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

Eduard Hovy (PI), Jaime Carbonell, Hans Chalupsky*,

Anatole Gershman, Alex Hauptmann, Florian Metze, Teruko Mitamura,

Zaid Sheikh, Ankit Dangi,

Aditi Chaudhary, Xianyang Chen, Xiang Kong, Bernie Huang, Salvador Medina, Hector Liu, Xuezhe Ma, Maria Ryskina, Ramon Sanabria, Varun Gangal

Carnegie Mellon University

*Information Sciences Institute of the University of Southern California

Abstract

The OPERA system of CMU and USC/ISI performs end-to-end information extraction from multiple media and languages (English, Russian, Ukrainian), integrates the results, builds Knowledge Bases about the domain, and does hypothesis creation and reasoning to answer questions.

1 System Summary

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

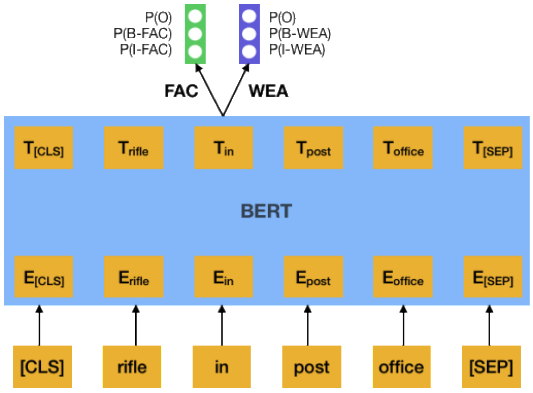
- **Existing high-performance media analysis (TA1)**: We build state-of-the-art information extraction technology for text, speech, and image/video data. Each TA1 analysis engine produces output in a uniform internal json notation.
- **Semantic representation and reasoning support (TA1 and TA2):** In OPERA, USC/ISI's PowerLoom knowledge representation and reasoning system 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. PowerLoom helps enforce consistency and improve coverage of the output of the TA1 extraction engines.
- **Cross-medium and cross-language integration (TA2)**: We develop methods to integrate the interpretations produced by the TA1 extractors, linking coreference clusters over text/speech and image/video, in various languages.
- **Hypothesis creation, management, and hypothesis exploration (TA3)**: OPERA incorporates a novel Belief Graph representation approach to represent alternatives that compose to form competing hypotheses about domain events. Alternatives' relative weights are captured in factor graphs whose values are derived from system-internal processing and other sources.
- **Framework (computational and semantic)**: A framework integrates all the modules and facilitates connection among OPERA modules and also with TA4 environments. Also, in conjunction with other program performers, we develop a top-level terminology ontology plus domain-specific detailed extensions of it. All knowledge elements are represented in an OPERA-internal frame notation, stored in OPERA's central semantic repository (CSR) using database technology, and converted to exportable Knowledge Bases in AIF notation.

2 Technical Components

In this section we describe the principal components. Throughout we refer to internal module evaluations conducted at various times using the domain texts annotated and released by LDC during the first half of 2019.

2.1 TA1 Text: Entity Processing

English entity extraction: The goal of this module is to identify and extract domain-relevant entities, assign them types (defined in the AIDA ontology, and deliver Knowledge Elements (KEs) for further processing. We define three sub-modules that respectively assign types from the ontology's type, subtype, and subsubtype levels. Type-level entity extraction uses BERT-based NER. Specifically, we fine-tune a pre-trained BERT (Devlin et al., 2018) model on KBP NER data and a small amount of data, annotated at the type level by ourselves. For deeper ontology levels, we work with YAGO (Suchanek et al.,, 2008)[1]. The YAGO knowledge base contains around 350K entity types. We map selected types to the AIDA ontology subtype or subsubtype levels manually. All YOGA entity types without a correspondence in the AIDA ontology are regarded as NULL. Since this would introduce noise if used to fine-tune BERT directly, we do multi-task learning, in which every subsubtype is regarded as an independent classification task with its own classifier. During training, each classifier is only updated by its corresponding type. During test time, for each token, all classifiers are applied and the class with the highest probability is assigned to this token. Should an entity be extracted at multiple levels, we first check ontology hierarchical consistency and if consistent, the results are merged; otherwise, the candidate with the higher probability (confidence score) is selected.



The multi-task architecture is schematically shown above. There are two main advantages to multi-task training: (1) different entity types may benefit from others, exploiting commonalities and differences across tasks; (2) we save memory with just a single shared encoder.

Performance: In internal mini-evaluations, the entity extraction engine achieves an F1-score = 0.65.

English entity linking: Here we perform sub-module ensembling. The extracted named entities are linked to the reference knowledge base via string matching. We index (*entity_id, entity_name, entity_info*) triples with Lucene for efficient searching and fuzzy matching. The reference KB contains the whole Geonames dataset)whose entities are mostly noisy, so we retain only the entities in Russia or Ukraine, or have a Wikipedia page). Given an extracted named entity, we search its mention string

[1] For this we follow the suggestion of Heng Ji from UIUC and give her great thanks.

within the reference KB and retrieve entities whose name contains the searched string. When multiple entities are retrieved, the results are filtered and ranked by several features:

1. Whether the reference KB type and the NER type match
2. Whether the reference KB name exactly matches the mention string
3. The edit distance between the entity name and the mention string
4. Whether the entity has a Wikipedia page
5. For GEO entities, whether it is a country/region level geoname
6. For GEO entities, whether it is in Russia/Ukraine
7. For GEO entities, whether it is in USA/Canada

For the last two features, we prioritize places in Russia and Ukraine since the evaluation's domain is about Russia-Ukraine conflict. On the other hand, the USA and Canada are two immigrant countries and many of their places are named after somewhere in Eurasia. This could cause much ambiguity so we de-prioritize them.

