections could be deemed consistent simply because their corresponding open sets do not overlap on non null facets, in which case they are scored very low.

Given a query graph and a designated set of “nodes of interest” (such as conflict-attack.Attacker in Figure 6), we developed a simple and effective methodology to produce these sets of alternate interpretations. We iterate over all subgraphs returned by the isomorphism query and project each subgraph based on the “nodes of interest”. For each projected node set, we substitute each node by their cluster Ids, and compute a string representation of the

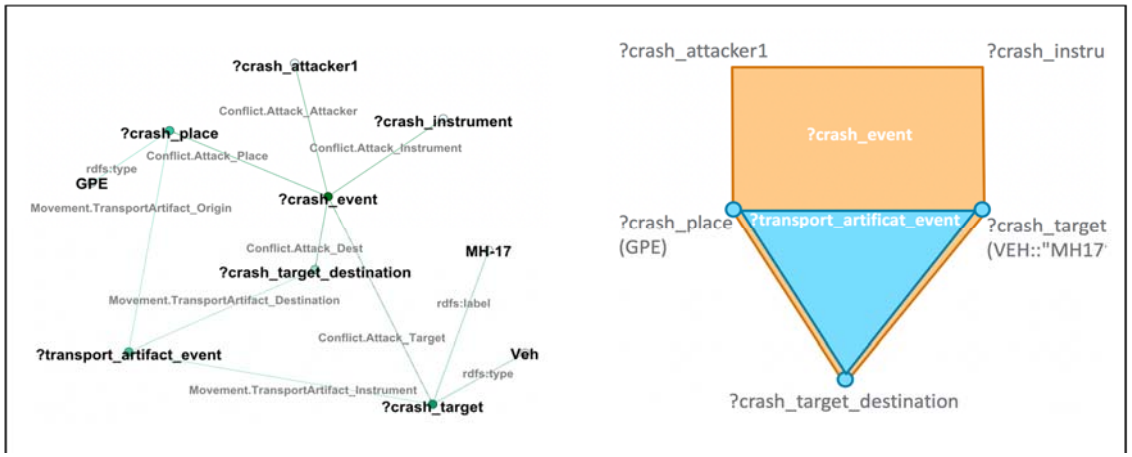


Figure 6 Transformation of a SPARQL query into a hypergraph query.

set using a canonical ordering. Finally, we simply group all answer subgraphs based on this computed key.

Finally, we rank each answer hypergraph using a vector embedding driven approach. We learn a vector representation for each node in the graph using an extension of the node2vec method. We export a bipartite graph from the Hypergraph connecting events to their arguments, and learn vector space representation of events and entities considering multi-relational properties of the bipartite graph. Given an answer hypergraph, we convert it to a walk or sequence of entities and compute a vector space divergence measure for these entities. This measure captures set (hyperedge)-level tightness of the elements, and not just emphasize on pairwise proximity in vector space.

Results

TA2

The TA-2 knowledge graph was generated from BBN’s TA-1 output and the background knowledge base provided by AIDA's BBN team. The incoming TA-1 input consisted of 36,942 event mentions, 536,587 entity mentions and 58,414 relation mentions. 231K entity mentions participated in events and had a valid label, and these were used for further processing during construction of TA-2 graph. Relations were treated as events for all subsequent analysis. The entity mentions where distributed as 24.6K images, 482.1K text, and 29.9K video.

The entity disambiguation algorithm was run on a PNNL institution computing center, using 64 nodes and running for 12 hours , processing all of the entity mentions. Out of the 231K entity mentions, 48.5K entity mentions were matched to one of the entities in base KB using the (Han, 2011) disambiguation algorithm that had a high confidence match in base KB. The entity clustering further lead to creation of 51K cluster nodes for the remaining entities. This reduced the total number of entities to 23% (54K) of the 231K original mentions. Figure 7**Error! Reference source not found.** shows the distribution of disambiguated entity mentions and node types in the TA2-KB.

