# GAIA - A Multi-media Multi-lingual Knowledge Extraction and Hypothesis Generation System

**Tongtao Zhang**[1], **Ananya Subburathinam**[1], **Ge Shi**[1], **Lifu Huang**[1], **Di Lu**[1], **Xiaoman Pan**[1], **Manling Li**[1], **Boliang Zhang**[1], **Qingyun Wang**[1], **Spencer Whitehead**[1], **Heng Ji**[1]
[1] Rensselaer Polytechnic Institute
jih@rpi.edu
**Alireza Zareian**[2], **Hassan Akbari**[2], **Brian Chen**[2], **Ruiqi Zhong**[2], **Steven Shao**[2], **Emily Allaway**[2], **Shih-Fu Chang**[2], **Kathleen McKeown**[2]
[2] Columbia University
sc250@columbia.edu, kathy@cs.columbia.edu
**Dongyu Li**[3], **Xin Huang**[3], **Kexuan Sun**[3], **Xujun Peng**[3], **Ryan Gabbard**[3], **Marjorie Freedman**[3], **Mayank Kejriwal**[3], **Ram Nevatia**[3], **Pedro Szekely**[3], **T.K. Satish Kumar**[3]
[3] Information Sciences Institute, University of Southern California
mrf@isi.edu
**Ali Sadeghian**[4], **Giacomo Bergami**[4], **Sourav Dutta**[4], **Miguel Rodriguez**[4], **Daisy Zhe Wang**[4]
[4] University of Florida
daisyw@ufl.edu

## 1 Introduction

An analyst or a planner seeking a rich, deep understanding of an emergent situation today is faced with a paradox - multimodal, multilingual real-time information about most emergent situations is freely available but the sheer volume and diversity of such information makes the task of understanding a specific situation or finding relevant information an immensely challenging one. To remedy this situation, the Generating Alternative Interpretations for Analysis (GAIA) team at DARPA AIDA program aims for automated solutions that provide an integrated, comprehensive, nuanced, and timely view of emerging events, situations, and trends of interest. GAIA focuses on developing a multi-hypothesis semantic engine that embodies a novel synthesis of new and existing technologies in multimodal knowledge extraction, semantic integration, knowledge graph generation, and inference.

In the past year the GAIA team has developed an end-to-end knowledge extraction, grounding, inference, clustering and hypothesis generation system that covers all languages, data modalities and knowledge element types defined in AIDA ontologies. We participated in the evaluations of all tasks within TA1, TA2 and TA3. The system incorporates a number of impactful and fresh research innovations:

- **Generative Adversarial Imitation Learning for Event Extraction:** We developed a dynamic mechanism – inverse reinforcement learning – to directly assess correct and wrong labels on instances in entity and event extraction (Zhang and Ji, 2018). We assign explicit scores on cases – or **rewards** in terms of Reinforcement Learning (RL). This framework naturally fits AIDA's future setting of streaming data because we adopt discriminators from generative adversarial networks (GAN) to efficiently assign different reward values to instances on different difficulty levels and different epochs.

- **Global Attention:** Popular deep learning methods have modeled Information Extraction (IE) as sequence labeling problems based on distributional embedding features, and led to improvements in quality for some low-level IE tasks such as name tagging and event trigger labeling (Chiu and Nichols, 2016; Lample et al., 2016; Zeng et al., 2014; Liu et al., 2015; Nguyen and Grishman, 2015b; Yang et al., 2016; Nguyen and Grishman, 2015b; Chen et al., 2015; Nguyen and Grishman, 2015a, 2016; Feng et al., 2016b; Huang et al., 2017a; Zhang et al., 2017a; Lin et al., 2018; Shi et al., 2018; Zhang et al., 2018a). However, more advanced extraction problems such as relation extraction and event argument labeling require us to cap-

ture global constraints that go beyond the sentence level, as well as inter-dependency among knowledge elements (Li et al., 2013; Li and Ji, 2014; Li et al., 2014). In the GAIA system we tackle this challenge by introducing a novel document-level attention mechanism (Zhang et al., 2018b) along with linguistic patterns and global constraints.

- **Multi-media Multi-lingual Common Semantic Space:** We developed a novel common semantic space for unifying cross-media and cross-language semantics, and applied it to encode visual attention to improve entity extraction (Lu et al., 2018) and event extraction (Zhang et al., 2017b), and ground entities detected from texts into images and videos.

- **Hierarchical Inconsistency Detection:** We developed a novel inconsistency detection method. Hierarchical entity matching and ontologically induced functional dependencies are used to detect logical inconsistencies. Such logical features are also summarised into an inconsistency metric showing the degree of inconsistency that each hypothesis bares.

- **Path Ranking Based Query Answering:** We are developing a novel hypothesis mining approach based on the path ranking algorithm (Lao et al., 2011). Our method creates a dual path-node embedding space which can be used to represent complex subgraphs, e.g., hypothesis.

- **Embedding Based Graph Clustering:** We are developing graph clustering algorithms based on various features and distance metrics. We also explore a novel approach inspired by "entity+graph" hybrid embedding methods. Such a method can be used to cluster similar/repeating subgraphs in the KG.

## 2 Multimodal Ontology Creation

A sufficiently expressive and refined ontology is paramount to the various IE tasks as it defines the target information that should be extracted. The ontology must offer a suitable target, with types that are reasonably disjoint and have a level of granularity such that a system can feasibly distinguish between them. Existing ontologies can be too coarse-grained (ACE or ERE) to offer clarity in settings that have conflicting information as in AIDA or too fine-grained (YAGO, WordNet, FrameNet, or VerbNet) to offer systems a clear target when performing the extraction. Therefore, we carefully design a multimodal ontology that is able to accurately capture the AIDA scenario relevant entities, relations, and events across multiple data modalities. We approach the development of the ontology by first creating a text-based ontology to serve as the foundation for the multimodal ontology. In parallel, we identify a set of salient visual types to form a visual ontology that is most relevant to AIDA. We then refine these ontologies and merge the textual and visual ontologies to form our final multimodal ontology.

### 2.1 Textual Ontology

#### 2.1.1 Entity Ontology

We use the YAGO ontology as a base entity ontology which we refine, restructure, and merge with the AIDA ontology to form our textual ontology. YAGO offers a greatly expanded set of entity types that are incredibly expressive and has annotations from sources, such as Wikipedia, that potentially already include entities of interest in the AIDA scenario. We start from a set of 7,309 YAGO type ontology.

We take a data-driven approach to form our ontology. First, we apply our entity discovery and linking system (Pan et al., 2017) to the AIDA Seedling data. This system is able to extract entities from the text data and perform fine-grained typing using the YAGO ontology. With the results of this system, we compute the frequencies of each entity type in the AIDA Seedling data. We filter these types using these frequencies, only keeping the top 250 most frequent entity types.

The data-driven approach offers the advantage that it ensures that the entity types included in the ontology are relevant and occur with relatively high frequency. However, further refinement is needed to ensure that the types are robust and truly cover the knowledge elements important to the AIDA program. Additionally, it is best to have the structure of the entity ontology be easily amenable to the event and relation ontology arguments so as to ensure that these ontologies may be used in concert with one another. Therefore, we carefully: 1) refine the set of 250 types and append any relevant types from the YAGO ontology that were mistak-

enly filtered out; 2) restructure the ontology such that the top level of the ontology is the same as the coarse-grained AIDA ontology, which is based upon ERE.

Starting from the 250 types, we prune types from the 250 and add back YAGO entity types based upon four criteria. First, the importance and/or relevance of a type to the AIDA scenarios. As a result, any truly irrelevant types, such as "*Skidder110605088: A person who slips or slides because of loss of traction*", are filtered out. Second, the level of granularity of a type, which further removes types like "*MapProjection103720443: A projection of the globe onto a flat map using a grid of lines of latitude and longitude.*" Third, a type's level of coherence with the existing AIDA ontology and target information, so if a YAGO type overlaps with an AIDA ontology type, then we use the AIDA ontology type to prevent redundancy. Lastly, we consider whether or not a type can only be represented as an entity or if the information conveyed by the type could be best represented as a relation or event, such as "*MaleSibling110286084: A sibling who is male.*" We then map the set of refined types determined using these criteria to the existing AIDA ontology. Throughout this process, as mentioned above, we favor the AIDA ontology types and merge our refined types into this ontology. This merging step ensures that the entity, event, and relation ontologies can be coherently used together.

The result of this process is a set of 163 fine-grained entity types that serve as the foundation for the multi-modal ontology. These types fit within the framework of the AIDA ontology, while still expanding the expressiveness of the ontology. Furthermore, due to our combination of a data-driven approach and careful design, we are able to ensure that the entity types are relevant, coherent, and are able to capture the entities most important to AIDA.

#### 2.1.2 Event Ontology

For events, we follow a similar procedure to the entity ontology. We use the AIDA event ontology as the base ontology into which we merge external event ontologies. The external event ontologies we consider are: CauseEx, FrameNet (Baker et al., 1998), VerbNet (Schuler, 2005), and PropBank (Palmer et al., 2005).

In contrast to the entity ontology, a data-driven approach to forming the event ontology is less plausible since the performance of event extraction systems (Li et al., 2013; Nguyen and Grishman, 2015a; Huang et al., 2017b; Zhang et al., 2017b) is markedly lower than entity discovery and linking (Pan et al., 2015; Lample et al., 2016; Pan et al., 2017). Thus, we manually merge and merge the external ontologies into the AIDA ontology.

Our first step is to map the external event types and argument roles to the AIDA ontology. During this process, it is likely that many of the types overlap between ontologies. In such instances, we set an order of precedence for overlapping event types and argument roles. This approach enables use to verify that the event types being incorporated are coherent and fit well within the AIDA framework. The following ordering is used to resolve these cases: 1) AIDA ontology; 2) CauseEx; 3) FrameNet; 4) VerbNet; 5) PropBank. For example, if CauseEx has an event type that is matches a FrameNet type, then we select the CauseEx type. After merging all 5 event ontologies, we prune and refine the entire ontology in a similar fashion to the entity ontology. Specifically, we only keep types that are: 1) relevant and salient to the AIDA scenario and 2) detectable by a system and human. This merging and refining process for the event ontology yields 114 salient event types along with their arguments.