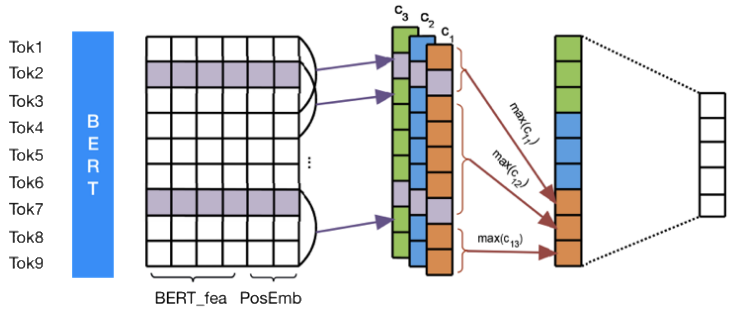
The entities with the highest scores are output. If no string match is found, we rerun the search but allow fuzzy matching with edit distance 1. We keep increasing the edit distance until some matched are found, or the edit distance reaches a certain threshold.

For entities that are not linked to the reference KB, we propagate the linking result via coreference. If none of the entities of a corefent cluster are linked, a temporary KB entry is created. If the same entity appears in the following documents, it might be linked to the temporary entry.

English Relation Extraction: The goal of this module is to extract relations between entities or events: input is two entities or events and the output is the relation between them. Our module contains two parts:

1. BERT (Devlin et al., 2018) as a feature extractor
2. Piecewise Convolutional Neural Network (PCNN) for relation extraction

The system (shown at right) consists of four main parts: vector representation, convolution, piecewise max pooling, and softmax output. A pre-trained BERT is employed to extract useful features from the input tokens and transform word tokens into low-dimensional vectors. Since an input sentence that contains the target entities and events expresses only a single relation type and does not predict labels for each token, it is necessary to merge all local features and perform this prediction globally. So 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 is able to capture the structural information between two entities. Finally, the output from previous parts is fed into a softmax classifier to compute the confidence of each relation. This system is trained on ACE 2004, ACE 2005, KBP English fine-grained relation data, and achieves state-of-the-art performance on ACE 2004. This system is unique in its automated learning of features without complicated NLP preprocessing.

Performance: Our system achieves F1-score = 0.58 on the mini-evaluation data.

2.2 TA1 Text: Event Processing

English event detection and assembly: The goal of this module is to produce the set of domain and topic relevant event instances and their participants. We further predict the salience of the event mentions in order to help reveal the most relevant events in the document.

This module comprises the following submodules:

- English Event Detection
- English Event Coreference
- English Event Argument Detection
- English Entity Coreference
- English Event/Entity Salience Detection

Given plain-text input, this module produces:

- Event mention annotations, including the event type and provenance
- Attributes of events: realis status and salience ranking
- Event arguments: a list of core participants and their types
- Event Coreference clusters: event mentions are organized into clustered groups

The main system is built on top of the event mention detection (ED) system developed at CMU in the DEFT project under the DARPA program preceding AIDA. The core of the ED system is a sequence tagger based on Conditional Random Field (CRF). The features for the ED system is a mixture of discrete features such as WordNet and Framenet sense, and some continuous features such as word embeddings. This base system is trained on the TAC-KBP event nugget detection data from 2015 to 2017. The base system is consistently ranked as the top 1 or 2 systems in the TAC workshop.

The event types are predicted jointly with the event spans. We further train a realis classifier to predict the realis status using the TAC-KBP event nugget data.

To adapt to the new event ontology in the AIDA program, we employ two methods:

1. We curated a mapping from the TAC-KBP and FrameNet ontology to the AIDA ontology, enabling the reuse of the existing data and models.
2. To improve coverage, we further developed a new zero-shot matching system to map the remaining predicates to the AIDA ontology based on word embedding similarity between the predicates and the AIDA types.

Our event coreference (EC) system is also adapted from the DEFT project. The EC system is based on the pairwise similarities of the two event instances hence is less sensitive to the change of domain. This allows us to simply port the existing coreference system. Our coreference system is a tree-based coreference system that holds the current state-of-the-art performance on the TAC-KBP event track. Our tree-based coreference decoding structure, the Latent Antecedent Tree (LAT) model, enables the system to effectively rank and combine the pairwise decisions.

The intuition behind the LAT model is that not all decoding structures are equally easy. A good decoding strategy is to decode the easiest structure first. Towards this goal, at training time, the LAT model tries to learn the easiest tree structure to represent the resulting coreference clusters. This is done via updating the parameters to make the system decode a tree that is maximally similar to the "gold standard tree". However, the gold standard tree structure is not given in the annotation (only the cluster is given, which corresponds to a set of different trees). The model then chooses the highest-scoring tree in the set of gold trees as the pseudo gold tree (hence the name Latent Tree). The fact that the tree has the highest score shows that this tree is considered to be the easiest for the system.

In order to perform event argument linking, we choose the Semantic Role Labeling approach. To ensure coverage, we try to merge the results from two sources. The first source is the results of a couple Semantic Role Labeling systems: Semafor and AllenNLP SRL. The Semafor parser is a frame-based parser with broad coverage in terms of predicate diversity (e.g., it includes nouns and adjectives). The AllenNLP system is currently the best SRL system for verb predicates. In order to adapt to the domain and improve coverage on the target domain, we use an additional source of information, namely a system based on a set of curated rules that encode the argument types based on lexical evidences and distributed similarities. Finally, to create the final argument extraction result defined by the ontology, we manually created a mapping from the SRL roleset (Framenet and Propbank) to the AIDA ontology.

