AFRL-RI-RS-TR-2023-006

OPERA: OPERATIONS-ORIENTED PROBABILISTIC EXTRACTION, REASONING, AND ANALYSIS

CARNEGIE MELLON UNIVERSITY

JANUARY 2023

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2023-006 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
ALEKSEY V. PANASYUK
Work Unit Manager

/ S /
SCOTT D. PATRICK
Deputy Chief
Intelligence Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
JANUARY 2023	FINAL TECHNICAL REPORT	DECEMBER 2017	SEPTEMBER 2022

4. TITLE AND SUBTITLE

OPERA: OPERATIONS-ORIENTED PROBABILISTIC EXTRACTION, REASONING, AND ANALYSIS

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8750-18-2-0018	N/A	62303E

5d. PROJECT NUMBER	5e. TASK NUMBER	5f. WORK UNIT NUMBER
		R2F9

6. AUTHOR(S)

Yonatan Bisk, Hans Chalupsky

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Carnegie Mellon University
5000 Forbes Ave
Pittsburgh PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Air Force Research Laboratory/RIEA 525 Brooks Road Rome NY 13441-4505	AFRL/ RI	AFRL-RI-RS-TR-2023-006

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The OPERA system (for Operations-oriented Probabilistic Extraction, Reasoning, and Analysis) developed jointly by CMU and USC/ISI is an integrated solution to the challenges of DARPA's Active Interpretation of Disparate Alternatives (AIDA) program in the form of: (i) high-performance media analysis (TA1) for text, speech, and image/video data, (ii) semantic representation and reasoning support (TA1 and TA2), (iii) cross-medium and cross-language integration (TA2), and (iv) hypothesis creation, management, and hypothesis exploration (TA3). Given that all required components of such a system are still active areas of research, the creation of a single system (pipelined or otherwise) has the potential for a substantial rate of compounded errors. Early versions of the system created had strong abstraction boundaries for limited information sharing between systems. Later incarnations benefited from allowing for the output of extractors to be coupled with raw text strings and embedding vectors. These prove especially advantageous in the presence of large-scale language models that encode world knowledge, and when aligning predictions to an open-domain ontology, like that of WikiData.

15. SUBJECT TERMS

Deep Exploration and Filtering of Text (DEFT), Automated Low-Level Analysis and Description of Diverse Intelligence Video (ALADDIN), Global Autonomous Language Exploitation (GALE), Broad Operational Language Translation (BOLT)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	70

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
ALEKSEY V PANASYUK	N/A

TABLE OF CONTENTS

LIST OF FIGURES

1.0 SUMMARY

The goal of DARPA's Active Interpretation of Disparate Alternatives (AIDA) program is to produce a system that can develop or provide multiple hypotheses (alternative interpretations) of events and statements from unstructured sources. Such a system would be able to both digest multiple accounts for news of events, situations, and claims and uncover the underlying common truth, despite the highly variable representation of the information and therefore propose how much events might be portrayed. To achieve this, the system aggregates knowledge from multiple languages and modalities to build an explicit representation that can be queried to produce hypotheses about said events with appropriate provenance for the claims. A classic example of this task might include merging characterizations like ``killed" versus "murdered" or "freedom fighter" versus "terrorist". Fundamentally, in either case there is a death, a cause, and a party responsible. These details can be mapped to knowledge entities (locally or via WikiData).

The OPERA system (for *Operations-oriented Probabilistic Extraction, Reasoning, and Analysis*) developed jointly by CMU and USC/ISI is an integrated solution to the challenges of DARPA's AIDA program. It combines the following (TA1/2/3 are technical focus areas of the AIDA program):

- High-performance media analysis (TA1) for text, speech, and image/video data
- Semantic representation and reasoning support (TA1 and TA2)
- Cross-medium and cross-language integration (TA2)
- Hypothesis creation, management, and hypothesis exploration (TA3)
- Integration framework (computational and semantic)

The system also produces standardized representations for sharing with other teams that digest either of TA1 or TA2 predictions. Given that all required components of such a system are still active areas of research, the creation of a single system (pipelined or otherwise) has the potential for a substantial rate of compounded errors. An incorrect entity impacts the creation of a role or event, which in turn limits cross-document, cross-medium, or cross-language clustering, and ultimately prevents the creation of a unified hypothesis.

Early versions of the system created had strong abstraction boundaries for limited information sharing between systems and relied on a custom LDC ontology. Later incarnations benefited from allowing for the output of extractors to be coupled with raw text strings and embedding vectors. These prove especially advantageous in the presence of large-scale language models that encode world knowledge, and when aligning predictions to an open-domain ontology, like that of WikiData. Importantly, these changes strengthen the resulting system and increase robustness for use in real deployment settings.

2.0 INTRODUCTION

The goal of DARPA's Active Interpretation of Disparate Alternatives (AIDA) Program was to develop a multi-hypothesis semantic engine that can generate explicit alternative interpretations of events, situations, and trends from a variety of unstructured sources. Such events might be natural disasters or international conflicts where analysts are often quickly inundated with large amounts of noisy, conflicting and possibly deceptive information, which makes it difficult to understand what is relevant and how to respond appropriately. The AIDA engine must be capable of automatically extracting knowledge elements from multiple languages and media sources, aggregate information derived from those sources, and generate and explore multiple alternative hypotheses about ongoing events, which can then be presented to and interactively explored by a user such as an intelligence analyst.

CMU's role in this collaboration focused on TA1 and TA2 – the creation of inter- and intra-document or modality representations.

- **Intra-document** Within individual documents, the system must produce accurate extractions of entities, events, and relations. This thresholds for precision vs recall must be appropriate for rich hypothesis formulation without undue noise propagation. Extractors need also operate in multiple languages and from vision. Both high precision expert annotated approaches and high recall learning based techniques are employed here.
- **Inter-document** Once local knowledge entities are built, they are clustered across document and modality to begin unification of like evidence.

USC/ISI's role in this collaboration was to design and develop representation mechanisms and software components to provide the following high-level functionalities:

- **Common Semantic Repository** which provides a representation formalism, ontologies, repository, inference engine and APIs to store, access, map, disambiguate and link knowledge elements (KEs) generated by TA1 modules or entered directly by analysts.
- **Hypothesis Generation and Management** to generate and manage semantically coherent hypotheses that are supported to some minimal degree by evidence available to OPERA, and to record and manage alternatives and allow backtracking and retraction under guidance of the Hypothesis Reasoner.

The AIDA program was very ambitious in its goals and posed extremely difficult technical as well as engineering challenges to the various teams addressing them. Given all the different input modalities, languages, media types, source and extraction noise, and requirements for knowledge element extraction, inference, linking, querying, cross-component and cross-team integration, containerization and end-to-end system automation, the resulting systems turned out to be extremely complex and difficult to build, test and debug, particularly in an academic, non-commercial research environment. The final components amount to massive code-bases, each in the 10s of thousands of lines. A large portion of this very sizable code base focuses on engineering and integration issues, which often turned out to be more important for overall system performance than the many interesting technical problems that also had to be solved.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

This section is broken down into four main sections, corresponding to the above innovations and components.

3.1 Intra-Document: Multilingual Extraction of Entities and Events

Event and entity extraction are the seed for the entirety of AIDA. Specifically, we have both a learned model and a high precision approach (COMEX).

3.1.1 COMEX

COMEX is a shallow semantic parser that uses hand-crafted rules instead of machine learning to transform incoming text into KE structures. We leverage cross-lingual domain knowledge to produce these rules and reconcile normalizations. We extract common patterns for phrasings and write rules for such templates (e.g. "X verb Y" for common verbs). For entities, we can also use cultural saliency to resolve ambiguities (e.g. Moscow being the Russian capital, not a city in Ohio, when referenced in Russian or Ukrainian texts).

Specifically, The COMEX system aims to produce KE frames from the output of off the shelf parsers (e.g. StanfordNLP and the Universal Dependency parser UDPipe 1.2). Event trigger terms are manually mapped to the ontology (either by direct matching of a manually curated list of trigger words, or using English triggers in translation or through WordNet/dictionary lookup). We perform some annotation, starting with LDC's seedling corpus and adding our own manual annotation. Shallow semantic / conceptual extraction is a form of rule-based inference. The rules are attached to lexical entries or to ontology nodes, and the lexical entries are attached to ontology nodes. The ontology and lexical entries are semi-manually created from the training data. The COMEX ontology is a superset of the NIST/LDC ontology, augmented as needed for shallow semantic frames. The COMEX ontology supports far greater detail than even the AIDA program ontology and allows multiple inheritance. A fragment is shown below.

Lexical links connect wordsenses to ontology concepts. They provide rules for instantiating and connecting concepts into a mention graph. The semantic requirements for slot fillers are specified in the ontology. While there may be numerous lexical items, the number of different rules is small thanks to the semantic similarity of many entities and the expressive power of the rule notation. An example definition is shown in Figure 1. These rules are handle constructed in each language, guaranteeing high precision. Coverage is improved by use of automatic machine translation. Azure translation services allows for English language models to be run on other languages and provides gloss alignments for projecting back to the original source text. This allows for accurate detection of events and filling of argument roles. We will leverage this approach as a slightly noisier but high recall approach for all TA1 extractors.

Finally, to aid in the efficient construction of COMEX rules, we extract minimal cases from the available texts. These simple examples (a verb and its necessary arguments) are base cases that can be efficiently converted to rules and which generalize to more complex or compound sentences thanks to accurate syntactic parsers that handle long-range dependencies.

```
W, атаковать, WS:attack-physical, WS:attack-verbal
S, WS:attack-physical,  *attack-physical,  VERB
A, WS:attack-physical,      Attacker = Pull:active-subj;     Pull:passive-subj
A, WS:attack-physical,      Target    = Pull:active-dir-obj; Pull:passive-dir-obj
A, WS:attack-physical,      Instr     = Pull:active-subj
A, WS:attack-physical,      Place     = Pull:obl-in
#
R, Pull:active-subj,     nsubj,    Trigger->Voice=Act
R, Pull:passive-subj,    obl,      Trigger->Voice=Pass,   Target->Case=Ins
```

Figure 1: Example lexical item for the event "attack"

3.1.2 ML Extraction

We use learned models for both entities and events. For our entity model, we leveraged multilingual-BERT (Devlin et al 2018), a pre-trained multilingual language model as the backbone. Then, training data with a similar ontology to AIDA is collected and preprocessed. Specifically, we build a coarse-to-fine hierarchy to mirror LDC. For the most fine-grained levels, we lack training data due to natural sparsity.

For events, we employ a two-step approach for the implicit argument detection, which enhances our event pipeline and makes it able to collect more arguments, including certain implicit ones that can go across sentence boundaries. More specifically, given an event trigger, we first detect all the possible head words of its arguments, with the intuition that the head words of the argument spans already contain enough information for the argument linking. In a second-step, we expand from the head words to the full spans. We train our system on the recent RAMS dataset and get a F1 score of around 70 if given gold argument spans and around 40 if not. This module can be directly integrated in our event pipeline after the basic sentence-level processing. Events are also processed with a multilingual BERT backbone. This approach works best with English and Spanish. We further enhance the models by translating English text and aligning the ontology to the new tokens. This augmentation strategy leverages the superior size and quality of English data.

For English and Spanish our system is trained on TAC KBP 2015-2017. Since we have non-English languages, preprocessing is kept minimal apart from tokenization. No linguistic features or annotations (e.g. dependency edges or POS tags) are used to augment the input. This also allows us to easily produce shared featurization across languages. This same multilingual BERT is than applied to Russian and Ukrainian in a zero-shot manner at test time.

To encode the input, we use a series of in-context mention pairs connected with [SEP]. The inputs are arranged as *<left-context1> <mention1> <right-context1> [SEP] <left-context2> <mention2> <right-context2>*, with the left and right contexts being capped at 128 tokens. An additional binary task layer (linear+sigmoid) is added to predict whether the mention pair is coreferent or not

coreferent. Rather than <mention1>/<mention2> just being the mention token sequence, we augment it with non-textual attribute information available as part of the input, such as mention type. This is done by way of html tag-enclosed sequences placed before or after the mention text. For instance, the mention Gamal Abdel Nasser would be represented as *Gamal Abdel Nasser <type> per politicalFigure HeadOfState </type>* . For event mentions, the augmented information is considerably more, given the presence of event arguments, realis etc. Next, to decode, as is practice, the scorer is not directly used to predict coreference - its probabilities are used to construct a pairwise similarity matrix which are then clustered together - a procedure typically called *best-first-clustering*. As noted above, we observe that using both EN+ES supervision works slightly better than just using ES supervision on the in-domain ES test set. Our model also generalizes reasonably decently to russian examples from the task set - an observation further corroborated in later quarters, especially from the TA1 Leaderboard eval, where we exhibit robust performance on Russian.

Merging Entities Across Teams There are two key concerns with transfer of embeddings between teams: When should one prediction be trusted? and How can the two embedding spaces be rectified. We looked to integrate GAIA embeddings. Where our representation is 768 dimensional (standard for BERT), GAIA appears to have trained an additional transformation that outputs a 2048 vector. This vector is produced by two 1024 vectors which we assume to lie in the same space and therefore pool to 1024. Next, we will estimate a linear projection $\{W,b\} : X_{gaia} \rightarrow X_{aida}$ using L1+L2-regularized linear regression. For training an alignment we require "seed" pairs. We consider entities from both spaces which have the *same refKB Id* - the inductive bias is that the mean entity embedding should remain invariant on projection. In other words, we minimize $L(X^r_{gaia}, X^r_{aida})$. The number of pairs we get this way is not as many as those in the multilingual mapping case - they're about 1-2% of the total cumulative number of entities. Given that the number of supervised examples is still fewer than the number of parameters (1025*768), we go for a parsimonious parameterization, with an additional L1 penalty to encourage sparsity. This approach gives us non-trivial cross-clusters (i.e the GAIA and AIDA entities aren't just simply clustered into two disparate sets of clusters), and at least 5-10% of the non-singleton clusters are heterogeneous (containing one entity from both sets). Specifically, for the final results received from our submission at the time of the post-eval remedial hackathon, of 6,526 non-singleton clusters formed, 381(i.e 5.83%) were heterogeneous.

Linking to WikiData After extracting the entities, we first employ BLINK (Wu et al 2020), an entity linking python library that uses Wikipedia as the target knowledge base, to link these entities to Wikipedia, then the wikipedia items are converted to their corresponding wikidata nodes. Since BLINK is a fine-tuned BERT architecture, it is relatively slow and not predictable. To ensure some entities of interest in this domain can be linked to the correct qnode, some simple rules are added. If linking process can not be done be these rules, BLINK will be used then.

3.1.3 Question Answering for Claim Frames

In our TA1 text Information Extraction (IE) pipeline, we also develop an initial claim frame extraction module, which takes the previously extracted IE graphs and extracts initial claim frames

as answers to the queries. Specifically, we divide this by two major steps, which are described below.

In the first step, we utilize a Question Answering (QA)-based model to directly deal with the queries themselves and extract the X variables. For each of the query templates, we first convert it to a question according to syntactic rules: 1) Locate the Noun Phrase (NP) that contains the "X", replace things with "Wh" words and move to the front. For example, for the query of "Treatment X cures COVID", we identify the NP of "Treatment X" and replace it with "What treatment". 2) For the cases where the NP is not the subject of the main sentence, we further change the word order according to English grammar, for example, from "Where coronavirus was created?" to "Where was coronavirus created?". We find that these simple rules are enough to handle most of the queries that we have and we can have reasonable questions for them. In the second step, we utilize a QA model to extract the answer, which we treat as the X variable. With the above conversions, the inputs are exactly the same as a standard QA problem: a question and a paragraph (we chunk the document into several groups of sentences which we feed at one time). Nevertheless, the outputs will be slightly different since we want to score our extracted entities or events, therefore we output a binary probability score for each token and train the model with binary loss. At testing time, we re-rank all the entities and events by the scores of the tokens inside their mentions and decide the final results with certain thresholds. With the around 500 English documents in the practice set, we are able to extract around 3k answers for the queries, which we form as our initial claim frames. In the next steps, we plan to adapt the model to the covid domain with domain-specific datasets.

In the second step, we aim to fill in the other parts of the claim frames, of which the epistemic status and the claimer are the most important fields. For the epistemic status, we plan to utilize a Natural Language Inference (NLI) model to judge the polarity (positive or negative) of the extracted answer according to the query template. We can form a premise by taking the surrounding context of the answer mention and put the filled template as the hypothesis. In this way, we can judge the polarity of the extracted template. For the judgement of certainty, we will utilize our labels obtained from our FactBank model. For the extraction of the claimer, we plan to perform another round of QA, and simply query with the question of "Who said that …", which is a straightforward way to identify the claimer. We will start with these simple schemes and later extend to more advanced techniques.

3.1.4 Multimodal Representations

The available multimodal content shifted throughout the program. We began with a focus on video and speech data, but later shifted with the program to primarily focus on images.

3.1.4.1 Audio

For all video content we employ Automatic Speech Recognition (ASR) to extract speech. We use several Kaldi based systems, individually. For the Ukrainian recognizer, we use LDC2018E73 and LDC2018E74 (126.3 hours of training data), for Russian LDC2018E75 (43.5 hours of training data), and for English a dataset of 1700 hours from Switchboard Fisher and 2000 hours of medical

conversations. In our experiments, these configurations performed best overall on diverse test data. Language ID is done by comparing the language model perplexities, with the most fluent (lowest perplexity) being chosen as the correct language. The results are: Russian: 34.8% WER and Ukrainian: 32.9% WER

3.1.4.2 Faces

To determine if an image or video is corroborating content in the text, we extract entities from the images to be aligned.

For object detection we first obtain bounding boxes through a Faster-RCNN (Ren et al., 2015) model pre-trained on the MS-COCO dataset (Lin et al., 2014). The object detection model is then fine-tuned on the Visual Genome dataset (Krishna et al., 2017). The class pool size for this dataset is 1600 different object classes. The performance achieved through this work is close to the current state-of-the-art on the VQA V1 challenge. We use the region-proposal network to extract type-specific bounding boxes. Then to classify the objects found within the detected bounding boxes we used an Inception V3 (Szegedy et al., 2016) with a class pool filtered from Freebase Entities and Google KG.

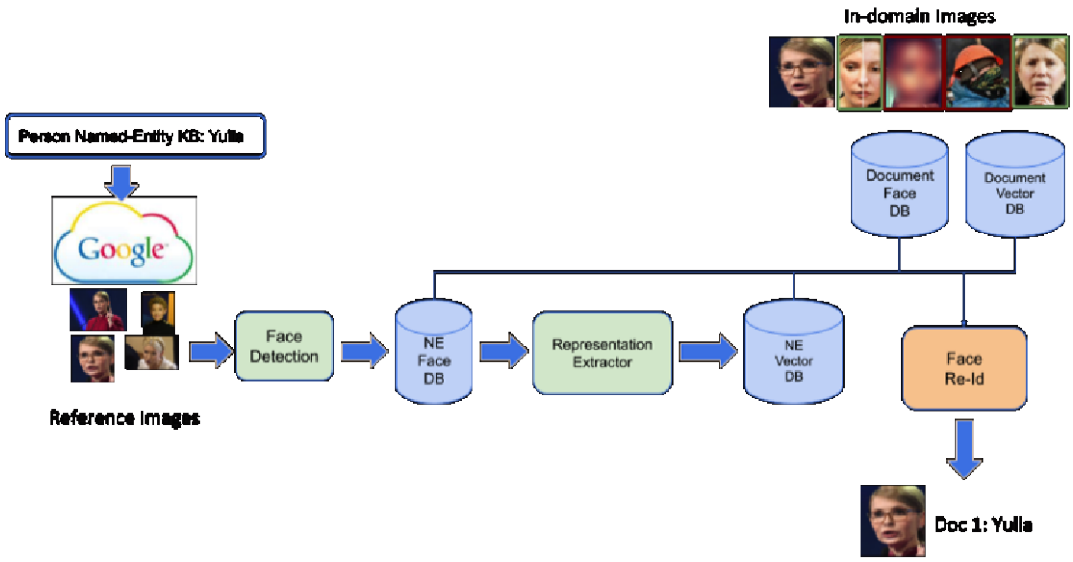

Figure 2: OPERA visual processing pipeline

To recognize the whole image semantics, as with the object detection and classification module, we took the same approach by using an Inception V3 model pre-trained on the ImageNet dataset and fine-tuned with 3-scaled Open Image dataset. The class pool is 500 after filtering once again from Freebase Entities and Google KG.

The visual pipeline uses a two-stage mapping for parsing the concepts from the aforementioned Freebase Entities and Google KG. In the first stage, we map from 500 concepts to a 220 visual type pool defined by Columbia-RPI-CMU, which later on in the second stage, we map the 220 visual types to the AIDA ontology. This final mapping is manually defined.

