OPERA: Operations-oriented Probabilistic Extraction, Reasoning, and Analysis

Eduard Hovy (PI), Taylor Berg-Kirkpatrick, Jaime Carbonell, Hans Chalupsky*,
Anatole Gershman, Alex Hauptmann, Florian Metze, Teruko Mitamura,
and students
Aditi Chaudhary, Xianyang Chen, Bernie Po-Yao Huang, Hector Zhengzhong Liu,
Xuezhe Ma, Shruti Palaskar, Dheeraj Rajagopal, Maria Ryskina, Ramon Salabria

Carnegie Mellon University
*Information Sciences Institute of the University of Southern California

Abstract

This paper describes CMU and USC/ISI's OPERA system that performs end-to-end information extraction from multiple media, integrates results across English, Russian, and Ukrainian, produces Knowledge Bases containing the extracted information, and performs hypothesis reasoning over the results.

1 System Summary

OPERA is an integrated solution to the challenges of DARPA's AIDA program:

- **Existing high-performance media analysis (TA1)**: We use and build upon CMU's existing information extraction and understanding technology, developed in prior DARPA, IARPA, and other programs, that processes text (various genres), images, video, and speech. Each medium includes one or more TA1 analysis engines that produce output in a consistent uniform json formalism.
- **Semantic representation and reasoning support (TA2 and TA3):** OPERA employs USC/ISI's PowerLoom knowledge representation and reasoning system, developed and used in numerous DARPA programs, that provides an expressive predicate logic representation language, several powerful deductive and abductive inference mechanisms, contextual and hypothetical inference, inference justifications and truth maintenance, and database integration. This system accepts the TA1 engines' json output and stores the results into OPERA's central semantic repository (CSR) using database technology.
- **Cross-medium and cross-language integration (TA2)**: We develop approaches to integrate the interpretations of the TA1 engines. This includes coreference of terms across the various domain languages (produced by the TA1 text engines) as well as across the outputs of speech, image/video, and text interpretation technology.
- **Hypothesis creation, management, and utility-based hypothesis space exploration (TA3)** (both standalone and guided by the analyst): OPERA includes two alternative pathways for hypothesis creation and hypothesis-driven (re)interpretation and reasoning. One uses PowerLoom, with the addition of a probabilistic hypothesis reasoning engine with sophisticated belief calculation and propagation. The other is a novel Belief Propagation system built for the project that uses factor graphs to represent interpretation alternatives and larger hypotheses concisely and effectively.

- **Framework (computational and semantic)**: for OPERA we build a computational framework that integrates all the modules and facilitates connection with TA4 environments. Also, in conjunction with other program performers, we develop a top-level terminology ontology plus domain-specific detailed extensions of it.

2 Technical Components

2.1 TA1 Text: Entity Processing

This component focuses on the identification, extraction, and linking/coreference of entities within individual texts. It also includes work on identifying and extracting the properties (= relations) of entities.

Entity extraction: For OPERA we re-implemented our EDL system, built in 2013–16 under DARPA's DEFT program. It is now faster, and re-trained on a selection of entities more suited to the AIDA domain. As part of this work we deployed a new NER system, built at CMU, that combines character-level and word-level representations with a CRF model (Ma and Hovy, 2016). On example AIDA domain texts, this system produced an F-score of 54.9 (though on the entities relevant to the domain, the score was 71.1), with the score for nominals being 32.5.

Relation/property extraction: We have implemented and built a relation extractor that identifies and extracts entity properties. We designed the system of four parts: vector representation, CNN convolution, piecewise max pooling, and softmax output. The inputs to the network are raw word tokens, which are transformed into low-dimensional vectors. Because an input sentence that is marked as containing the target entities corresponds only to a relation type and it does not predict labels for each word, it might be necessary to utilize all local features and perform this prediction globally. Therefore, the convolution approach is employed to extract and merge all local features. In piecewise max pooling, the input feature is split into three segments according to the positions of the entity pair, and piecewise max pooling returns the maximum value in each segment instead of a single maximum value. Hence, piecewise max pooling can capture structural information between two entities. Finally, the output is fed into a softmax classifier to compute the confidence of each relation. The system achieves state-of-the-art performance.