To rank the importance of the events and entities, we also adopted our salience prediction system. The salience prediction system is trained on an automatic harnessed dataset from Annotated New York Times. We first automatically created the salience labels by comparing entities and events in the abstract and the full text documents. We then trained a Kernel based Centrality Estimation (KCE) model to predict the salience of the discourse elements (DEs, including entities and events). The KCE model utilizes features from two perspectives. First, we compute the similarity scores between the document DEs and the target DE. The similarity scores are aggregated via a Kernel estimation function, as described in our SIGIR (Liu et.al. 2018) paper. Our system then concatenated the kernel scores with a set of discrete feature values to create the input to the final classification layer, which is a simple linear regression layer that produce the salience ranking score of the DE (the discrete features include common features such as frequency, sentence location). In experiments, we found that the Kernel scores can effectively capture script-like information and greatly improve the accuracy of salience ranking. And importantly,

the large dataset size allows us to create a good estimate for salience prediction, and make our system to be generalizable to different domains.

Performance: Without sufficient in-domain annotation and evaluation, we here report the performance of our systems on the test data in the original domain. The event nugget detection system achieves an F-score = 0.45 in event and type classification on the TAC-KBP 2016 test. The event coreference system achieved an average score on 4 metrics (B-cued, CEAF-E, MUC and BLANC) of 0.78. The event salience system achieves Precision@1 = 0.5 on the Annotated New York Times test data.

Russian/Ukrainian event, entity, and relation extraction: We are developing a multi-lingual module for extracting concept mentions of events, relations and entities in given domains. Our approach is designed for domains where sufficiently large annotated corpora are not available and purely statistical methods produce poor results. This is especially true for less resource-rich languages such as Ukrainian.

COMEX uses a domain-specific ontology that is common to all three the languages and corresponds at highest levels to the AIDA ontology. The ontology defines and organizes concepts such as *tank or *attack. We use the '*' prefix to distinguish concepts from English words. Concepts are common to all languages. Concepts *event, *relation and *entity are the top-level concepts and mapped to the AIDA ontology. All other concepts have one or more parents. For example, *tank has two parents: *vehicle and *weapon. Concepts can have attributes that are either literals such as strings or numbers or instances of other concepts. For example, *attack has an attribute Attacker which must be an instance of *person-like. The ontology specifies the allowed attribute types. COMEX entities are often more detailed than the LDC AIDA entities. For example, we have the *SU-25 concept which is a specific type of a military jet. Such specific concepts help coreference entities in TA1.

COMEX mentions are nodes in a mention graph. Each mention points to a concept and the mentions for the extracted attributes. It also records the span of the original text corresponding to the mention.

COMEX is driven by its lexicons. While the COMEX code is language-independent, the lexicons are language-specific. The lexicons connect words and phrases to concepts and specify the rules for word-sense disambiguation and attribute attachment. Currently, the lexicon entries are created manually, which is time-consuming. However, this effort is incremental and may compare favorably to the resources required for semantic annotation of sufficiently large corpora. We are working on the ways to partially automate this task.

The extractor tokenizes the text using an external tokenizer. After trying several we selected the NLTK tokenizer because it does not break up words such as "SU-25" and works better with Russian and Ukrainian. Note that COMEX can use any tokenizer that outputs CoNLL format.

Next, the extractor uses an external Universal Dependency (UD) parser. After trying several we selected UDpipe (Straka et al., 2016) because it produces significantly

better results for Russian and Ukrainian. Unfortunately, the UD parses are not always correct. COMEX fixes some of the systemic errors that we found (e.g., systematic assignment of the nominative case to Russian and Ukrainian nouns that end with a number, such as Boeing777). We may be able to find some fixes for incorrect prepositional phrase attachments, but other parsing errors are difficult to detect and to fix.

Given a UD parse tree, COMEX overlays it with a concept graph. In the first pass, we look up each word or its lemma in the lexicon. If the lexical entry has more than one sense for the part of speech (as specified in the UD parse tree), we apply the disambiguation rules listed in the word sense entry in the lexicon. We also have provisions for word sense disambiguation using BERT (Devlin et al., 2018) embeddings but, so far, didn't have to use it. The result of the first pass is the UD parse tree in which some of the nodes have concepts. Next, we go down the tree and connect concepts through their attributes. For example, the lexical entry for the phrase "shoot down" specifies the concept *attack and the attributes Attacker, Target, Instrument and Place. The Attacker attribute further specifies the rules of how to find the attacker in the UD parse tree starting from the node corresponding to "shoot down" (the "trigger" node). One of these rules says that it could be found in the "nsubj" position, provided that the voice of the trigger is "Act" (active). When we find a candidate for the Attacker, we verify if it fits semantically by checking if its concept descends from the concept specified in the lexicon (in our example, if it descends form *person-like). We found that the vast majority of the connection can be accomplished using a fairly small number of rules (so far, about 25). The rules, of course, are language-specific.

Recently we have been working on enabling appropriate light verb constructions, for example the ability for COMEX to infer that in "Separatists launched an attack on Ukraine", the attacker in the *attack event must be inferred from the launcher (i.e., that essentially the *launch event is empty).