Our visual pipeline also includes an optical character recognition (OCR) module that is not currently being used. For this module we adopt an end-to-end two-stage OCR model to extract characters or text in an image. The model has a text-alignment layer is proposed to extract sequence features within a detected quadrilateral of multi-orientation. A character attention layer is applied to for decoding. At the time, this model achieved state-of-the-art performance on the ICDAR2013 and ICDAR2015 challenges.

Finally, the visual pipeline can detect faces and re-identify them based on a reference database formed from Google Images queries. The reference database is built by downloading the top images from Google Images of the persons found in a list of domain names detected by our system's Named Entity Recognizer (NER) on domain text. The faces are then detected on the images and video keyframes by using an MT-CNN model pre-trained on the Menpo dataset and recognized by processing the images through Oxford's VGGFace2 model, from which we obtain a vector representation of the faces. Face similarity is obtained by calculating the L2 distance among the vector. To recognize the faces, we use a nearest-neighbor metric.

In a 10% randomly selected subsample of last year's LDC2018E52 AIDA Scenario 1 Seeding Corpus testing data we found that the face recognizer achieves an F1 score = 0.827.

As the program progressed, we began utilizing Microsoft celebrity detection API, which was effective with heads of state and other prominent individuals. The initial pipeline remains the same, but now we can retrieve a second vote for certain individuals.

3.1.4.3 Charts

A new phenomenon in the COVID space was chart processing. Charts contain information ideally expressed in tables, but not available to us in that form as the bars and labels are more intuitive for humans. We implemented the chart-to-text work of Obied and Hoque (2020) to run on our domain. This provides summaries of the charts to be used as additional documents within the broader system. Note, that this does not handle general visual reasoning as we were initially assuming would be required in this domain (hence QA work), but relies very heavily on OCR style extractions from the images. This follows the general paradigm of images corroborating rather than supplementing the knowledge found in text.

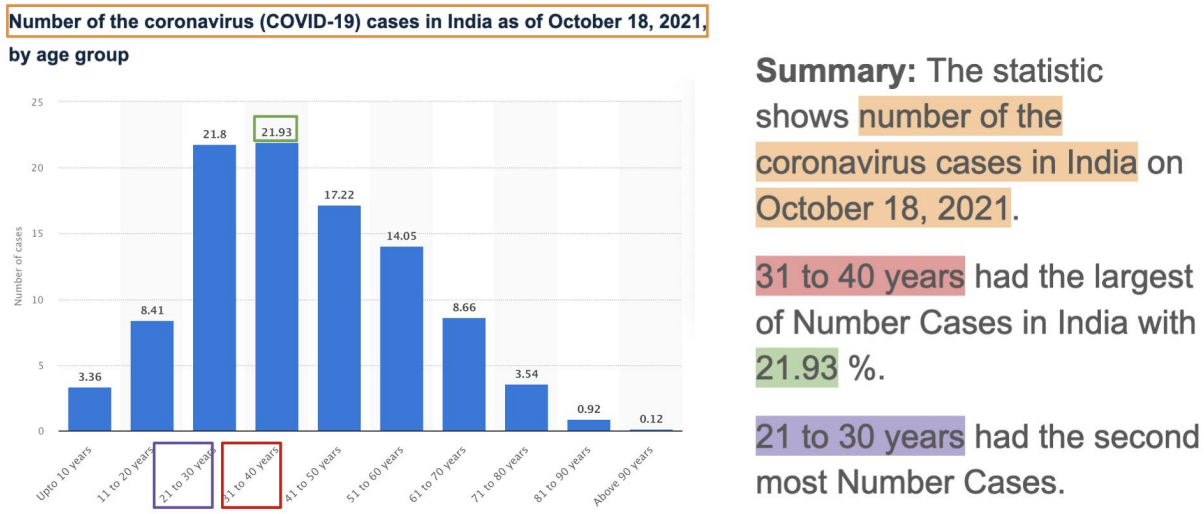

Figure 3: Example Chart Parse for producing a document from the key facts and trends

Once the summary is produced, it can be used as a standard document by the text-based pipelines in our system.

3.1.4.4 Extraction of Open-Domain and Novel Concepts

To extract events and relevant entities in images, our primary pipeline assumes ResNet pretrained backbones and detectors. Given the caption "***Medical personnel wearing personal protective equipment*** out of concern for the coronavirus remove ***a person from an ambulance*** near an entrance to Massachusetts General Hospital, in Boston" the first phrase grounds to the left heat map while the second to the right. Note that while generally accurate, a number of issues arise related to precision and overly broad categorizations. This is a natural result of receptive fields, even when cropped more narrowly (a secondary process). A second issue of concern to what extent the pretrained detector is limiting the regions of interest. The detector, as is standard, is pretrained on ImageNet categories – which are not reflected here (i.e. there is no label for PPE or most verbs). We therefore explored a novel patch based vision-language alignment model trained using Masked AutoEncoders (MAE). Specifically, we extend the paradigm introduced by ViLT (Kim et al, 2021) which uses a combination of image-text matching and masked language modeling to a novel MAE initialization.

Figure 4: Example COVID image

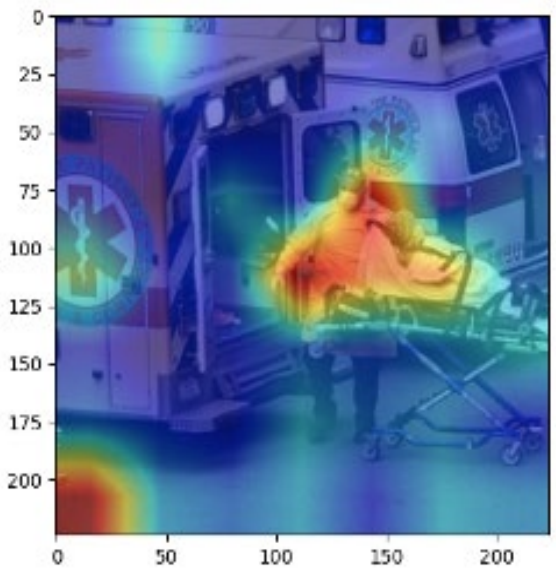

Figure 5: CNN Based Attribution

Patch visual transformers convert the image into small 16x16 or 32x32 pixel patches that take the place of visual tokens for which the model learns embeddings and cross-modal alignment. Where ViT trains the model discriminatively to predict a visual class, the MAE loss (Masked Image Modeling here) focuses on reconstruction of local patches so the patches retain local visual information and context. This provides better representations for the language to align with. This model exhibits a number of beneficial properties over our previous visualizations. First, the patches do not

require local coherence. In the aligned branch or leaves, individual patches can be isolated for more accurate predictions.

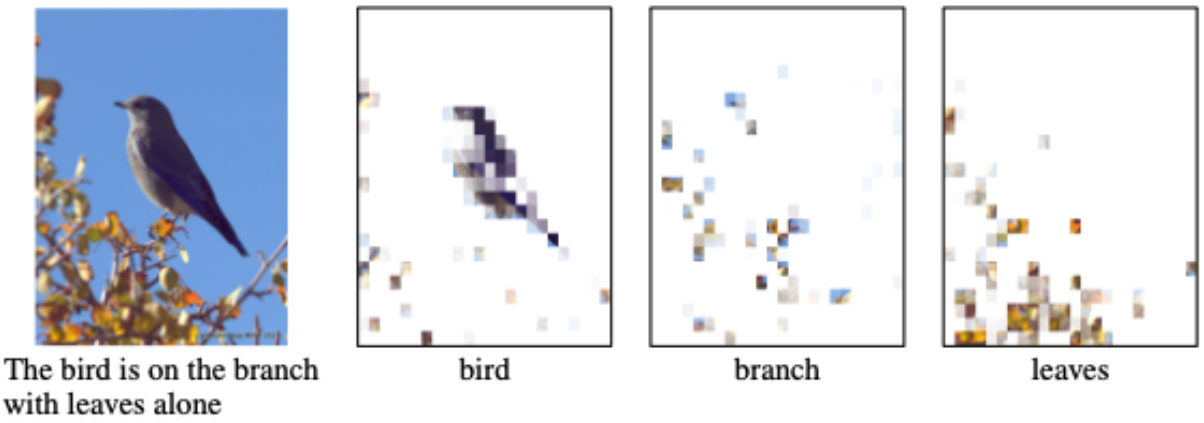

Figure 6: Example Text-to-Patch alignment

The secondary goal of a model trained directly on these alignments, is to provide a richer alignment model for large scale retrieval. Specifically, if a QNode contains an image, then entities in both the document and the qnode should ideally be linked via their visual appearance. To this end we introduce both an evaluation domain/test bed in WebQA (Chang et al 2022) and a novel Knowledge augmented transformer (Gui et al. 2022). We used CLIP (Radford et al, 2021) as part of our retrieval mechanism, and then take the corresponding WikiData as a document that can be integrated with a question for information extraction via GPT-3.

To begin building models that actually progress on knowledge based multimodal QA, we constructed a model which aggregates knowledge from detections, implicit model parameters, and explicit knowledge bases. Given a source image like the plane presented here, 1. a detection can be run for an ImageNet class (a very limited set). 2. A caption and detection tags can be passed to a large language model like GPT-3 to get "common sense" knowledge. Given a question like "What type of plan is this?", GPT-3 provides the most likely answer (e.g. jet). And 3. regions of the image can be presented to a retrieval mechanism like CLIP to compare against the multimodal content of a knowledge source like WikiData. CLIP believes that the most similar entities are the Avro Shackleton, MC-130 Hercules, and P-3B Orion – all military plans and two with substantial fuselages. Multimodal retrieval remains the primary bottleneck for advances. In Figure 6 we show how performance on the popular Outside Knowledge Visual Question Answering (OK-VQA) benchmark scales with the number of retrieved entities for two different backbones. Note, that it is not clear how much benefit can be gained from increasingly large pools if the retrieval model is not able to accurately identify the necessary sources.

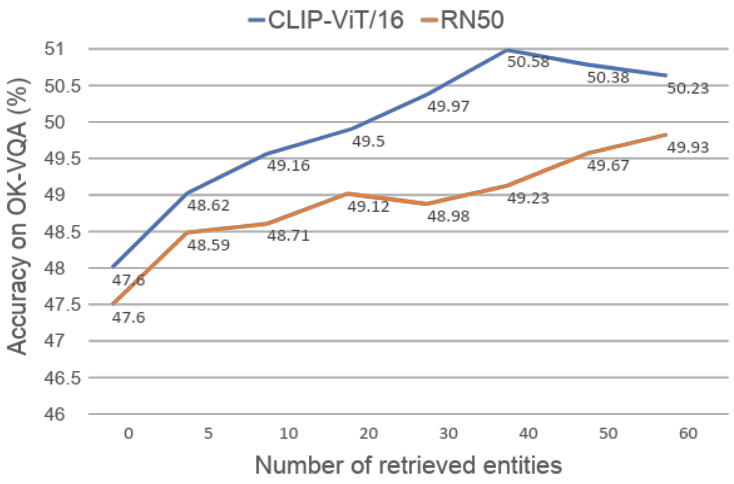

Figure 7: OK-VQA performance based on retrieval backbone

We introduced a new knowledge based benchmark to explore the strength/limitations of these techniques. An example query, answer, and set of sources from our WebQA resource are presented in Figure 7. Because this testbed contains ground truth source selection – the content required by humans to answer the question – we can evaluate the retrieval mechanism directly. We find that CLIP is a fundamental bottleneck due to its two unimodal encoders. Crossmodel encoders perform much better on retrieval tasks but are computationally infeasible to run at scale, O(n^2) as compared to O(2*n) for very large values of n.

Figure 8: Example open-domain cross-modal retrieval

This is further motivation for building better joint encoders, as noted above. We continued to adapt this patch alignment and began to see promising results on the image in Figure 4. Specifically, we can juxtapose Figure 9 with Figure 10. These figures contain the same image but now the model identifies different patches when we reference the entities that are medical personal in PPE versus when we reference the event of "removing". The latter focuses on the stretcher itself.

Figure 9: Patches for the phrase "*Medical personnel wearing personal protective equipment*"

Figure 10: Patches for "*remove a person from an ambulance*"

3.2 Inter-document KB aggregation

The goals of this component are to: (1) merge the individual per-document knowledge bases (mini-KBs) into a single graph via cross-document event and entity linking, and (2) to match knowledge extracted from a visual media document to knowledge extracted from text, and create co-reference links across documents in different media. We create and deploy technology to perform coreference of entity and event KEs drawn from different text documents (and hence also different lan-

guages). The basic approach is to represent each coreference chain as an embedding vector produced with BERT – initial versions relied on FastText. Rather than aggregating token representations, a single ``sentence" representation is built for the entire chain. For example:

[START] *Ukraine* *Ukraine government* *Ukraine* [END]

These vectors can then be clustered (agglomerative) to link mentions across documents. We further bias the clustering by leveraging the entity links in the KB (aka entities that known to be similar). Below we show that both FastText lexical embeddings and native BERT representations perform comparably, but FastText alone degrades quickly, and additional sentential context around entities was generally weaker. Though there may be a path to leveraging this information.

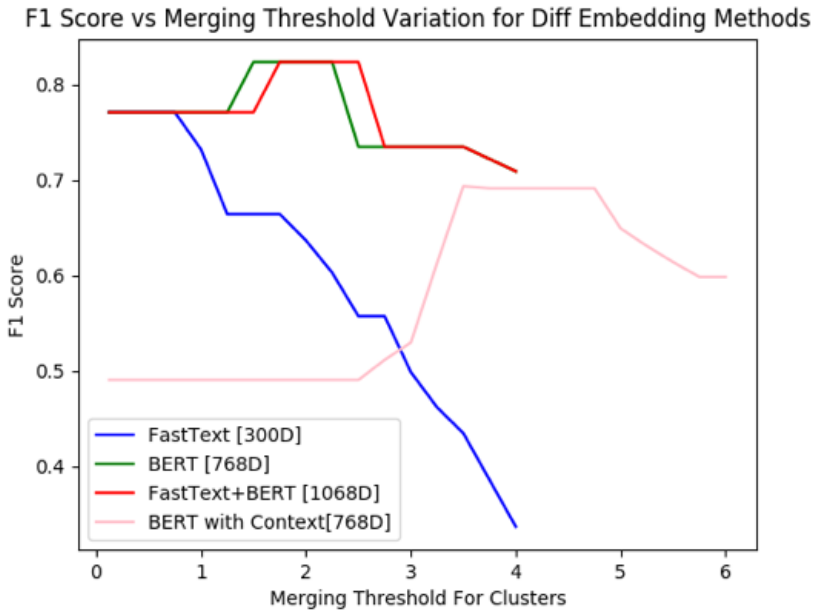

Figure 11: Coref performance using different embedding methods.

To confirm that we are not just doing sophisticated string matching, we performed a simple experiment. Instead of using FastText or BERT, we simply used a large 0/1 feature vector representing all character-trigrams in the mention chain:

[Ukrainian president Yanukovich] yesterday announced ...

—>{Ukr, kra, rai, ain, ini, nia, ian, an p, … }

This resulted in lower F1 scores than the learned methods, indicating they are capturing more than lexical overlap.

Efficiency A core challenge in this clustering process is the time complexity of execution. Every chain in every document is eligible for clustering, which explodes the computational requirements. Manual constraints are imposed to create smaller viable comparisons. Additionally, the merger of two entities also implies the merger of relations. The code must maintain bookkeeping for the prediction confidences and provenance of the original components.

Hyperparameters There is no ground truth clustering that can be determined at this stage. Instead, we can set intuitive guides for the entropy we would expect at different levels of the ontology or across sizes. For example, keeping the number of singletons low but also the cluster size low (low entropy). These are at odds as all singletons would be the perfect low entropy configuration for size but also not a useful clustering.

An additional note, is that while throughout the program a substantial number of new model architectures with increased scaling were introduced. We did not see noticeable differences from the basic BERT configuration. This lends evidence to the theory that the models are aggregators but not imbuing substantial prior or external knowledge to the final representations.

3.2.1 Listicles

One common problem is that many "documents" do not adhere to a traditional structure of prose with relevant context. In fact, "listicles" are a common resource, as they link broad categories of topics and entities. Unfortunately, this also creates false context as now entities that are adjacent to one another in the list are conflated inappropriately. A particularly nefarious version of this is when there are multiple events in the news which are discussed in a single article, perhaps as one is conceptually similar to the other, they are contemporaneous, or for editorial purposes. However, this provides evidence that two events should be linked. At a low-level this makes entity/event recognition fail due to complex coreferents. Afterwards, Once unrelated entities/events are grouped together under the same coreference cluster even in one or few documents, the errors propagate as the cross-document clustering is designed not to split within-document chains.

We identify that approximately 0.5% of documents contain this structure, though propagate widely. In articles which are constructed of bulleted lists, we may find 100s of events (e.g. assassinations) all become linked to a single cluster. These super clusters break the entire pipeline. To address this, we create a set of synthetic documents from a single document. This allows for the local coreference chains to process unperturbed. Specifically, we rely on a package "textsplit" https://github.com/chschock/textsplit. One concern, is that this approach does introduce yet another hyperparameter for thresholding the splits. The best approach we found was to nearly equate a paragraph with a document, therefore setting a "reasonable" threshold for local consistency. The goal is to avoid single sentence segmentations without merging where inappropriate. A similar balancing act to cluster entropy concerns.

3.3 Common Semantic Repository

The *Common Semantic Repository* provides a representation formalism, ontologies, repository, inference engine and APIs to store, access, map, disambiguate and link knowledge elements (KEs) generated by TA1 information extraction modules or entered directly by analysts.

3.3.1 OPERA Interchange Format

At the heart of the Common Semantic Repository (CSR) is the OPERA Interchange Format (OIF). OIF is a frame-based representation and interchange language based on JSON-LD[1] used by all components of the OPERA system. OIF uses an easy to read and manipulate JavaScript Object Notation (JSON) syntax, while at the same time being a legal Resource Description Framework (RDF) syntax which makes it easy to process and query with RDF/SPARQL engines like Blazegraph.

Picking the right data model and representation language at the beginning of a project is always a difficult decision, fortunately, OIF turned out to be a resounding success. The familiar JSON syntax allowed OPERA team members to easily ingest and generate data in OIF, even if they were not familiar with RDF-style data and knowledge representation (as was true for most of them), and without having to use any libraries which are often somewhat arcane and difficult to use. Nevertheless, since JSON-LD is really just another RDF syntax, all the tools from the RDF ecosystem were immediately applicable. For example, it was easy to write and execute complex structured queries over large sets of OIF documents using a triple store such as Blazegraph. OIF also stood the test of time and could adapt to new representation requirements as they emerged over the course of the project (for example, to represent TA3 claim frames as needed during the last phase of AIDA).

A main design goal of our language was to remain as light-weight and readable as possible. We felt that an interchange language should tread lightly in terms of representational commitment to not force the CSRs and other producers and consumers into any specific direction. It should primarily allow components to tell each other what they need to know, without biasing them too much on how to represent that information internally.

We also proposed OIF as a candidate for the program-wide interchange language but lost out to an RDF-based representation based on multi-layered annotation graphs which eventually became the AIDA Interchange Format (AIF). AIF relies heavily on reification to achieve its goals which led to very verbose and complex representations that plagued performers and evaluators throughout the program. We strongly believe that a simpler, JSON-based representation such as OIF would have been a better choice.

3.3.1.1 Framework

OIF's basic representational structures are *frames* represented as nested key/value lists in JSON-LD. Using JSON syntax allows us to easily represent nested structures and to associate confidences and meta-information with minimal reification. Using JSON-**LD** further enables simple definition of name spaces and datatypes plus preserving relatively easy translation to traditional RDF while still maintaining our readability goal. There is a large amount of tool support available for both JSON and also JSON-LD, but even without that, it is very easy to write simple readers and generators. The same is not true for RDF in any of its standard notations.

We distinguish between the following core frame types in OIF:

- **Evidence frames** encode pieces of source material and corresponding probabilistic interpretations from a single extraction software component

[1] json-ld.org

- **Instance frames** group coreferent evidence frames into knowledge elements (KEs) such as entities, events and relations that cluster coreferent mentions into a knowledge graph
- **Hypothesis frames** are sets of semantically coherent frames relevant to some topic or higher-level narrative of interest
- **Auxiliary frames** are document objects that describe some aspects of the hierarchical structure of source documents, frame collections and other OPERA-internal frames

OIF frame objects are generally divided into a meta-information section with an object ID, document provenance, engine provenance, document extent, etc., and an interpretation section that describes the higher-level semantics of the object. The latter will often be soft or probabilistic with multiple interpretations possible.

