GAIA at SM-KBP 2019 - A Multi-media Multi-lingual Knowledge Extraction and Hypothesis Generation System

Manling Li[1], **Ying Lin**[1], **Ananya Subburathinam**[1], **Spencer Whitehead**[1], **Xiaoman Pan**[1], **Di Lu**[1], **Qingyun Wang**[1], **Tongtao Zhang**[1], **Lifu Huang**[1], **Heng Ji**[1]
[1] University of Illinois at Urbana-Champaign
hengji@illinois.edu
Alireza Zareian[2], **Hassan Akbari**[2], **Brian Chen**[2], **Bo Wu**[2], **Emily Allaway**[2], **Shih-Fu Chang**[2], **Kathleen McKeown**[2]
[2] Columbia University
sc250@columbia.edu, kathy@cs.columbia.edu
Yixiang Yao[3], **Jennifer Chen**[3], **Eric Berquist**[3], **Kexuan Sun**[3], **Xujun Peng**[3], **Ryan Gabbard**[3] **Marjorie Freedman**[3], **Pedro Szekely**[3], **T.K. Satish Kumar**[3]
[3] Information Sciences Institute, University of Southern California
mrf@isi.edu
Arka Sadhu[4], **Ram Nevatia**[4]
[4]University of Southern California
nevatia@usc.edu
Miguel Rodriguez[5], **Yifan Wang**[5], **Yang Bai**[5], **Ali Sadeghian**[4], **Daisy Zhe Wang**[5]
[5] University of Florida
daisyw@ufl.edu

1 Introduction

In the past year the GAIA team has improved our end-to-end knowledge extraction, grounding, inference, clustering and hypothesis generation system that covers all languages (English, Russian and Ukrainian), data modalities and knowledge element types defined in new AIDA ontologies. We participated in the evaluations of all tasks within TA1, TA2 and TA3 and achieved highly competitive performance. Our TA1 system achieves top performance at both intrinsic evaluation and extrinsic evaluation through TA2 and TA3. The system incorporates a number of impactful and fresh research innovations:

- **Attentive Fine-Grained Entity Typing Model with Latent Type Representation:** We propose a fine-grained entity typing model with a novel attention mechanism and a hybrid type classifier. We advance existing methods in two aspects: feature extraction and type prediction. To capture richer contextual information, we adopt contextualized word representations instead of fixed word embeddings used in previous work. In addition, we propose a two-step mention-aware attention mechanism to enable the model to focus on important words in mentions and contexts. We also develop a hybrid classification method beyond binary relevance to exploit type interdependency with latent type representation. Instead of independently predicting each type, we predict a low-dimensional vector that encodes latent type features and reconstruct the type vector from this latent representation.

- **Cross-media Structured Common Semantic Space for Multimedia Event Extraction:** We propose and develop a new multimedia Event Extraction (M2E2) task that involves jointly extracting events and arguments from text and image. We propose a weakly supervised framework which learns to encode structures extracted from text and images into a common semantic embedding space. This structured common space enables us to share and transfer resources across data modalities for event extraction and argument role labeling.

- **Cross-lingual structure transfer for relation and event extraction:** We investigate the suitability of cross-lingual structure transfer techniques for these tasks. We exploit relation- and event-relevant language-universal features, leveraging both symbolic (including part-of-speech and dependency path) and distributional (including type representation and contextualized representation) information. By representing all entity mentions, event triggers, and contexts into this