The final step is the production of mentions from the concept graph. This is relatively straightforward: every node in the concept graph whose concept descends from an event produces an event mention. Relation and entity mentions are produced similarly. The attributes of the event and relation mentions are collected via the attribute links of the concept graph. The attributes of entities typically imply relations; for example, "Kramatorsk airport" produces the following concept graph fragment:

*airport ~related-to *Kramatorsk

The attribute "~related-to" implies a relation between *airport and *Kramatorsk (as a convention, we prefix the attributes that imply a relation with a '~'). COMEX uses rules to transform these attributes into implied relations. For example, a *location related to a *facility implies a "*located-near" relation.

COMEX was designed to favor precision over recall and to work as a complement to other extraction approaches.

2.3 TA1 Speech Processing

The goal of this module is to segment, diarize, and transcribe speech signals present in videos. The speech on such videos can be in three languages: Russian, Ukranian, or English. Therefore, this module should also be able to identify the language of each video to generate a proper transcription. Within OPERA, the output of the transcription is then piped to the appropriate TA1 text processing engine.

The input of our module is raw videos with audio. The output is time-segmented transcriptions accompanied by four different possible labels: Ukrainian, Russian, English, and Silence (non-speech). These labels are used by OPERA's downstream modules to interpret the text correctly. The Silence label is used when our module hypothesize that the segment does not contain speech information (e.g., background noise, music, silence).

We first construct three Automatic Speech Recognition (ASR) systems based on Kaldi. Each ASR system is trained to recognize each language 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. We train each recognizer using a chain model (Povey et al., 2016) that obtains comparable performance to the state-of-the-art in publicly released datasets (e.g., WFJ, switchboard). Although the chain model uses closed-vocabulary setting; we are looking at ways of expanding this module towards open vocabulary by using character-based sentence piece units.

In addition to speech-to-text, we also need a module that performs language classification on the spoken data. This module analyzes the transcriptions of the whole video of all three systems and classify on one of the four possible classes: Ukrainian, Russian, English and Silence. First, we classify all silent segments by computing the word rate produced by all three recognizers. If the average number of words per segment is lower than three we classify a video as silence. Then, in the rest of the videos, we tried two strategies, assuming in both that the speech of each video belongs to one of the languages. In the first strategy, we computed the perplexities of the outputs (normalized by length) on the text generated by each recognizer for a whole video. These perplexities were computed with an ngram language model trained with text belonging to each individual language. The language model obtaining the lowest perplexity was selected as final language. In the second strategy, we summed the posterior probabilities generated by each individual recognizer without applying any normalization strategy. Because we observed that systems with matching languages generate longer transcriptions, this strategy promotes systems that generate more words on each segment, and presents an improvement over a similar strategy that proved successful in earlier work (Metze et al., 2000). While not computationally efficient, this approach is robust and fault-tolerant. This approach also stores text lattices that can be re-scored quickly with a new language model, so that new hypotheses can be derived if and when new context information becomes available, as required by project goals.

We have also developed several experimental ASR systems that use text and visual features as context for adaptation and re-interpretation of the original audio. The main issue of such systems is that are based on an end2end framework and in some cases (such as insufficient in-domain training data), the performance can be lower than the performance of well optimized conventional baselines, such as the systems used here. Still, we document their performance in English language and publications such as (Caglayan, 2019; Sanabria et al., to appear).

Performance: Regarding individual WER performance for each ASR language specific system, we created two development sets, one for Russian (LDC2018E75) and another for Ukrainian (LDC2018E73 and LDC2018E74). (Because we did not have any test or development set for language identification, we could not run a formal experiment on it.) We obtained the following word error rate performance: Russian: 34.8% WER and Ukrainian: 32.9% WER.

2.4 TA1 Image and Video Processing

The purpose of the visual pipeline is to process the images and videos provided by the current challenge to identify visual entities that are added into the knowledge base to be further used as possible evidence for hypothesis resolution in a downstream task.

The visual pipeline processes the multimedia from the documents by detecting objects in images and video keyframes, infers semantics from whole images, maps the image concepts to the AIDA ontology, detects characters to extract pieces of text in the images, and performs person re-identification through facial recognition. It is shown in Figure 1.

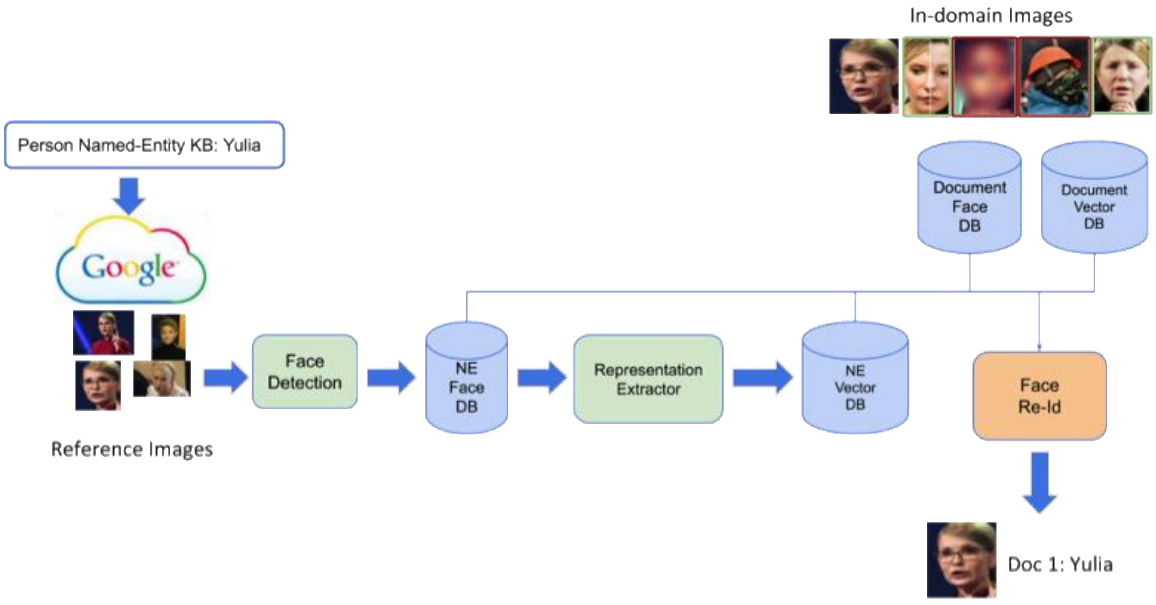


Figure 1. OPERA visual processing pipeline.