JSON-LD objects use context definitions (indicated by a `@context` keyword) which control the interpretation of elements in the body of an object. Contexts are a powerful mechanism for defining how the body of a JSON-LD object gets translated into RDF, while retaining maximum brevity and readability of the JSON structures. For OIF we employ the following two contexts:

1. frames.jsonld describes the various object types defined and used by OIF
2. resources.jsonld defines a number of resource name space mappings for OIF data

Below we give an example for a mention-type object, since those are best-understood and have been used and formalized in many related efforts in the past. For a more complete overview of OIF frames the reader is referred to the appendix (4.1). Textual mentions are evidence provided in some source document for the (hypothetical) existence of an object of some type such as an entity. For this reason, we represent mention information in OIF with *evidence frames* which describe pieces of source material and corresponding probabilistic interpretations by a single extraction component. For example, entity mentions from a document are linked to their text span in a document, images are linked to an image file and a bounding box, etc.

Here is an example for an entity mention for the text "Ukraine" in one of the Bellingcat investigative stories on the crash of Malaysia Airlines flight 17. JSON and JSON-LD do not define a comment syntax, but for ease of exposition, we use a Python-style #-syntax below to annotate the meaning of certain fields.

```
{"@context": ["http://www.isi.edu/isd/LOOM/opera/jsonld-contexts/resources.jsonld",
        "http://www.isi.edu/isd/LOOM/opera/jsonld-contexts/ail/0.3/frames.jsonld"],
 # syntactic type of a JSON-LD object or subobject:
 "@type":    "entity_evidence",
 # string identifying the software component that created this object:
 "component": "opera.EDL",
 # a globally unique, readable object ID:
 "@id":      "data:ment-bellingcat-text-cmu-r2-11-2-4",

 # document extent relative to closest enclosing reference object:
 "provenance": {"@type": "text_span",
          "reference": "data:sent-bellingcat-cmu-r2-11-2",
          "start": 149, "length": 7,
          # surface text string of this mention:
          "text": "Ukraine",
          # sentence containing this mention:
```

```
            "parent_scope": "data:sent-bellingcat-cmu-r2-11-2"},

    # interp captures the meaning of this entity mention such as its type and referent;
    # values of interpretation slots can either be hard and unqualified, or softened
    # via confidence values or possible alternative interpretations:
    "interp": {"@type": "entity_evidence_interp",
            # type of the described entity from some agreed-upon vocabulary:
            "type":  "tac:GPE",
            # links to external databases and resources (Freebase, Wikipedia, etc.):
            "xrefs": [{"@type": "db_reference", "@id": "freebase:m.07t21"}],
            # link to frame representing the entity for this mention:
            "entity": "data:ent-bellingcat-cmu-r2-42",
            # part-of-speech:
            "pos":   "NAM"}
}
```

In the example a hard semantic type of tac:GPE was used (a geopolitical entity type taken from one of the early program ontologies), however, often we will be unsure about the correct type of a KE. To represent soft or alternative interpretations in OIF, we can wrap values in slot facet objects which contain the original hard value with some additional annotations such as a confidence or probability. Alternative interpretations can be captured via xor and similar facets. See the appendix (4.1) for more details on that.

3.3.2 RDF translation and querying

We use the rdflib Python package to translate OIF frames into RDF N-triples format which can then be loaded directly into our Blazegraph triple store. Each JSON dictionary object becomes an RDF subject URI based on its @id field or through introduction of a blank node if no explicit ID is provided. Key value pairs are then asserted as predicate/object triples about the respective subject ID. Lists are treated as sets by default and simply become multi-valued assertions, but mapping onto ordered lists is also possible through special context directives.

Once loaded, OIF frames can be queried with SPARQL as in the example below where we access entity mentions based on their extraction component and semantic type. The unnamed nested objects such as interp become blank nodes in the RDF translation, but they are hidden in the query below by using a path syntax such as ail:interp / rdf:type:

```
>>> bg.ppquery(
    """SELECT * WHERE
        { ?entity ail:component "opera.EDL" .
          ?entity ail:interp / rdf:type tac:GPE .
          ?entity ail:provenance / ail:text ?text .
        }""")

?entity=data:ment-bellingcat-text-cmu-r2-11-2-4          ?text=Ukraine
>>>
```

3.3.3 Knowledge Aggregator inference and integration toolkit

The Knowledge Aggregator (or KAgg) is a toolkit developed by USC/ISI to address a number of different tasks in the OPERA system. At its core it is based on a logic-based knowledge representation and reasoning system called PowerLoom[2] that provides an expressive predicate logic representation language, several deductive and abductive inference mechanisms, contextual and hypothetical inference, inference justifications and truth maintenance, and database integration. For OPERA PowerLoom was integrated with a triple store and graph database called Blazegraph[3] to support storage and querying of very-large-scale structured and heterogeneous data, while at the same time supporting PowerLoom's sophisticated logic-based inference. PowerLoom and Blazegraph are then wrapped in a set of Python tools that support data format translation, TA1 mini-KB generation, TA2 KB generation and various integration tasks such as cross-media linking. KAgg relies on TA1 extraction outputs as well as TA1 and TA2 cross-language and cross-media coreference and linking information provided by other OPERA components to assemble TA1 and TA2 KBs.

3.3.3.1 TA1 mini-KB generation

For this task KAgg accepts the TA1 engines' JSON output and stores the results into OPERA's central semantic repository (CSR) using Blazegraph database technology. Mini-KB generation produces consistent per-document KBs in OPERA OIF and TAC-KBP AIF formats for each document in the corpus. An important advantage of this scheme is scalability, since it allows us to use more expensive inferencing on a smaller, focused, per-document basis, which in addition can be performed in parallel, since documents can be processed independently. The disadvantage is that it prevents us from performing more fine-grained adjudication of conflicts when looking across documents. This phase takes entity, event and relation mentions together with equivalence information from within-document coreference and EDL links as input, and then links equivalent mentions into KE instances which form an initial raw knowledge base. However, once equivalences are introduced, type information from equivalent mentions starts propagating which can commonly lead to conflicts. To address this, all annotations coming from text extraction components are treated as separate *instance, type, relation* and *event hypotheses*. When a conflict is detected (e.g., an entity having both type PER and ORG), we aggregate the underlying evidence and adjudicate based on component provenance and confidence values. For example, a type inferred from a relation argument constraint is viewed as weaker evidence than a type predicted by an entity detector, even if they have the same confidence value, since relation detection is a more difficult task. Finally, the refined TA1 mini-KB is output as a set of instance and provenance frames in OPERA's OIF format.

3.3.3.2 TA2 KB generation

For this task, document-level TA1 mini-KBs are combined into a global raw KB which is then merged, refined and deconflicted. A main challenge here is scale, since we have to integrate and refine O(10,000) or more mini-KBs. For this reason, KAgg's TA2 KB generator performs most of its work with the help of the Blazegraph triple store. Since TA1 KBs are represented in OPERA's

[2] https://www.isi.edu/isd/LOOM/PowerLoom/

[3] https://www.blazegraph.com/

JSON-LD format (which is just another RDF syntax), they can be directly loaded into a Blazegraph instance. We also load cross-document coreference information produced by OPERA's TA2 clustering components. Now a large set of queries is run to retrieve KE instances and merge them based on the TA2 coreference information. During such merges type conflicts might emerge which at the moment are resolved strictly based on majority vote. Merging of entities might also lead to merging of relations and necessitates a host of other bookkeeping operations such as aggregation of confidences, propagation of provenance, merging of informative justifications, etc. At the end, a set of merged and deconflicted KE frames is output in OPERA JSON-LD format to represent the TA2 KB.

3.3.3.3 Translation and integration

To translate OPERA KBs into the required TAC-KBP AIF format (AIDA Interchange Format), KAgg provides a number of translators that are used for TA1, TA2 and TA3 data and result translation. KAgg also has a number of other tools to facilitate integration such as, for example, a multimedia linker that links visual and audio-extracted frames with text frames based on reference KB links, and a DBPedia tool to perform cross-language Wikipedia and Freebase linking.

3.3.3.4 Blazegraph backend

We experimented with a number of different database backend technologies to support storage and querying of very-large-scale structured and heterogeneous data, while at the same time allowing sophisticated logic-based inference provided by our PowerLoom KR&R system to be applied to this data. Some of the candidate systems we considered are listed here (a '*' means we had past experience from using those systems):

- **RDBMS**: MySQL*, Postgres*, Oracle*
- **NoSQL**: MongoDB, Neo4J, Elasticsearch, Lucene*, RethinkDB, Crate DB,...
- **RDF**: Fuseki*, Parliament*, GraphDB, STARDOG, Blazegraph, RDF4J, AllegroGraph,...
- **Other**: Dremio, Apache Drill, Google's Dremel, Vertica,…

After some initial analysis we took a closer look at Elasticsearch, Elasticsearch + Dremio and Blazegraph.[4]

Elasticsearch uses JSON-document oriented storage using Lucene which immediately fits our OPERA data model based on JSON-LD, it is easy to define mappings, has fast ingest, structured / unstructured and combined search and efficient data & index storage. Its biggest drawback was that it did not have good join-query support.

Dremio is based on a suite of Apache tools such as Calcite, Arrow and Drill. It has a very powerful SQL engine that allows one to run queries over any combination of Elasticsearch, HBase, MongoDB, Oracle, Postgres, Hive, S3, JSON, NAS, files, etc. (no RDF at the time we looked at it). Its biggest problem was fairly high latency, which resulted in O(100) ms query times for simple count queries, which stems from its focus on very large data lakes, parallelization, and cross-source data

[4] This analysis was conducted in 2018 and might not hold anymore for the most recent versions of these systems.

integration. This made it less suited for integration with PowerLoom inference which requires very nimble API calls and fast query times.

While there was no obvious winner among the technologies we investigated, our use of JSON-LD for the OPERA CSR and interchange language steered us towards a solution that fluently supported RDF. For this and other reasons, we eventually decided to use a triple store and graph database called Blazegraph.[5] Blazegraph has a number of features that make it well-suited for our purposes:

- RDF/SPARQL and Apache TinkerPop APIs
- Supports 50B edges on a single machine, 1T+ edges scale-out in a cluster
- Used by WikiMedia to implement the SPARQL endpoint for Wikidata, a large-scale knowledge graph with over 1.4B edges, various commercial clients, "allegedly" the basis for Amazon's Neptune graph database[6]
- REST API, direct-call Java-based SESAME API
- Can get close to "bare metal" for fast PowerLoom integration
- Fast ingest, lookups, querying
- 2 min to load 7.3M triples, 0.7ms random 2-step lookup on embedded server through direct-call Java API (using an SSD drive)
- JSON support through JSON-LD to RDF mapping
- Open source, GPLv2

After settling on Blazegraph, we wrote a Python library to support easy interaction, querying, experimentation and integration with Blazegraph.

PowerLoom / Blazegraph integration

We also built a full integration with PowerLoom that allows us to transparently map PowerLoom relations onto complex SPARQL queries that call out to Blazegraph and that transparently translate results between RDF and PowerLoom representations. In this way relations and rules can be defined that are used by PowerLoom inference just like other native logic rules, but that in fact do their work by calling out to the query engine of the underlying Blazegraph store. This integration has a number of important features:

- Combines PowerLoom inference capabilities with large-scale Blazegraph data storage and querying
- Allows application of PowerLoom ontologies and rules to arbitrary mix of Blazegraph and native PowerLoom data
- Enables KR&R functionality such as inference explanation and hypothetical, multi-contextual reasoning over triple-store data

[5] www.blazegraph.com

[6] https://en.wikipedia.org/wiki/Blazegraph

- Allows export of inference results to external components via materialization and Blazegraph API

The implementation uses PowerLoom's Python API to connect the C++ version of PowerLoom with Blazegraph's REST and direct-call Java APIs. It dynamically maps RDF URIs onto PowerLoom logic objects via namespace-to-module mappings. The following example shows how a Framenet frame Arrest qualified by a Framenet URI in RDF data is mapped onto a corresponding PowerLoom logic object qualified by a PowerLoom module:

```
http://framenet.icsi.berkeley.edu/1.5/Arrest  =>  FRAMENET/Arrest
```

These mappings allow assertions in RDF space and in PowerLoom-space to seamlessly interact, which means we can easily augment them, map them, add rules for certain types of inferences, etc. A new PowerLoom Blazegraph query specialist (a computed predicate) calls out to Blazegraph via SPARQL queries and then maps back results. Initial bindings to query variables are pushed down to the Blazegraph query engine for most efficient select queries. Memoization and caching is used for efficient query reuse during inference.

For example, here is a PowerLoom relation definition based on a rule that queries Blazegraph and seamlessly integrates that with type restrictions and normalization relations. Note that the variables ?link, ?type, etc. following the SPARQL query in the query-blazegraph clause might have initial bindings depending on where this is called within a query or inference tree. These bindings are used to instantiate the respective SPARQL query variables before the query is run to ensure it to be maximally selective:

```
(defrelation csr-coref-link-frame (?link ?type ?argtype ?arg ?comp ?score)
  :documentation "Select non-singleton CSR coref frames of ?type."
  :<<= (exists (?comp_ ?types)
       (and (rdbms/bind-as
            (setof aida/|event_coreference| aida/|entity_coreference|) ?types)
          (rdbms/query-blazegraph blazegraph
            "SELECT distinct ?link ?type ?comp_ ?score ?argtype ?arg
             WHERE { ?link rdf:type ail:relation_evidence .
                  ?link ail:component ?comp_ .
                  ?link ail:interp ?interp .
                  ?interp rdf:type ?types .
                  ?interp rdf:type ?type .
                  ?interp ail:score ?score .
                  ?interp ail:args ?arg_ .
                  ?arg_ rdf:type ?argtype .
                  ?arg_ ail:arg ?arg .
                  FILTER EXISTS {?interp ail:args ?arg2_ .
                           FILTER(?arg_ != ?arg2_) }
            }" ?types ?link ?type ?comp_ ?score ?argtype ?arg)
          (normalized-component ?comp_ ?comp))))
```

3.3.4 Chameleon 2.0: Integrating Neural and Symbolic Reasoning in PowerLoom

All AIDA tasks such as TA1 information extraction, TA2 knowledge base construction and TA3 hypothesis generation and management pose very significant technical challenges. AIDA systems addressing these tasks need to handle large and complex ontologies, representation of and inferencing with domain knowledge, noise and uncertainty, as well as handling of vector representations such as text or image embeddings. There is no system available today that "can do it all".

Our own PowerLoom knowledge representation and reasoning system which is at the core of the Knowledge Aggregator toolkit can already handle many of these challenges, but not all of them. By itself it is primarily focused on structured representations and logic-based inferencing. In this section we describe our work on Chameleon 2.0 which adds the missing pieces to PowerLoom to create this unified functionality.

PowerLoom's Chameleon 2.0 reasoner combines traditional symbolic reasoning in a first-order logic framework with neural network learning and inference to make PowerLoom's inference more general, flexible and robust. PowerLoom[7] is a logic-based knowledge representation and reasoning system that allows the representation of complex knowledge in a declarative, logic-based language and supports a variety of reasoning mechanisms to make implicit knowledge explicit. It has a query engine to retrieve asserted and logically implied statements from the knowledge base, provides persistent storage, a context and module system to organize large KBs, and has an extensive API for integration into other applications.

The basic representational units are predicates for types and relations taken from some ontology. Facts describe instances in terms of the ontology and rules specify relevant dependencies, constraints, computations, axioms, and so on. A knowledge base is then the sum of ontology plus facts plus rules. Logical inference makes implicit relations explicit. For example, Figure 12 shows how from a small base of facts and a simple domain rule we can infer the approximate location of some entity.

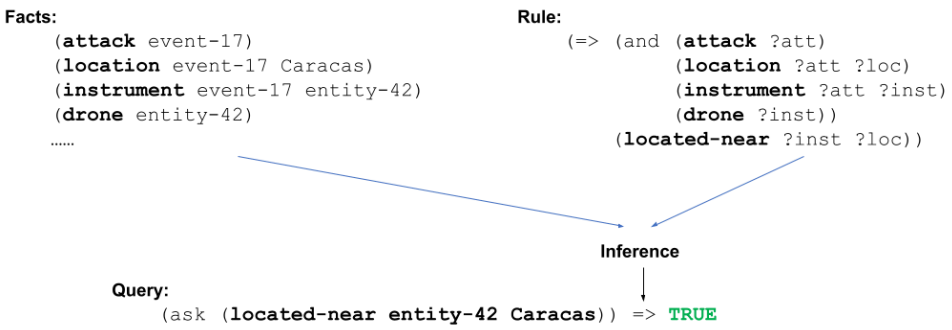

Figure 12: Rule-based logical inference in PowerLoom

While these constructs allow one to build very complex and sophisticated knowledge bases (e.g., (Lenat 1995)), a main and valid criticism of the approach has been its rigidity and brittleness along a number of dimensions:

1. Hard truth values: something is either true or false, there are no gradations
2. Complete preconditions: if just one of the preconditions in a rule cannot be satisfied, the rule cannot be applied
3. Fixed vocabulary: if an instance or relationship does not fit into any of the predefined types and relations, it cannot be represented

[7] http://www.isi.edu/isd/LOOM/PowerLoom/

4. Fixed, limited rule set: the rules are generally hand-coded and always incomplete, new rules are often difficult to add without disturbing other inference paths

There is a large body of research that tries to address these issues, for example, fuzzy logic (Zadeh 1975) and probabilistic logics such as MLNs and PSL (Richardson and Domingos 2006; Bach et al. 2017) allow truth values to be soft or probability estimates, non-deductive inference such as abduction can address missing preconditions (Stickel 1990), and ontology and rule learning approaches such as inductive logic programming can automatically learn or extend vocabulary and rule bases. These approaches are often focused primarily on probabilistic inference which can lead to inefficiency, since KB evaluation needs enumeration of large number of rule groundings. These systems also lack a lot of the machinery to build large knowledge-based systems such as query language, incremental updates, explanation, etc. They also generally do not handle embeddings.

PowerLoom's partial matcher is an abductive inference engine that supports soft truth values and that can handle unsatisfied preconditions. It has been applied to support debugging of large KBs (Chalupsky and Russ 2002) as well as for case-based reasoning (Moriarty 2000) and activity recognition from noisy data (Adibi et al. 2004). For example, if in the rule in Figure 12 only the first three conjuncts were satisfied, we might conclude the located-near relation with a score of 0.75 using a simple weighted average as shown in Figure 13.

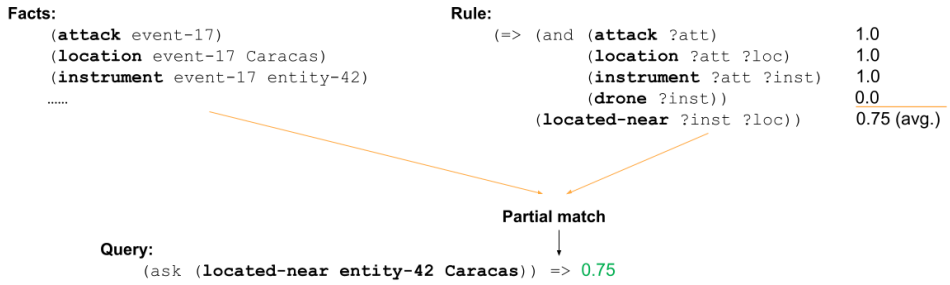

Figure 13: Partial-match inference in PowerLoom

To avoid such ad-hoc score computation schemes, Moriarty and McGregor (Moriarty and MacGregor 2000) developed the first version of Chameleon (1.0) which made these computations more principled and learnable from data. To do so, a neural network was associated with each rule which

would take the current soft truth values of the antecedent clauses as inputs and then computed a corresponding soft result value. Figure 14 shows how this is done for our running example.

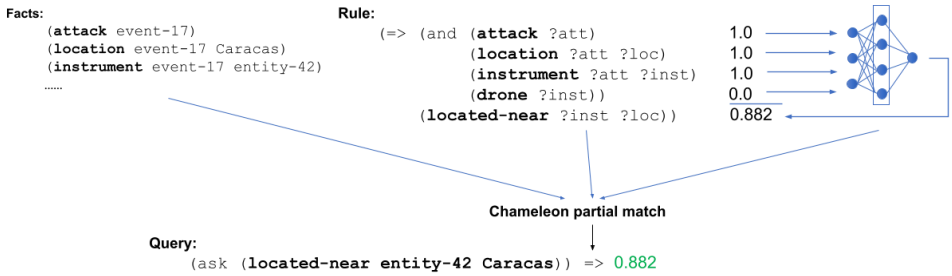

Figure 14: Partial-match inference with Chameleon 1.0

The networks were trained on possibly recursive partial match proof trees derived from training examples to minimize the total error over all examples. Back-propagation of error from one rule network to another was performed through the connections in the proof tree. Depending on the training examples, the networks would learn different weight combination semantics, thus the name Chameleon. Figure 15 shows a larger inference tree and how inference results change after some training has been performed. The dotted red line indicates error back-propagation.