2.2 TA1 Russian and Ukrainian Processing

Our initial goal was to parse Russian and Ukrainian sentences into Universal Dependencies (UD) trees with native words referring to entities, events, and relations, each one substituted with its appropriate ontological reference(s), thereby enabling the English event extraction module (Section 2.3) to produce workable KEs for coreference and subsequent inclusion into the CSR. We wanted to avoid using MT tools for two reasons: (1) sufficiently accurate MT tools may not be available for every language of interest and (2) training an MT tool is a harder problem than extraction itself. We largely achieved this goal although the accuracy is far from perfect. For the next iteration of the project, we are exploring alternative approaches focusing on the creation of domain-specific semantic parsers that elicit the necessary knowledge from the users without requiring annotated texts.

Entities: In the interim our solution is as follows. After POS tagging the sentence, rules delimit NP chunks and then identify the head noun of each chunk. Linking the head noun into the OPERA ontology (currently using manually created mappings specific to the domain lexicon) allows the extractor to return a unit (the chunk) with its ontology node as its type, but without any detailed further information.

NER: A principal challenge during the initial phase is the absence of decent named entities extractors for Russian and Ukrainian. To meet this challenge we relied on extensive lists of named entities specific to the Russia-Ukraine conflict (we created a Named Entity glossary from our pre-ontology analysis of the domain corpus word/type distributions) and on "fuzzy" matching heuristics. We believe that this is not an expediency trick but a legitimate approach—there is a finite number of Ukrainian towns with a finite number of variations of their spelling.

Parsing: We identified several Russian parsers, tested them, and after some exploration selected the Universal Dependency parser *UDPipe 1.2* (Straka and Strakova, 2017). We used it with the Babylon lexicon and the Extended Open Multilingual Wordnet. After producing an initial parse, with appropriate linguistic features, we then apply the above Entity and NER systems. Given an input sentence, each NP or V chunk is returned to the core entity assembler described in Section 2.3. Each such chunk contains:

- the text span
- its type
- its head
- the head's ontology item
- other relevant linguistic features (e.g., case and number for NPs, or tense for Vs).

The English event extractor is still unable to use Russian and Ukrainian case markers. We are now designing a truly multi-lingual event/relation extractor that uses universal dependencies and universal grammatical features.

Lexical and other resources: We have found reasonably high-quality Russian lexical resources, including the AOT Russian-English dictionary, the Babylon Russian-English dictionary (BGL), the Extended Open Multilingual Wordnet (RUS and UKR), and a transliteration package (transliterate 1.10.1) for names.

Ukrainian: We searched extensively for adequate Ukrainian technology and resources, but found nothing useful. In the mean time, we tried MT from Ukrainian to Russian and then using the Russian solution. It worked surprisingly well.

2.3 TA1 Text: Event Processing

This module plays a central role in the text-based OPERA pipeline. It accepts input from entity processing (Section 2.1) and Russian and Ukrainian analysis (Section 2.2), and then performs event detection and frame selection. It then combines all the information to produce instantiated frames. It outputs KEs for events, entities, and relations, and uploads them into the Central Semantic Repository CSR.

Migrating and extending the argument extractor: Since the arguments for the AIDA knowledge base are also governed by an ontology, we convert the found arguments to this ontology to allow knowledge propagation. We also migrate our prior argument extractor to the new scenarios. We have extended our existing argument extractor, are based on Propbank, to now use FrameNet, which is more robust for our purposes.

Traditional argument extraction follows the semantic role labeling approach, where the arguments are extracted locally, often with the help of a syntactic tree. We extend our previous work to also cover some implicit arguments, using scripts. Currently, we have trained some basic implicit argument detection models. They can be easily integrated with the system since the implicit arguments are also based on Propbank and FrameNet.

Identifying the important content in documents: In order to deal with an semi-open domain, we obtain word-frequency statistics from the Gigaword news articles and their human-written abstracts. We have successfully built a feature- and a neural network model for this task. Precision@1 reaches 50% and Precision@10 is as high as 36%, showing that salience estimation is quite accurate for the top events.

Event assembly: We extended coverage and performance of the top-level event detector and event structure integrator. It now accepts inputs from the entity extractor and the relation extractor (Section 2.1) and the multilingual entity and event extractors (Section 2.2), integrates them all into event structures (frames) and performs limited within-document coreference on entities and events.

2.4 TA1 Images/Video Processing

The goals of this component are: (1) knowledge extraction from visual media (images and video), (2) conversion and harmonizing output into a format suitable for connecting into the AIDA and/or OPERA ontology, and (3) integration into the OPERA pipeline and the AIDA evaluations.