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) model that was pre-trained on the ImageNet dataset (Deng et al., 2005), and fine-tuned with 3-scale images from Open Image dataset (Kraslin et al., 2017). The class pool for this dataset is 500 that were filtered from Freebase Entities and Google KG.

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. This model is currently achieves a state-of-the-art performance on the ICDAR2013 and ICDAR2015 challenges.

Finally, the visual pipeline is capable of detecting faces and re-identifying 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 by downloading 15 images per person and using DBSCAN (Khan et al., 2014) to filter outliers.

The entities detected such as persons and objects while processing the images and videos are considered to be part of the TA1 output, while the cross-image and cross-medium coreference is part of the TA2 output of our system.

2.5 TA1: Building the KBs

Knowledge Aggregator: Inference and Integration Toolkit: The Knowledge Aggregator (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 (www.isi.edu/isd/LOOM/PowerLoom/) 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 we integrated PowerLoom with a triple store and graph database called Blazegraph (www.blazegraph.com) 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.

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 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 hypothesis*. 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 JSON-LD format.

2.6 TA2 Cross-Document/Cross-Language Coreference

Cross-document coreference: The purpose of this module is to accept as input clusters of (supposedly) coreferent entities or events, each cluster from a single

document, and to produce larger clusters that group together those input clusters whose entities and events also corefer across documents.

In summary our approach is to first represent each coreference chain as an embedding vector (using the BERT contextual embedding model) and merge them by agglomerative clustering. To improve performance we employ the reference KB entity links to 'warp' the embedding space and hence bias the distance metric for clustering.

Given a collection of mention chains $C = \{c_1; c_2; \ldots c_M\}$, where each c_i is a (possibly singleton) l_i-sized sequence of mentions $m_{1i};m_{2i}...m_{li}$ and a language O ∈ {Russian, Ukrainian, English}, we want to learn a partition of exhaustive and non-overlapping sets (i.e., clustering). Additionally, we are provided link information from multiple sources such as DBPedia (Auer et al., 2007), $K = \{(c_i; n_{ij}; K_i; p_{ij})\ldots\}$, where K_i is the i-th knowledge source, n_{ij} is one of its nodes, and p_{ij} is a confidence score. Our algorithm broadly proceeds as follows:

1. Condense Chain: Represent each mention chain c_i as a concatenation of mention span tokens, separated by an appropriate separator token [*SEP*].

2. Token Representation: Use a pretrained contextual representation function f to obtain a per-token representation for each chain token. Specifically, we use the BERT (Devlin et al., 2018) mechanism, as described below. Based on the language O, the method of representation may vary; specifically, we use plain BERT for English but BERT-multilingual for Russian and Ukrainian.

3. Pooling: Use an inverse-frequency weighted average pooling to get a single representation for each mention chain. Specifically, we use the method from (Arora et al., 2016). We refer to this space of mention-based representations as R, and distance in this space as $D_R(c_i; c_j)$.

4. Incorporate Knowledge: Given two mention chains, we find the fraction of knowledge sources in which they are co-referring, along with their respective confidences. We compute a distance $D_K(c_i; c_j)$ based on the Jenson-Shannon Divergence between the node-sets of c_i and c_j. This is used to scale the distances in the text representation space R. The whole process is outlined below. This gives us final distances $D_{final}(c_i; c_j)$.

5. Agglomerative Clustering: Agglomerative clustering iteratively merges the current closest clusters in bottom-up fashion, starting with singleton clusters of each element. Specifically, we use the *scikit-learn* library's implementation of hierarchical linkage-based clustering[2]. This clustering is done on the $D_{final}(c_i; c_j)$ distance matrix.

We experimented with various ways to improve the performance by adjusting the embedding-space representation.

Contextual embeddings use a pretrained model to represent each token of a sentence in context, represented in a space we denote as *R.* They are pretrained

[2] scipy.cluster.hierarchy.linkage

using language modeling-like losses on large corpora. This is a much richer form of representation to the earlier word embeddings such as FastText (Bojanowski et al. 2017), since different token occurrences of the same word type can be represented differently. Here, we use BERT (Devlin et al., 2018) for contextual embedding. Specifically, we use the BERT-base-uncased and BERT-base-uncased-multilingual models from the canonical pytorch-transformers[3] library for English and Russian or Ukrainian mention chains respectively. The "sentence" we provide as input to BERT is the concatenated sequence of mention span tokens from the mentions in the chain.

Representation	% Seedling Pair F-score
BERT	82.37
+FULL-CONTEXT	−2.9
+WINDOW-CONTEXT	−1.3
FASTTEXT	77.10
STRING-MATCH	76.6

Table 1. Performance of various representation encodings.

We observe from Table 1 that BERT is significantly superior to using word embeddings such as FastText. We also observed that BERT representations make the clustering algorithm have lower variance w.r.t. its hyperparameters, making it much simpler to tune.