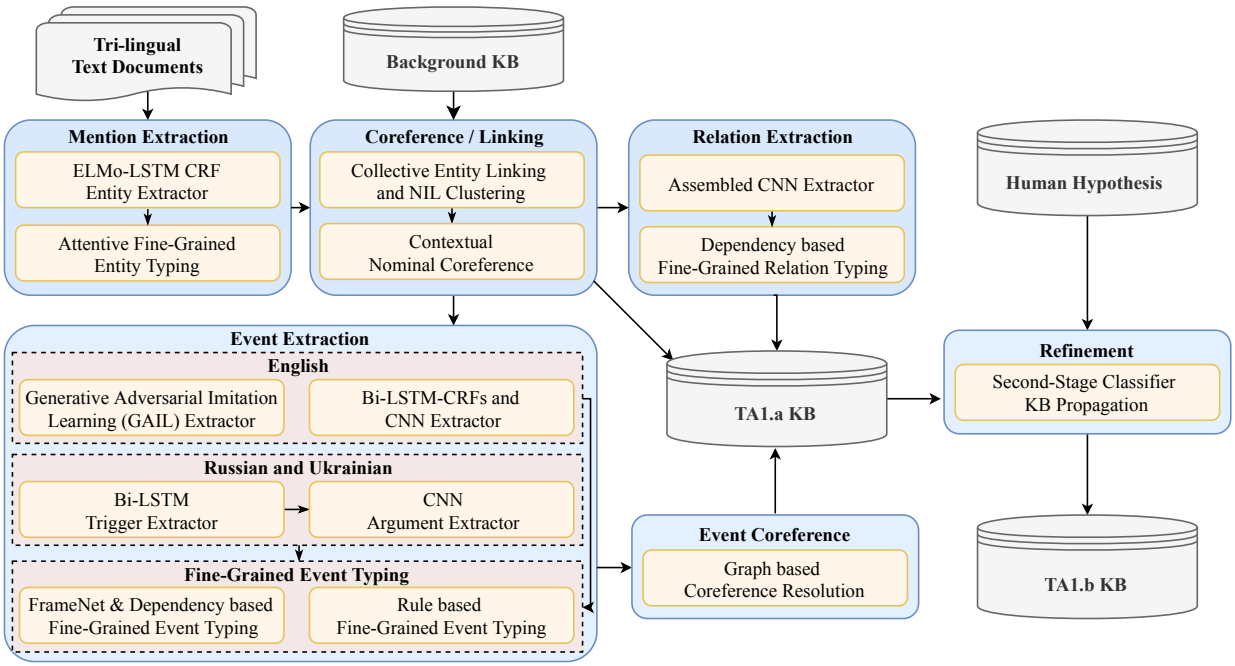

Figure 1: The Architecture of GAIA Text Knowledge Extraction

complex and structured multilingual common space, using graph convolutional networks, we can train a relation or event extractor from source language annotations and apply it to the target language.

2 TA1 Text Knowledge Extraction

2.1 Approach Overview

We build an end-to-end knowledge extraction, grounding, inference, clustering and hypothesis generation system that covers all languages, data modalities. It supports the extraction of 187 entity types, 49 relation types, and 139 event types defined in AIDA ontology. Figure1 shows the overview framework.

The details of each component will be presented in the following sections.

2.2 Mention Extraction

2.2.1 Coarse-grained Named Mention Extraction

We developed a neural model for name mention extraction. The basic model consists of an embedding layer, a character-level network, a bidirectional long-short term memory (LSTM) layer, a linear layer, and a conditional random fields (CRF) layer. In this architecture, each sentence is represented as a sequence of vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_L\}$, where $\boldsymbol{x}_i$ represents features of the i-th word. We use two types of features in our model: 1. *Word embedding* that encodes the semantic information of words. 2. *Character-level representation* that captures subword information.

The LSTM layer then processes the sentence in a sequential manner and encodes both contextual and non-contextual features of each word $\boldsymbol{x}_i$ into a hidden state $\boldsymbol{h}_i$. After that, we decode the hidden state into a score vector $\boldsymbol{y}_i$ with a linear layer. The value of each component of $\boldsymbol{y}_i$ represents the predicted score of a label. However, as the label of each token is predicted separately, the model may produce a path of inconsistent tags such as [BEGIN-GPE, INSIDE-GPE, SINGLETON-GPE]. Therefore, we add a CRF layer on top of the model to capture tag dependencies and predict a global optimal tag path for each sentence. Given a sentence $\boldsymbol{X}$ and scores predicted by the linear layer $\boldsymbol{Y} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_L\}$, the score of a sequence of tags is computed as:

$$s(\boldsymbol{X}, \hat{\boldsymbol{z}}) = \sum_{i=1}^{L+1} \boldsymbol{A}_{\hat{z}_{i-1}, \hat{z}} + \sum_{i=1}^{L} y_{i, \hat{z}_i},$$

where each entry $\boldsymbol{A}_{\hat{z}_{i-1}, \hat{z}_i}$ is the score of jumping from tag $\hat{z}_{i-1}$ to tag $\hat{z}_i$, and $y_{i, \hat{z}_i}$ is the $\hat{z}_i$ dimension of $\boldsymbol{y}_i$ that corresponds to tag $\hat{z}_i$. We append two special tags `<start>` ($\hat{z}_0$) and `<end>` ($\hat{z}_{L+1}$) to denote the beginning or end of a sentence. Finally, we maximize the sentence-level log-likelihood of

the gold tag path $\boldsymbol{z}$ given the input sentence by

$$\log p(\boldsymbol{z}|\boldsymbol{X}) = \log\Big(\frac{e^{s(\boldsymbol{X},\boldsymbol{z})}}{\sum_{\hat{\boldsymbol{z}}\in Z} e^{s(\boldsymbol{X},\hat{\boldsymbol{z}})}}\Big) = s(\boldsymbol{X},\boldsymbol{z}) - \log\sum_{\hat{\boldsymbol{z}}\in Z} e^{s(\boldsymbol{X},\hat{\boldsymbol{z}})},$$

where Z denotes the set of all possible paths.

For English, we improve the model by incorporating ELMo contextualized word representations. We use a pre-trained ELMo encoder to generate the contextualized word embedding $\boldsymbol{c}_i$ for each token and concatenate it with $\boldsymbol{h}_i$.

We train separate models for name, nominal, and pronominal mentions and merge their outputs into the final mention extraction result.

We also explore a reliability-aware dynamic feature composition mechanism (Lin et al., 2019) to obtain better representations for rare and unseen words. We design a set of frequency-based reliability signals to indicate the quality of each word embedding. These signals control mixing gates at different levels in the model. For example, if a word is rare, the model will rely less on its pre-trained word embedding, which is usually not well trained, but assign higher weights to its character and contextual features.

2.2.2 Fine-grained Mention Extraction

We develop an attentive classification model that takes a mention with its context sentence and predicts the most possible fine-grained type. Unlike previous neural models that generally use fixed word embeddings and task-specific networks to encode the sentence, we employ contextualized word representations (Peters et al., 2018) that can capture word semantics in different contexts.

After that, we use a novel two-step attention mechanism to extract crucial information from the mention and its context as follows

$$\boldsymbol{m} = \sum_{M}^{i} a_i^m \boldsymbol{r}_i,$$

$$\boldsymbol{c} = \sum_{C}^{i} a_i^c \boldsymbol{r}_i,$$

where $\boldsymbol{r}_i \in \mathbb{R}^{d_r}$ is the vector of the i-th word, d_r is the dimension of $\boldsymbol{r}$, and attention scores a_i^m and a_i^c are calculated as

$$a_i^m = \text{Softmax}(\boldsymbol{v}^{m\top} \tanh(\boldsymbol{W}^m \boldsymbol{r}_i)),$$

$$a_i^c = \text{Softmax}(\boldsymbol{v}^{c\top} \tanh(\boldsymbol{W}^c(\boldsymbol{r}_i) \oplus \boldsymbol{m} \oplus p_i)),$$

$$p_i = \Big(1 - \mu\big(\min(|i-a|, |i-b|) - 1\big)\Big)^+,$$

where parameters $\boldsymbol{W}^m \in \mathbb{R}^{d_a \times d_r}$, $\boldsymbol{v}^m \in \mathbb{R}^{d_a}$, $\boldsymbol{W}^c \in \mathbb{R}^{d_a \times (2d_r+1)}$, and $\boldsymbol{v}^c \in \mathbb{R}^{d_a}$ are learned during training, a and b are indices of the first and last words of the mention, d_a is set to d_r, and μ is set to 0.1.