### 2.2 Visual Enrichment

Visual data conveys knowledge elements that rarely appear in text. For instance, a news article may refer to an *explosion*, but not the *flames* and *smoke* that appear in a related image. Similarly, text may refer to an *organization*, which may only be visually represented by a *logo*. Text may mention *rescuing* in a *disaster*, but not refer to the *medical stretcher* that is used to carry an injured person.

In order to extract and represent multimodal knowledge in a unified language, and discover links between them, our ontology should cover visual concepts as well. In this section, we report how we enriched a text-driven ontology by adding visual concepts. Overall, we proposed a set of 150 entity types and 25 event types. Out of 150 entity types, only 26 were already covered by the text-driven concepts, while 124 were new and were added to the ontology.

To expand the ontology, we look for concepts

that are: (1) visually expressible, (2) important, and (3) related to the scenario of interest. If a concept has all the three characteristics, we determine if it already exists in our ontology, and if not, determine where we can add it in the ontology hierarchy. In this section, we review three techniques that we used to discover such concepts, and we share some examples and insights.

#### 2.2.1 Ontology-driven expansion

To expand the ontology, we search existing visual ontologies to discover new visual concepts. More specifically, given a list of visual concepts from a reference visual ontology, we select the ones that are important and relevant to the scenario, and add those to our target ontology. Here we use the list of 20,000 Open Images v3 (OI3) categories as our reference ontology, and the 9,000 YAGO types as our target ontology.

To measure the relevance of OI3 concepts to the seedling scenario, we extract a doc2vec (Le and Mikolov, 2014) representation from the AIDA Seedling Data Collection and Annotation Plan V2.0 and match it to the word embeddings of the reference ontology. We use cosine similarity from each concept embedding to the document embedding to sort the concepts in order of relevance. We choose a cut-off threshold by subjectively assessing the relevance of the sorted list to the scenario, which leads to the top 800 concepts being selected. Then we manually verify the relevance of each of the top 800 to the scenario, based on subjective intuition. Out of the 800 selected concepts, 306 were verified to be relevant, which indicates a precision of 0.38. We further prune this list by manually removing concepts that are too specific or infrequent, leading to 120 visual concepts.

The selected 120 visual concepts are all important and relevant visual concepts, and can be good choices for the target ontology. In the next step we try to link each of the 120 to the YAGO concepts. To this end, for each of the 120 selected concepts, we extract semantic embeddings and compare to all YAGO concept embeddings, matching to the highest similarity. We do this using 4 different methods for extracting semantic embeddings, which results in 4 possibly different matches for each selected concept. We manually assess the matches and select the best match. We consider 5 possible cases:

1. The selected visual concept and the YAGO match are identical, which makes it redundant.

2. The selected concept is a subtype of the YAGO match, e.g. police car which was matched to car. In these cases, the selected concept can be added as a child node to the YAGO ontology.

3. The YAGO match is a subtype of the selected concept, e.g. Emergency Vehicle which was matched to Ambulance. In these cases, the selected concept can be added as a node between the YAGO match and its current parent.

4. The selected concept and its YAGO matches are sibling concepts, e.g. Flight Engineer which was matched to Pilot. In these cases, the selected concept can be added as a child node to the parent node of the YAGO match.

5. The selected visual concept does not have any of the aforementioned relationships with any of the 4 matches from YAGO. In that case, the selected concept is completely new and should be manually added to YAGO. Examples are Missile Destroyer and military robot.

Out of the 120 selected visual concepts, 49, 45, 4, 11, and 11 fall under each of the 5 cases respectively. This means, while 49 of the selected concepts already exist in YAGO, 71 can be added to enrich YAGO. Moreover, discovering 45 child nodes compared to 4 parent nodes suggests that the visual concepts that are not covered by YAGO are usually more fine-grained than YAGO concepts and belong to the lower levels of the hierarchy.

#### 2.2.2 Data-driven expansion: bottom up

Whilst the first approach is limited to a predefined source ontology, the second approach starts from a corpus of scenario-related documents. We use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) along with a phrase mining tool (sha) to get a list of keywords. Each keyword should be frequent enough in the corpus to be picked. Therefore, they tend to be important and relevant to the scenario. We follow (Chen et al., 2014) to determine whether each keyword represent a visual concepts or not. For each keyword, we use the Google search API to crawl 100 images, and remove outliers using the method proposed in (Chen

et al., 2014). then we split the remaining images into training and test parts, and train a classifier on the training portion, to separate those images from other keywords. We determine a keyword as a visual concept if the accuracy of the trained model on the test images is above a threshold.

Using LDA and (sha), we find 238 and 336 concepts, out of which 61 and 72 were determined as visual concepts. Those selected concepts were linked and added to YAGO using the same approach described in section 2.1. out of 61 and 72 selected concepts, 19 and 39 concepts were new (not existing in YAGO), and therefore can be used to enrich the YAGO ontology. Examples are riot police, missile launcher, and anti-aircraft.

#### 2.2.3 Data-driven expansion: multimodal pattern mining

In this method, we utilize our recent work on multimodal pattern mining (Li et al., 2016). This work aims to find frequent image-text accompaniment patterns and assign them to a name. It extracts both visual and textual features (by an AlexNet and Word2Vec model, respectively), and then generates transactions to be used in association rule mining. The transactions are a 1000-D vector (with 0 or 1 entries each standing for a cluster presenting in the text), and a 256-D vector (with 0 or 1 entries each standing for activation of features maps of the pool5 layer in AlexNet). The feature map activations are calculated for each of 6x6 regions of the pool5 layer. Thus, there are 36 transactions for each image-caption pair. After running association rule mining, the most frequent patterns are named by TF-IDF and the area corresponding to that transaction is chosen for the visual representation for that name.

We ran this model on a dataset of 61k image-caption pairs collected in-house from the VOA website. Since association rule mining needs a discrimination annotation, we ran event extraction by trigger words on VOA dataset and grouped it to 33 ACE events. The model found 225 event-specific concepts. By manually looking, we chose 28 of them by the following criteria:

1. Event relatedness: The discovered concepts (pattern names and related images) are relevant to the event, and would be useful entities or attributes in an event schema for that event.

2. Visual semantic coherence: The visual pattern associated with the name is semantically consistent. Namely, the images shown under the pattern depict a coherent semantic concept, and not a mix of many different concepts. If the majority of images in a pattern are consistent, with few outliers, the pattern is considered to exhibit this property.

3. Text-visual matching: The pattern name correctly describes the semantic concept of the visual pattern.


Figure 1: Examples of discovered multimodal patterns.

Among the 28 hand-picked concepts, 14 of them were not present in the YAGO ontology. Examples are water cannon, protesters, and tear gas, as shown in Figure 1.

## 3 TA1 Text Knowledge Extraction

### 3.1 Approach Overview

As shown in Figure2, the text knowledge extraction system is a comprehensive end-to-end trilingual system. Note that all components without specifying language is language independent. Given tri-lingual text documents, we firstly extract named and nominal mentions using Bi-LSTM-CRFs extractors, and determine the coreference of named mentions based on collective entity linking and NIL clustering, followed by nominal coreference using Bi-LSTM Attention Network. After that, relations are extracted using Assembled CNN extractor, with post-processing via pattern ranking. Meanwhile, with entities as input, English events are extracted through GAIL, Bi-LSTM-CRFs and CNN, followed by attribute extraction through Logistic Regression hedge detection and syntax-based negation detection. For Russian and Ukrainian events, their triggers are extracted by Bi-LSTM extractor, followed by a CNN argument extractor. After that, coreference between

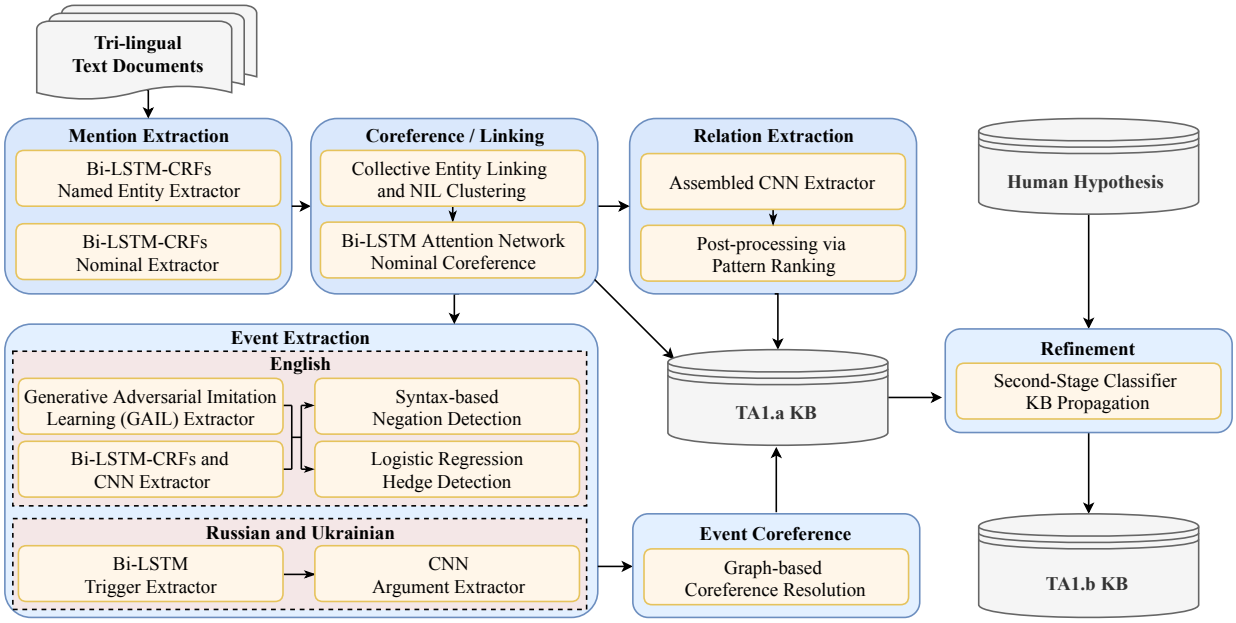


Figure 2: The Architecture of Text Knowledge Extraction

all events is determined with graph-based resolution. In the end, extraction results of entities, relations and events composite a document-level KB for TA1.a. Furthermore, this KB is refined by human hypothesis based on second-stage classifiers to generate TA1.b KB. The details of each component will be illustrated in following sections.