How much context is required? We find that including context beyond the immediate mention spans, such as entire mention sentences is detrimental. Another alternative explored was using a small window of context, although this is better than the entire mention sentences case, it is still worse than using just mention spans.

Are we doing better than string-match? To sanity-check that our methods are not simply doing simple string-match, we experiment with a baseline where each mention chain is represented as a bag of character trigrams, encoded as 0–1 features. Our results establish that we are able to surpass StringMatch.

Incorporating knowledge: We have knowledge from multiple sources, each with associated confidence scores. Given a pair of mention chains c_i and c_j, we first represent both as a distribution over nodes, combining the links and confidence scores from different knowledge sources, e.g., {0.91,0.08,0.01} where the dimensions represent BARACK OBAMA, USA, and CHICAGO. We then find the Jensen-Shannon (JS) Divergence between c_i and c_j, denoted as $D_K(c_i; c_j)$.

Using this newly found knowledge-based distance, we then scale the pairwise distances of the respective pair in the original representation space. The final distance is given by $D(c_i; c_j) = D_R(c_i; c_j) \times (1 + D_K(c_i; c_j))$ — thereby perturbing the space to incorporate knowledge information.

[3] https://github.com/huggingface/transformers

This method also aids in improving our cross-lingual correspondence. Since slightly different representation methods are used to project mention chains from different languages in the original text space *R*, there might be limited alignment between chains of different languages. However, since they are linked to the same knowledge sources, they would get aligned better when scaled using D_K.

2.7 TA2: Building the KBs

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 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.

2.8 TA3 Belief propagation for hypothesis creation

Belief propagation: LEAPFROG (Choudhary et al., 2019) is a novel probabilistic Belief Propagation framework built for the project 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. The goal of this work is not to judge the veracity of an information source but to provide the end user with a multifaceted view of an event. Along with presenting a ranked list of the most probable alternatives, we also provide the provenance/evidence pointers that lend support to the presented ranking.

The underlying probabilistic model of LEAPFROG is a belief graph (BG), which is our internal version of the KB. The belief propagation engine performs inference by calculating the marginal probabilities of the KB nodes (each one a Knowledge Element KE or cluster) to update the belief graph. LEAPFROG generates one “mini” belief graph per document as an intermediate step in the incremental BG construction. The BG nodes are KEs that represent entities, events and relations. A natural way to connect KEs to their uncertain properties is through random variables. The uncertainty is reflected in the variable's probability distribution. To propagate and, hopefully, reduce the uncertainties observed in the data, we construct factor graphs over the BG.

A factor graph is a probabilistic graphical model that represents factorization over several variables. It allows for efficient computation of marginal distributions using a sum-product algorithm, belief propagation (BP) (Pearl, 2014).

Dependencies between random variables are captured through factors. Suppose an observer reports seeing a truck but not sure if the truck is Spanish or French. We capture this with an observed entity *Obs* and two variables corresponding to the observed properties: *Obs.name* with distribution {truck: 1.0**}** and *Obs.affiliation* with distribution {Spanish: 0.5, French: 0.5}. Assume that we have two existing grounding candidates *Truck1* and *Truck2*. We also consider the possibility of the observed entity being a new truck (*Truck3*). *Obs.target* is the grounding variable for the observation with a uniform prior, {Truck1: 0.33, Truck2: 0.33, Truck3: 0.33}. We form a factor that connects these three variables as depicted on Figure 4. The local function of this factor takes three arguments —the instances of *Obs.name, Obs.a_liation, Obs.target*— and returns 1 if the entity-name and affiliation of the observation match the entity-name and affiliation of the grounding candidate and 0 otherwise. LEAPFROG has several common observation models with pre-built factors and local functions.

Belief propagation (BP) is a message-passing algorithm for performing inference over graphical models. It efficiently computes the posteriors of the target variables and the algorithm stops when the prior and the posterior distributions of the target variables converge, as measured by their Kullback-Leibler distance. We explicitly list only the highest posteriors, relegating the lower ones to **other* or rejecting them. For example, if the posterior probability of the new truck in the previous example is below a threshold, we do not create the new entity in the KB. Observation target variables serve as a co-reference resolution mechanism: given two observations, we can look at their grounding targets and calculate the probability of an intersection. LEAPFROG additionally performs knowledge completion for event roles as part of this process. If after BP, the model decides that the observation is an alternative interpretation, a union of the observed mention's arguments and arguments of the existing event KE to which the observation was resolved to, is taken. The update equations (Gormley and Eisner, 2014) can be seen below, where x_i are variables, α are the factors, $N(x)$ are neighboring nodes of x, ψ_α are the factor potentials, which in our case are the identity local functions, and $\mu_{x\to y}$ are incoming messages from node x to node y.

$$\text{Belief of variable } x_i\text{: } b_i(x_i) = \prod_{\alpha \in N(i)} \mu_{\alpha \to i}(x_i)$$

$$\text{Outgoing message from variable } x_i\text{: } \mu_{\alpha \to i}(x_i) = \prod_{\alpha \in N(i) \setminus \alpha} \mu_{\alpha \to i}(x_i)$$

$$\text{Factor belief: } b_\alpha(x_\alpha) = \psi_\alpha(x_\alpha) \prod_{i \in N(\alpha)} \mu_{i \to \alpha}(x_\alpha[i])$$

$$\text{Factor message: } \mu_\alpha(x_i) = \sum_{x_\alpha : x_\alpha[i] = x_i} \psi_\alpha(x_\alpha) \prod_{j \in N(\alpha)_\alpha} \mu_{j \to \alpha}(x_\alpha[i])$$

LEAPFROG constructs the KB in an incremental fashion and resolves the observations sequentially. This causes the beliefs to be biased towards interpretations observed early in the process.