Next, we adopt a hybrid type classification model consisting of two classifiers. We first learn a matrix $\boldsymbol{W}^b \in \mathbb{R}^{d_t \times 2d_r}$ to predict type scores by

$$\tilde{y}^b = \boldsymbol{W}^b(m \oplus c),$$

where $\tilde{y}_i^b$ is the score for the i-th type.

We also learn to predict the latent type representation from the feature vector using

$$\boldsymbol{l} = \boldsymbol{V}^l(\boldsymbol{m} \oplus \boldsymbol{c}),$$

where $\boldsymbol{V}^l \in \mathbb{R}^{2d_r \times d_l}$. We then recover a type vector from this latent representation using

$$\tilde{\boldsymbol{y}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{l},$$

where $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ are obtained via Singular Value Decomposition (SVD) as

$$\boldsymbol{Y} \approx \tilde{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{L}^\top,$$

where $\boldsymbol{U} \in \mathbb{R}^{d_t \times d_l}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{d_l \times d_t}$, $\boldsymbol{L} \in \mathbb{R}^{N \times d_t}$, and $d_l \ll d_t$. Finally, we combine scores from both classifier

$$\tilde{y} = \sigma(\boldsymbol{W}^b(\boldsymbol{m} \oplus \boldsymbol{c}) + \gamma \boldsymbol{W}^l \boldsymbol{l}),$$

where γ is set to 0.1. The training objective is to minimize the cross-entropy loss function as

$$J(\theta) = -\frac{1}{N}\sum_{i}^{N} \boldsymbol{y}_i \log \tilde{\boldsymbol{y}}_i + (1 - \boldsymbol{y}_i)\log(1 - \tilde{\boldsymbol{y}}_i).$$

Furthermore, we obtain the YAGO fine-grained types by linking entities to the Freebase (LDC2015E42), and map them to AIDA entity types. Besides, for GPE and LOC entities, we link them to GeoNames [1] and determine their fine-grained types using GeoNames attributes *feature_class* and *feature_code*. Considering that most nominal mentions can not be linked to Freebase or GeoNames and the lack of training data,

[1] http://geonames.org/

we develop a nominal keyword list for each type. We compute a weighted score for these typing results and normalize the scores as typing confidence. Moreover, some entity types are related to events (e.g., *PER.Protester* is usually a person argument played as a protester role in a protest event), and thus we label them based on event extraction results.

2.2.3 Entity Filler Mention Extraction

We use Stanford CoreNLP (Manning et al., 2014) to extract values and titles from English raw documents. We also utilize the time normalization function included in the CoreNLP by adding the date when the document was posted as the reference time. For time expressions in Russian and Ukrainian, we use a deterministic rule-based system based on SUTime (Finkel et al., 2005).

For `color` entity types, we simply assemble a list of common colors in English, Russian, and Ukrainian, and then perform string matching to tag these entities.

2.2.4 Informative Justification Extraction

For each named entity, we rank all mentions in terms of length and frequency. For each nominal entity, we select the noun phrase chunk containing the most frequent mentions. For pronoun entities, we select the most frequent mention. The confidence of informative justification is the weighted sum of all mentions, with higher weights for name mentions. If one entity only has nominal and pronoun mentions, we reduce its confidence so it will be ranked after other entities with name mentions.

2.3 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).For Russian and Ukrainian, we use translation dictionaries collected from Wikipedia (Ji et al., 2009) to translate each mention into English first. If a mention has multiple translations, we merge the linking results of all translations at the end. 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., 2017), 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.