The disambiguated graph was further processed for event clustering and created 20K unique event clusters. TA2 clustering created a reified graph of ~85M triples, 27K clusters, and 90K members of clusters. The background TAC-KB provided 48.5K disambiguated matches to cluster member elements from 3447 unique KB nodes. Figure 8 shows the distribution of different entity and event types in reified TA-2 KB.

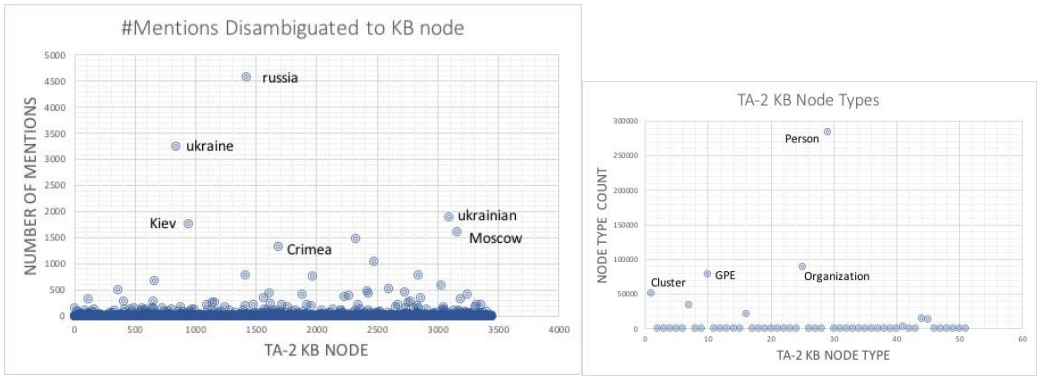


Figure 8 Disambiguation and Linking of Event and Entity Mentions in TA-2 KB

TA3

SheafBox

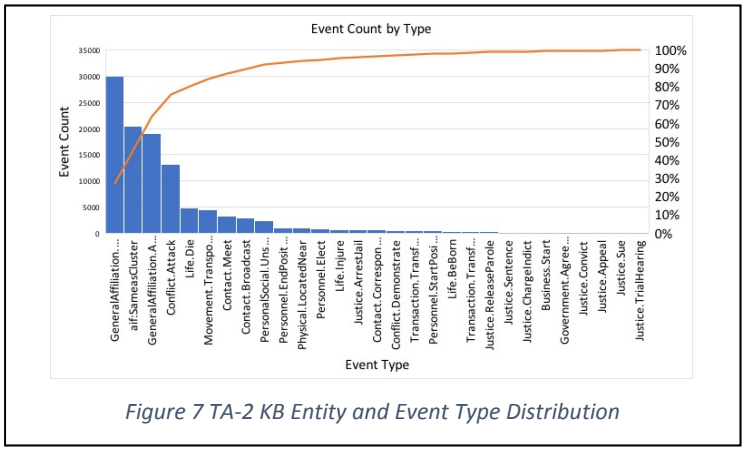


Figure 7 TA-2 KB Entity and Event Type Distribution

We queried our own TA2-created knowledge base to retrieve events and relations. Our combinatorial optimization effort restricted us to a subgraph of the TA2 graph corresponding to mentions found in the set of documents listed in the information request. This reduced the data to 25.9K triples, producing 839 events and relations.

For each frame the queries returned between 1 and 72 disambiguated events. The events were compared for consistency and assigned to one or more hypotheses. SBX produced hypotheses for all seven frames found in the information request. Most of these were disconnected graphs due to the number of null facets in the events and relations retrieved causing the overlaps to be empty. This was expected as not all of the entry points could be queried. We will reduce the number of empty slots in the next evaluation by improving the TA2 querying pipeline and by using an optimization approach from PySheaf to fill in missing slots.

For this evaluation disconnected and fragmented hypotheses received low scores due to the lack of overlapping events. A summary of one run of SBX against the TA2 graph is given below. Scoring was done by taking the sum of two measures on each hypothesis.