### 3.2 Mention Extraction

We consider mention extraction as a sequence labeling problem, to tag each token in a sentence as the Beginning (B), Inside (I) or Outside (O) of a name mention with one of seven types: Person (PER), Organization (ORG), Geo-political Entity (GPE), Location (LOC), Facility (FAC), Weapon (WEA) or Vehicle (VEH). Predicting the tag for each token needs evidence from both of its previous context and future context in the entire sentence. Bi-LSTM networks (Graves et al., 2013; Lample et al., 2016) meet this need by processing each sequence in both directions with two separate hidden layers, which are then fed into the same output layer. Moreover, there are strong classification dependencies among tags in a sequence. For example, "I-LOC" cannot follow "B-ORG". A CRF model, which is particularly good at jointly modeling tagging decisions, can be built on top of the Bi-LSTM networks. External information like gazetteers, brown clustering, etc. have proven to be beneficial for mention extraction. We use an additional Bi-LSTM to consume the external feature embeddings of each token and concatenate both Bi-LSTM encodings of feature embeddings and word embeddings before the output layer.

We set the word input dimension to 100, word LSTM hidden layer dimension to 100, character input dimension to 50, character LSTM hidden layer dimension to 25, input dropout rate to 0.5, and use stochastic gradient descent with learning rate 0.01 for optimization.

### 3.3 Entity Linking

Given a set of name mentions $M = \{m_1, m_2, ..., m_n\}$, we first generate an initial list of candidate entities $E_m = \{e_1, e_2, ..., e_n\}$ for each name mention $m$, and then rank them to select the candidate entity with the highest score as the appropriate entity for linking.

We adopt a dictionary-based candidate generation approach (Medelyan and Legg, 2008). In oder to improve the coverage of the dictionary, we also generate a secondary dictionary by normalizing all keys in the primary dictionary using a phonetic algorithm NYSIIS (Taft, 1970). If an name mention $m$ is not in the primary dictionary, we will use the secondary dictionary to generate candidates.

Then we rank these entity candidates based on three measures: salience, similarity and coherence (Pan et al., 2015).

We utilize Wikipedia anchor links to compute

salience based on entity prior:

$$p_{prior}(e) = \frac{A_{*,e}}{A_{*,*}} \quad (1)$$

where $A_{*,e}$ is a set of anchor links that point to entity $e$, and $A_{*,*}$ is a set of all anchor links in Wikipedia. We define mention to entity probability as

$$p_{mention}(e|m) = \frac{A_{m,e}}{A_{m,*}} \quad (2)$$

where $A_{m,*}$ is a set of anchor links with the same anchor text $m$, and $A_{m,e}$ is a subset of $A_{m,*}$ which points to entity $e$.

Then we compute the similarity between mention and any candidate entity. We first utilize entity types of mentions which are extracted from name tagging. For each entity $e$ in the KB, we assign a coarse-grained entity type $t$ (PER, ORG, GPE, LOC, Miscellaneous (MISC)) using a Maximum Entropy based entity classifier (Pan et al., 2017). We incorporate entity types by combining it with mention to entity probability $p_{mention}(e|m)$ (Ling et al., 2015):

$$p_{type}(e|m,t) = \frac{p(e|m)}{\sum_{e \mapsto t} p(e|m)} \quad (3)$$

where $e \mapsto t$ indicates that $t$ is the entity type of $e$.

Following (Huang et al., 2017c), we construct a weighted undirected graph $G = (E, D)$ from DBpedia, where $E$ is a set of all entities in DBpedia and $d_{ij} \in D$ indicates that two entities $e_i$ and $e_j$ share some DBpedia properties. The weight of $d_{ij}$, $w_{ij}$ is computed as:

$$w_{ij} = \frac{|p_i \cap p_j|}{\max(|p_i|, |p_j|)} \quad (4)$$

where $p_i$, $p_j$ are the sets of DBpedia properties of $e_i$ and $e_j$ respectively. After constructing the knowledge graph, we apply the graph embedding framework proposed by (Tang et al., 2015) to generate knowledge representations for all entities in the KB. We compute cosine similarity between the vector representations of two entities to model coherence between these two entities $coh(e_i, e_j)$. Given a name mention $m$ and its candidate entity $e$, we defined coherence score as:

$$p_{coh}(e) = \frac{1}{|C_m|} \sum_{c \in C_m} coh(e, c) \quad (5)$$

where $C_m$ is the union of entities for coherent mentions of $m$.

Finally, we combine these measures and compute final score for each candidate entity $e$.

### 3.4 Within-document Coreference Resolution

#### 3.4.1 Name Coreference Resolution

For name mentions that cannot be linked to the KB, we apply heuristic rules described in Table 1 to cluster them within each document.

| Rule | Description |
|---|---|
| Exact match | Create initial clusters based on mention surface form. |
| Normalization | Normalize surface forms (e.g., remove designators and stop words) and group mentions with the same normalized surface form. |
| NYSIIS (Taft, 1970) | Obtain soundex NYSIIS representation of each mention and group mentions with the same representation longer than 4 letters. |
| Edit distance | Cluster two mentions if the edit distance between their normalized surface forms is equal to or smaller than $D$, where $D = length(mention_1)/8 + 1$. |
| Translation | Merge two clusters if they include mentions with the same translation. |

Table 1: Heuristic Rules for NIL Clustering.

For each cluster, we assign the most frequent name mention as the document-level canonical mention.

#### 3.4.2 Nominal Coreference Resolution

We takes in the output of both the entity linking and NIL Clustering system to predict the nominal coreference. Our main architecture is based on the End-to-End Neural Coreference Resolution(Lee et al., 2017). However, since we already know the mention types, we further simplify the coreference system by only computing scores between nominal and corresponding named entity with same type.

Given an document $D = \{y_1, y_2, ...y_n\}$, named entity span $S_a = \{s_{a_1}, s_{a_2}, ..., s_{a_m}\}$ with its type $T_a = \{t_{a_1}, t_{a_2}, ..., t_{a_m}\}$ and its cluster $C_a = \{c_{a_1}, c_{a_2}, ..., c_{a_m}\}$, nominal mention span $S_o = \{s_{o_1}, s_{o_2}, ..., s_{o_l}\}$ with its type $T_o = \{t_{o_1}, t_{o_2}, ..., t_{o_l}\}$, where $y_i$ is individual words, $n$ is the length of the document, $m$ is the number of named entity, $c_{a_i} \in \{1, 2, ..., v\}$, $v$ is the cluster size, $n$ is the number of the nominal mention, and each span $s_{a_i}$ or $S_{o_i}$ contains two indices START$(i)$ and END$(j)$. We compute word representation from a pre-trained Word2Vector(Mikolov et al., 2013) embeddings with 1-dimensional convolution neural networks

for character embeddings with window sizes of 2,3,and 4 characters. The character embeddings are random initialized and optimized during training. compute span representation, we first use BiLSTMs as encoders to get hidden states $h_{k,1}, h_{k,-1}$ of words where $1, -1$ represent the directional of LSTM. For word at position $k$, its representation $y_k^* = [h_{k,1}, h_{k,-1}]$. For each sentence, the LSTM is independent to improve the program efficiency.

Then for each span either named entity or nominal mention, we compute the soft attention(Bahdanau et al., 2014) over each word in the span:

$$\alpha_k = \omega_\alpha \cdot \text{FFNN}_\alpha(y_k^*) \quad (6)$$

$$\alpha_{t,k} = \frac{\exp(\alpha_k)}{\sum_{n=\text{START}(t)}^{\text{END}(t)} \exp(\alpha_t)} \quad (7)$$

$$\hat{y_k} = \sum_{n=\text{START}(k)}^{\text{END}(k)} \alpha_{t,k} \cdot y_t \quad (8)$$

where $\hat{y_k}$ is a weighted sum of word hidden states in span $k$, FFNN is the feed-forward neural network. Then the representation of $g_k$ a span $k$ is computed as:

$$g_k = [y^*_{\text{START}(k)}, y^*_{\text{END}(k)}, \hat{y_k}, \Phi_g(t)] \quad (9)$$

where feature vectors $\Phi_g(t)$ encode size of span $k$, and type pf span $k$.

Finally, to calculate nominal clusters, we first group each named entity span according to their cluster. Then for each nominal span $g_{o,i}$, we check with every name span $g_{a,j}$ with same type. For each pair of nominal span $g_{o,i}$ and name span $g_{a,j}$, we compute score $x(i,j)$:

$$x(i,j) = w_x \cdot [g_{o,i}, g_{a,j}, g_{o,i} \circ g_{a,j}, \Phi_x(i,j)] \quad (10)$$

where $\circ$ is the element-wise similarity feature vectors $\Phi_x(i,j)$ encode distance between two span. We first combine score of same cluster $q_e$ for span $s_{o_i}$:

$$P_{o_i}^{q_e} = \sum_{q_e | q_e = c_{a_j}} x(i,j) \quad (11)$$

where $q_e \in \{1, 2, ..., v\}$ We then compute conditional probability $P(o_i = q_e|D)$ for each cluster:

$$P(o_i = q_e|D) = P(o_i = q_e|C_a) \quad (12)$$

$$= \frac{\exp(P_{o_i}^{q_e})}{\sum_{e=1}^{v} \exp(P_{o_i}^{q_e})} \quad (13)$$

We then compute marginal log-likelihood of all correct clusters by the gold clustering:

$$\prod_{i=1}^{l} P(o_i = q'_e|D) \quad (14)$$

where $q'_e$ is the gold cluster. For English training, we used ACE2005, EDL2016, EDL2017 to train the system.