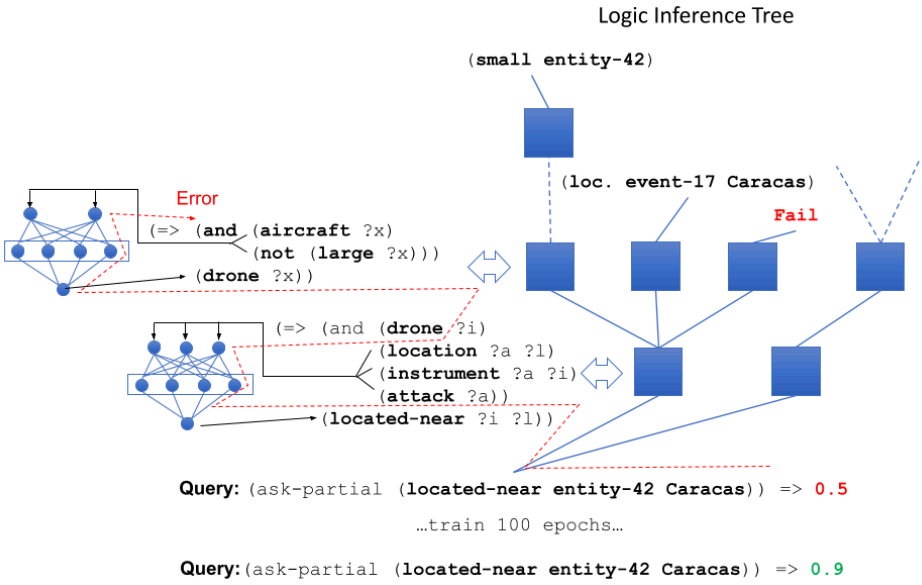

Figure 15: Chameleon 1.0 inference tree

Chameleon 1.0 provided a very significant neural-inference extension to PowerLoom long before *neuro-symbolic reasoning* became the buzz-word it is today. Nevertheless, it had a number of significant limitations:

- Soft truth values but "hard" vocabulary
- Learning at the type and predicate level only
- Fixed neural network structure and implementation
- Fixed rule-combination strategies
- No explanation

For Chameleon 2.0, we reimplemented and extended the existing Chameleon system along a number of dimensions. We particularly focused on the following aspects (not all of which are finished yet):

- Soften type / predicate vocabulary through embeddings
- Learning at instance level through embeddings and arbitrary instance vectors (e.g., images)
- Flexible neural network structure and inference through integration with TensorFlow
- Learn rule-combination strategies, e.g., to handle defaults with exceptions
- Provide explanation capability leveraging PowerLoom's proof tree explanation

3.3.4.1 Softening vocabulary through embeddings

Maybe the most significant restriction of Chameleon 1.0 was that network inputs are handled purely at the clause level. In our example rule above, the associated network's input 4 would simply consider the soft truth of (drone ?inst) but nothing about a specific instance such as entity-42 binding the variable ?inst. As such, vocabulary such as drone is still "hard", either satisfied or not, albeit softened through inference and weight computations. For Chameleon 2.0, we are introducing embedding relations that allow for rich high-dimensional similarity spaces that can be taken into account by rule networks. An overview of this approach is given in Figure 16.

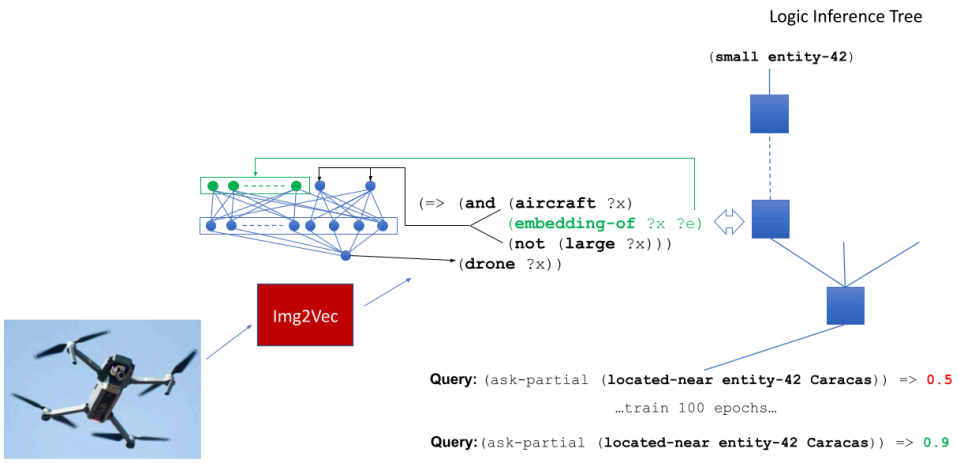

Figure 16: Chameleon 2.0 inference exploiting image embeddings

Here embedding-of is a user-defined PowerLoom relation that given an instance ?x can look up or compute an embedding vector ?e. When Chameleon encounters this rule for the first time, it looks up meta-information such as dimensionality and type of the embedding vector, as well as how to access the actual numeric information, and then builds the appropriate network with inputs activated by the embedding vector's dimensions in addition to the soft truth value information as before. Multiple embedding relations can be defined which can take one or more input arguments.

Embeddings might come from natural language resources such as Word2Vec (Mikolov et al. 2013), other media inputs such as images or video, or be computed directly from knowledge graphs. Arbitrary neural network vectors from a system such as Img2Vec can also be used (as illustrated in Figure 16). Meta-information about the particular embedding relation tells the network builder how to take such vectors into account.

Embedding relations are functions whose input arguments need to be defined to compute the embedding, which will be ensured by PowerLoom's clause optimizer. Moreover, even though they are antecedent clauses in the rule, their truth value will be ignored by the neural network machinery.

3.3.4.2 Integration with TensorFlow

Chameleon 1.0 was written long before the deep learning revolution of the last decade or so. Therefore, it used its own, hand-coded implementation of multi-layer perceptrons coded directly as part of the PowerLoom code base. To allow us to take advantage of all the latest and greatest neural network architectures, inference mechanisms and high-performance CPU and GPU computation, we integrated Chameleon with TensorFlow. The overall architecture of this integration is shown in Figure 17.

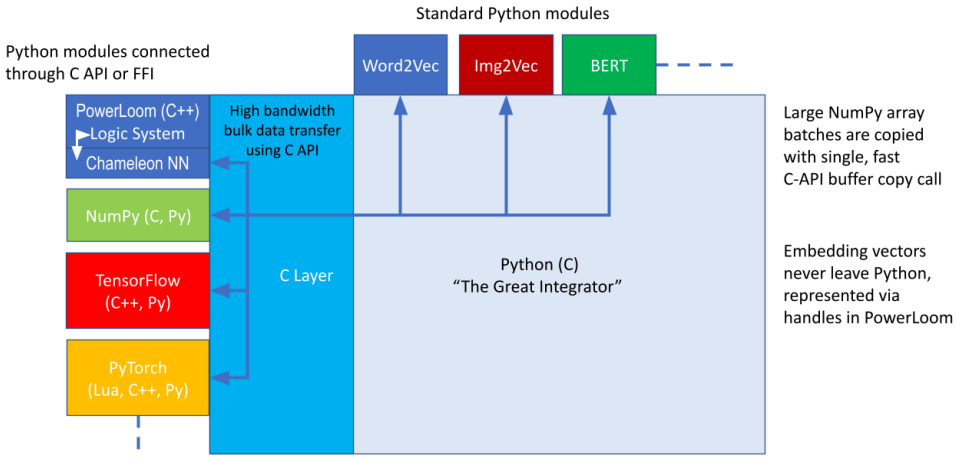

Figure 17: Chameleon 2.0 system architecture

Using TensorFlow allows us to leverage all the high-performance parallelism, GPU support and learning machinery provided by TensorFlow, but it also opens the door to construct arbitrarily

complex networks whose structure could be controlled not just by the shape of rules but by arbitrary logical and/or Chameleon inference.

Integrating a system such as TensorFlow with PowerLoom's inference engine has its challenges. One very significant one is that Chameleon inference operations and neural network computations are tightly interleaved. This leads to one network operation at a time in sequence which is very inefficient when using TensorFlow due to the significant overhead on each invocation. What we need instead are large batches of neural network operations, but such batches are not easy to come by. Our current solution is to use a queuing algorithm that exploits training example parallelism. Instead of one inference tree at a time, we propagate through many or all of them in parallel, exploiting that the same rule networks are invoked in different trees and places. When a rule network has all its inputs available, it adds those to its batch queue. When batch queues are full or nothing else can be done without propagating, batches execute in TensorFlow and dependent goals are notified of updates, and so on. This enables us to run efficiently on CPUs and GPUs as well.

3.3.4.3 Status and discussion

To date we have completed a first fully functioning version of Chameleon 2.0 that is integrated with TensorFlow and that performs neuro-symbolic inference in a tight integration between the various systems. We successfully tested Chameleon 2.0 on standard learning tasks and implemented demos that show a number of interesting capabilities (e.g., to perform multi-modal word sense disambiguation combining rules, text and image embeddings). Despite our best intentions, Chameleon 2.0 was in the end not being used by any of the systems we built for the various AIDA evaluations. Many other pressures and frequent changes in requirements made it difficult to focus on these more research-oriented aspects of our work.

Chameleon 2.0 is not yet finished and we are still working on a number of topics such as learnable rule combinations, explanation, integration with PyTorch, and others. For example, when a particular relation can be inferred by multiple rules, Chameleon 1.0 uses an ad-hoc combination strategy such as "max" or "noisy-or" to combine evidence from different rules. For 2.0 we are generalizing this to allow the system to use rule networks to learn how to combine multiple pieces of positive and negative support. This allows the system to learn, for example, how to handle conflicting rules such as "birds fly" but "penguins do not".

Another important topic is inference explanation. Neural networks are highly opaque and their inferences difficult to explain. Logic proofs on the other hand are structured and can be rendered into useful explanations. For Chameleon 2.0 we plan to extend PowerLoom's explanation machinery to additionally describe pertinent aspects of the neural networks' score computations, e.g., where they significantly deviate from a more standard logic-based interpretation of a rule. This is helped by the fact that we are dealing with a somewhat modular system of many small neural networks that are only sparsely connected and associated with small, generally "understandable" rules, compared to the case of a single, complex multi-layered neural network where everything is connected to everything else.

3.3.5 Transition to Wikidata

During the last phase of the AIDA program, the shared program ontology used by all performers was transitioned from the sizable yet limited annotation ontology developed primarily by LDC, to a DARPA version of Wikidata (or DWD). Wikidata (Vrandečić and Krötzsch 2014) is a popular, broad-coverage knowledge graph (KG) which contains circa 100 million entities described with

over 1.4 billion statements.[8] One of the reasons for the switch was the hope that a very large, broad-coverage ontology such as Wikidata, that is actively used and developed by a large world-wide user community, might survive beyond a single DARPA program and be reused in other efforts beyond AIDA.

Wikidata is a single KG that contains both a large ontology of entity, event and relation or property types, and a large number of *items* or instances described by these types. The ontology portion of Wikidata is quite large with over 2.5 million entity and event types and around 10,000 different property types. It is also not a true ontology, rather an open taxonomy developed by many independent editors that is both redundant and incomplete, and also contains major flaws such as terminological cycles. Nevertheless, the richness of the Wikidata ontology was a great value and provided new opportunities for information extraction systems to map unstructured content such as text onto a structured vocabulary.

Use of such a large ontology opened the new challenge that systems might pick many similar but equally valid types to classify entities, events and relations, which makes it more difficult to evaluate correctness relative to a gold standard or human assessment. For example, both `Q5` (human) and `Q215627` (person or being with personhood) and various other types might be chosen for something that previously was simply typed as `PER`. It was therefore assumed and mandated by program management from the start, that type "correctness" be evaluated by some automated means such as a similarity measure, rather than based on comparison to a gold standard or human judgment.

The change of ontology had two major impacts on our CSR software components:

1. Components actively using ontological information such as our KAgg TA1 mini-KB generator had to be adjusted to work with the new types and hierarchy information coming from Wikidata.
2. The very large scale of Wikidata demanded new methods to effectively interact with it during TA1, TA2 and TA3 processing.

The next sections give some details on how we addressed these issues.

3.3.5.1 Adapting KAgg for Wikidata

Any software that uses an ontology in some non-trivial way to exploit type hierarchies, relation argument types, type compatibility, disjointness or more complex domain rules, becomes intricately linked to that ontology. For this reason, swapping in one ontology for another is generally a non-trivial exercise that involves touching many places in the code and doing lots of redevelopment and retesting. Standard good software engineering practices such as modularization and abstraction of well-defined APIs do not apply in this context, since ontologies are generally large non-modular collections of highly interlinked definitions that do not have small, well-defined API surfaces. This means they cannot be easily swapped in and out compared to, for example, some authentication module that has a small, well-defined API.

Our KAgg module was no exception. Particularly for TA1 processing, it used a number of ontology-specific schemes to model entities, events and relations and to define type compatibility and

[8] https://grafana.wikimedia.org/d/000000175/wikidata-datamodel-statements

disjointness to filter noisy extractions. These representations all had to be adjusted or replaced due to the transition to Wikidata.

The initial step in our team-wide transition within OPERA was to have all our TA1 entity, event and relation extractors generate Wikidata types instead of the old types from the LDC annotation ontology. For some types this was fairly straight forwards and could be done using a mapping developed by the Cross-Program Ontology (XPO) committee. For others, models had to be re-trained to find new distinctions that were not relevant before.

This new vocabulary was quite complex and diverse. We observed over 6,000 different Wikidata type symbols in our TA1 dry run and evaluation data, and the underlying ontology structure supporting these types was complicated and difficult to exploit. Adapting our old methods of type-based checking and filtering was not an option given available resources. For this reason, we eliminated all that for Phase-3 and simply passed through Wikidata types, leaving only a handful of places where actual Wikidata type names surfaced in the code. Our AIF-to-OIF-to-AIF translators also had to be adjusted to support this new vocabulary, eventually allowing us to generate TA1 and TA2 KBs based on the new ontology as required.

3.3.5.2 PowerLoom / KGTK - Kypher integration

The second challenge from the transition to Wikidata was the unprecedented scale of the KG we had to work with, which was much larger than what we had addressed previously by using Blazegraph as the database backend for the CSR (see 2.2.4). Experience from working on the team of USC/ISI's KGTK project had shown that loading an RDF translation of Wikidata into Blazegraph can take 7-10 days on a high-end compute server, requiring about 1 TB of disk space. This was not an acceptable footprint for our AIDA evaluation pipelines. Alternatively, we could have used ISI's DWD Wikidata SPARQL endpoint, however, this resource was shared among AIDA performers and would have likely led to serious performance bottlenecks. While ISI's SPARQL service had a much larger query timeout compared to the public Wikidata endpoint, just a few bad queries by us or others could have brought the service to its knees which was too much of a risk to take.

Instead, we chose to go a different route and use a local installation of the Knowledge Graph Toolkit (KGTK) (Ilievski et al. 2020) with its Kypher query engine to host and query Wikidata. From separate work on the KGTK project (Chalupsky et al. 2021) we knew that this was a viable and much more efficient and reliable option to provide sophisticated access to DWD for our CSR than using RDF and Blazegraph.

To provide access to this local KGTK-based installation of DWD Wikidata, we built a full integration between PowerLoom and KGTK's knowledge graph data model and its Kypher query engine, somewhat similar to what we had done before for Blazegraph. This allowed us to transparently map PowerLoom relations onto complex Kypher queries that call out to KGTK and that transparently translate results between KGTK and PowerLoom representations. In this way relations and rules can be defined that are used by PowerLoom inference just like other native logic rules, but that in fact do their work by calling out to the Kypher query engine with its SQL-based graph cache mechanism to store large-scale knowledge graphs.

The Knowledge Graph Toolkit (KGTK) (Ilievski et al. 2020) is a comprehensive framework for the creation and exploitation of large hyper-relational KGs, designed for ease of use, scalability, and speed. KGTK represents KGs in tab-separated (TSV) files with four columns: edge-identifier,

head, edge-label, and tail. All KGTK commands consume and produce KGs represented in this format, so they can be composed into pipelines to perform complex transformations on KGs. KGTK provides a suite of import commands to import Wikidata, RDF and popular graph representations into the KGTK format. We exploited these tools to easily translate OPERA KBs in OIF format into KGTK format. This could be done more or less out of the box by using KAgg's existing OIF-to-RDF/NTriples format translator first and then using KGTK's import-ntriples command to translate further into KGTK format. All we had to do in addition was to handle the mapping of RDF namespaces onto short prefixes similar to the way prefixes can be used in RDF's Turtle format. For example:

```
http://framenet.icsi.berkeley.edu/1.5/Arrest => framenet:Arrest
```

Kypher (kgtk query) is one of 55 commands available in KGTK. Kypher stands for *KGTK Cypher* (Chalupsky et al. 2021). Cypher (Francis et al. 2018) is a declarative graph query language originally developed at Neo4j which uses an ASCII-art pattern language that makes it easy even for novices to express complex queries over graph data. Kypher adopts many aspects of Cypher's query language, but has some important differences. Most importantly, KGTK and therefore Kypher use a hyper-relational quad-based data model that explicitly represents edges as first-class objects which is more general than the labeled property graph (LPG) model used by Cypher. This is also an important feature when translating and querying AIDA KBs in KGTK format. To implement Kypher queries, they are translated into SQL and execute on SQLite, a lightweight file-based SQL database that is very efficient and scalable.

Kypher queries are designed to look and feel just like other file-based KGTK commands. They take tabular file data as input and produce tabular data as output. There are no servers and accounts to set up, and the user does not need to know that there is in fact a database used underneath to implement the queries. A cache mechanism makes multiple queries over the same KGTK files very efficient. In fact, the graph cache for DWD is "only" about 110 GB and can easily be installed and queried on a laptop. This is large but many times smaller than what we would have needed for a Blazegraph installation.

The integration between PowerLoom and KGTK / Kypher has a number of important features:

- Combines PowerLoom inference capabilities with Wikidata-scale data storage and querying
- Allows application of PowerLoom ontologies and rules to arbitrary mix of KGTK / Wikidata and native PowerLoom data
- Enables KR&R functionality such as inference explanation and hypothetical, multi-contextual reasoning over KGTK / Wikidata
- Allows export of inference results to external components via materialization and Kypher API

The implementation uses PowerLoom's Python API to connect the C++ version of PowerLoom with KGTK which is written in Python. Specifically, the KGTK Kypher API is used to call out to Kypher from PowerLoom's inference engine. The integration also dynamically maps namespace prefixes (originally translated from RDF namespace prefixes) onto PowerLoom logic objects via namespace-to-module mappings. The following example shows how a Framenet frame Arrest qualified by a Framenet prefix in KGTK data is mapped onto a corresponding PowerLoom logic object qualified by a PowerLoom module:

```
framenet:Arrest  =>  FRAMENET/Arrest
```

These mappings allow assertions in KGTK space and in PowerLoom-space to seamlessly interact, which means we can easily augment them, map them, add rules for certain types of inferences, etc. A new PowerLoom query specialist (a computed predicate) calls out to KGTK via Kypher queries and then maps back results. Initial bindings to query variables are pushed down to the Kypher query engine for most efficient select queries. Memoization and caching is used for efficient query reuse during inference.