We put in place the visual (image and video processing) component of the OPERA pipeline. An overview is shown in Figure 1. After preprocessing, the visual content is passed through four visual knowledge element extraction units, including classification (finding WHAT is inside), detection (finding WHERE is the entity), OCR (recognizing characters within an image or video) and extracting metadata. The pipeline takes raw image (jpg, png, bmp files) and video (mp4 files) directly and generates OPERA's json CSR output.

Table 1 shows results of the OPERA visual pipeline. For quantitative analysis, we manually labeled 1103 in-domain images with 10 selected classes and evaluated the model with standard metrics. As can be seen, both results are reasonable, and the model is robust against in-domain noise. However, we think the contribution of the visual part is still ambiguous without building the link to the text content. Therefore, we conducted cross-modal linking/coreference (in the current phase) and plan to explore visual name entity recognition in the next phase.

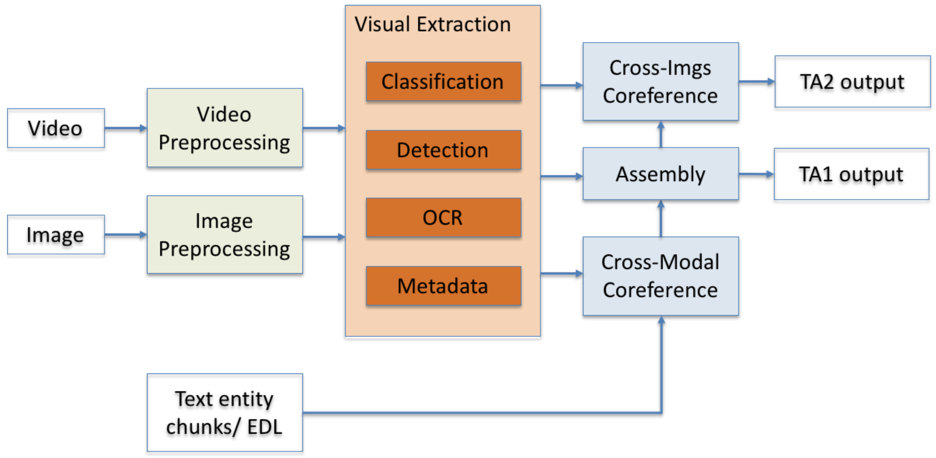

Figure 1. OPERA image/video processing pipeline: I/O, preprocessing, extraction, merging/coreference in white, green, orange, blue color respectively.

Class	Person	Vehicle	Snow	Military	Soldier	Troop	Air_force	Fire	Weapon	Tank
P	0.85	0.85	0.7	0.75	0.75	0.32	0.82	0.8	0.7	0.75
R	0.7	0.65	0.6	0.6	0.61	0.21	0.5	0.7	0.32	0.72
F1	0.77	0.74	0.65	0.67	0.67	0.25	0.62	0.75	0.44	0.73

Table 1. Quantitative result for OPERA in-domain visual entity extraction.

Harmonizing visual ontology with the OPERA ontology: The image detection/classification model is based on the ResNet101 model pretrained on the Open Image dataset. It contains 5000+ visual concepts. The video classification model is pretrained on Youtube8M and Kinetics and contains 5000+ visual concepts. For OPERA, we further manually selected 2,100+ visual concepts of the union of two concept pools as the final pool for visual entity classification. In this pool, 800+ of the visual concepts are localizable, which means we can further generate bounding boxes to localize the visual entities. For OPERA in-domain data, we further trained 24 unique new concepts (e.g., the Ukraine national flag) to localize critical information. After merging/trimming/filtering based on the visual ontology, we finalized a visual pool with 224 visual concepts. The extraction result is then sent to cross-modal coreference for assembly with other TA1 outputs. We also use clustering to generate cross image coreference for TA2 visual outputs.

Cooperating with other parts of OPERA, we built the visual ontologies based on the WordNet hierarchy and Google's knowledge graph. As shown in Figure 2, we have a three- and four–level hierarchy for the video and image ontology respectively. We facilitate these two ontologies to define the further shrink to a 224-type ontology on which we then perform type matching to the OPERA text ontology.