#### 3.4.3 Additional Russian/Ukrainian name coreference resolution

For Russian and Ukrainian, we performed additional name coreference as a post-process. We built a graph where each AIF cluster in the input was a node. In order, we applied the following rules to add edges between clusters if:

1. there was an exact match in string or stem between any name mentions in either cluster
2. there existed two strings from the two clusters where the Levenshtein distance between them was less than or equal to 2 and the size of the string or stem was at least 6 or 5, respectively.
3. (person names only) there existed a single-token string in one cluster which appeared as the second token of a two-token string in another cluster and did not appear as the second token of any other two-token string (e.g. *Smith* to *John Smith* unless *Jane Smith* also appears in the document).
4. (person names only) there existed a two-token string in one cluster which matched a two-token string in another cluster, accounting for morphological inflection of both strings.

A coreference cluster was made from each connected component of the resulting graph. "Stems" were determined in a simple, heuristic way by stripping a list of common Russian and Ukrainian inflectional affixes from the end of the string.

### 3.5 Relation Extraction

Most AIDA relation types can be mapped to ACE/ERE ontologies. For these types, we adopt a convolutional neural network (CNN) based model (Section 3.5.1). For those new types, such as `Genafl.Sponsor`, `Measurement.Count`, `Genafl.Orgweb`, we implement a rule based system (Section 3.5.2).

### 3.5.1 Relation Extraction for types in ACE/ERE

Our relation system takes named entity coreference results as input, predicting relations between each entity mention pairs occurred within sentences. We utilize a typical neural network architecture that consists of Convolution Neural Network (CNN) with piece-wise pooling as our underlying learning model.

Formally, given a source sentence $(s, e_1, e_2, r)$ where $s = [w_1, ..., w_m]$, for each word $w_{ik}$, we generate a multi-type embedding: $\tilde{v}_i = [v_i, p_i, \tilde{p_i}, t_i, \tilde{t_i}, c_i, \eta_i]$ where $v_i$ denotes a word embedding. Specifically, we use the Skip-Gram model to pre-train the word embeddings. $p_i$ and $\tilde{p_i}$ are position embeddings indicating the relative distance from $w_i$ to $e_1$ and $e_2$ respectively. $t_i$ and $\tilde{t_i}$ are entity type embedding of $e_1$ and $e_2$. $c_i$ is the chunking embedding, and $\eta_i$ is a binary digit indicating whether the word is within the shortest dependency path between $e_1$ and $e_2$. All these embeddings except pre-trained word embedding are randomly initialized and optimized during training. Thus the input layer is a sequence of word representations $V = \{\tilde{v}_1, \tilde{v}_2, ..., \tilde{v}_n\}$. We then apply the convolution weights $W$ to each sliding n-gram phrase $g_j$ with a biased vector $b$, i.e., $g_j^{'} = \tanh(W \cdot V) + b$. Considering different segments do not share the same weights in composing the semantic of sentences, we split all the $g_j^{'}$ into three parts based on the two entity mentions and perform piecewise max pooling, after which we concatenate the representations of the three segments and feed them into a fully connected layer and obtain a dense vector, then we use a linear projection function with a softmax as the relation classifier to determine the relation type.

**Training Details** For English training, we manually map DEFT Rich ERE and ACE2005 to AIDA schema and combine all the filtered resources. For Ukrainian and Russian, we employed a native speaker to annotate part of the seedling corpus. To further improve the system performance, we also adopted majority vote strategy to assemble the models for all three languages. Besides, we found that our model can hardly capture the global information due to instance level training. Thus, we extract some high confidence frequent relation patterns (Table 2) from training data as hard constraints to tackle this problem.

| Relation type | Relation pattern |
|---|---|
| Employment | Person of Organization |
| Organization_origin | Organization in Geography |
| Leadership | Geography prime Person |
| located_near | Person at Facility |
| Part_whole | Organization of the Organization |
| Manufacture | Geography Weapon |

Table 2: Examples of frequent relation patterns.

### 3.5.2 Relation Extraction for New Relation Types

We implement a rule-based component for `Genafl.Sponsor`, `Measurement.Count`, `Genafl.Orgweb` and relations based on the rules as follows (for English only):

**`Genafl.Sponsor`** We first use spacy to extract the shortest dependency path (SDP) between two entity mentions. If keywords like "pledge allegiance," "draws support from," "behind" occurred in the SDP and the length of SDP is shorter than a threshold, we will tag this relation as `Genafl.Sponsor`. We augmented a seed set of manually determined keywords with trigger words computed by our sentiment system for sentiment relations. The sentiment system was developed by training a relation extraction system on the Good-For/Bad-For corpus (Deng et al., 2013) to discover benefactive relations between entities and events. We split the data into 5 folds and each time a model is trained on 4 folds, leaving one of the folds out. To achieve high precision, we focus on those relation candidates where all of the 5 models predict positive. We also used the attention mechanism, trained using cue words in GFBF, to extract the trigger words: we consider the most attended word by each of the 5 model, and then take the majority vote across models.

**`Measurement.Count`** We first apply Stanford CoreNLP to extract all `NUMBER` $\mathcal{N}$. Then for $n_i \in \mathcal{N}$, we check if the word after $n_i$. If it is a part of a named entity extracted by our EDL system, we tag the relation between between them as `Measurement.Count`.

**`Genafl.Orgweb`** Our EDL system can detect urls as `ORG` and cluster them with other name mentions. We tag those urls as `Genafl.Orgweb` of the `ORG` named entities.

### 3.6 Event Extraction

#### 3.6.1 Event Extraction with Generative Adversarial Imitation Learning

We use Generative Adversarial Imitation Learning (GAIL) – an inversed reinforcement learning framework – to extract event triggers and detect argument roles. Event extraction consists of two steps: 1. trigger labeling: the system detects trigger words or phrases from natural language sentences; 2. argument role labeling: the system determines the relation between the triggers and entities detected from Section 3.2.

**Q-Learning for trigger labeling** We use Q-learning to label the triggers from sequences of raw text tokens.

We have Q-tables and Q-values

$$Q_{sl}(s_t, a_t) = \boldsymbol{f}_{sl}(s_t|s_{t-1}, \ldots, a_{t-1}, \ldots), \quad (15)$$

where $s_t$ denotes a state (of feature) for a token $t$.

$$s_t =< \boldsymbol{v}_t, a_{t-1} > . \quad (16)$$

and $a_t$ denotes an action (or label) from the agent (or extractor).

$$\hat{a}_t = \arg\max_{a_t} Q_{sl}(s_t, a_t). \quad (17)$$

The $v_t$ in Equation 16 is the context embedding of the token $t$. We use local embeddings concatenated with the following: 1. 200-dim **Word2Vec** embeddings, 2. 50-dim representation for Part-of-Speech (PoS) tags, 3. 100-dim of token embeddings randomly initialized and updated in the training phase, 4. 32-dim character embeddings. We adopt a Bi-LSTM to calculate $v_t$. We utilize another mono-directional LSTM to calculate Equation 15 and determine the labels from Equation 17.

In the training phase, we use Bellman Equation to update Q-values and Q-tables:

$$Q^{\pi^*}_{sl}(s_t, a_t) = r_t + \gamma \max_{a_{t+1}} Q_{sl}(s_{t+1}, a_{t+1}), \quad (18)$$

where $r_t$ denotes a reward based on the current state $s_t$ and action $a_t$.

The word embeddings and parameters on LSTMs are updated by minimizing the object

$$L_{sl} = \frac{1}{n} \sum_{t}^{n} \sum_{a} (Q'_{sl}(s_t, a_t) - Q_{sl}(s_t, a_t))^2, \quad (19)$$

where $Q'$ denotes the updated Q-table from Equation 18.

**Policy-gradient for argument role labeling** For the argument role labeling, we use another RL algorithm, policy gradient.

In this scenario, we also define a state (feature) for the agent (extractor)

$$s_{tr,ar} =< \boldsymbol{v}_{t_{tr}}, \boldsymbol{v}_{t_{ar}}, a_{t_{tr}}, a_{t_{ar}}, \boldsymbol{f}_{ss} >, \quad (20)$$

where $v$s are the context embedding as in Equation 16 and $\boldsymbol{f}_{ss}$ denote a sub-sentence embedding calculated using a Bi-LSTM which consumes the sequence of context embedding between the trigger and the argument.

We feed the state representation into an MLP to obtain Q-table which represents probability distribution of actions and we can determine the argument role with

$$\hat{a}_{tr,ar} = \arg\max_{a_{tr,ar}} Q_{tr,ar}(s_{tr,ar}, a_{tr,ar}). \quad (21)$$

We also assign a reward $R$ for the action (argument role) and we train the models by minimizing

$$L_{pg} = -R \log P(a_{tr,ar}|s_{tr,ar}). \quad (22)$$

**Dynamic Reward Estimation with GAN** Our framework includes a reward estimator based on GAN to issue dynamic rewards with regard to the labels committed by the event extractor as shown in Figure 3. The reward estimator is trained upon the difference between the labels from ground truth (expert) and extractor (agent). If the extractor repeatedly

The original GAN has a pool of real data, a generator $G$ and a discriminator $D$, while in our framework, the real data is ground truth (expert), the generator $G$ is the agent, and the discriminator $D$ is a reward estimator.

To train the discriminator, we minimize the following object function

$$L_D = -(\mathbb{E}[\log D(s, a_E)] + \mathbb{E}[log(1 - D(s, a_A))]), \quad (23)$$

where $s$ is the state representation from Equation 16 and 20.