2.4 Within-document 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.

2.4.1 English Coreference Resolution

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

Given a document $D = \{y_1, y_2, ...y_n\}$, we receive its mention spans $S = \{s_1, s_2, ..., s_l\}$ with their mention types $M = \{m_1, m_2, ..., m_l\}$, entity types $E = \{e_1, e_2, ..., e_l\}$, and clusters $C = \{c_1, c_2, ..., c_l\}$ from entity linking and NIL Clustering components, where y_i is an individual word, n is the length of the document, m is the number of mentions, $c_i \in \{1, 2, ..., v\}$, v is the cluster size, and each span s_i contains two indices START(i) and END(i). To compute a word representation at position k, we use a pre-trained ELMo (Peters et al., 2018) embedding layer to obtain its representation y^*_k.

Then for each span i, we compute the soft attention(Bahdanau et al., 2014) over each word in the span:

$$\alpha_k = \omega_\alpha \cdot \text{FFNN}_\alpha(y^*_k) \tag{6}$$

$$\alpha_{i,k} = \frac{\exp(\alpha_k)}{\sum_{k=\text{START}(i)}^{\text{END}(i)} \exp(\alpha_k)} \tag{7}$$

$$\hat{y}_i = \sum_{k=\text{START}(i)}^{\text{END}(i)} \alpha_{i,k} \cdot y^*_k \tag{8}$$

where $\hat{y}_i$ is a weighted sum of all word hidden states in span i, and FFNN is the feed-forward neural network. Then the representation g_i of a span i is computed as:

$$g_i = [y^*_{\text{START}(i)}, y^*_{\text{END}(i)}, \hat{y}_i, \Phi_g(i), \Gamma_g(i)] \tag{9}$$

where a feature vector $\Phi_g(i)$ encodes the size of span i, a feature vector $\Gamma_g(i)$ encodes the entity type of span i, and $[,]$ is the concatenated vector.

Finally, to compute clusters, for each span g_i, we check all precedent mentions g_j where $j < i$ has the same entity type. For each pair of span g_i and g_j, inspired by the dynamic feature composition (Lin et al., 2019), we introduce two dynamic span gates to update each span respectively:

$$\hat{g}_i = \sigma(\text{FFNN}_{\sigma 1}([g_i, g_j]])) * g_i \tag{10}$$

$$\hat{g}_j = \sigma(\text{FFNN}_{\sigma 2}([g_i, g_j]])) * g_j \tag{11}$$

where σ is the sigmoid function. Then we can compute the pairwise score $x(i, j)$ as:

$$x(i, j) = [w_x \cdot \hat{g}_i, \hat{g}_j, \hat{g}_i \circ \hat{g}_j, \Phi_x(i, j)] \tag{12}$$

where $\circ$ is the element-wise similarity feature vector, and $\Phi_x(i, j)$ encodes the distance between two span.

For each span s_i, following Lee et al. (2017), we learn a conditional probability distribution for the corresponding span s_i over all its antecedent $a_i \in A(i) = \{\epsilon, 1, ..., i-1\}$ where ϵ is a dummy antecedent:

$$P(s_1, ..., s_l|D) = \prod_{i=1}^{l} \frac{\exp(x(i, a_i)))}{\sum_{a' \in A(i)} \exp x(i, a')} \tag{13}$$

Similarly, we then optimize the marginal log-likelihood of all correct antecedents implied by gold-standard clustering:

$$\log \prod_{i=1}^{l} \sum_{a' \in A(i)(i)} P(a') \tag{14}$$

where $GOLD(i)$ is the set of spans in the gold-standard cluster containing span i. In the test phrase, we only use the cluster prediction results from the system generated span s_i whose original cluster c_i either only contains nominal mentions or a single mention.

For English training, we used ACE2005, EDL2016, EDL2017, CoNLL-2012 to train the system.