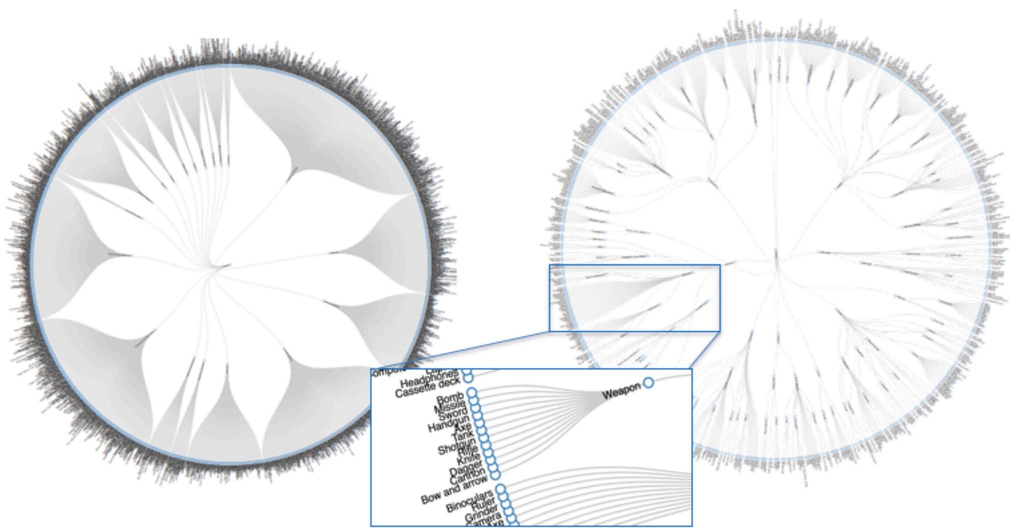

Figure 2. Visualization of OPERA video ontology (left) and image ontology (right).

2.5 TA1 Speech/Audio Processing

We are building Automatic Speech Recognition (ASR) systems to recognize Russian, Ukrainian and English speech in the AIDA domains. We are exploring the use of a "context" frame to help disambiguate decisions, a procedure we have demonstrated in recent publications helps with for English How-To videos.

Initially, we transcribed videos in English, Ukrainian, and Russian, by setting up three separate pretrained systems. We next conditioned these ASR systems to recognize Russian, Ukrainian, and English speech using a "context" frame. We found more speech in the evaluation test video than we could computationally handle and made selections based on the apparent relevance of each segment.

Language and segment detection: We built a speech-based language detector by using the posterior probabilities of each recognizer, assigning each video to the language with the highest average posterior. We also integrated speech segmentation into the pipeline, to separate non-speech parts from speech.

Pipeline: We integrated the ASR systems and the speech-based language detection in a more general pipeline where the input was a list of video files and the output a list of text files containing the transcriptions of each video in the predicted language. Finally, we parallelize the whole pipeline so that we could automatically transcribe a considerable quantity of videos in a decent amount of time.

2.6 TA2 Cross-Medium Coreference

The goals of this component are to: (1) match knowledge extracted from a visual media document to knowledge extracted from text, and (2) create co-reference links across documents in different media. A specific challenge is to identify individual people and connect them to their activities described in the textual documents. We did not deploy this component in the September pre-evaluation.

2.7 TA2 Cross-Document/Cross-Language Coreference

The goal of this module is to merge the individual per-document knowledge bases (mini-KBs) into a single graph via cross-document event and entity linking. We create and deploy technology to perform coreference of entity and event KEs drawn from different text documents (and hence also different languages).

We view the cross-document entity linking as a clustering problem, where instances to be clustered are established coreference chains within each document, including singleton mentions. We tried a simple clustering approach, representing each coreference chain as its embedding vector and then merging them with agglomerative clustering.

To compute an embedding for a coreference chain, we concatenated the individual mentions and then created a sentence embedding of the sequence. Since we do not have a large in-domain corpus to train our own sentence embedding method, we used a simple approach of Arora et al. that does not require training. Sentence embeddings were computed by averaging word embedding vectors, weighted by smooth inverse frequency. We computed the embeddings for a sequence of contexts of individual mentions in the same fashion, and averaged the two together to create a more balanced representation. We used pretrained FastText embedding vectors as word embeddings for each of the three languages.

We used a simple agglomerative clustering method, empirically setting the distance threshold for merging the clusters. Since we have decided to focus on precision, the distance threshold has to be set quite low, so most of the effect could have been captured by simple string matching features. However, embeddings let us go beyond string matching: to some extent, they capture synonyms (e.g., *troops* and *armed forces*), and subword information in FastText gives us some flexibility in spelling (such as merging mentions of *Kiev* and *Kyiv*).