We use a linear transform to estimate rewards

$$R(s, a) = 20 * (D(s, a) - 0.5), \quad (24)$$

#### 3.6.2 Event Extraction with Bi-LSTM

**Trigger Labeling** Event trigger detection remains a challenge due to the difficulty at encoding word semantics and word senses in various contexts. Previous approaches heavily depend on language-specific knowledge. However,

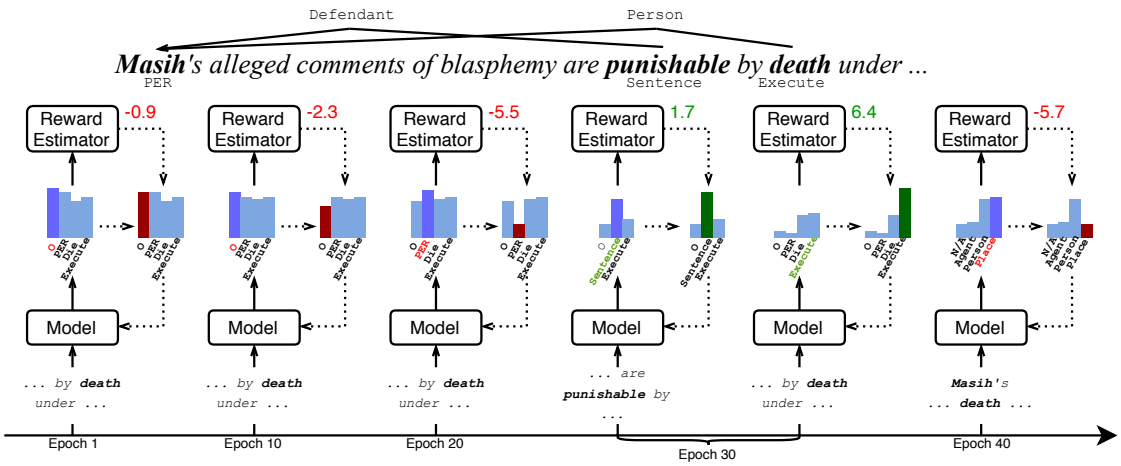


Figure 3: A diagram illustrating the dynamic rewards estimated by the GAN in the GAIL event extractor.

compared to English, the resources and tools for Chinese are limited and yield low quality. A more promising approach is to automatically learn effective features from data, without relying on language-specific resources.

We developed a language-independent neural network architecture: Bi-LSTM-CRFs, which can significantly capture meaningful sequential information and jointly model nugget type decisions for event nugget detection. This architecture is similar as the one in (Yu et al., 2016; Feng et al., 2016a; Al-Badrashiny et al., 2017).

Given a sentence $X = (X_1, X_2, ..., X_n)$ and their corresponding tags $Y = (Y_1, Y_2, ..., Y_n)$, $n$ is the number of units contained in the sequence, we initialize each word with a vector by looking up word embeddings. Specifically, we use the Skip-Gram model to pre-train the word embeddings (Mikolov et al., 2013). Then, the sequence of words in each sentence is taken as input to the Bi-LSTM to get meaningful and contextual features. We feed these features into CRFs and maximize the log-probabilities of all tag predictions of the sequence.

**Argument Labeling** For event argument extraction, given a sentence, we first adopt the event trigger detection system to identify candidate triggers and utilize the EDL system to recognize all candidate arguments, including Person, Location, Organization, Geo-Political Entity, Time expression and Money Phrases. For each trigger and each candidate argument, we select two types of sequence information: the surface contexts between trigger word and candidate argument or the shortest dependency path, which is obtained with the Breath-First-Search (BFS) algorithm over the whole dependency parsing output, as input to our neural architecture.

Each word in the sequence will be assigned with a vector, which is concatenated from vectors of word embedding, position and POS tag. In order to better capture the relatedness between words, we also adopt CNNs to generate a vector for each word based on its character sequence. For each dependency relation, we randomly initialize a vector, which holds the same dimensionality with each word.

We encode the following two types of sequence of vectors with CNNs and Bi-LSTMs respectively. For the surface context based sequence, we utilize a general CNNs architecture with Max-Pooling to obtain a vector representation. For the dependency path based sequence, we adopt the Bi-LSTMs with Max-Pooling to get an overall vector representation. Finally we concatenate these two vectors and feed them to two softmax functions: one is to predict whether the current entity mention is a candidate argument of the trigger, and the other is to predict final argument role.

**Training Details** For all the above components, we utilize all the available event annotations from DEFT Rich ERE and ACE2005 for training.

### 3.6.3 Event Extraction for New Event Types

We implemented a system, for the event types not included in English training data, to map framenet to AIDA ontology (Table 3).

We tag frame elements as event arguments only

| Framenet event type | AIDA event type |
|---|---|
| Conquering | Transaction.TransferControl |
| Criminal_investigation | Justice.Investigate |
| Sign_agreement | Government.Agreements |
| Inspecting | Inspection |
| Destroying | Existence.DamageDestroy |
| Damaging | Existence.DamageDestroy |

Table 3: Mapping from Framenet to AIDA event ontology.

if they overlap with named entities in our EDL outputs.

For `Government.Vote` and `Government.Spy` event types, which cannot be mapped from Framenet, we implement a rule-based system. We detect event triggers based on fuzzy string match with keywords [1]. And we further decide event arguments based on dependency trees (Table 4).

| Event Type | Syntactic Relation | Argument Role |
|---|---|---|
| Government.Vote | nsubj | Voter |
| | nmod:for | Candidate |
| | dobj | Candidate |
| | nmod:in | Place_Date |
| Government.Spy | nsubj | Agent |
| | nmod:for | Beneficiary |
| | dobj | Target |
| | nmod:in | Place_Date |

Table 4: Dependency Tree-based Argument Role identification for `Government.Vote` and `Government.Spy` events.

#### 3.6.4 Russian and Ukrainian Event Extraction

Event extraction for Russian and Ukrainian textual data is performed using a two-stage language-independent model. The first stage uses a Bi-LSTM (Bi-directional Long Short Term Memory network) which takes pre-trained word embeddings of each word in the sentence as input and generates a vector representation of each word in the context of the sentence. To capture the structured label sequences, such as *B-attack* followed by *I-attack* in the case of event triggers like *shot down*, we use the vectors from the Bi-LSTM with a Softmax classifier to obtain tag predictions for each word.

The second stage uses a Convolutional Neural Network (CNN) followed by a Softmax classification (Kim, 2014) to label the arguments of the event mention. Each input to the CNN is a portion of the sentence ('sub-sentence') between a named entity mention and the trigger word. If there are multiple named entities present in a sentence containing a trigger, all such sub-sentences are retrieved. Here, each word in an input sub-sentence be represented by its word embedding. We concatenate an additional dimension to each word here to represent its trigger type if it is a trigger word. This forms the input to the CNN. We use filters of size 3, 4, and 5 in the convolution layer, followed by a max-pooling layer, and a fully-connected layer to generate a single vector representation of the input sub-sentence. We then use this vector and perform a Softmax classification to identify and label arguments.

**Training Details** For Russian and Ukrainian languages, due to the absence of relevant training data, we use a native speaker to annotate three hundred documents in each language. This annotation is performed using the Brat annotation tool (Stenetorp et al., 2012), which we set up to allow annotations using the AIDA Event Ontologybrat. These annotated files serve as the training and development sets for the above event extraction system.

#### 3.6.5 Event Attribute Extraction

**Hedge Detection** We apply an uncertainty classifier (Vincze, 2015) to detects the uncertainty of each sentence. **Negation Detection** We apply negation detection toolkit (Gkotsis et al., 2016), given a sentence and a key-word (e.g. event trigger) as inputs.

### 3.7 Event Coreference Resolution

We apply our language-independent within-document Event Coreference model to all English, Russian and Ukrainian. The model is based on our previous work (Al-Badrashiny et al., 2017; Yu et al., 2016; Hong et al., 2015; Chen and Ji, 2009). We view the event coreference space as an undirected weighted graph in which the nodes represent all the event nuggets and the edge weights indicate the coreference confidence between two event nuggets. And we apply hierarchical clustering to classify event nuggets into event hoppers. To compute the coreference confidence between two events, we train a Maximum Entropy classifier using the features listed in Table 5.

[1] `Government.Vote`: vote, ballot, poll; `Government.Spy`: spy

| Features | Remarks(EM1: the first event mention, EM2: the second event mention) |
|---|---|
| type_subtype_match | 1 if the types and subtypes of the event nuggets match |
| trigger_pair_exact_match | 1 if the spellings of triggers in EM1 and EM2 exactly match |
| Distance_between_the_wordembedding | quantized semantic similarity score (0-1) using pre-trained word embedding () |
| token_dist | how many tokens between triggers of EM1 and EM2 (quantized) |
| Argument_match | argument roles that are associated with the same entities |
| Argument_conflict | Number of argument roles that are associated with different entities |

Table 5: Event Coreference Features.

### 3.8 Refinement with Human Hypotheses

A human hypothesis is a small topic-level connected knowledge graph. Given five human hypotheses, we generate TA1.b KBs conditioned upon each hypothesis through entity refinement, relation refinement, and event refinement.

**Entity Refinement** For each entity in human hypothesis, we link it to the entities in TA1.a KB based on its name string. If one entity appears in hypothesis but not in TA1.a KB, we propagate KB by adding new entities. If the entity has several types in one hypothesis, we will select the type of highest frequency. If there are conflicts between human hypothesis and TA1.a KB, we trust the one with longer name string. For example, if an entity in human hypothesis is 'Kramatorsk airport' and the corresponding entity mention in TA1.a KB is 'airport', we will update 'airport' to 'Kramatorsk airport'. Besides, to better support relation event refinement, we also update name translations from English to Russian and Ukraine after entity refinement.

**Relation Refinement** Given a relation in a human hypothesis, we propagate relations based on the co-occurrence sentences of its head and tail entities. There are four steps to generate these sentences. Firstly, we enrich hypothetical relations using entity clusters in human hypothesis, i.e., entity coreference information. Namely, more hypothetical relations are constructed by substituting each entity with its coreferential entities. Secondly, three-way translations are done to further enrich hypothetical relations. Specifically, for each language, apart from the original hypothetical relations, we also add the translated ones from other two languages. Note that English serves as a bridge for the translation between Russian and Ukraine. Thirdly, we link head and tail hypothetical entities to our system's entities following Section 5.1. Fourthly, we gather all entity mentions of the above linked entities to find co-occurrence sentences, including named entity mentions, nominal mentions and pronominal mentions.