For example, here is a PowerLoom relation definition based on a rule that queries DWD's p279star table, which is a precomputed table for fast test of sub/superclass relationships. In DWD Wikidata, P279 represents the sub/superclass relationship similar to RDF's rdfs:subClassOf. Note that the variables ?node1 and ?node2 following the Kypher query in the query-kypher clause might have initial bindings depending on where this is called within a query or inference tree. These bindings are used to instantiate the respective Kypher query variables before the query is run to ensure it to be maximally selective. In fact, in the definition below we enforce that at least one of the two node variables is bound, since otherwise the result set would be the whole table with close to 90 million entries:

```
(defrelation dwd-p279star-edge (?node1 ?node2)
  :<<= (and (or (bound-variables ?node1)
                (bound-variables ?node2))
            (rdbms/query-kypher dwddb
              "-i p279star
                 --match 'p279star: (x)-[]->(y)'
                 --where ' x=?node1 and y=?node2 '
                 --return 'x, y'"
              ?node1 ?node2)))
```

Then we can use these rules in PowerLoom queries or other rules to define a custom domain model based on DWD Wikidata. For example, here we retrieve five of the superclasses for Q121998 which is the occupation "ambassador". Note that these are just five random superclasses out of the full set of 61 superclasses for this type, they are not ordered by specificity here:

```
|= (retrieve 5 (and (dwd-p279star-edge "Q121998" ?super)
                   (dwd-english-label ?super ?label)))
There are 5 solutions so far:
#1: ?SUPER="Q103940464", ?LABEL="continuant"
#2: ?SUPER="Q1162163", ?LABEL="director"
#3: ?SUPER="Q1190554", ?LABEL="occurrence"
#4: ?SUPER="Q11997597", ?LABEL="representative"
#5: ?SUPER="Q1207505", ?LABEL="quality"
```

For scalability reasons, we map all DWD Wikidata identifiers onto strings and therefore no namespaces are shown in the results above. In the next query we access the top-level slots of a Phase-3 TA3 claim frame which does show some of the mapped namespaces. Note how nested objects such as the content of the claim frame point to (the KGTK representation of) anonymous blank nodes:

```
|= (retrieve all (?l ?v)
       (and (= ?claim DATA/|claim-L0C04958D-text-cmu-r202111202356-0|)
            (aidakb-kb-edge ?id ?claim ?l ?v)))
There are 9 solutions:
```

```
#1: ?L=/XMLNS/AIL/|claim_template|, ?V=/XMLNS/DATA/C322
#2: ?L=/XMLNS/AIL/|component|, ?V="opera.cf.qa"
#3: ?L=/XMLNS/AIL/|content|, ?V=|_:njPZiLDVEgkj8xoGXaFV27:c14n124|
#4: ?L=/XMLNS/AIL/|document|, ?V=/XMLNS/DATA/L0C04958D
#5: ?L=/XMLNS/AIL/|epistemic_status|, ?V="unknown"
#6: ?L=/XMLNS/AIL/|importance|, ?V=1
#7: ?L=/XMLNS/AIL/|label|, ?V="Pro-Kremlin outlets claimed the United States was behind the spread of
coronavirus in China."
#8: ?L=/XMLNS/AIL/|sentiment|, ?V="neutral-unknown"
#9: ?L=/XMLNS/RDF/|type|, ?V=/XMLNS/AIL/|claim|
```

For the final Phase-3 evaluation runs we built a KGTK version of DWD using the relevant DWD files provided in KGTK format by USC/ISI's KGTK project. Specifically, we used the claims, labels, p279star and pagerank graph files of DWD-v2 which is a DWD Wikidata snapshot taken on 2021/02/15. The resulting Kypher graph cache for this data was approximately 110 GB. Note that we get significant savings from only importing what we need, for example, we do not need to load references, descriptions, aliases, statement qualifiers, etc., allowing us to create a custom KG for our application that does not contain any unnecessary bloat. For each KAgg TA3 run, we also translate all of the TA1 mini-KBs used as input into KGTK format first and then load it into KGTK Kypher, which for the evaluation runs results in a graph cache of about 6.5 GB.

3.3.5.3 KGTK similarity service

We also employed the KGTK similarity service provided by USC/ISI's KGTK project for some of our KAgg TA3 processing. This service is very resource intensive. It needs about 400 GB of disk space to store all the required database and embedding files, and also needs a large-scale compute server to operate effectively. For this we decided to simply use its Web-API instead of trying to host it locally, since that would have been very expensive. This service was also less critical to our processing and less prone to unexpected crashes, since it did not execute arbitrary user queries. For this reason, using the Web-based API constituted a much smaller risk than using the DWD SPARQL endpoint.

Figure 18 shows the KGTK similarity GUI[9] with some example similarities for DWD QNode Q1930187 which represents the occupation of "journalist". The KGTK similarity service used by the GUI computes various different similarity measures that are either ontology-based such as Class or JC (for Jiang-Conrath), embedding-based such as ComplEx, TransE and Text, or aggregation measures such as TopSim. Each measure captures different aspects of similarity and leads

[9] https://kgtk.isi.edu/similarity

to different rankings which is why we chose TopSim for our work, since it aggregates the different measures along multiple dimensions for more robust similarity values.

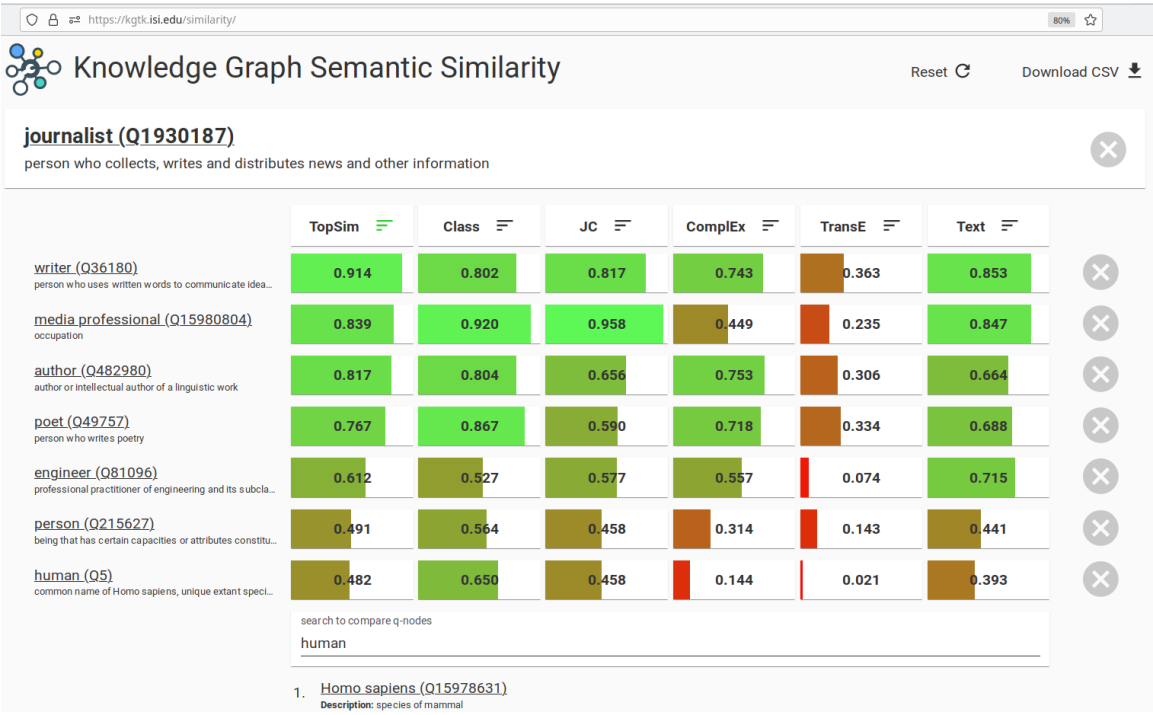

Figure 18: Exploring similarities to "journalist" in the KGTK similarity GUI

The various similarity measures shown in Figure 18 can also be accessed programmatically through a Web API.[10] For our KAgg TA3 processing we use this API in conjunction with our KGTK DWD backend to explore high-PageRank neighbors of a seed concept that have some minimal similarity to the seed for the purposes of query expansion. For example, here is a Python call that retrieves additional descriptors for QNode Q1930187 using these mechanisms:

```
>>> getQnodeDescriptors("Q1930187")
{'media professional', 'author', 'journalist'}
```

3.4 Hypothesis Generation and Management

3.4.1 Overview

USC/ISI's Knowledge Aggregator (or KAgg) toolkit initially focused heavily on the TA1 and TA2 areas of AIDA, but as the program progressed, it became more and more involved in the TA3 area of hypothesis generation as well. Initially, the main pathway within OPERA for creation of hypotheses was a system called *HypGen* which used a novel Belief Propagation system based on factor graphs to represent and reason with interpretation alternatives and larger hypotheses. Starting with that there were three distinct phases of KAgg development and application for TA3 which

[10] https://kgtk.isi.edu/similarity_api

stretched from a purely supportive role for HypGen, to a hybrid system to a purely KAgg-based solution:

1. HypGen-based hypothesis generation using KAgg support for hypothesis KB assembly and AIF-translation (Phase 1 Month 9, Month 18 and Phase 2 Month 36 evaluations)
2. Hybrid HypGen/KAgg-based document-centric hypothesis generation and refinement (Phase 3 Month 36 Post-evaluation Hackathon)
3. KAgg-based claim frame generation and refinement (Phase 3 Month 54 evaluation)

Below we describe each of these phases and systems in more detail together with relevant evaluation results.

3.4.2 KAgg support for HypGen-based hypothesis generation

Plausible hypotheses need to take into account the uncertainties in the underlying knowledge. To support generation, management and evaluation of hypotheses, the OPERA team at CMU developed LEAPFROG (Choudhary, Gershman, and Carbonell 2019; Hovy et al. 2018, 2019), a novel probabilistic Belief Propagation framework that uses factor graphs to represent interpretation alternatives and larger hypotheses concisely and effectively. It works over graphs of alternative probabilistic interpretations for entities, events and relations.

LEAPFROG used belief graphs as a complementary knowledge base in OPERA in all three stages: TA1 for mini-KB construction and its conditioning by an analyst's hypotheses, TA2 for coreference resolution, and TA3 for hypotheses construction. Given the output of LEAPFROG as a graph of weighted alternatives, the TA3 module accepts a query from the user or evaluator, extracts the subgraphs anchored at the query's entry point, and constructs the separate alternatives as hypotheses which are then further ranked and deduplicated. This TA3 system is called Hypothesis Generator or *HypGen*.

HypGen operates in two basic modes: (1) fully natively where it starts with the OPERA CSR coming from the various TA1 extraction components and then assembles TA3 hypotheses from TA1 and TA2 KBs that were built by LEAPFROG based on its probabilistic belief graph representation and inference. In this mode all inputs and outputs are represented in OPERA's OIF format and additionally embellished with belief graph information. (2) As part of a larger AIDA system pipeline where a TA2 KB is the sole input provided in AIF format lacking any LEAPFROG-specific belief graph information.

In both modes KAgg implemented and provided a number of important support functions that enabled the OPERA HypGen module to participate successfully in AIDA TA3 evaluations. We only give a brief summary of the various support functions here:

- translation of the AIDA domain ontology from RDF Turtle format to OPERA JSON-LD for use by HypGen
- generation and use of special-purpose TA2 provenance KBs stored in Blazegraph, so relevant provenance objects could be added to the hypotheses generated by HypGen without requiring HypGen to handle these large-scale KB objects directly
- generation of a type consistent LDC TA2 KB from LDC's manual TA1 annotations, which required a new event reclustering module (M18 eval only)

- various patching and reformatting of hypotheses to address specific evaluation requirements, data problems, type and role inconsistencies and AIF validation issues
- integration of PNNL's SheafBox as a hypothesis reranking component (M36 eval only)
- final translation of hypothesis KBs from OIF to AIF format

3.4.2.1 Phase 1 Month 18 evaluation

While all of the TA3 work in the initial phases of AIDA was performed at CMU, we give here a brief synopsis of the basic approach and evaluation results to provide context for later TA3-related work done at ISI. Given an AIDA TA2 KB created by OPERA or some other system (translated from AIF to OIF if necessary), HypGen basically builds hypotheses of interest bottom-up in the following way:

- Starting from a user-provided statement of information need (or *SIN*) which simulates an analyst's query, it tries to match SIN entry points to KEs in the TA2 KB based on reference KB IDs established by entity linking as well as relaxed string matching on names associated with KEs.
- Starting at entry point matches it then tries to find connected candidate event and relation matches for each role in a SIN frame.
- HypGen matching is extremely relaxed and generates event and relation candidate matches even if an entry point is not in requested role.
- It then exhaustively combines linked matches via cross product generation into core hypotheses (since for each role we might have multiple alternatives reported in the TA2 KB).
- It then prunes that set based on how well hypotheses overlap with a particular SIN frame.
- It then further grows core hypotheses with linked events & relations informed by belief graph inference (however, that aspect became somewhat obsolete in later evaluations, since belief graph information was either not generated by other performers or could not easily be passed through an AIDA TA2 KB).
- Finally, once a set of hypothesis candidates is available, a collective ranking algorithm using Maximum Marginal Relevance (MMR) (Carbonell and Goldstein 2017) ranks the candidates to optimize both goodness-of-fit to a SIN but also overall hypothesis diversity.

It is important to note that HypGen was optimized for recall. It became clear early on that given the high amount of noise in TA1 extractions which was further amplified when TA1 KBs were combined into TA2 KBs based on noisy coreference, it was extremely challenging to find *any* reasonable matches for the sometimes very complex SINs developed by the evaluation team. These problems were further amplified by a complex AIDA ontology and type system that might make two KEs look very different from each other based on type and role information, even though the underlying language giving rise to their extraction might have been very similar. For this reason, HypGen uses relaxed matching and back-off everywhere, otherwise, nothing relevant would have ever been found.

This very relaxed matching is evident in the results reported in Figure 19 - Figure 21 from three TA3 task variants of the Month 18 TA3 evaluation. Each result is from five different team submissions (only OPERA is identified) based on different types of TA2 KBs used as input. OPERA

results rank well for all three variants which are ranked based on coverage, that is how much (on average) reported hypotheses overlap with one of LDC's prevailing theories that summarize the topics of different scenario narratives AIDA systems are supposed to find in the evaluation document corpus (see (Dang 2019) for more details on the evaluation and its criteria). The high recall approach of HypGen generated high coverage results, even though correctness and coherence scores which are more precision-type measures are lower than for some of the other teams.

Hyps submitted	Theories matched	Correctness	Edge coherence	KE coherence	Relevance strict	Relevance lenient	Coverage	
24	6	0.4393	0.4834	0.6655	0.2832	0.6554	0.0320	
42	4	0.2607	0.2894	0.423	0.1343	0.4192	0.0127	OPERA
7	1	1.0000	1.0000	1.0000	1.0000	1.0000	0.0035	
2	1	0.4167	0.4167	1.0000	0.0000	1.0000	0.0032	
42	1	0.3864	0.4475	0.5851	0.3295	0.4836	0.0032	

Figure 19: Month 18 Evaluation Task 3a: using team's own or any TA2 KB

Hyps submitted	Theories matched	Correctness	Edge coherence	KE coherence	Relevance strict	Relevance lenient	Coverage	
34	2	0.1042	0.2178	0.3711	0.1002	0.3512	0.0079	OPERA
20	2	0.5107	0.6072	0.8145	0.3823	0.7973	0.0061	
7	1	1.0000	1.0000	1.0000	1.0000	1.0000	0.0035	
2	1	0.4167	0.4167	1.0000	0.0000	1.0000	0.0032	
42	1	0.3864	0.4475	0.5851	0.3295	0.4836	0.0032	

Figure 20: Month 18 Evaluation Task 3a: using other teams' TA2 KB

Hyps submitted	Theories matched	Correctness	Edge coherence	KE coherence	Relevance strict	Relevance lenient	Coverage	
42	2	0.5961	0.6249	0.8390	0.3829	0.8390	0.0238	OPERA
45	6	0.8065	0.8069	0.9589	0.6263	0.9589	0.0153	
42	4	0.8266	0.8612	0.8979	0.8312	0.9034	0.0100	
24	2	0.8207	0.8406	1.0000	0.5905	1.0000	0.0060	
29	1	0.8925	0.9063	1.0000	0.7421	1.0000	0.0051	

Figure 21: Month 18 Evaluation Task 3b: using LDC KB

3.4.2.2 Phase 2 Month 36 evaluation

For Phase 2 of the AIDA program, the evaluation domain shifted from the Ukraine-Russia conflict in 2014-15 to Crisis in Venezuela. This significantly affected the AIDA program ontology to handle new topics such as elections and disease outbreaks. Unfortunately, during this new phase the

main developer of HypGen left the CMU OPERA team, at which ISI took over as the main proprietor of the code base, responsible for any necessary adaptations and extensions which fell into the following main categories:

- Analyzing, refactoring, cleaning up and testing of the existing code so we could start extending and further developing it
- Adapting the code to various changes necessitated by the updated ontology and Month 36 evaluation plan
- Collaborating with PNNL's HypertThesis team to integrate their SheafBox system into a separate OPERA TA3 pipeline
- Full dockerization of HypGen and its various configuration and run scripts so it could run within the fully automated AIDA evaluation pipeline developed by the TA4 team
- Preparing our Chameleon reasoner for integration into HypGen

Since the original HypGen developer had left the project, we considered a number of options going forward, one of which was to start from scratch. However, since the module has quite complex functionality, this was really not an option given our available resources and other commitments for the evaluation. A lot of the original code was written under "evaluation duress" so code quality had suffered and a clean-up and refactoring was necessary before we could start to actually adapt and extend the code. This was a significant and difficult undertaking, since we had to make sure that we only fixed existing problems and did not introduce new ones or jeopardized the core functionality.

After this refactoring was completed, HypGen needed a number of updates particularly for the new ontology. Appropriate role abstractions for 577 event role types had to be defined, so those could be mapped onto the more abstract agent, patient, instrument, location and time roles assumed by HypGen. In the latest ontology events could serve as arguments to events and relations (e.g., to handle reporting and communications), which required appropriate generalizations in HypGen together with handling of zero-argument events which could now be arguments to events and relations. It also had to handle new temporal properties, generalizations to the schema for statements of information need (SINs), and various other changes.

Another significant effort in our preparation for the Month 36 evaluation was a collaboration with the PNNL HypertThesis team which was suggested to us by program management. Specifically, we investigated how we could use PNNL's SheafBox reasoner in our TA3 hypothesis generation pipeline. A core functionality of SheafBox is to partition a set of hypotheses into a set of maximally consistent subsets which are then ranked by a number of metrics they developed.

Given the complexity of such a collaboration both technically as well as from an engineering perspective, we initially looked to see if we could simply use SheafBox as a reranking component for hypotheses generated by HypGen. To that end we specified an API and data format for communication between HypGen and SheafBox, investigated how to communicate rules between the two systems in KIF (the Knowledge Interchange Format), and implemented various low-level adjustments to HypGen to address PNNL requirements such as linking entities and events to SIN variables, linking event and relation roles to their respective SIN edge IDs, and eliminating the renaming of KEs to make sure IDs match those from the underlying TA2 KB. Finally, we worked out a fully dockerized pipeline including SheafBox invocation for hypothesis reranking based on HypGen's

original hypotheses. This also required some other additions such as a hypothesis resizer to deal with some of SheafBox's scaling issues. In the end we decided to have two separate pipelines, one purely using HypGen and one with HypGen plus SheafBox, the architecture of which is shown in Figure 22.

Input and output translators translate between AIDA AIF format (RDF) and OPERA's OIF format (JSON-LD) which is also now what SheafBox expects and generates in this collaboration. TA2 Merge and recluster is needed to generate TA2 KBs with singleton entity and event clusters that aggregate information from cluster members (this is needed by HypGen). Finally, event clusters are split into type-consistent subclusters to avoid role heterogeneity.

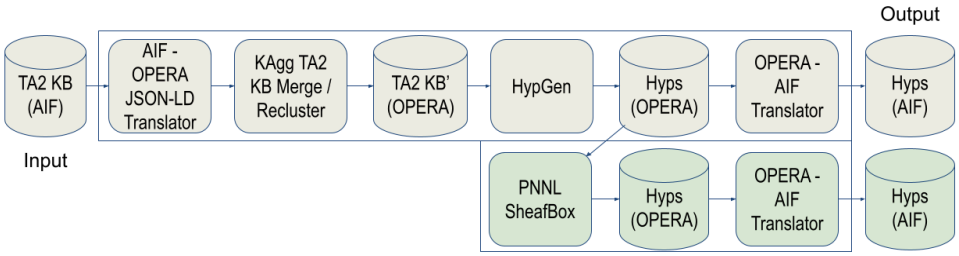

Figure 22: Month 36 Evaluation: OPERA TA3 fully dockerized pipelines

The Month 36 Evaluation required all AIDA systems to run fully automated in a docker-based evaluation infrastructure developed by the TA4 team. This was a very significant hurdle to climb, since it required a high level of "robustification" of various highly experimental code plus development of scripts that could reliably invoke that code in a docker environment. Testing of these dockerized components was also very challenging, particularly once they were transferred onto the external evaluation infrastructure where we could not immediately observe the errors that occurred and had to rely on the evaluation team to communicate any issues to us. Overall, while dockerization of end-to-end AIDA pipelines was an important goal of the AIDA program, it came with a very significant cost that preempted other important work.

One of the casualties of the extensive engineering requirements was our plan to integrate our new Chameleon 2.0 reasoner into HypGen, which now has become a goal for future work.