We implement a merging strategy for merging existing KEs in light of new knowledge. This new knowledge can be external (e.g. cross-lingual co-reference) or acquired by the model in the process of KG construction. Consider two KEs Spanish truck (KE**1**) and French truck (KE**2**). Some later events/relations reveal that these were in fact the same truck and hence need to be merged. However, it is not enough to just assign KE**1** = KE**2**, because these entities could be arguments of other event and relation KEs. Therefore, for every merge we follow these steps:

1. Substitute KE**2** with KE**1** across all corresponding events and relations where KE**2** is the argument.
2. For each updated event and relation KE, run the BP again by selecting relevant candidates from existing KG. This is done as merging entity KEs could lead to the merging of event and relation KEs as well. For example, the two separate event KEs Transport by Spanish truck and Transport by French truck should also get merged. We leverage the same factor graph model for merging these events/relations as described above.

To keep the BP computation tractable and to avoid probability underflow, we prune the graph nodes by removing instances having very low probability. Currently, we only prune entity nodes as the number of entities is much larger than the number of events and relations. Periodically, all entity nodes are pruned removing interpretations that have probability less than the specified threshold. Period *k* is a hyperparameter that should be chosen based on the domain or quality of corpus, as having too small a k could run the risk of pruning away possibilities that could be important later. This risk is higher for event and relation arguments, and that is another reason why we focus only on entity nodes.

Evaluation: We used LDC domain documents (news articles and social media data from Twitter) in English, Russian, and Ukrainian. These documents are centered around three conflicting scenarios: T101 (Crash of Malaysian Air Flight MH17), T102 (Flight of Deposed Ukrainian President Viktor Yanukovych), T103 (Who Started the Shooting at Maidan?). Table 2 shows the number of root documents, including both text and images within that document, for each topic and language. We use the following LDC-provided annotations: entity, event and relation mentions; observation uncertainties (hedging and negation) at both the event and argument level. Although this dataset is relatively small, comprising only 432 documents in total, to the best of our knowledge this is the first of its kind that contains significant amount of conflicting information and has annotation for both event and relation arguments as well as the observation uncertainties. In addition to mention annotations, LDC provided a KG with Knowledge Elements (KEs) representing entities, events and relations. The LDC mentions were grounded to these KEs. We used the LDC grounding as the “gold standard” to evaluate our system's grounding accuracy.

In Table 2 we report results. The metrics for evaluation are precision and recall, which check the grounding for each mention for entity, event, and relation, where Exact Mention Match is an indicator function that returns 1 if all the mentions grounded to a particular KE match exactly the gold KE's mentions. We see that as expected for entities precision is close to 100%. For events and relations, we observe a similar performance. We note that we use gold mentions but we do not use the gold KEs, so within and cross-document entity/event/relation co-reference is purely handled by LEAPFROG along with cross-lingual entity/event/relation co-reference.

	Entity	**Event**	**Relation**
doc-sp KG	79.78 / 98.58 / 88.18	52.91 / 79.50 / 63.53	41.89 / 83.22 / 55.72
doc-sp KG w/o NIL	98.51 / 98.47 / **98.48**	77.11 / 80.21 / **78.62**	77.02 / 85.07 / **80.84**

Table 2. Results of mini-evaluation with and without NIL clusters.

Error analysis: Qualitative analysis reveals two main possible grounding errors:

- Commonsense and world knowledge: Currently, the model uses only the provided information for constructing the KG. However, there are entities that have multiple identifiers that cannot be captured without external knowledge. For instance, the *flight MH17* is referred as {*MH17, Malaysian passenger plane, Malaysian 17, Flight AOJ92C*}. Without prior knowledge, it is difficult to conflate these entities, further affecting the event and relation resolution.
- Merging knowledge elements: Currently, LEAPFROG merges existing KEs based on external knowledge in the form of cross-lingual entity co-reference, but doesn't leverage the knowledge it has acquired so far. So, KEs like *BUK missile* and *SA-11 Buk missile systems* that refer to the same missile are not merged in the current setting, even though they always seem to occur under similar circumstances. Some higher-order inference is required on top of LEAPFROG's current implicit coreferencing to identify such merges.

Hypothesis formation and matching: Given the output of LEAPFROG as a graph of weighted alternatives, our 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. These are then evaluated by AIDA assessors.

2.9 Assorted Representational Issues

Ontology: We used the AIDA Ontology but eventually focused, for the evaluation, on just the LDC Annotation Ontology. For entities and events we manually created mappings from our vocabulary and the SRL roleset (Framenet and Propbank) to the AIDA ontology.

As described above we found it more convenient in the COMEX parser to develop our own internal ontology and map that periodically into the program ontologies.

Translation and integration: For OPERA we developed a frame-based CSR representation and interchange language based on JSON-LD (json-ld.org) which uses

an easy to read and manipulate JSON syntax, while at the same time being a legal RDF syntax which makes it easy to process and query with tools like Blazegraph. 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 multi-media 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.