Info Request Frame	# Events/ Relations	# Hyp	Highest Score	Mean Score	Top Hypothesis
Q002_F1	1	1	0	0	{'Transaction.TransferOwnership_Giver': ['Russian', 'Russia']}
Q002004_F1	38	61	12 (3)	6.8	{'Conflict.Attack_Place': ['KRAMATORSK', 'Kramatorsk'], 'Conflict.Attack_Target': ['Krutov', 'Vasyl Krutov'], 'Transaction.TransferOwnership_Giver': ['Russian', 'Russia']}
Q002004005_F1	25	67	5 (2)	2	{'Conflict.Attack_Attacker': ['Ukrainian', 'Ukraine'], 'Conflict.Attack_Place': ['Russian', 'Donetsk', 'Russia'], 'Conflict.Attack_Target': ['Mariupol']}
Q002004005_F3	10	7	2 (2)	0.6	{'Conflict.Attack_Target': ['Mariupol'], 'Transaction.TransferControl_Giver': ['Kramatorsk']}
Q004_F1	19	40	16	3.7	{'Conflict.Attack_Place': ['Kramatorsk'], 'Conflict.Attack_Target': ['Krutov', 'Vasyl Krutov']}

Q005_F1	**72**	**1212**	**43 (2)**	**7.3**	**{'Conflict.Attack_Attacker': ['Moscow', 'Russias', 'Russian GRU', 'Russian', 'Russia', 'russian'], 'Conflict.Attack_Place': ['Kramatorsk'], 'Conflict.Attack_Target': ['Ukrainian', 'Ukraine\nDonetsk', 'Kyiv', 'Ukraine']}**
Q005_F3	**11**	**8**	**2 (2)**	**0.5**	**{'Transaction.TransferControl_Giver': ['Kramatorsk'], 'Transaction.TransferControl_Recipient': ['Ukrainian', 'Ukraine']}**

The data represented here is not a perfect reflection of the subgraph we generated. We believe this is due to an inconsistent use of the identifiers to distinguish mentions and clusters. We can draw some conclusions from the data, however. Hypotheses are combinations of events and relations. The number of hypotheses is a factor of the number of distinct events and relations available and their overlaps. If a pair of events overlap but are inconsistent in even one place they generate distinct hypotheses. If they don't overlap at all they may be combined into a single hypothesis. This explains the huge number of hypotheses generated from the 72 events and relations for Q005_F1. Ideally we would like a tighter fit of events and fewer hypotheses. For the next phase we plan to implement metrics to permit a non-zero tolerance for combining overlapping events that are 'close' if not exact. Note that Q002004_F1 has quite a few more events than Q002004005_F1 but they have a comparable number of hypotheses. This is explained by the Mean Score. The higher score implies more consistent overlaps and hence fewer hypotheses generated by a set of events. Our scoring technique is appropriate for rating the hypotheses, and we will improve it in M18 by factoring in confidence values for the events and entities making up a hypothesis.

HyperQA

The 5 information need queries provided by the AIDA program were run on the reified TA-2 KB for hypothesis generation. Each query consisted of an information need frame describing the basic structure of the query and multiple specification of entry points , which served as a filer on a particular kind of event or entity mention. The number of entry points per query is shown below:

- P103_Q002.xml : 104
- P103_Q004.xml : 39
- P103_Q005.xml : 103
- P103_Q002Q004.xml : 143
- P103_Q002Q004Q005.xml : 63

We created final queries using a combination of information need frame and entry point. However, we observed the entry points provided in queries severely limited the number of events/entities matched in the graph, where most of the entry points seem to be randomly created and did not match any entity and event properties in the final TA-2 reified graph. The table below shows the number of hypothesis generated per infoneed query and entry point. The maximum matches for the query Q005 with frame need specifying a conflict attack. The hypothesis generated contained following attack place as candidate matches, where support indicates the number of events in the graph supporting the hypothesis.