Figure 23 shows a summary of the Month 36 evaluation results for the TA3 task for various submissions using single team runs or combining the results of multiple teams in different stages. Given a structured SIN using the AIDA ontology that can be paraphrased as "Who killed Paola Andreina Ramírez Gómez on April 19, 2017 at The Mother of All Marches protest in Venezuela?", systems were tasked to derive internally consistent hypotheses related to this SIN. The evaluation used three high-level SINs which had several components like the example above. Systems were judged how well the returned results aligned with one of the previously formulated prevailing

theories formulated by the evaluation team (see TAC SM-KBP 2020 for more details on this evaluation.

Prec	Recall	F1	Run
0.3667	0.5789	0.4490	Team1_TA1_Team1_TA2_Team1_TA3_Relax
0.1667	0.2632	0.2041	Team2_TA1_Team2_TA2_Team2_TA3
0.1458	0.3333	0.2029	Team1+OPERA_TA1_OPERA_TA2_Team1_TA3
0.1379	0.3077	0.1905	Team1_TA1_OPERA_TA2_Team2_TA3
0.1000	0.1579	0.1224	Team1_TA1_Team1_TA2_Team1_TA3
0.0457	0.3810	0.0816	Team1_TA1tv_OPERA_TA2_Team3_TA3
0.0476	0.1053	0.0656	OPERA_TA1_OPERA_TA2_OPERA_TA3
0.0377	0.1905	0.0630	Team1_TA1_Team1_TA2_OPERA_TA3
0.0274	0.2857	0.0500	OPERA_TA1_Team4_TA2_Team3_TA3
0.0345	0.0526	0.0417	OPERA_TA1_OPERA_TA2_Team5_TA3

Figure 23: Month 36 Evaluation TA3: lenient micro-averaged F1

The two runs that used HypGen are marked in green. Overall the scores for this evaluation were quite low (even with *lenient* matching performed by the assessors), but they were particularly bad for Team OPERA. Post hoc analysis revealed a number of different reasons for this such as errors and bugs in our data (missing event arguments, bad links to the reference KB, issues when importing other teams' TA2 KB data, etc.). We also had deficiencies in our querying, for example, missing location inference that might have led us from "El Valle neighborhood" to "Caracas" to "Venezuela". Due to the restrictions on the number of runs teams could submit, one of the high-coverage runs we produced that combined OPERA TA1 data with that from another team could not get assessed.

Most importantly, however, our recall-oriented very-relaxed-matching approach was susceptible to lossy and low-precision TA2 KBs (note that it performed best in Month 18 on the high-precision manually constructed LDC KB). Particularly problematic were noisy event clusters that might come from "listicles" (articles that combine a long list of different stories into a single document which confuses TA1 coreference resolution), and then TA1 coreference cluster noise that gets amplified in TA2 KBs once such noisy clusters get linked to each other. The good news is that this led us to completely rethink our approach which led to the *document-centric* approach described in the next section, and which eventually improved our results quite dramatically.

3.4.3 Hybrid HypGen/KAgg-based document-centric hypothesis generation

A key insight from post hoc analysis of our Month 36 evaluation results was the following: When assembling hypotheses HypGen was often connecting KEs that were either not strongly enough linked, or they were linked due to some incorrect coreference between entities or events in the TA2 KB. HypGen often used such weak connections as a last resort to improve recall when no other connections could be found.

Figure 24 shows an example hypothesis for SIN E202D produced by HypGen and displayed by the OPERA hypothesis pretty printer. The architecture of the system is shown in Figure 25 which is just a different graphical view on the pipeline displayed in Figure 26 above.

The particular information need HypGen was trying to address was "Who killed Miguel Castillo Bracho on May 10, 2017 at the Our Shield is the Constitution protest?". This output is based on an OPERA TA2 KB over a combined OPERA + GAIA TA1 input dataset which was not assessed during the evaluation. The system did find KEs matching Miguel Castillo but no further linked events which is why he does not appear in this hypothesis. Instead, HypGen connected various unrelated death and demonstration events - all in Venezuela, that otherwise only matched on type to elements of the SIN but nothing else. Further filtering based on temporal constraints was also not possible, since the temporal information extracted by TA1 was too noisy.

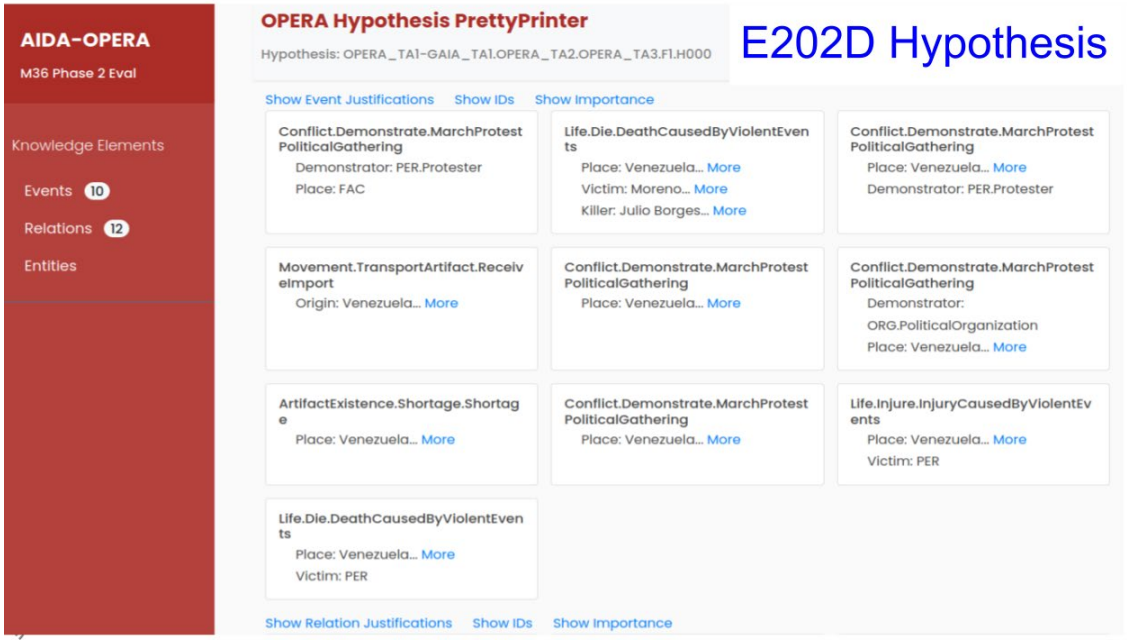

Figure 24: Example hypothesis for SIN E202D produced by OPERA TA3 System 1 at the Month 36 evaluation

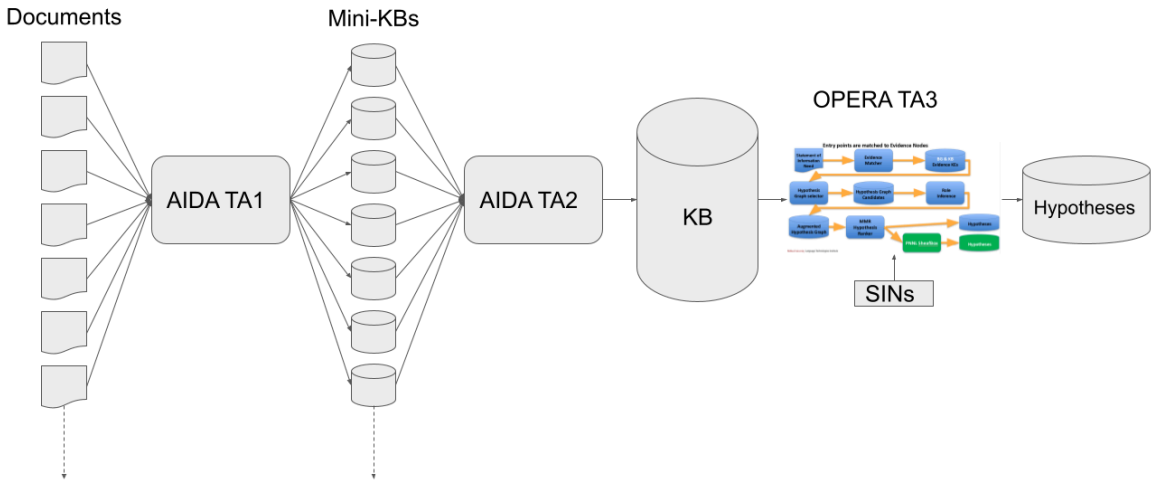

Figure 25: OPERA TA3 System 1 at the Month 36 evaluation

Another insight was that those unrelated pieces connected by HypGen came from many different documents, which naturally made it difficult to assemble a coherent "story". If we instead looked at events described in a single document, the descriptions and KEs extracted from it were naturally coherent and mostly relevant to each other. These difficulties were compounded by the AIDA *membrane*, which was a barrier introduced to prevent specific textual document information to flow from TA1 to TA2 and beyond to simulate security considerations an AIDA system would encounter in an operational setting. So, a system operating in TA2 KB space did really not have any strong TA1 document context to work with besides name strings and possibly embedding vectors.

The post-evaluation discussion of the subpar performance of all TA3 systems in the Month 36 evaluation led to what came to be called the *Month 36 Post-evaluation Hackathon*. The hackathon was an effort to give researchers the opportunity to explore possible solutions to these problems in a less structured and more free-flowing environment. One of the important outcomes of the initial discussions was that people wanted to "puncture the membrane" to allow more textual document context to flow between components, and program management eventually agreed to this request.

Given our error analysis from above and the new opportunity to use document text in a more liberal way, we decided to develop a new *document-centric* version of HypGen whose architecture is shown in Figure 26. The dotted red arrow going from documents all the way to the text-based merger illustrates how textual information can now flow through the opened membrane.

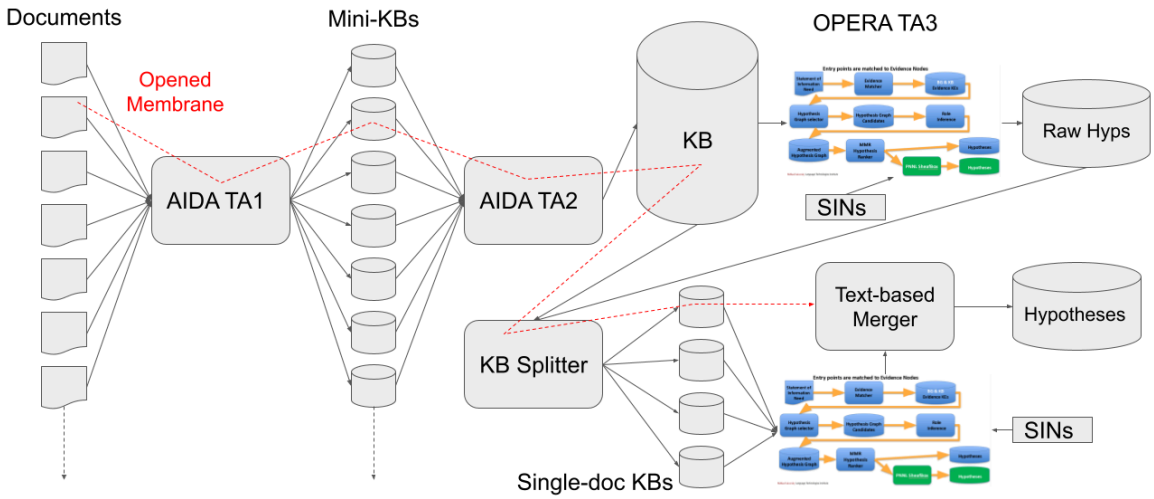

Figure 26: OPERA TA3 System 2 at AIDA M36 Post-eval Hackathon

Document-centric HypGen starts with the same pipeline as the original HypGen from Figure 27. However, the outcome this time is only a set of raw hypotheses. The KEs from all of these hypotheses are then combined and split into per-document buckets which are the single-document KBs output by the KB Splitter. These single-document KBs are somewhat similar to TA1 mini-KBs, however, they are much smaller and only contain KEs that were part of one of the raw hypotheses generated by the initial HypGen run. One could consider them a hypothesis-relevance-filtered set of KEs. The main reason for this somewhat convoluted architecture is that at the time of the hackathon, systems were still required to work from a TA2 KB only, and were not allowed to look directly at the TA1 mini-KBs that gave rise to the TA2 KB. Hence, we had to "reverse-engineer" a document-centric view of the TA2 KB first. This restriction was further relaxed in Phase 3 of the program which eventually made this unnecessary.

Next HypGen is run again for each SIN and for each single-document KB separately which might lead to a number of equivalent hypotheses which should be deduplicated or merged. Our next innovation was to perform this deduplication in *text space* instead of KB space which was now made possible due to the opening of the membrane. This solved another problem our and other systems had run into before: many of the returned hypotheses looked very similar or identical to assessors but were not recognized as such by the system. It turns out that finding true duplicates in KB space is quite difficult. Things that might look very similar at the surface text level might wind up looking quite different in the KB, for example, an event might have been classified by one system as an attack and by another system as a death event. Also, nominals such as "truck" led to unnamed KEs that are different from each other, and it is not clear when such entites can be considered the same. Even if we have two structurally identical events such as "person P killed somebody in country C", the prudent approach is to keep those events as separate, since we do not have enough information to consider them the same. It turns out that many of those problems go away if we compare textual descriptions of events, and once these descriptions are large enough, that method becomes robust enough for deduplication across documents. To do this we built a text-based merger that constructs textual descriptions of hypotheses (using machine translation to normalize non-English documents to English), and then merges hypotheses purely based on textual

similarity. Figure 27 shows an example textual description for a hypothesis addressing the same SIN E202D shown in Figure 24.

```
('K0C03BDKB', 'E202D', 'F1', '_0.json')
[[ 'GNB murdered Miguel Castillo Bracho in Las Mercedes this May 10',
    ('Instrument', 'firearm'),
    ('Killer', 'Polyclinic'),
    ('Place', 'The Mercedes'),
    ('Victim', 'Miguel Castillo Bracho')],
 [ 'May 10',
    ('EntityOrFiller', 'Miguel Castillo Bracho'),
    ('Measurement', '10')],
 [ 'May 10, 2017',
    ('EntityOrFiller', 'Miguel Castillo Bracho'),
    ('Measurement', '10')],
 [ 'Miguel Castillo Bracho, 27, was killed on Wednesday, May 10, during the repressed
    opposition protest on the highway',
    ('Demonstrator', 'Miguel Castillo Bracho'),
    ('Place', 'the highway')],
 [ 'Miguel Castillo Bracho, 27, was killed on Wednesday, May 10, during the repressed
    opposition protest on the highway',
    ('Place', 'the highway'),
    ('Victim', 'Miguel Castillo Bracho')],
 [ 'The boy was transferred to The Mercedes Polyclynic, where he arrived',
    ('Transporter', 'Miguel Castillo Bracho')]]
```

- Description generated by textual summarizer
- Correct parts underlined
- Noisy but **no unrelated clutter**

Figure 27: Example hypothesis for SIN E202D produced by textual summarizer of OPERA TA3 System 2 from the Month 36 Post-evaluation Hackathon

The main features of the Month-36 Post-evaluation Hackathon HypGen TA3 system can be summarized as follows:

- Uses newly available textual features for hypothesis clustering and duplicate detection
  - Summarizes events and relations based on their (machine-translated) textual provenance
  - Clusters events based on textual similarity
  - Case in point: Reduces E202D mostly duplicate hypotheses from 14 to 2
- Completely ignores TA2 cross-document coreference
  - Uses standard HypGen run to determine relevant document space
  - Re-runs HypGen against TA1-document-centric TA2 KB islands for each SIN
  - Deduplicates and merges hypotheses across runs using textual features
  - Retains within-document event coherence
  - Avoids noise blow-up from TA2 KB graph

An interesting and somewhat counter-intuitive conclusion from all this is that we can actually use text to normalize highly variable structural knowledge representations (KR) used in the TA2 KB. This is really the opposite from KR normalizing text, which is what we usually think of as the purpose of KR for NLP.

Figure 28 - Figure 30 show evaluation results from the evaluation rerun performed after the hackathon. For this rerun three sequestered SINs were used that were not shared with teams during the hackathon, so the results shown are true test results. Results marked in yellow (or green) with a postfix 'Rerun' are compared to results on the same SIN during the original Month 36 evaluation. Results for OPERA are marked in green and the improvements are quite significant, not just in relative performance across teams but also in terms of absolute F1 value. This demonstrates that document-centric hypothesis generation without use of TA2 is a successful strategy. This was also born out during the last Phase-3 TA3 evaluation which is described later in this report.

SIN	Prec	Recall	F1	Run
E201	0.6000	1.0000	0.7500	Team1+OPERA_TA1_Team4_TA2_Team3_TA3_Rerun
E201	0.5455	1.0000	0.7059	OPERA_TA1_OPERA_TA2_OPERA_TA3_Rerun
E201	0.4286	0.5000	0.4615	Team1_TA1_Team4_TA2_Team3_TA3_Rerun
E201	0.3750	0.5000	0.4286	Team1_TA1_OPERA_TA2_Team1_TA3_Rerun
E201	0.3750	0.5000	0.4286	Team1_TA1_Team1_TA2_Team1_TA3_Rerun
E201	0.1429	0.3333	0.2000	Team1to_TA1tv_Team1_TA2_OPERA_TA3
E201	0.0833	0.6667	0.1481	Team1tv_TA1_OPERA_TA2_Team3_TA3
E201	0.1250	0.1667	0.1429	Team1+OPERA_TA1_OPERA_TA2_Team1_TA3
E201	0.0000	0.0000	0.0000	OPERA_TA1_OPERA_TA2_OPERA_TA3
E201	0.0000	0.0000	0.0000	OPERA_TA1_Team4_TA2_Team3_TA3
E201	0.0000	0.0000	0.0000	Team1_TA1_Team1_TA2_Team1_TA3_im_relax
E201	0.0000	0.0000	0.0000	Team1_TA1_Team1_TA2_Team1_TA3_im

Figure 28: Month 36 Post-evaluation Hackathon TA3 rerun results: E201

SIN	Prec	Recall	F1	Run
E202C	0.5000	1.0000	0.6667	Team1_TA1_OPERA_TA2_Team1_TA3_Rerun
E202C	0.5000	1.0000	0.6667	Team1_TA1_Team1_TA2_Team1_TA3_Rerun
E202C	0.5000	1.0000	0.6667	Team1_TA1_Team1_TA2_Team1_TA3_im_relax
E202C	0.3636	1.0000	0.5333	OPERA_TA1_OPERA_TA2_OPERA_TA3_Rerun
E202C	0.2500	0.5000	0.3333	Team1_TA1_Team4_TA2_Team3_TA3_Rerun
E202C	0.2500	0.5000	0.3333	Team1+OPERA_TA1_OPERA_TA2_Team1_TA3
E202C	0.1429	1.0000	0.2500	Team1+OPERA_TA1_Team4_TA2_Team3_TA3_Rerun
E202C	0.1429	0.5000	0.2222	OPERA_TA1_OPERA_TA2_OPERA_TA3
E202C	0.1250	0.2500	0.1667	Team1_TA1_Team1_TA2_Team1_TA3_im
E202C	0.0000	0.0000	0.0000	OPERA_TA1_Team4_TA2_Team3_TA3
E202C	0.0000	0.0000	0.0000	Team1to_TA1tv_Team1_TA2_OPERA_TA3
E202C	0.0000	0.0000	0.0000	Team1tv_TA1_OPERA_TA2_Team3_TA3

Figure 29: Month 36 Post-evaluation Hackathon TA3 rerun results: E202C

SIN	Prec	Recall	F1	Run
E203	0.5000	1.0000	0.6667	OPERA_TA1_OPERA_TA2_OPERA_TA3_Rerun
E203	0.4286	1.0000	0.6000	Team1+OPERA_TA1_Team4_TA2_Team3_TA3_Rerun
E203	0.3750	1.0000	0.5455	Team1_TA1_Team1_TA2_Team1_TA3_im_relax
E203	0.3333	0.6667	0.4444	Team1_TA1_Team4_TA2_Team3_TA3_Rerun
E203	0.5000	0.3333	0.4000	Team1_TA1_Team1_TA2_Team1_TA3_Rerun
E203	0.2500	0.6667	0.3636	Team1+OPERA_TA1_OPERA_TA2_Team1_TA3
E203	0.2500	0.3333	0.2857	Team1_TA1_OPERA_TA2_Team1_TA3_Rerun
E203	0.1304	1.0000	0.2308	OPERA_TA1_Team4_TA2_Team3_TA3
E203	0.1250	0.3333	0.1818	Team1_TA1_Team1_TA2_Team1_TA3_im
E203	0.0400	0.6667	0.0755	Team1tv_TA1_OPERA_TA2_Team3_TA3
E203	0.0000	0.0000	0.0000	OPERA_TA1_OPERA_TA2_OPERA_TA3
E203	0.0000	0.0000	0.0000	Team1to_TA1tv_Team1_TA2_OPERA_TA3

Figure 30: Month 36 Post-evaluation Hackathon TA3 rerun results: E203

3.4.3.1 Hypothesis refinement

After the conclusion of the post-evaluation hackathon reruns, we implemented a number of additional refinement strategies to improve the precision of the generated hypotheses. These improvements were particularly targeted towards the AIDA Black Box Evaluations in the Venezuela domain where system outputs were presented to analysts to evaluate the usefulness of an AIDA system in a more realistic task setting. Specifically, we added the following refinements:

- Removal of garbage roles, which are event and relation roles for which the system cannot generate some non-trivial textual description.
- Removal of weakly connected or disconnected events and relations, which are those that do not have any argument connected to either a strong entry point or other well-connected KE.
- Eliminate fragmentary hypotheses, for example, events that only have a single, trivial role such as a location.
- Eliminate duplicates based on textual similarity of descriptions.
- Rerank filtered hypotheses based on SIN coverage.
- Add/improve handles, for example, replace empty handles with textual descriptions where possible (handles are text strings associated with KEs and their arguments that are used to generate readable textual descriptions of hypotheses for a user such as an analyst).
- Remove unused frames that are now dangling since we removed some components and roles.