We trained a second-stage binary classifier which predicts whether these exists meaningful relation type in a given sentence. Compared with the first-stage classifier(depicted as S1) (Section 3.5.1), most parts of the second-stage classifier (depicted as S2) including features are the same except for the output layer. In testing phase, we adopt three different strategies to refine relations. (1) Regarding conflict results which S1 and S2 both predict meaningful relations (relation types defined in AIDA schema), we trust the binary classifier due to the high quality of human hypothesis; (2) In these cases which S1 predicts meaningful relation type but S2 predicts no relation, we remove the S1 relation results; (3) For cases in which S1 predict no relation, We adopt the S1 results because the S2 classifier tends to be too aggressive.

**Event Refinement** Event refinement is based on the sentences which are extracted according to the co-occurrence of each event type and argument provided by human hypothesis. The occurrence of each event type is based on the triggers detected by our event extraction systems. For each sentence containing an event trigger, we link each argument entity to entity mentions in our generated knowledge graph similar to relation refinement, i.e., through entity cluster-based enrichment, three-way translations, entity linking and entity mention linking. If one of the entity mentions is also in the sentence, the sentence will be extracted. Based on these sentences, we find some rules to propagate events. For example, for event type 'Transaction.TransferOwnership', if the argument role is 'Thing', we add this argument to our extracted event.

### 3.9 Experiments

#### 3.9.1 Performance Overview

A table to show component, benchmark data set, results.

| Components | | Benchmark | # of Training Sents | # of Dev Sents | Our F1 (%) | SOTA F1 (%) |
|---|---|---|---|---|---|---|
| Mention Extraction | | CoNLL-2003 | 23,499 | 5,942 | 91.81 | 91.35 |
| Nominal Coreference | | ACE2005&EDL | 721 | 21 | 67.6 | N/A |
| Relation Extraction | English | ACE&ERE | 61,857 | 6,879 | 65.6 | N/A |
| | Russian | AIDA Seedling | 37,179 | 4,137 | 72.4 | N/A |
| | Ukraine | AIDA Seedling | 17,001 | 1,896 | 68.2 | N/A |
| GAIL Event Extraction | Trigger | ACE2005 | 14,837 | 863 | 72.9 | 69.6 |
| | Argument | ACE2005 | 14,837 | 863 | 59.0 | 57.2 |
| Bi-LSTM Event Extraction | Trigger | ERE | 45,801 | 5,088 | 65.41 | N/A |
| | Argument | ERE | 3,240 | 1,056 | 85.02 | N/A |
| Russian Event Extraction | Trigger | AIDA Seedling | 3,703 | 1,233 | 56.15 | N/A |
| | Argument | AIDA Seedling | 3,456 | 1,480 | 58.16 | N/A |
| Ukrainian Event Extraction | Trigger | AIDA Seedling | 3,500 | 1,166 | 58.98 | N/A |
| | Argument | AIDA Seedling | 3,268 | 1,398 | 61.13 | N/A |

Table 6: List of Components for Text Knowledge Extraction

#### 3.9.2 What Works

We conducted ablation experiments on each feature components for relation extraction. Among the linguistic features we used, the entity type and position features contribute the most to the performance. For example, the relation extraction performance decreases by about 8% if removing the entity type feature. We analyze the reasons and find that the entity type feature is vital to ensure the types of two entity mentions to be consistent with the hard entity type constraint of each relation type defined in AIDA schema.

For instances with ambiguity, our dynamic reward function can provide more salient margins between correct and wrong labels: *e.g.*, "... *they sentenced him to* ***death*** ...", with the identical parameter set as aforementioned, reward for the wrong `Die` label is $-5.74$ while correct `Execute` label gains $6.53$. For simpler cases, *e.g.*, "... *submitted his* ***resignation*** ...", we have flatter rewards as $2.74$ for `End-Position`, $-1.33$ for `None` or $-1.67$ for `Meet`, which are sufficient to commit correct labels.

#### 3.9.3 Remaining Challenges

Losses of scores are mainly missed trigger words and arguments. For example, the `Meet` trigger "***pow-wow***" is missed because it is rarely used to describe a formal political meeting; and there is no token with similar surface form – which can be recovered using character embedding – in the training data.

We observe some special erroneous cases due to fully biased annotation. In the sentence "***Bombers*** *have also hit targets ...*", the entity "***bombers***" is mistakenly classified as the `Attacker` argument of the `Attack` event triggered by the word "***hit***". Here the "***bombers***" refers to aircraft and is considered as a `VEH` (Vehicle) entity, and should be an `Instrument` in the `Attack` event, while "***bombers***" entities in the training data are annotated as `Person` (who detonates bombs), which are never `Instrument`. This is an ambiguous case, however, it does not compromise our claim on the merit of our proposed framework against ambiguous errors, because our proposed framework still requires a mixture of different labels to acknowledge ambiguity.

## 4 Visual Knowledge Extraction

Recent advances in computer vision have made it possible to extract various types of knowledge from images and videos, e.g. object detection and tracking, face detection and recognition, human activity understanding, etc. Nevertheless, it is challenging to create a framework that extracts knowledge from various types of media including visual, communicates information in a unified language for higher-level analysis, and cross-references knowledge elements between modalities.

Our first step toward such a unified system was the creation of a multimodal ontology that covers essential visual concepts. In this section, we summarizes the system we developed for parsing visual knowledge from images and videos. We utilize an ensemble of state-of-the-art object detection models to detect, localize, and categorize a variety of entity types in still images and video keyframes. Next, we use face verification and object

instance matching models to link and coreference the detected entities. The methods are elaborated in the following.

### 4.1 Entity Detection

Entities can appear in an image, either as physical objects (e.g. Vehicle), or as scenes (e.g. Airport). To detect physiscal objects, we utilize Faster RCNN (Girshick, 2015), which is the current state of the art in object detection. We use three Faster RCNN models trained on Open Images (Krasin et al., 2017) v2 (600 classes), MSCOCO (Lin et al., 2014) (80 classes), and Pascal VOC (Everingham et al., 2010) (20 classes) datasets. These models can accurately detect, localize, and categorize hundreds of object types. Nevertheless, supervised models like Faster RCNN come with a considerable limitation: they require a massive dataset of images with full bounding box annotation, which is not available for every visual concept.

To detect objects types that are not covered by publicly available bounding box datasets, we use a Weakly Supervised Object Detection (WSOD) technique (Zhou et al., 2016), which requires only image-level annotation for training. We train that model on a subset of 250 classes from OpenImages v4, that are selected using the techniques described in Section 2.2.1. Since this method works based on image-level labels and is not limited by bounding boxes, it can be used to recognize scenes and image-level events as well. Therefore, we include scene and event types in the 250 selected classes.

To integrate these 4 models, we perform several post-processing steps. The first challenge is that each model generates a distribution across its own classes, and since the list of classes are not shared between models, the probability values cannot be compared to each other. To fix this, we rescale confidence scores generated by each model to a unified distribution that is shared for all models. We use a reference dataset, randomly sampled from OpenImages, and apply all models on that. Then for each object type, for each model, we collect all the correctly detected bounding boxes and fit a Gaussian distribution to the confidence scores. We rescale the confidence values by a constant multiplication and addition to adjust mean and standard deviation to a fixed value for all models and categories.

Furthermore, we merge bounding boxes with the same type if their overlap is higher than a threshold. We also propagate class labels up the hierarchy tree and remove those detections that are outside our ontology.

In addition to our object detection ensemble, we utilize MTCNN (Zhang et al., 2016) which is a 3-stage CNN, to detect and align faces. Detected faces are linked with any person object detected by the ensemble if the intersection of the two bounding boxes has a high ratio over the total face area.

### 4.2 Face Recognition

We extract facial features from each detected face using the FaceNet (Schroff et al., 2015) model trained on the CAISA dataset. These features are compared with our database of facial features and matched to the closest person if the distance between features are below a threshold. To construct a database of facial features, we first create a list of people that are helpful to detect. We collect named person annotations from the seedling data and use as a seed to discover other relevant names from DBPedia. To do that, we find all incoming and outgoing relations to/from each seed name, and repeat for 2 hops. We filter based on a set of rules to only maintain highly relevant names. This results in a list of 388 people that are potentially relevant to the scenario.

After that, we use the Google search API to collect images for each named person. Finally, we extract FaceNet features from each image and store in the database. In test time, we compare the features of a query face to all entries in our database and compute the average distance for each person. We link the query to the closest person if the average distance is less than a threshold.

### 4.3 Visual Entity Coreference

We use the DBSCAN clustering algorithm (Ester et al., 1996) on features extracted from entities of same type, to link mentions of the same entity instance. The linking accuracy highly depends on the quality of features. For entities of type person, we used FaceNet features, which are proven to be highly accurate in various face recognition and verification tasks (Schroff et al., 2015). For other entity types, instance-level feature representation is still an unsolved research issue.

We use a similar approach to FaceNet, training a CNN using triplet loss, to learn a Euclidean metric where images of the same instance are closer to

each other compared to other instances of the same type. To come up with such triplets, we use the Youtube BoundingBoxes dataset which was originally intended for object tracking. Each video comes with bounding box annotations for certain objects at every frame. For each triplet, we randomly sample two frames of an object tracklet as positive images of the same object instance, and randomly choose a bounding box from other tracklets (of the same type), as a negative example.

Another type of coreference resolution was done by merging highly overlapping bounding boxes in an image. For example, if a face bounding box falls within a human body bounding box, we link those to indicate they represent the same person entity. Moreover, if a grounded text entity overlaps with a bounding box that is detected by our object detection system, we link those.

## 5 Cross-Media Coreference

In this section, we demonstrate how visual knowledge is integrated with textual knowledge via cross-modal grounding. We propose a model for visual grounding of text phrases that outperforms the state of the art. The input to this model is an image and sentence pair, and the output will be a localization and relevance score for each word in the sentence. The model can also localize phrases instead of words, if phrase boundaries are provided. Figure reffig:grounding-diag illustrates the model block diagram. It consists of visual and textual branches, each extracting local and global features.