Finally, we established entity links across the languages. Since projecting the embedding spaces of different languages into the same shared space may be noisy and unreliable, we chose to rely on the cross-lingual mapping of the database links extracted with DBpedia Spotlight and the translations of named entities extracted from manually collected glossaries. However, since both cross-lingual mappings and monolingual clustering are noisy, we may end up with conflicting evidence (for example, a coreference chain that contains links to different DBPedia entries, such as *Donetsk Republic*, an organization, and *Donetsk People's Republic*, a proto-state). We implemented two ways to handle these discrepancies: a more precision-oriented one that intersects the cross-lingual and monolingual links, breaking up clusters into parts that all map to a single database entry and linking those; and one that allows for more recall, combining all information from both sides and accepting all merging decisions. The latter might introduce clustering noise but also might allow the two components correct each other's mistakes, while the former may be too sparse.

Our current implementation establishes links between entities and thus relies on the linking between event arguments as a proxy for event linking. We chose a precision-oriented approach, so we have decided to additionally restrict our scope to only named entities of certain categories (GPE, Person, Organization). The motivation here is that without an extensive use of context we would not be able to resolve certain ambiguities, and any noise in relation extraction or within-document coreference resolution could result in incorrect linking that could be very damaging to the hypothesis verification results (e.g., if two mentions of *army* are merged where in fact one is Russian and one is Ukrainian).

Since this system operates within state TA2, within-document coreference has already taken place. We therefore had two alternatives: (1) mapping individual KEs into the TA2 KB and then performing clustering across documents, using the within-doc coref groups/clusters to set the extent parameters for each cluster, or (2) mapping each distinct TA1 coreference cluster into the TA2 KB as a single unit and then performing cross-document clustering using these larger units. Eventually we chose the latter, for two reasons: the former is more delicate and potentially might introduce far more damaging results if a widespread but erroneous clusters were to be formed, and with the former it is more difficult to ensure that no recoverable document-level information is propagated into TA2, which would contravene the rules of the evaluation.

2.8 TA3 Hypotheses and Belief Graph Processing

This module is one of two hypothesis reasoners of OPERA. It complements the KR&R reasoner described in Section 2.9.

Plausible hypotheses need to take into account the uncertainties in the underlying knowledge. To support generation, management and evaluation of hypotheses, we developed Belief Graphs (BG)—a knowledge representation system that includes and manages uncertainties. We used BGs as a complementary knowledge base in OPERA in all three stages: TA1 for mini-KB and its conditioning by the analysts' hypotheses, TA2 for coreference resolution, and TA3 for hypotheses construction.

Belief graphs use random variables to represent uncertainty. These variables mediate between knowledge elements and their uncertain attributes. Many relations imply probabilistic dependencies creating uncertainties. For example, the relation "parent-child" implies a certain probabilistic relation between the age of the parent and the age of the child. If we know the parent's age we can estimate the probability distribution of the child's age. This knowledge of the child's age is uncertain but it is better than no knowledge.

The two major steps in Belief Graph construction are evidence graph formation and grounding.

Evidence graph construction: Each observed entity, event, and relation mention is represented as an evidence knowledge element with some properties. For entities, we consider only two properties: **name** and **entity-type**. Since events and relations have specific arguments, we map these arguments to a fixed set of semantic roles e.g., **Conflict.Attack-Attacker** to **Agent**, **Conflict.Attack-Targe**t to **Patient**, etc.

Evidence knowledge elements are intentional and need to be grounded to specific extensional elements in the KB.

Grounding: Since a KB may contain millions of knowledge elements, we use heuristics to select a small number of plausible candidates for the grounding of an evidence knowledge element. We create a grounding variable for each evidence element and form a factor graph to conduct belief propagation which results in the posterior probability distribution of the grounding variable. For entities, we get a set of potential candidates by retrieving entities that have the same entity type as the observed entity. For events, it is somewhat more complex because we have different arguments. As described above, for consistency and for the purpose of tractability we map all the event-specific arguments to a fixed set of semantic roles. In an unbiased scenario, all events receive uniform priors. Belief propagation is run over this factor graph to find the posterior distribution of the "target" variable.

TA1a inference: We run the procedure descried above for each document to generate a document-specific mini knowledge graph. The inputs to this are the entity, event, and relation mentions extracted by the TA1 module extractors. To this, the belief propagation module adds additional co-reference. The mini-KB starts empty and is populated by knowledge elements as a result of mention grounding. We also assume a uniform prior distribution among grounding candidates.