2.4.2 Russian/Ukrainian Name Coreference Resolution

For Russian and Ukrainian, we perform an additional name coreference step as post-processing. We construct a graph by considering each existing AIF cluster as a node, and apply the following rules to add edges between clusters if:

1. There is an exact match in the string or stem between any name mentions in either cluster.
2. There exists two strings from two clusters where the Levenshtein distance between them is less than or equal to 2 and the size of the string or stem is at least 6 or 5, respectively.
3. There exists an `appos`[2] relation between two name mentions, with the same entity type. We apply the UDPipe(Straka and Straková, 2017) models, which are pre-trained on the Universal Dependencies(Nivre et al., 2019) dataset, to extract the syntactic relations.
4. For person names, there exists a single-token string in one cluster which appeares as the second token of a two-token string in another cluster and does 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).
5. For person names, there exists a two-token string in one cluster which matches a two-token string in another cluster, accounting for morphological inflection of both strings.

A coreference cluster is constructed from each connected component of the resulting graph. For Russian and Ukrainian stemming, stems are determined in a simple, heuristic way by stripping a list of common Russian and Ukrainian inflectional affixes from the end of the string.

[2]appositional modifier

2.5 Relation Extraction

For fine-grained relation extraction, we map the AIDA subtypes to ACE/ERE ontology and apply a Convolutional Neural Network (CNN) based model to detect these mapped types, and then utilize entity type constraints as well as dependency patterns to re-categorize these detected relations into fine-grained types (Section 2.5.1). For types that cannot be mapped directly to ACE/ERE, such as `Genafl.Sponsor` or `Evaluate.Deliberateness`, we design dependency patterns and implement a rule-based system to detect these fine-grained relations directly from the text (Section 2.5.2).

2.5.1 ACE/ERE Relation Extraction and Fine-grained Typing

The relation extraction system predicts relations between each pair of entity mentions within the same sentence. We employ a CNN with piecewise pooling as our underlying model.

Model Given a source sentence $s = [w_1, ..., w_m]$ along with entity mentions e_1 and e_2, for each word w_i, we generate an embedding $\boldsymbol{v}_i = [\boldsymbol{w}_i, \boldsymbol{p}_i, \tilde{\boldsymbol{p}}_i, \boldsymbol{t}_i, \tilde{\boldsymbol{t}}_i, \boldsymbol{c}_i, \eta_i]$, where $\boldsymbol{w}_i$ denotes the word embedding of w_i from a set of pre-trained embeddings. $\boldsymbol{p}_i$ and $\tilde{\boldsymbol{p}}_i$ are position embeddings indicating the relative distance from w_i to e_1 and e_2, respectively. $\boldsymbol{t}_i$ and $\tilde{\boldsymbol{t}}_i$ are, respectively, the entity type embeddings of e_1 and e_2. Finally, $\boldsymbol{c}_i$ is the chunking embedding, and η_i is a binary digit indicating whether the word is within the shortest dependency path between e_1 and e_2. All embeddings besides the pre-trained word embedding are randomly initialized and optimized during training. The result of this embedding process is a sequence of word representations $V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$. We then apply a convolution filter, $\boldsymbol{W}$, with an added bias, $\boldsymbol{b}$, to each sliding n-gram phrase, g_j (*i.e.*, $\boldsymbol{g}_j = \tanh(\boldsymbol{W} \cdot V) + \boldsymbol{b}$). Since different segments of a sentence do not hold equal weight towards the underlying semantics of the sentence, we split all the $\boldsymbol{g}_j$ into three parts based on the two entity mentions and perform piecewise max pooling. Lastly, we concatenate the representations of these three segments, feed them into a fully connected layer, and apply a linear projection and softmax to classify the relations.

Training Details For English training, we manually map DEFT Rich ERE and ACE2005 to the AIDA ontology, and combine all the resources. For Ukrainian and Russian, we asked a native speaker to annotate some part of the seedling corpus. To further improve the system performance, we also adopted a majority vote strategy to assemble the models for all three languages. Additionally, we find that our model cannot well capture global information due to instance-level training. Accordingly, we extract some high-confidence, frequent relation patterns (Table 2) from the training data and treat them 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.