More specifically, we use a PNASNet (Liu et al., 2017) and an ELMO (Peters et al., 2018) as our visual and textual branches respectively. PNASNet produces a global image feature at its output and several levels of local feature maps at its intermediate layers. We use one of the middle layers where each local feature represents an image region. ELMO produces local features for each word while also a global feature for the sentence. We compute correlations between all word features and all image region features, which results in a heatmap for each word.

The model is trained on Flickr30k (Young et al., 2014), by maximizing overall image to sentence matching score between relevant image-caption pairs, and minimizing that for non-relevant image-caption pairs. Qualitatively, we observe great improvement over the previous baseline (Engilberge et al., 2018), which was the state of the art in visual grounding. Example results on the seedling data are illustrated in Figure 5.

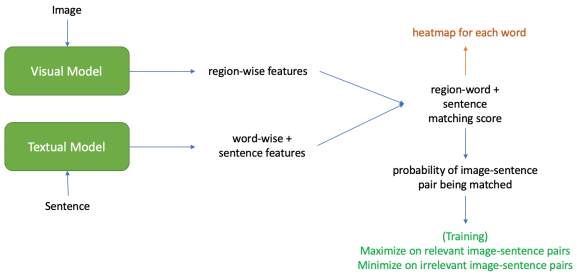


Figure 4: Block diagram of the proposed visual grounding method.


Figure 5: Example grounding results.

## 6 Cross Document Knowledge Element Linking

Our approach to cross-document knowledge element linking uses connected component detection. We formulate a graph relies heavily on information associated with the knowledge elements by TA1. For entities, we apply the following rules:

1. Within a single document, we link nodes that belong to the same TA1-identified cluster (i.e. we respect within document coreference decisions from TA1).

2. We link those entities for which TA1 associated the same external link.

3. We use string similarity to cluster named entities. Because pair-wise string matching of all named entities is not scalable, we apply blocking and only compare those entities that meet one of the following conditions: (a) they share the first three characters;(b) they meet a min-hashing requirement. Within a block, two entities are linked if they meet all the following criterion: (a) they are of same type

and that type is one of: GPE, LOC, PER, ORG, Facilities; (b) neither entity has a reliable external link The jaro-distance is above a threshold(0.9).

## 7 Question Answering

Both the document level and the corpus level tasks are evaluated via question answering using three types of queries: 1. Class-based queries: Find all instances of a known class in a document (for TA1) or the corpus (for TA2). For example, find all organizations. 2. Zero-hop queries: Given a specific instance of an entity, find all instances of that entity in a document (for TA1) or the corpus (for TA2). For example, find all instances of the PERSON entity indicated by bounding box A,B,C,D. 3. Graph-queries: Given an instance (where instance is specified using offsets/bounding box as in a zero-hop query) and a graph of relations connected to that instance, find the nodes that could be included in the graph. Class-based queries are answered with a simple SPARQL query of the KB. Zero-hop queries are answered with slight relaxation to the offsets and/or bounding box of the query (this accounts for errors in exact extent). Answering the graph queries is more complex, even with state-of-the-art extractions from TA1, we expect there to be missing information in the KB. Matching the full graph (often more than 50 edges) is unlikely. To match a graph query, we first use the relaxation strategies described for zero-hop queries on any entry-points. If simple, extent-based relaxation fails to match, we further relax the query by using exact string match the name attribute in the KB within a single document. Note that if all of these relaxations fail, we will not find any answers to the query. Assuming that we find an entrypoint, we attempt to match the full graph. Much of the time, this fails. Our first relaxation attempts to find a 'backbone' of the query by eliminating those nodes that are connected to only one other node in the graph. We then attempt to match the backbone. In some cases, nodes are connected to the graph by more than one edge. If we are unable to match the full backbone, we try an alternative relaxation strategy where we allow a match if any (rather than all) edges linking a node connect. As final relaxation strategy, we combine the backbone approach with the match any approach. We use the same strategies for both TA1 and TA2. For TA1, because we limit string based match to within a single document (to avoid e.g. conflating all John Smiths), we will not find answers where the entry point entity was not found in the relevant document. For TA2, we rely on system cross-document linking to enable answering questions across documents.

## 8 HypoGator: Alternative Hypotheses Generation and Ranking

In this section we provide an overview of HypoGator, our hypothesis generation system. HypoGator relies on the KB constructed from combining and aligning of the document level knowledge elements. Using multiple features and inconsistency detection methods, it extracts coherent and consistent hypothesis. It finally returns a sorted list of the alternative hypothesis relevant to the query. We describe the major components of our system in the following sections.

### 8.1 Hypotheses Generation and Ranking

The input KB is first converted from AIDA Interchange Format (AIF) to our internal format, where each edge in the KB is represented by `(event, role, entity, properties)`. We also find entry points and their roles from the information need `.xml` files.

The major components of Hypogator's pipeline include:

**Extraction:** Find all the nodes in the graph that match any of the entry points from the information need. We used string similarity metrics in combination with exact text offset matching to find all the matches. The set of matched entities are called *seeds*.

**Expansion:** Based on heuristical methods, each of the seed entities is locally expanded to a subgraph. This subgraph contains several nodes and events that are in the neighborhood of the seed.

**Exposition:** During the training phase (on the annotated data) multiple features that are indicative of coherence and relevance were developed and identified. For example, one of the features measures how similar a given subgraph is to the query; another feature approximates coherence using pair-wise vertex connectivity.

These features are then aggregated to calculate a score for each subgraph. Finally, the subgraphs are sorted using this score.

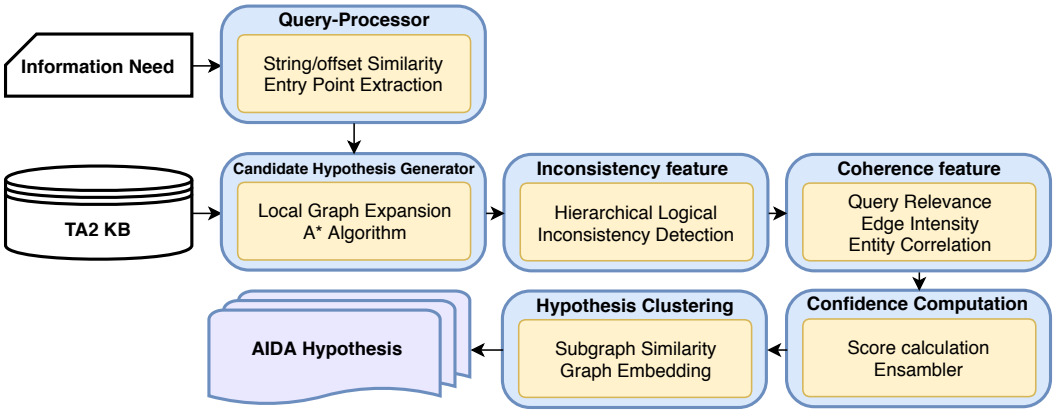


Figure 6: HypoGator: hypothesis mining and ranking, the Architecture.

### 8.2 Inconsistency Detection

The inconsistency detection focuses on identifying conflicting information in multiple event and relationships represented as tuples. Such inconsistency may be inferred from either negated tuples or conflicting information provided within the tuples' fields. Another source of inconsistency comes from event and filler types associated with the wrong field information according to the project's ontology. We then compare events after a transformation phase under ontology-extracted functional dependencies, which allows us to select the fields that need to be compared. The field inconsistency is detected through the use of hierarchical information extracted from common sense networks. Such data allows detecting whether all the entities and fillers occurring in the same event or relationships are compatible – that is when they appear within the same *is-a/part-of* generalisation path – or not.

Each hypothesis is associated to an inconsistency function $\mathcal{I}_{\texttt{MI}^c} \colon \mathcal{K} \rightarrow \mathbb{R}^{+\infty}_{\geq 0}$ (Hunter and Konieczny, 2008), which summarises the amount of the found inconsistencies over a real positive number. Such function weights the number of over the minimal inconsistent sets in $\texttt{MI}(H)$ for each hypothesis $H$, and it is defined as follows:

$$\mathcal{I}_{\texttt{MI}^c}(H) = \sum_{M \in \texttt{MI}(H)} \frac{1}{|M|}$$

We chose such function because it satisfies some relevant rationality postulates such as *super additivity*, *attenuation* and *irrelevance of syntax*. Moreover, such function overcomes the problems of the *lottery paradox* (Kyburg, 1961) by assigning a stronger weight to inconsistencies that can be drawn from few pieces of evidence.

### 8.3 Hypotheses Clustering

Due to the nature of AIDA's data (e.g., multiple documents about the same hypothesis, and noise in TA2s alignment), it is possible to have multiple subgraphs representing the same hypothesis. Our system uses subgraph clustering to prune duplicate hypothesis.

Our focus on the hypothesis and having a smaller data size compared to what TA2 has to work with, enables us to perform more expensive subgraph level clustering. We compute a similarity score for each pair of generated subgraph-hypothesis. Entity features, TA2's alignments and the graph structure, are used to compute this score. We finally use spectral clustering to cluster the set of hypothesis into K clusters.

## 9 Hypothesis Generation at ISI

TA2 generates a full Knowledge Graph (KG). Given such a KG and a file containing the "information need" that identifies a topic in the KG, the steps required to generate the top $k$ hypotheses in TA3 are as follows.

**Retrieve the knowledge elements relevant to the Entry Points**. We convert the XML file containing the information need to SPARQL and perform a query on the full KG generated by TA2.

**Convert the full KG to data in our own internal format**. The data in this new format only contains information of the knowledge elements that are used for reasoning, such as types of Entities, Events, and Relations, as well as the arguments of these Events and Relations. Each of these Entities, Events, and Relations has a confidence value.

**Generate mutual exclusion constraints**. Based on the LDCOntology, we generate two additional types of mutual exclusion constraints: (a) cardinality constraints, and (b) domain constraints. For some types of Events or Relations, some of their arguments can only have a limited number of distinct objects. For example, given an Event Life.Born, the argument Life.Born_Time can have only one object. Such constraints qualify as cardinality constraints. For each type of Event or Relation, each argument can accept only specific types of objects. Such constraints qualify as domain constraints.