TA1b inference: In this task, we are given human generated hypotheses and we use this knowledge to bias our knowledge base construction. We do this by seeding the mini KB with the events, entities and relations mentioned. We also assign higher grounding priors to these knowledge elements. This biases the interpretation of the document mentions. The remaining procedure is the same as for TA1a above.

TA2 inference: We run the same procedure as before, but instead of mentions as inputs, we use the document level mini knowledge graph as the input. If available, we add cross-document and cross-lingual entity co-references, which serve as anchors help reduce a potentially much larger set of grounding candidates. Belief propagation is then used as before to compute the marginals. We retain only a few high-probability candidates in each grounding variable distribution relegating the rest to "*other". Grounding helps identify likely cross-document coref candidates.

TA3 inference: Given an information need in the form of an incomplete graph, we retrieve the relevant event types from the knowledge graph. We match our KB entity nodes with the entry points in the information need statement. We then use these nodes to include the events and other role fillers. Usually this results in the required event graph. Since all knowledge element attributes are connected via random variables, we have a natural way of constructing alternative hypotheses and evaluating their likelihood.

2.9 TA3 Knowledge Representation and Reasoning in PowerLoom

This component developed a representation formalism, repository, inference engine and APIs to store, access, map, disambiguate and link knowledge elements (KEs) generated by TA1 modules or entered directly by analysts. It also generates and manages semantically coherent hypotheses that are supported to some minimal degree by evidence available to OPERA, and records and manages alternatives to enable backtracking and retraction under the Hypothesis Reasoner. (This module includes one of the two hypothesis reasoners of the OPERA engine; the other is described in Section 2.8.) This module is implemented in large part in PowerLoom.

CSR representation: We developed a CSR representation and interchange language for the OPERA team; our json-LD-based OPERA interchange language is used internally and is translated into AIF (AIDA Interchange Format) as required, as well as Web hosting of context definitions to support translation into RDF.

Database backend technologies: 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. We decided to use a triple store and graph database called Blazegraph (www.blazegraph.com). 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 scaleout in a cluster
- Used by WikiMedia / WikiData, some commercial clients
- 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. We also completed an initial 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.

Query processing: We developed a query-to-SPARQL converter to handle NIST's pre-evaluation queries, as well as an OPERA-json-format-to-AIF-format converter for delivery of final evaluation output to NIST.

The pre-evaluation in September highlighted numerous aspects and alternative inference paths that we did not have time to explore. One aspect is the deployment of a probabilistic soft logic (PSL) reasoner. We have an initial framework to export entity/event/relation hypotheses represented in PowerLoom into PSL and then apply probabilistic PSL inference over them. To evaluate this we so far have some anecdotal results, nothing yet quantitative. This aspect appears to be a good complement to the kind of Belief Graph reasoning described in Section 2.8.

3 The Pre-Evaluation

The recent pre-evaluation (September–October 2018) was an enormous effort on everyone's part. In summary, OPERA submitted runs as follows. We uploaded our final TA1a, TA1b, TA2 and TA3 KBs, and query results for them all, including TA3. We validated them as much as possible with NIST's validator code. In particular we

- partially validated our TA1 results by focusing on the top-20 largest response files from each class and each KB and successfully validating those. The problem there is that for single-file validation, the validator has a large per-file overhead (about 2.5min for graph query responses on our machine), which makes it somewhat prohibitive to validate 1000s of response files. Even just focusing on the core documents would take a long time for 3 + 5 TA1 KBs. If there is a work-around for that, we'd be happy to run validation again on a larger scale;
- fully validated all TA2 results;
- did not validate our TA3 results, since it seems the NIST validator doesn't yet support that and breaks on our hypothesis file names.

We also uploaded the various KBs in our own OPERA json-LD format, and have translated them into the optional AIF.

4 References

Chalupsky, H. 2018. CMU OPERA Team Proposal for AIDA Interchange Language, Version 0.3. July 5, 2018.

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*.

Huang, P-Y., X. Chang, and A. Hauptmann. 2018. CMU-AML Submission to *Moments in Time, ActivityNet workshop at CVPR conference*.

Huang, P-Y., J. Liang, J-B. Lamare, and A. Hauptmann . 2018. Multimodal Filtering of Social Media for Temporal Monitoring and Event Analysis. *Proc. ICMR 2018*.

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

Ma, X. and E.H. Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. *Proc. ACL conference*. Berlin, Germany.

Ma, X., Z. Hu, J. Liu, N. Peng, G. Neubig and E.H. Hovy. 2018. Stack-Pointer Networks for Dependency Parsing. *Proc. ACL conference 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*.