- Kramatorsk : Support 15
- Kramatorsk Airport : Support 2
- Lugansk Airport : Support 1

Infoneed Frame Id	Number of Hypothesis
Q004	5/39
Q005	5/103

Table 1: Number of hypothesis for infoneed queries

Conclusions and Future Work

This paper reports on our experiences leading up to the M9 evaluation for AIDA. The severe time pressures of the AIDA resulting in this preliminary report, and our team is honing its understanding of results. Future work includes:

- **TA2/NOUS Enhancements:** Our M18 plans are primarily focused on verification and validation. We plan to develop an inference query benchmark to evaluate the quality of any given KB (TA2) and hypothesis queries on the KB (TA3). The benchmark would be a collection of inference chains collected from LDC annotation datasets by graph walk from specific entities of interest. Currently, the AIDA program only has annotated datasets, and without any notion of "absolute correctness" measured in terms of presence of specific entity/events or their cardinality, we cannot evaluate our algorithms in a principled fashion. Articulating the problem space in terms of query metrics, and the ability to reason about where we do well and why (or where we do not fare well and why) will be valuable. Given the complexity and noisiness of the data, we hope that a commonly agreed upon query benchmark will help the AIDA community to precisely enumerate program objectives and measure progress.
- **SBX/PySheaf Enhancements:** The initial version of SBX provided a first attempt at hypotheses generation from a subgraph of focused semantic data, and was limited by subgraph generation and the metrics used to compare events for consistency, which reduced the number of entities and events drawn from the TA2 graph. With only a discrete metric we were forced to generate more hypotheses than needed from a small set of events. These events were mostly incomplete, so our hypotheses provided incomplete descriptions of events and were often disconnected. We will utilize optimization capabilities from PySheaf to enrich the queried events with additional events by filling empty slots in the TA2 generated events with likely solutions. We will identify appropriate metrics for the datatypes in order to cluster hypotheses with close meaning. Finally we will use role alignment to better interpret the hypotheses generated.
- **HyperQA Enhancements:** We will make algorithmic enhancements for M18:
    - **Query expansion**: Seed hypothesis queries beyond specified information need entry points to include additional mentions that are closely related to mentions in those entry points.
    - **Inference:** Generate new event structures from the collection of TA2-graph events returned by M9 query. We will compress answer set by inferring new events with filled slots as opposed to many events with partially filled slots.
    - **Ranking via Coherence**: Rank events generated by the inference phase via different coherence /subgraph divergence measures.
    - **Generate hypothesis groups:** group hypotheses into multiple groups based on entities participating in pivotal roles in query, where each group represents an alternate explanation.

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. All data were supplied by the DARPA AIDA program. Any hypotheses algorithmically derived from this data were not judged for their correctness. Our claims of coherence are also derived algorithmically rather than by which are actually correct or incorrect. PNNL-SA-140712. Distribution Statement "A": (Approved for Public Release, Distribution Unlimited).

References

Barmak, J. A. (2011). *Algebraic Topology of Finite Topological Spaces and Applications.* Berlin Heidelberg: Springer-Verlag.

Choudhury, S., Ag*arwal, K., Purohit, S., Zhang, B., Pirrung, M., Smith, W. and Thomas, M., (2017) Nous: Construction and querying of dynamic knowledge graphs. ICDE Workshops.*

Cordasco, G. and Gargano, L., 2012. Label propagation algorithm: a semi-synchronous approach. *International Journal of Social Network Mining*, *1*(1), pp.3-26.

Ghrist, R. (2014). *Elementary Applied Topology.* CreateSpace Independent Publishing Platform.

Han, X., Sun, L. and Zhao, J., (2011). Collective entity linking in web text: a graph-based method. ACM SIGIR.

Hatcher, A. (2001). *Algebraic Topology.*

Praggastis, B. (2016). Maximal Sections of Sheaves of Data over an Abstract Simplicial Complex, arXiv:1612.00397v1.

Robinson, M. (2016). Sheaves are the canonical datastructure for sensor integration {arXiv:1603.01446v3 [math.AT]}.

Robinson, M. (2018). Assignments to sheaves of pseudometric spaces { arXiv:1805.08927v2 [math.AT]}.