2.10 Challenges

Without a doubt, AIDA is one of the most, if not the very most, challenging DARPA program that any of the OPERA team members has ever participated in. While the individual TA1/TA2/TA3 tasks are very technically challenging, an even greater challenge is their inherent joint complexity, which is of numerous types:

- Task complexity: This includes text extraction, audio processing, multi-lingual, image and video extraction, KB construction and refinement, query processing, conflict detection, hypothesis generation, explanation, and more.
- Data complexity: The data includes news, discussion forums, tweets, videos, images, multitude of input and result formats, etc.
- Ontology complexity: The ongoing disagreements about ontology contents and organization, and the continuing discrepancy between what the program scientists request and what the data providers produce, is a constant source of difficulty. This is worsened by the relatively large number of types (187 entity types, 139 event types, 49 relation types, 604 role types) and their complex subtyping.
- Code complexity: At time of writing the OPERA system comprises millions of lines of code, most of it written by others and imported and retrained as needed. This includes WordNet, Framenet, Wikipedia, DBPedia, Freebase, parsers, NER and coref engines, single- and multi-lingual embeddings, gazetteers, and numerous more.
- System complexity: OPERA is a distributed system, only loosely integrated via files. The university-based environment is not susceptible to more professionally managed integration.
- Processing complexity: The size and complexity of the data makes processing expensive. For example, EDL on 100K docs took several days to run, and much longer to retrain.
- Integration complexity: The requirement to not only produce a running system, but also to absorb results produced by other teams, and to p produce modules that can be run by the integrator TA4 team, adds significantly to the effort and complexity of the work.

What has helped are several fortunate factors. There is now a tremendous amount of high-quality resources and code at your fingertips right now and for free; as well as a lot of support for component interoperation, integration and deployment (including Web services, REST APIs, Java RMI, Python, Dockerization, UIMA, ADEPT,

ARGO, WINGS, and more). Large-scale processing is much easier now thanks to Hadoop, Spark, Amazon EC2, S3, Pegasus,, and data integration is also easier thanks to schema mapping, query translation, the semantic web, and more.

Still, it is not helpful to build very complex systems merely because we can. While adding another 100,000 lines of code is just a few clicks or git commands away, building large-scale software systems requires careful, well-managed engineering. In a university setting most of our work remains component-focused, developed by individual students in somewhat isolated projects, with the goal not only of building OPERA but also doing publishable research. The general focus is on better components, not so much on putting them together better. That is, we still don't exactly know how to make *more* into *better.* And significantly hampering our research is the continued lack of reliable and consistent data against which to evaluate.

2.11 Conclusion

OPERA has exercised the team at CMU and USC/ISI to a degree we have not yet experienced. We believe it is the same experience for other teams in the AIDA program. We look forward to working with them, and with others in the AIDA family, collaboratively in order to explore multi-lingual multi-media information extraction, integration, and hypothesis formation together.

References

Arora, S., Y. Liang, and T. Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Auer, S. C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pp. 722–735. Springer.

Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, vol 5, pp. 135–146.

Çaglayan, O., R. Sanabria, S. Palaskar, L. Barrault, and F. Metze. 2019. Multimodal grounding for sequence-to-sequence speech recognition. *Proc. ICASSP*, IEEE.

Choudhary, A., A. Gershman, and J. Carbonell. In prep. LEAPFROG: Adapting Belief Propagation for Knowledge Graph Construction. *Submitted to the AKBC Conference.*

Deng, J. et al. 2009. Imagenet: A large-scale hierarchical image database. *IEEE conference on computer vision and pattern recognition*. IEEE.

Devlin, J., M-W. Chang, K. Lee, and K. Toutanova. 2018. Bert: Pretraining of deep bidirectional transformers for language understanding. arXiv preprint 1810.04805.

Gormley, M. and J. Eisner, 2014. Structured belief propagation for NLP. *Proceedings of the 52nd ACL conference: Tutorials*, pp. 9–10.

Khan, K., et al. 2014. DBSCAN: Past, present and future. Proc of t*he Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*. IEEE.

Krasin I., Duerig T., Alldrin N., Ferrari V., Abu-El-Haija S., Kuznetsova A., Rom H., Uijlings J., Popov S., Veit A., Belongie S., Gomes V., Gupta A., Sun C., Chechik G., Cai D., Feng Z., Narayanan D., Murphy K. 2017. *OpenImages: A public dataset for large-scale multi-label and multi-class image classification,* Available from https://github.com/openimages

Krishna, R., et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision,* vol 123.1, pp. 32–73.

Lin, T-Y., et al. 2014. Microsoft coco: Common objects in context. *European conference on computer vision*. Springer.

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

Liu, Z., Xiong, C., Mitamura, T., and Hovy, E. 2018. Automatic Event Salience Identification. *Proceedings of the EMNLP conference*.

Liu, Z., Araki, J., Mitamura, T., and Hovy, E. 2016. CMU-LTI at KBP 2016 Event Nugget Track. *Proceedings of the 2016 Text Analysis Conference (TAC'16).*

Pearl, J. 2014. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Elsevier.

Metze, F., T. Kemp, T. Schaaf, T. Schultz, and H. Soltau. 2000. Confidence measure based Language Identification. *Proc. ICASSP*. IEEE.

Povey, D., et al. 2016. Purely sequence-trained neural networks for ASR based on lattice-free MMI. *Proc. Interspeech*.

Ren, S., et al. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*.

Sanabria, R., S. Misra and F. Metze. (to appear) The Printed Word For Contextualized Speech Recognition.

Straka, M., J. Hajic, and J. Straková. 2016. UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. *Proceedings of the LREC conference,* pp. 4290–4297.

Suchanek, F.M., G. Kasneci, and G. Weikum. 2008. Yago: A large ontology from Wikipedia and Wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, vol *6*(3), pp. 203–217.

Szegedy, C., et al. 2016. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*.