These refinements filtered out about 75% of the original unrefined hypotheses that were returned by the OPERA system during the post-evaluation hackathon rerun, leaving a much cleaner, high-precision set of results to be presented to analysts. We do not have any formal evaluation results for these runs, since they were executed in a separate effort by the TA4 evaluation team without any direct involvement of performers.

3.4.4 KAgg-based claim frame generation and refinement

For the final Phase-3 of the AIDA program, the evaluation domain and process changed very significantly. The domain moved from Crisis in Venezuela to the COVID-19 pandemic, and instead

of corpus-level hypotheses, TA3 systems were now required to return *document-level claims* related to the domain, together with details such as precise semantics of a claim, claimer, location, time, claim medium, and so on.

Another very significant shift was from the AIDA program ontology developed mainly internally by LDC with some help from performers, to the use of a DARPA version of Wikidata (or DWD). Wikidata (Vrandečić and Krötzsch 2014) is a popular, broad-coverage knowledge graph (KG) which contains circa 100 million entities described with over 1.4 billion statements.[11] Use of such a large ontology opened the new challenge that systems might pick many similar but equally valid types to classify entites, events and relations, which makes it more difficult to evaluate correctness relative to a gold standard or human assessment.

In this new setup claim frames play the role of hypotheses, but it is now also important to establish the *source* of a claim in addition to what it means. Claims are now by definition *document centric* (something we explored with our post-evaluation hackathon system) and are extracted by TA1 systems, not TA3. The role of TA3 has shifted to claim clustering and deduplication, claim filtering, negation and epistemic status interpretation, and the detection of inter-claim relations such as "supports" or "refutes". TA3 does not anymore focus on claim assembly by querying TA2 KBs guided by SINs.

The combination of these factors made it clear that our HypGen system could not be reasonably retrofitted for this new task. HypGen focused primarily on bottom-up hypothesis assembly from low-level KEs in the TA2 KB, which was now not needed anymore. Furthermore, the AIDA program ontology was entrenched in its code and difficult to excise without breaking large amounts of code. For this reason, we decided to build a new TA3 system from scratch within the existing KAgg framework, using PowerLoom and newly available KGTK search and similarity tools (Ilievski et al. 2020, 2021; Chalupsky et al. 2021) to deal with the challenges of Wikidata, and reuse some of the text-based hypothesis deduplication ideas from the previous system.

[11] https://grafana.wikimedia.org/d/000000175/wikidata-datamodel-statements

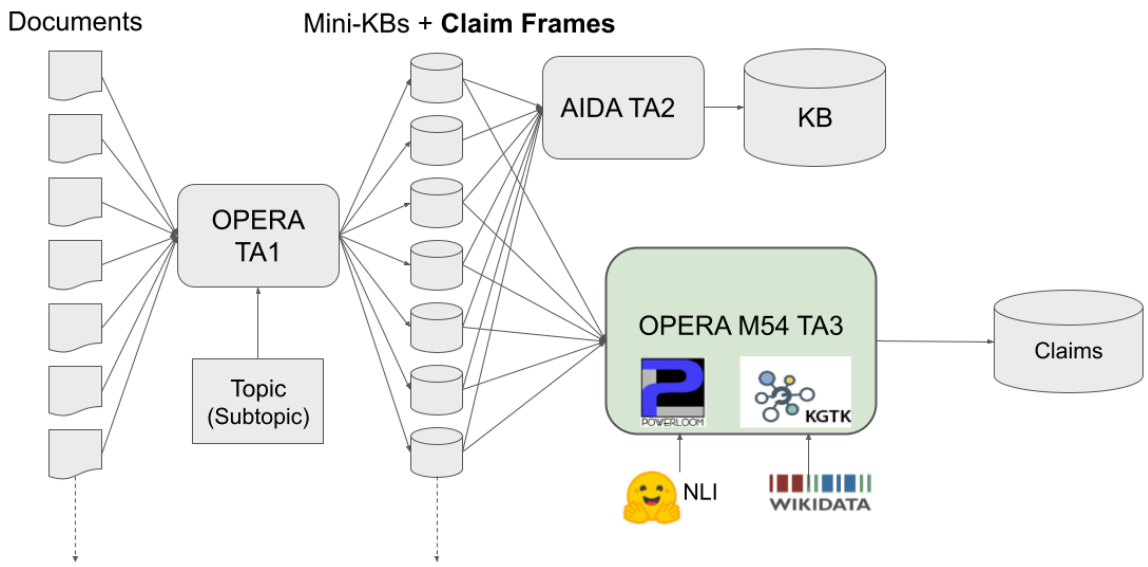

Figure 31: OPERA TA3 System 3 at Month 54 evaluation overview

Figure 31 gives a high-level overview of the architecture used by OPERA systems to address the Phase-3 evaluation tasks. Note that in this final evaluation there is no membrane at all. All TA1, TA2 and TA3 systems get access to all task-relevant inputs such as source documents, TA3 user queries, etc. Furthermore, the task is by definition document-centric, which allows us to go directly from TA1 to TA3 without any involvement of TA2 at all. We still output a TA2 KB, since that was an evaluation requirement, but we do not use that KB or TA2 KBs produced by other teams for any of our TA3 processing. Any necessary cross-document deduplication can be achieved purely based on TA1 entity linking, similarity and text-based inference.

End-to-end generation of claim frames goes through the following steps in OPERA:

- TA1 claim frame generation using question answering (QA)
- TA1 input data translation and import
- Topic-based querying and answer pool selection
- X-Value clustering
- Cluster ranking and within-cluster sorting
- Query expansion, relevance filtering and result generation

Figure 32 shows the various steps of our TA3 processing which we describe in more detail below.

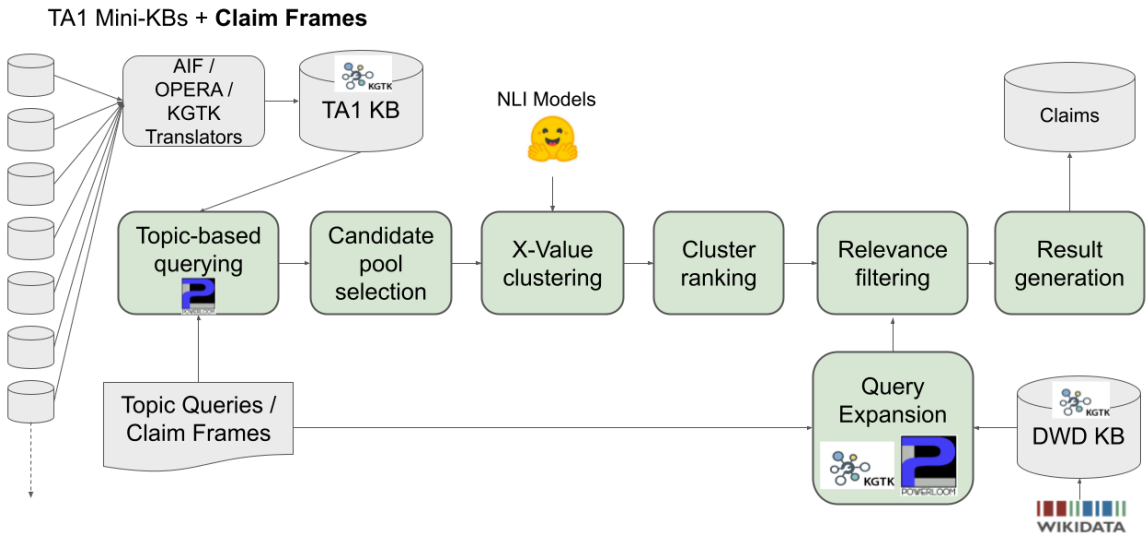

Figure 32: OPERA TA3 System 3 at Month 54 evaluation details

3.4.4.1 TA1 claim frame generation and translation

While this is not really part of the TA3 system, we give a very brief outline of the process here. More details can be found in the TA1 section of the report for the CMU team. Claims are generated by AIDA systems based on user queries that describe some topic/subtopic combination. For example, a user query with topic, subtopic and query template might look like this:

Topic: "Curing/Preventing/Destroying the Virus"
Subtopic: "Curing the virus"
Template: "X cures COVID-19"

The OPERA TA1 system built by the CMU team uses a question answering (QA) approach to find claims relevant to a user query by trying to find answers to the provided query template in the document corpus. Since query templates are not really questions but templatized abstractions of claims, it uses the following approach to first generate questions from the templates and then find relevant fillers for the X-variable slot:

- Convert templates into questions using a rule-based converter operating on dependency trees resulting from parsing the template.
  - Example: “X cures COVID-19” becomes “What cures COVID-19?”
- Use an off-the-shelf QA module to extract X variables and score the candidate answers.
  - Example: “What cures COVID-19?” might find “Lemon is effective against COVID” as a potential answer.

After mini-KB assembly through the KAgg TA1 component, a resulting claim frame might look like this in OIF. Note that it references and links to relevant entity and event KEs such as the claimer and the supporting event encapsulating the core semantics of the claim:

{ "@type": "claim",
"@id": "data:claim-L0C04959A-text-cmu-r202201220050-0",
"component": "opera.cf.qa",

```
"claimer": "data:entity-instance-L0C04959A-r202201220050-5",
"claimer_text": "Facebook Messenger",
   .....
"claim_template": {
   "@type": "claim_template",
   "topic": "Curing/Preventing/Destroying the Virus",
   "subtopic": "Destroying the virus",
   "template": "X destroys COVID-19",
   ..... },
"content": [{
   "claim_template": "data:C306_c8gyXauW6dVZ9Vh386Gkba",
   "value": ["data:entity-instance-L0C04959A-r202201220050-76"],
   "score": 0.9073742628097534,
   "text": "hot lemon juice",
   "support": ["data:event-instance-L0C04959A-r202201220050-3"],
   "support_text": "Facebook Messenger video falsely claims hot lemon juice can kill the coronavirus"
   .....
}],  ..... }
```


3.4.4.2 TA1 input data translation and import

The first step of claim frame generation is carried out purely within TA1 and the resulting mini-KBs containing KEs and claim frame candidates can come either from OPERA TA1 or some other AIDA TA1 system that can generate claim frames. If the former, no further translation into OPERA OIF is needed. If the latter we need an additional AIF-to-OIF translation step to convert mini-KBs into OPERA format.

To support this translation, we extended our AIF-to-OIF translator to also handle translation of claim frames represented in AIF format. We do this in two steps: (1) we first use our Java-based AIFConv translator to translate TA1 mini-KBs into OIF as done previously, simply ignoring any claim frames. (2) In a second step we use a new claim-frame focused translator added to KAgg to perform the remaining translations using the initially translated mini-KBs as a starting point. This separation was done purely based on pragmatic reasons, since extending AIFConv to also handle claim frames was deemed too time-consuming given that the original developer of AIFConv had left the project.

Next, we translate TA1 mini-KBs and claim frames as well as user query claim frames (provided by the evaluation team) into a uniform KGTK data model (Ilievski et al. 2020)[12] and import them into a KGTK Kypher graph cache database. This is a departure from our previous approach where we used an RDF triple store based on Blazegraph as the database backend for AIDA KBs. There is one primary reasons for this: We want to be able to transparently query and perform inference across TA1 mini-KBs and DWD (the DARPA version of Wikidata). DWD is very large, but using a KGTK translation and the KGTK toolkit it can be hosted and queried on a laptop (needing a 110GB graph cache DB). Staying with RDF would have required us to use a KGTK service or use a very large compute server, but either way, query times over Wikidata-size knowledge graphs would have been very slow (Chalupsky et al. 2021).

[12] https://kgtk.readthedocs.io/en/latest/data_model/

We do not use any TA2 KBs for OPERA TA3, so nothing needs to be translated or loaded in that respect. We do, however, exploit TA1 entity linking on named entities to link claim frames across document mini-KBs.

3.4.4.3 Month 54 evaluation conditions and ranking criteria

Before we describe the core claim frame generation components of our TA3 system, it is helpful to understand the tasks the system is trying to solve, and the evaluation criteria it is trying to optimize for. The Month 54 evaluation task was the following: given a user query, compute an importance-ranked list of claim frames relevant to the query which can be any of the following:

1. A fully instantiated claim frame on some topic and subtopic in AIF format (evaluation Condition 5).
2. A topic/subtopic pair in natural language with an associated template as shown in the example above (evaluation Condition 6).
3. A topic in natural language without subtopic and template, for example, "COVID-19 vaccine is causing people to die" (evaluation Condition 7). This last condition was considered a *stretch goal* and the most difficult.

Regardless of the type of query, the answer the system needed to provide was always a ranked list of claim frames. A second part of the answer was to compute inter-claim relationships such as "supports" or "refutes" or "related", however, that aspect was only assessed and evaluated for Condition 5.

The most difficult aspect to understand and optimize for were the ranking criteria for the resulting claim frames. Some of them were straight-forward such as claim frames had to be correct (that is based on correct TA1 extractions), relevant to the topic (and subtopic), complete (contain all required fields such as claimer and as many optional fields as possible), and they should not contain duplicates. The next consideration was *importance* which was some combination of a user-provided ranking of claim frame fields (e.g., that claimer was more important than claim medium) with the systems confidence in those fields it had extracted for a claim.

Finally, the most difficult aspect to optimize for was *diversity*. The claim frames returned by the system had to be significantly diverse and not just be minor variations along some dimension. This diversity could come from different values of the X-variable (e.g., "lemon juice" vs. "Remdesivir"), different epistemic status such as "lemon juice cures COVID" vs. "lemon juice doesn't cure COVID", different claimer, and differences in other aspects of the claim frame. While diversity of results is intuitively a valid goal to strive for, it was difficult to optimize for without any training data or gold standard. The problem was exacerbated by the fact that only a small number of returned claim frames would actually be assessed by LDC assessors (some number greater than 1 but much smaller than 50). So, whether one of those precious result slots should be allocated for "lemon juice" since that had a different epistemic status, vs. "Remdesivir" since that was a different treatment option, was not a clear-cut decision. In the end we implemented some ad hoc heuristics that we could not really test rigorously, but that seemed to have performed quite well nevertheless.

Full details of the evaluation conditions for the AIDA Phase-3 evaluation in Month 54 can be found in the NIST Phase-3 evaluation plan (Dang 2022).

3.4.4.4 Topic-based querying and answer pool selection

The first stage in our KAgg TA3 processing pipeline is a simple topic and subtopic-based querying of claim frame candidates. In this stage we try to establish a smaller pool of candidates which is then subject to further refinements. We select claim frames with matching topics (and subtopics), and filter out claim frames that are below a certain score threshold or lack any of the mandatory fields. This yields an initial list with $O(100)$ to $O(1000)$ results. We then rank this list by TA1 claim frame importance score, which basically ranks the claim frames the extraction system was most confident in first. From that we select a small pool of $O(100)$ high-scoring candidates which will be subject to further refinement.

Refinement is expensive, so we pick an initial pool size big enough to contain variety but small enough to not be too expensive to process (we basically pick a pool size several times the size of the result set we are aiming for). These initial filtering steps are very significant, for example, in the TA3 evaluation run on OPERA TA1 data, we started with over 30,000 extracted claim frames, of which only about 150 were reported in the final result.

3.4.4.5 X-Value clustering

One of the most important processing stages is X-value clustering which groups equivalent or similar claims based on the retrieved X-value for a claim template. Detecting such claims allows us to avoid redundancy, but also enables us to find supporting claims (e.g., same or entailing claim from a different claimer) or refuting claims (same or entailing claim with a different truth value or polarity).

When we look at the KEs instantiating X-values of claim frames, we find very high variability at the type level, particularly with a large and somewhat idiosyncratic ontology such as Wikidata. This makes it very difficult to cluster X-values purely based on type information. Instead, we use textual implication and equivalence on X-value descriptors determined by natural language inference (NLI) models.

In particular, we use the following off-the-shelf Hugging Face NLI models[13] and aggregate their entailment and similarity judgements for robustness:

roberta-large-mnli (entailment)
microsoft/deberta-xlarge-mnli (entailment)
bert-base-cased-finetuned-mrpc (paraphrase)

For example, here are some clusters the system derived for X-values of claims that match this template: “Federal government is diverting PPE etc. from X”:

{ the states / states / state authorities / states and hospitals / the state }
{ the Federal Emergency Management Agency / FEMA }
{ hospitals / doctors / the hospitals }
{ communities / homes / other municipalities / humans }
{ heavy-load helicopters / military aircrafts }

[13] https://huggingface.co/

It can easily be seen that these clusters are heterogeneous enough to make it difficult to declare them equivalent based on type information alone (for example, hospitals are organizations while doctors are people working at those hospitals), while they are similar enough at a textual level to make two claims just differing in these values equivalent or at least similar. This is another example of "text normalizing KR" as we already discussed in Section 3.3.

When finding clusters, the system is careful not to equivalence different instances, which are X-values that have been linked to different DWD entities by TA1 entity linking. Mapping instances to types is allowed, however.

Performing NLI inference with large neural models is expensive in CPU/GPU time, moreover, we have to perform $O(N^2)$ comparisons to find all potential equivalences in a set of N elements. For this reason, it is important that the pool of claim frame candidates for a query is reasonably small, otherwise computing these clusters can take a very long time.

3.4.4.6 Cluster ranking and within-cluster sorting

After a pool of candidate claim frames has been clustered, we score and rank the clusters so we know which clusters to focus on during result generation. We compute a cluster score based on the following:

- Size of the cluster (which uses mention frequency as a proxy for topic importance)
- Aggregated importance scores of each individual claim frame in the cluster
- Amplified by diversity (variance) of epistemic status values in the cluster

We then sort elements within clusters based on the following:

- Epistemic status, where certain truth and falsity rank above uncertain falsity and truth
- Importance of the claimer, e.g., named country over named organization over named person, however, this proved difficult and for now we just rank named over unnamed claimers

One interesting side-observation from implementing this was that classifying claimers into three groups of (named) country over (named) organization over (named) person using the Wikidata ontology proved to be quite involved. After collecting examples from training runs, it turned out that there were a lot of different types to consider, and that finding appropriate supertypes that could have been used for taxonomic inference was a significant challenge and required analysis of a large number of different paths in the ontology. Figure 33 presents a type cloud cartoon from this domain to illustrate the diversity and complexity. While this is certainly a solvable problem, we were pressed for time and decided on a simpler method instead. This is another example that using an ontology and particularly a large one like Wikidata comes with its own challenges and does not always immediately help.

Figure 33: Wikidata type cloud for claimers

3.4.4.7 Query expansion, relevance filtering and result generation

Next we list a number of additional steps needed to produce the final result. First, if we are given a full claim frame as a user query (as for Condition 5), we sometimes need to elaborate its natural language descriptors to facilitate textual inference via NLI. To do this we take the Wikidata reference and/or type of a KE, retrieve a small set of high-PageRank super classes as candidates using our KGTK DWD backend, then compute semantic similarity between the source node and each candidate using the KGTK similarity service[14] and threshold for minimum similarity. For example, here is a Python call that retrieves descriptors for QNode Q1930187:

```
>>> getQnodeDescriptors("Q1930187")
{'media professional', 'author', 'journalist'}
```

For Condition 5 we also compute the claim frame relation between the query claim frame and each result claim frame using textual entailment between the two claim frames and their epistemic statuses. For example, if claim frame A is textually equivalent to claim frame B, and A is claimed as true and B as false, we will say that B refutes A and A refutes B. When we generate results, we promote claim frames that are in an interesting relation to the query claim frame such as "supports" or "refutes", leaving simply related claim frames farther down the list.