Fine-grained Typing We take the relations extracted from the above component as input to a rule-based component that assigns fine-grained relation types, if possible. This component only matches high confidence patterns and largely relies on entity type constraints. For example, if a `Leadership` relation type is detected between a `PER` mention and an `ORG` mention, and the `ORG` mention is a law enforcement agency or military organization, then we assign a fine-grained type of `Leadership.MilitaryPolice`.

In conjunction with entity type constraints, we also employ dependency patterns to ensure the correctness. For instance, if two entities fit the argument constraints for `LocatedNear` and the location is an object of a "*surround*" keyword, then we assign the fine-grained type `LocatedNear.Surround`. We base these patterns on observations from training data across English, Ukrainian, and Russian.

2.5.2 Unmapped Relation Types

We implement a rule-based relation extraction component for `Evaluate.Deliberateness/Legitimacy`, `Measurement.Size`, `Genafl.Orgweb`, and `Information.Make/Color` based on the rules as follows (for English only):

`Evaluate.Deliberateness/Legitimacy` We extract this relation in a similar fashion to the fine-grained typing. We first utilize argument type constraints, such as the event having a `StartPosition` type. Then, we employ keywords and dependency patterns to determine who is the holder of the evaluation.

`Measurement.Size` We first apply Stanford CoreNLP to extract all `NUMBER` $\mathcal{N}$. Then for $n_i \in \mathcal{N}$, we check if the word appears after n_i. If it is a part of a name mention extracted by our EDL system, we tag the relation between them as `Measurement.Count`. We then attempt to assign a fine-grained type to these relations based upon whether or not there are units mentioned in the sentence.

`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.

`Information.Make/Color` Any entities that are detected that fit the argument constraints for these types and satisfy the dependency pattern criteria are assigned one of these types. Only entities of type `Color` may be assigned `Information.Color`.

2.5.3 Sponsorship and AssignBlame Relation Extraction

We implement a rule-based component for `GeneralAffiliation.Sponsorhip` and `ResponsibilityBlame.AssignBlame` using manually collected trigger words in English and Russian. A seed set of trigger words is collected from the training data and then augmented to include synonyms and related words. We used a native Russian speaker to augment the set of Russian trigger words. Each trigger is associated with a fine-grained type and accompanied by a confidence score that accounts for whether we have seen the trigger, a translation of the trigger, or a synonym of the trigger in a relation.

For English, we extract relations by finding the shortest dependency path (SDP) between two entity mentions and checking for trigger words along this path. If any triggers occur in the SDP, we extract the relation associated with the trigger word between the two entity mentions. For Russian, we look at the plain text between the two entity mentions, instead of the SDP, and for Ukrainian we first translate to Russian and then follow the same procedure as for Russian.

2.5.4 Sentiment Relation Extraction

We train a targeted-sentiment system (Zhong et al., 2019) for the `Evaluate.Sentiment` relation. The system is developed for English and then adapted to Russian and Ukrainian.

The English system features a technique for training attention to focus on human rationales and can distinguish between pairs of entities that have sentiment relations and pairs that do not. We define human ground-truth attention as uniform over words in a rationale, selecting rationales from the MPQA dataset (Wilson, 2008). The model's attention is then made to match the human attention through a new KL-divergence loss component, giving significant improvements and outperforming a wide range of competitive baselines (Zhong et al., 2019).

We adapt the English sentiment system to Russian and Ukrainian by training the model using pre-trained multi-lingual English-Russian-Ukrainian word embeddings (Conneau et al., 2017) as input features. Since the English sentiment system uses positional embeddings derived from the POS, we also use Russian POS tags (Kopotev and Ianda, 2006). Due to the linguistic similarities between Ukrainian and Russian, we assume a one-to-one relationship between a Ukrainian sentence and its Russian translation and use this to project Russian POS tags to Ukrainian for the sentiment system. In this way, we only need labeled training instances for English in order to train a system for all three languages. The model achieves comparable F1-score to the original system (Zhong et al., 2019) trained using pre-trained English only word embeddings (Pennington et al., 2014).