**Represent the new data as weighted constraints**. In our approach, an assertion in the KG is not automatically deemed to be true in a hypothesis. This is because it has a certain confidence value associated with it. Instead, a hypothesis fixes the truth value of each assertion, i.e., it is a subset of the full KG. We therefore treat each assertion as a Boolean variable with domain $\{0, 1\}$. Here, '0' and '1' represent 'False' and 'True', respectively. For each Boolean variable $v$ corresponding to an assertion with confidence value $p$, we add a unary weighted constraint with weights $-\log\ (1-p)$ and $-log\ (p)$ against the assignments $v = 0$ and $v = 1$, respectively. For each mutual exclusion constraint $c$ between two variables $v_1$ and $v_2$, we add a binary weighted constraint with weights $-\log\ (p/2)$, $-\log\ (1-p/2)$, $-\log\ (1-p/2)$ and $-\log(p/2)$ against the assignments $(v_1 = 0, v_2 = 0)$, $(v_1 = 0, v_2 = 1)$, $(v_1 = 1, v_2 = 0)$, and $(v_1 = 1, v_2 = 1)$, respectively.

**Solve a Weighted Constraint Satisfaction Problem (WCSP) defined on the weighted constraints**. We use the `WCSP-LIFT` solver based on the idea of the Constraint Composite Graph (CCG)(Xu et al., 2017) for exploiting structure in combinatorial optimization problems. The WCSP solver finds an optimal assignment of values to all Boolean variables that minimizes the total weight. This optimal assignment is the top hypothesis of the original problem. To generate the $k^{th}$ hypothesis, we construct a new WCSP that includes all the weighted constraints described above as well as additional constraints that disallow the previously generated $k-1$ hypotheses.

**Focus hypothesis generation to a subgraph**. Step 1 identifies the Entry Points. From the full KG, we can retrieve a subgraph of knowledge elements that are within $n$ hops from these Entry Points, for some user-specified value of $n$. For each of the top $k$ hypotheses, we discard the elements that do not exist in the subgraph to produce a hypothesis that is relevant to the topic of interest specified in the "information need".

**Scoring the hypotheses**. Each time we solve a WCSP instance, we get an optimal assignment as well as a total weight $w$. The score of the hypothesis is then $e^{-w}$, using the transformation rule that defined weighted constraints from confidence values.

## 10 Conclusion and Future Work

## Acknowledgments

This work was supported by the U.S. DARPA AIDA Program No. FA8750-18-2-0014, LORELEI Program No. HR0011-15-C-0115, Air Force No. FA8650-17-C-7715, NSF IIS-1523198 and U.S. ARL NS-CTA No. W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

## References

Automated phrase mining from massive text corpora.

Mohamed Al-Badrashiny, Jason Bolton, Arun Tejasvi Chaganty, Kevin Clark, Craig Harman, Lifu Huang, Matthew Lamm, Jinhao Lei, Di Lu, Xiaoman Pan, et al. 2017. Tinkerbell: Cross-lingual cold-start knowledge base construction. In *TAC*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

D. Blei, A. Ng, and M. Jordan. 2003. Latent dirichlet allocation. *the Journal of Machine Learning Research*, 3:993–1022.

Jiawei Chen, Yin Cui, Guangnan Ye, Dong Liu, and Shih-Fu Chang. 2014. Event-driven semantic concept discovery by exploiting weakly tagged internet

images. In *Proceedings of International Conference on Multimedia Retrieval*, page 1. ACM.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Zheng Chen and Heng Ji. 2009. Graph-based event coreference resolution. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*, pages 54–57. Association for Computational Linguistics.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. In *Transaction of Association for Computational Linguistics*.

Lingjia Deng, Yoonjung Choi, and Janyce Wiebe. 2013. Benefactive/malefactive event and writer attitude annotation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 120–125.

Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. 2018. Finding beans in burgers: Deep semantic-visual embedding with localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3984–3993.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Bing Qin, Heng Ji, and Ting Liu. 2016a. A language-independent neural network for event detection. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 66.

Xiaocheng Feng, Heng Ji, Duyu Tang, Bing Qin, and Ting Liu. 2016b. A language-independent neural network for event detection. In *Proceeddings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

George Gkotsis, Sumithra Velupillai, Anika Oellrich, Harry Dean, Maria Liakata, and Rina Dutta. 2016. Don't let notes be misunderstood: A negation detection method for assessing risk of suicide in mental health records. In *Proceedings of the Third Workshop on Computational Lingusitics and Clinical Psychology*, pages 95–105.

Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding, 2013 IEEE Workshop on*.

Yu Hong, Di Lu, Dian Yu, Xiaoman Pan, Xiaobin Wang, Yadong Chen, Lifu Huang, and Heng Ji. 2015. Rpi blender tac-kbp2015 system description. In *Proc. Text Analysis Conference (TAC2015)*.

Lifu Huang, Heng Ji Avirup Sil, and Radu Florian. 2017a. Improving slot filling performance with attentive neural networks on dependency structures. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP2017)*.

Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R. Voss. 2017b. Zero-Shot Transfer Learning for Event Extraction. *arXiv preprint arXiv:1707.01066*.

Lifu Huang, Jonathan May, Xiaoman Pan, Heng Ji, Xiang Ren, Jiawei Han, Lin Zhao, and James A. Hendler. 2017c. Liberal entity extraction: Rapid construction of fine-grained entity typing systems. *Big Data*, 5.

Anthony Hunter and Sébastien Konieczny. 2008. Measuring inconsistency through minimal inconsistent sets. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning*, KR'08, pages 358–366. AAAI Press.

Yoon Kim. 2014. Convolutional neural networks for sentence classification.

Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. 2017. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*.

Henry E. Kyburg. 1961. Probability and the logic of rational belief. Wesleyan University Press.

Guillaume Lample, Miguel Ballesteros, Kazuya Kawakami, Sandeep Subramanian, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceeddings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197. Association for Computational Linguistics.

Hongzhi Li, Joseph G Ellis, Heng Ji, and Shih-Fu Chang. 2016. Event specific multimodal pattern mining for knowledge base construction. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 821–830. ACM.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proc. the 52nd Annual Meeting of the Association for Computational Linguistics (ACL2014)*.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Proc. the 2014 Conference on Empirical Methods on Natural Language Processing (EMNLP2014)*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. the 51st Annual Meeting of the Association for Computational Linguistics (ACL2013)*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proc. The 56th Annual Meeting of the Association for Computational Linguistics (ACL2018)*.

Xiao Ling, Sameer Singh, and Daniel Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3.

Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2017. Progressive neural architecture search. *arXiv preprint arXiv:1712.00559*.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, and Ming Zhou. 2015. A dependency-based neural network for relation classification. In *Proceeddings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Di Lu, Leonardo Neves, Vitor Carvalho, Ning Zhang, and Heng Ji. 2018. Visual attention model for name tagging in multimodal social media. In *Proc. The 56th Annual Meeting of the Association for Computational Linguistics (ACL2018)*.

O. Medelyan and C. Legg. 2008. Integrating cyc and wikipedia: Folksonomy meets rigorously defined common-sense. In *Proc. AAAI 2008 Workshop on Wikipedia and Artificial Intelligence*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Thien Huu Nguyen and Ralph Grishman. 2015a. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Thien Huu Nguyen and Ralph Grishman. 2015b. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of NAACL Workshop on Vector Space Modeling for NLP*.

Thien Huu Nguyen and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proc. the 55th Annual Meeting of the Association for Computational Linguistics*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Karin Kipper Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon.

Ge Shi, Chong Feng, Lifu Huang, Boliang Zhang, Heng Ji, Lejian Liao, and Heyan Huang. 2018. Genre separation network with adversarial training for cross-genre relation extraction. In *Proc. 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP2018)*.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, Avignon, France. Association for Computational Linguistics.

Robert L Taft. 1970. *Name Search Techniques*. New York State Identification and Intelligence System, Albany, New York, US.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Veronika Vincze. 2015. *Uncertainty detection in natural language texts*. Ph.D. thesis, szte.

Hong Xu, T. K. Satish Kumar, and Sven Koenig. 2017. The Nemhauser-Trotter reduction and lifted message passing for the weighted CSP. In *Proceedings of the 14th International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR)*, pages 387–402.

Yunlun Yang, Yunhai Tong, Shulei Ma, and Zhi-Hong Deng. 2016. A position encoding convolutional neural network based on dependency tree for relation classification. In *Proceedings of the Empirical Methods on Natural Language Processing*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Dian Yu, Xiaoman Pan, Boliang Zhang, Lifu Huang, Di Lu, Spencer Whitehead, and Heng Ji. 2016. Rpi blender tac-kbp2016 system description.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceeddings of the 25th International Conference on Computational Linguistics*.

Boliang Zhang, Di Lu, Xiaoman Pan, Ying Lin, Halidanmu Abudukelimu, Heng Ji, and Kevin Knight. 2017a. Embracing non-traditional linguistic resources for low-resource language name tagging. In *Proc. the 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*.

Boliang Zhang, Spencer Whitehead, Lifu Huang, and Heng Ji. 2018a. Global attention for name tagging. In *Proc. the SIGNLL Conference on Computational Natural Language Learning (CONLL2018)*.

Boliang Zhang, Spencer Whitehead, Lifu Huang, and Heng Ji. 2018b. Global attention for name tagging. In *Proc. the SIGNLL Conference on Computational Natural Language Learning (CONLL2018)*.

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. 2016. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503.

Tongtao Zhang and Heng Ji. 2018. Event extraction with generative adversarial imitation learning. In *arxiv*.

Tongtao Zhang, Spencer Whitehead, Hanwang Zhang, Hongzhi Li, Joseph Ellis, Lifu Huang, Wei Liu, Heng Ji, and Shih-Fu Chang. 2017b. Improving event extraction via cross-modal integration. In *Proc. the 25th ACM International Conference on Multimedia (ACMMM2017)*.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929.