For the final result we generate a ranking with claims that are important, relevant and diverse in values and epistemic status. To do that we go through the computed clusters in order of their importance scores, possibly reranked according to relevance to the query claim frame. For each cluster we select the most important members (representatives with differing epistemic status). If after the first round we still have available result slots, we start again from the top and backfill results up to a threshold.

3.5 Manual Annotation Efforts

We also participated in the manual annotation efforts for transitioning the ontology and domains. The first step in this process was learning a mapping between WikiData Qnodes and the existing LDC entities. For this we built a list of about 7,500 Q nodes for mapping.

[14] https://kgtk.isi.edu/similarity_api

We also began the process for annotation of claim frames by constructing minimal examples (aka simplified sentences) that should trigger all necessary components. We then evaluated how these align to existing resources (e.g. VerbNet, PropBank) and looked for places where concepts in the new COVID domain may not align to existed resources that predate the pandemic.

Full manual claim frame annotations were performed for comparison to other teams and provided a groundtruth for comparison to system predictions. Perhaps the most interesting result being not whether the system is able to find/build claims, but instead the selection of the most important or overarching claim of the document. In manual human annotation, a document clearly has a purpose with a single most important claim. This was not true for the automated approach, which can, and did, extract many claims, even if they are irrelevant or superfluous.

4.0 RESULTS AND DISCUSSION

While most individual components are benchmarked in their appropriate sections, we briefly discuss the final evaluation of our entire pipelines here.

Phase 3 Month 54 TA3 evaluation results

Figure 33 and Figure 34 present TA3 evaluation results from the Month 54 evaluation for our OPERA system and TA3 systems of other performers. Overall our OPERA KAgg TA3 claim frame generator did very well which can be seen in Figure 34 highlighted in green. When using TA1 extraction inputs from Team1, it scored best overall for 6 out of the 10 variants of the evaluation measure, being a close second only to another team using the same TA1 extractions as input. When running on our own OPERA TA1, performance was in the middle of the pack but still respectable. TA3 is dependent on the quality and completeness of claim frames extracted by TA1, and this dependency is visible here.

The evaluation measure used here is normalized discounted cumulative gain (NDCG) which is a ranking measure from information retrieval designed for evaluating the quality of a ranking (such as a ranked list of search results). Details on how NDCG was computed from TA3 results can be found in the evaluation plan (Dang 2022).

Pipeline (TA3.TA2.TA1)	Condition 5								Condition 6	
	Refuting		Supporting		Related		Ontopic		Ontopic	
	V1	V2	V1	V2	V1	V2	V1	V2	V1	V2
Team1.Team4.Team1	0.0271	0.0271	0	0	0.0233	0.0578	0.0957	0.1195	0.0604	0.1201
Team1.Team1.Team1	0.112	0.112	**0.1308**	**0.1308**	**0.0657**	0.1166	**0.1445**	0.1564	0.0858	0.1502
OPERA.none.OPERA	0.0644	0.0644	0.0247	0.0247	0.0537	0.0981	0.0764	0.138	0.0742	0.133
OPERA.none.Team1	**0.1488**	**0.1488**	0.0679	0.0679	0.065	**0.1407**	0.1103	**0.2322**	**0.1**	**0.2072**
Team2.none.Team2+OPERA+Team1	0.0872	0.0872	0.0727	0.0727	0.025	0.0582	0.0559	0.1193	0.0292	0.067
Team2.none.Team2+OPERA	0	0	0.0101	0.0101	0.0134	0.0285	0.0404	0.0844	0.0116	0.0357
Team3.Team4.Team1	0	0	0.0117	0.0117	0	0	0.0096	0.0216	0.0506	0.1241
Team3.Team1.Team1	0	0	0.0222	0.0222	0	0	0.0099	0.0224	0.0664	0.1439

Figure 34: Month 54 TA3 evaluation results – NDCG (normalized discounted cumulative gain)

The evaluation team computed a large number of different measures and variants to look at the results from different angles. We averaged over these various measures by ranking our results within the eight submissions for each measure, where Rank = 1 for the best score and Rank = 8 for the worst. The average over these ranks is reported in Figure 35, which again shows that our OPERA submission based on Team 1 TA1 did very well and ranked best for Condition 6 and close to best for Condition 5.

Pipeline (TA3.TA2.TA1)	Condition 5	Condition 6
OPERA.none.OPERA	3.46	3.5
OPERA.none.Team1	1.52	**1.00**

Figure 35: Month 54 TA3 evaluation results - average rank across measures

5.0 CONCLUSIONS

The core goals of AIDA are admirable and interesting, as it aims to operationalize something that is generally trivial to humans. There is synonymy and paraphrase at the lexical level, but these equivalences are more abstract (e.g. at the event level) from multiple views – perspectives in the language of the program. This multi-view representation problem is not unique to language, but present in many contexts or across modalities. For example, language, audio, perception, and action may be views or ways to express the same underlying concept – even for primitive objects or tasks. This is an intuitively simple statement, as we experience the world through audio and vision, participate in action, and communicate in language or text. So, if every event has this property and a unique perspective, why then does this program reduce to text processing? And why is it so difficult?

Let's start with multimodality. Any original event was multimodal. The assassination was an event in the real world, experienced first-hand by audiences. But we don't have access to that experience, that input. Presumably, analysts don't either. Instead, we have interpretations and noisy descriptions. The event was singular. The parties perhaps unambiguous. But the news written about it, was interpreted and contorted or contextualized. With each subsequent article, the facts become harder to ascertain. Clip-art and other pathos inducing images are used in leiu of the originals. Figures are made with selections of the original scientific data, and so forth. In effect, by the time AIDA gets access to documents, we (like an analyst or reader of the news) have a stack of splinters, which we are tasked with sifting through to reconstruct or infer the original. At this point, multimodal processing is all but useless, and the hypothesis formulation challenge immense.

The difficulty in constructing a system (from my perspective) comes conceptually from the lack of a world model, and practically from pipelines and error propagation. I think these both manifest as the core themes of our technologies. The easier case to discuss is the pipeline error propagation. If an entity is missed, perhaps due to coreference, or its slot is incorrectly assigned in semantic role labeling and event parsing. Substantial effort is required to try and recover from this error. This lead to lots of bottom up and top down discussions. Should extraction happen more locally to limite errors, but doing so pushes more burder on aggregation, and vice versa. What's training and guiding these systems is the label space or ontology? This leads to the KB design and the difficulty of aligning to those entities. The KB/ontology are the static and impoverished "world model" that the system needs to adopt. Let's discuss a simpler case first.

An embodied agent wants to clear the table. How should they progress? They have a model of which objects fit in one another, what platters are strong enough to hold things, and so forth. With each action they have a prediction, thanks to their world model, of how all objects will interact and from that they can work backwards and form a plan about how to achieve their goal. This planner, is also how we read the news. We establish what the writer claims is occurring and use our model to predict which pieces, tools, steps, pre-conditions, and so forth would be necessary for this to be true. The ontology and related resources stand-in for this. Telling us which events are related, which entites can fill what slots, and so forth. It is an impressive attempt to outline the dynamics of the world and society. We leverage complex analytical tools to process the language and try to decipher these pieces and unify references.

So, should we simply build an embodied agent to solve this? No, that's equivalent to saying we need to solve AI to to solve this task. But, I think understanding the differences between how

we would approach this and how our models have to, elucidates some of what the next steps should look like for this research agenda. Specifically, real-time multimodal event recognition and captioning of a live event. Models should be able to construct understanding of an event they are observing, and then reconcile those with the written descriptions provided in the news. This will build an anchor to a ground truth that links all the disparate surface forms they are observing.

6.0 REFERENCES

Adibi, J., H. Chalupsky, E. Melz, and A. Valente. 2004. "The KOJAK Group Finder: Connecting the Dots via Integrated Knowledge-Based and Statistical Reasoning." In *Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI-04)*, 800–807.

Bach, S.H., M. Broecheler, B. Huang, and L. Getoor. 2017. "Hinge-Loss Markov Random Fields and Probabilistic Soft Logic." *Journal of Machine Learning Research (JMLR)* 18 (109): 1–67.

Carbonell, J., and J. Goldstein. 2017. "The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries." *SIGIR Forum* 51 (2): 209–10.

Chalupsky, H., and T. Russ. 2002. "WhyNot: Debugging Failed Queries in Large Knowledge Bases." In *Proceedings of the Fourteenth Innovative Applications of Artificial Intelligence Conference (IAAI-02)*, 870–77. Menlo Park: AAAI Press.

Chalupsky, H., P. Szekely, F. Ilievski, D. Garijo, and K. Shenoy. 2021. "Creating and Querying Personalized Versions of Wikidata on a Laptop." In *Proceedings of the 2nd Wikidata Workshop (Wikidata 2021) Co-Located with the 20th International Semantic Web Conference (ISWC 2021)*, edited by L.-A. Kaffee, S. Razniewski, and A. Hogan. CEUR-WS.org.

Chang, Y, Narang, Mridu, Suzuki, Hisami, et al. 2022. "WebQA: Multihop and Multimodal QA", CVPR

Choudhary, A., A. Gershman, and J. Carbonell. 2019. "LEAPFROG: Adapting Belief Propagation for Knowledge Graph Construction."

Dang, H.T. 2019. *TAC/TRECVID Streaming Multimedia KBP for AIDA 2019 Evaluation Plan*. V1.5 ed. Gaithersburg, MD: NIST. https://tac.nist.gov/2019/SM-KBP/guidelines.html.

Deng, J. et al., 2009. "Imagenet: A large-scale hierarchical image database". In 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255.

———. 2022. *TAC/TRECVID Streaming Multimedia KBP for AIDA 2022 Evaluation Plan*. Gaithersburg, MD: NIST. https://www-nlpir.nist.gov/projects/tv2022/smkbp.html.

Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". NAACL 2018

Francis, N., A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor. 2018. "Cypher: An Evolving Query Language for Property Graphs." In *Proceedings of the 2018 International Conference on Management of Data*, 1433–45.

Hovy, E., T. Berg-Kirkpatrick, J. Carbonell, H. Chalupsky, A. Gershman, A. Hauptmann, F. Metze, et al. 2018. "OPERA: Operations-Oriented Probabilistic Extraction, Reasoning, and Analysis." In *Proceedings of Text Analysis Conference (TAC 2018)*.

Hovy, E., J. Carbonell, H. Chalupsky, A. Gershman, A. Hauptmann, F. Metze, T. Mitamura, et al. 2019. "OPERA: Operations-Oriented Probabilistic Extraction, Reasoning, and Analysis." In *Proceedings of Text Analysis Conference (TAC 2019)*.

Ilievski, F., D. Garijo, H. Chalupsky, N.T. Divvala, Y. Yao, C. Rogers, R. Li, et al. 2020. "KGTK: A Toolkit for Large Knowledge Graph Manipulation and Analysis." In *International Semantic Web Conference*, 278–93. Springer.

Ilievski, F., P.A. Szekely, G. Satyukov, and A. Singh. 2021. “User-Friendly Comparison of Similarity Algorithms on Wikidata.” *arXiv* 2108.05410.

Kim, W, Son, B and Kim, I. 2021. "ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision" ICML

Krishna, R., Zhu, Y., Groth, O. *et al.* Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *Int J Comput Vis* **123**, 32–73 (2017). https://doi.org/10.1007/s11263-016-0981-7

Lenat, D. 1995. “CYC: A Large Scale Investment in Knowledge Infrastructure.” *Communications of the ACM* 38 (11): 32–38.

Lin, T-Y, Maire, M, Belongie, Serge, Bourdev, L, Girshick, R, Hays, J, Perona, P, Ramanan, D, Zitnick, C. L, and Dollar, P. 2014. “Microsoft COCO: Common Objects in Context”

Mikolov, T., K. Chen, G. Corrado, and J. Dean. 2013. “Efficient Estimation of Word Representations in Vector Space.” *arXiv* 1301.3781.

Moriarty, D.E. 2000. “Determining Effective Military Decisive Points Through Knowledge-Rich Case-Based Reasoning.” In *Proceedings of the Thirteenth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. Springer Verlag.

Moriarty, D.E., and R.M. MacGregor. 2000. “Neural Computation in Symbolic Knowledge Representation Systems.” *CiteSeer Preprint CiteSeerX.psu:10.1.1.36.8667*. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.8667.

Obeid, J, and Hoque, E. 2020. "Chart-to-Text: Generating Natural Language Descriptions for Charts by Adapting the Transformer Model". INLG

Radford, A, Kim, J W, et al. 2021. "Learning Transferable Visual Models From Natural Language Supervision" ArXiv

Ren, S, He, K, Girshick, R and Sun, J. 2015. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks” NeurIPS

Richardson, M., and P. Domingos. 2006. “Markov Logic Networks.” *Mach. Learn.* 62 (1-2): 107–36. https://doi.org/10.1007/s10994-006-5833-1.

Stickel, M.E. 1990. “Rationale and Methods for Abductive Reasoning in Natural Language Interpretation.” In *Natural Language and Logic*, edited by R. Studer, 233–52. Springer-Verlag.

Szegedy, C, Liu, W, Jia, Y, Sermanet, P, et al. 2015. "Going Deeper with Convolutions", CVPR

Vrandečić, D., and M. Krötzsch. 2014. “Wikidata: A Free Collaborative Knowledgebase.” *Communications of the ACM* 57 (10): 78–85.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, Luke Zettlemoyer. 2020. "Scalable Zero-shot Entity Linking with Dense Entity Retrieval" Proceedings of EMNLP

Zadeh, L. A. 1975. “Fuzzy Logic and Approximate Reasoning.” *Synthese* 30 (3): 407–28.

APPENDIX A – PUBLICATIONS AND PRESENTATIONS

Hovy, E., T. Berg-Kirkpatrick, J. Carbonell, H. Chalupsky, A. Gershman, A. Hauptmann, F. Metze, et al. 2018. "OPERA: Operations-Oriented Probabilistic Extraction, Reasoning, and Analysis." In Proceedings of Text Analysis Conference (TAC 2018).

Huang, P-Y. J. Liang, X. Chang, and A. Hauptmann. 2018. Combining discrete and continuous representations for cross-modal retrieval. *Proc. TRECVid workshop 2018*

Liu, Z., T. Mitamura, and E.H. Hovy. 2018. Graph-Based Decoding for Event Sequencing and Coreference Resolution. *Proceedings of the 27th International Conference on Computational Linguistics COLING*

Palaskar, S. and F. Metze. 2018. Acoustic-to-Word Recognition with Sequence-to-Sequence Models. *Proc. SLT 2018*. IEEE.

Sanabria, R. and F. Metze. 2018. Hierarchical Multi-Task Learning with CTC. *Proc. SLT 2018.* IEEE

Chang, X., P-Y. Huang,Y-D. Shen, X. Liang, Y. Yang, and A. Hauptmann. 2018. RCAA: Relational Context-Aware Agents for Person Search. *Proc. ECCV 2018*

Liu, Z., C. Xiong, T. Mitamura, and E.H. Hovy. 2018. Automatic Event Salience Identification. *Proc*. *EMNLP 2018*

Palaskar, S., R. Sanabria, and F. Metze. 2018. End-to-End Multimodal Speech Recognition. *Proc. ICASSP 2018*. IEEE

Xiong, C., Z. Liu, J. Callan, and TY. Liu. 2018. Towards Better Text Understanding and Retrieval through Kernel Entity Salience Modeling. *Proc*. *SIGIR 2018*

H. Chalupsky. Chameleon 2.0: Integrating Neural and Symbolic Reasoning in PowerLoom. In AKBC Workshop on Neural and Symbolic Representation and Reasoning, 2019.

Choudhary, A., A. Gershman, and J. Carbonell. 2019. "LEAPFROG: Adapting Belief Propagation for Knowledge Graph Construction."

Hovy, E., J. Carbonell, H. Chalupsky, A. Gershman, A. Hauptmann, F. Metze, T. Mitamura, et al. 2019. “OPERA: Operations-Oriented Probabilistic Extraction, Reasoning, and Analysis.” In Proceedings of Text Analysis Conference (TAC 2019).

Dang, H.T. 2019. TAC/TRECVID Streaming Multimedia KBP for AIDA 2019 Evaluation Plan. V1.5 ed. Gaithersburg, MD: NIST. https://tac.nist.gov/2019/SM-KBP/guidelines.html.

An Empirical Investigation of Structured Output Modeling for Graph-based Neural Dependency Parsing. Zhisong Zhang, Xuezhe Ma and Eduard Hovy. ACL conference.

Ma, X., C. Zhou, and E.H. Hovy. 2019. MAE: Mutual Posterior-Divergence Regularization For Variational Autoencoders. International Conference on Learning Representations. ICLR conference.

Kong, X., V. Gangal, and E.H. Hovy. 2020. Sentence Cloze Dataset with High Quality Distractors From Examinations. ACL 2020. Seattle, WA.

Zhang, Z., X. Kong, Z. Liu, X. Ma, and E.H. Hovy. 2020. A Two-Step Approach for Implicit Event Argument Detection. ACL 2020. Seattle, WA.

Kong X, Zhang Z, Hovy E. Incorporating a Local Translation Mechanism into Non-autoregressive Translation. EMNLP 2020

Zhang, Z., Kong, X., Levin, L., & Hovy, E. An Empirical Exploration of Local Ordering Pretraining for Structured Learning. EMNLP-Findings 2020

Zhang, Z., Strubell E., & Hovy, E. On the Benefit of Syntactic Supervision for Cross-lingual Transfer in Semantic Role Labeling. EMNLP 2021

Gangal, V., Feng, S. Y., Alikhani, M., Mitamura, T., & Hovy, E. (2021). Nareor: The narrative reordering problem. arXiv preprint arXiv:2104.06669 AAAI

Feng, S. Y., Lu, K., Tao, Z., Alikhani, M., Mitamura, T., Hovy, E., & Gangal, V. (2021). Retrieve, Caption, Generate: Visual Grounding for Enhancing Commonsense in Text Generation Models. arXiv preprint arXiv:2109.03892. AAAI

Jhamtani, H., Gangal, V., Hovy, E., & Berg-Kirkpatrick, T. (2021). Investigating Robustness of Dialog Models to Popular Figurative Language Constructs. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (pp. 7476-7485).

Feng, S., Huynh, J., Narisetty, C. P., Hovy, E., & Gangal, V. (2021). SAPPHIRE: Approaches for Enhanced Concept-to-Text Generation. In Proceedings of the 14th International Conference on Natural Language Generation (pp. 212-225).

Dang, H.T. 2022. TAC/TRECVID Streaming Multimedia KBP for AIDA 2022 Evaluation Plan. Gaithersburg, MD: NIST. https://www-nlpir.nist.gov/projects/tv2022/smkbp.html.

Ma X, Kong X, Wang S, et al. Luna: Linear Unified Nested Attention, NeurIPS 2021

Mekala, D., Gangal, V., & Shang, J. (2021). Coarse2Fine: Fine-grained Text Classification on Coarsely-grained Annotated Data. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (pp. 583-594).

Mille, S., Dhole, K., Mahamood, S., Perez-Beltrachini, L., Gangal, V., Kale, M., ... & Gehrmann, S. (2021). Automatic Construction of Evaluation Suites for Natural Language Generation Datasets. (NEURIPS '21 Datasets Track)

Liangke Gui, Qiuyuan Huang, Alex Hauptmann, Yonatan Bisk, Jianfeng Gao (2022) Training Vision-Language Transformers from Captions Alone https://arxiv.org/abs/2205.09256

Liangke Gui, Borui Wang, Qiuyuan Huang, Alex Hauptmann, Yonatan Bisk, Jianfeng Gao (2022) KAT: A Knowledge Augmented Transformer for Vision-and-Language, North American Chapter of the Association for Computational Linguistics

Yingshan Chang, Mridu Narang, Hisami Suzuki, Guihong Cao, Jianfeng Gao, Yonatan Bisk (2022) WebQA: Multihop and Multimodal QA, CVPR

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

DOD	Department of Defense
AIDA	Active Interpretation of Disparate Alternatives
OPERA	Operations-oriented Probabilistic Extraction, Reasoning, and Analysis
LDC	Linguistic Data Consortium
BERT	Bidirectional Encoder Representations from Transformers
IE	Information Extraction
KE	Knowledge Elements
OIF	OPERA-internal frames
QA	Question Answering
NP	Noun Phrase
NLI	Natural Language Inference
ASR	Automatic Speech Recognition
NER	Named Entity Recognizer
MAE	Masked AutoEncoders
CSR	Common Semantic Repository
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
JSON	JavaScript Object Notation
AIF	AIDA Interchange Format
KAgg	The Knowledge Aggregator
KGTK	Knowledge Graph Toolkit