2.6 Event Extraction

Event extraction consists of two subtasks: *trigger labeling* identifies the trigger words of events and determines the event types, *e.g.*, `Conflict.Attack`; *argument role labeling* assigns the relation between entities (or arguments) and triggers, *e.g.*, an `Attacker` in a `Conflict.Attack` event.

In this section, we will introduce our English event extractor in Section 2.6.1 and Section 2.6.2, and then we will present our approach in Russian and Ukrainian event extractor in Section 2.6.3, followed by fine-grained typing methods in 2.6.4.

2.6.1 Event Extraction with Generative Adversarial Imitation Learning

Trigger labeling The trigger labeling module still follows the sequence labeling model, which includes a Bi-LSTM model and a CRF decoder (Feng et al., 2018). This is different from the previous work (Zhang et al., 2018), which utilizes the Q-learning algorithm from (Zhang et al., 2019). We notice that under the definition of AIDA schema, some triggers are phrases (more than one word), and we find that the performance with the Bi-LSTM-CRF model for such phrases is better and more stable.

The input of the Bi-LSTM framework is word embedding concatenated with the following categories.

- Token surface embeddings: We initialize a random embedding dictionary for each token, this embedding will be updated with the in the training phase.
- Character-based embeddings: Similar to the token surface embeddings, each character has an randomly initialized embedding. We feed the embeddings into a token-level Bi-LSTM network and concatenate the final hidden representations from both directions for the token. These embeddings are also updated during the training phase.
- POS embeddings: We apply Part-of-Speech tagging (POS) on the sentences with spaCy. Similarly, the each POS label has a trainable embeddings which is intialized with random variables.
- Pre-trained embeddings: We also utilize the pre-trained Word2Vec embeddings from Wikipedia article dump (a version on January 1st, 2017). These embeddings are fixed and not updated in the training phase.

Argument role labeling Traditionally, argument role labeling is considered as classification problem. In this work, we model it in a Reinforcement Learning scenario, and consider a label as an action from an agent (the model/extractor).

In this scenario, we define a state (feature)

$$s_{tr,ar} =< \boldsymbol{v}_{tr}, \boldsymbol{v}_{ar}, a_{tr}, a_{ar}, \boldsymbol{f}_{ss} >, \qquad (15)$$

where $\boldsymbol{v}$s are the context embedding from the Bi-LSTM network, tr denotes triggers, ar denotes the argument candidate (typically, detected entities from Section 2.2), as are the actions (labels)

of event types (with tr footnote) and entity type (with ar footnote), and $\boldsymbol{f}_{ss}$ denote a sub-sentence embedding which consists of output of three Bi-LSTM networks that consume the left, middle and right subsentences seperated by the trigger and argument candidate.

We feed the state representation into an MLP to obtain the probability distribution $Q_{tr,ar}$ of actions and we determine the argument role $a_{tr,ar}$ with

$$\hat{a}_{tr,ar} = \arg\max_{a_{tr,ar}} Q_{tr,ar}(s_{tr,ar}, a_{tr,ar}). \quad (16)$$

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

$$L_{pg} = -R \log P(a_{tr,ar}|s_{tr,ar}). \quad (17)$$

The footnote pg denotes the RL algorithm called policy gradient, which focus on optimizing the actions on the probability distribution.

Dynamic Reward Estimation with GAN From Equation 17 we notice that, if we impose a fixed reward R, there is no difference between a classification model with cross-entropy loss and an RL model. Hence, we introduce a reward estimator based on GAN to issue dynamic rewards with regard to the labels committed by the event extractor. If the extractor repeatedly make mistakes on an instance, then the extractor will increase harshness on penalty, and the reward will also increase as the target instance is considered a difficulty one. The expanded difference between rewards and penalties deviate the extractor from wrong actions (labels) and push them to correct ones.

We train the reward estimator based on the combination of the feature vector $s_{tr,ar}$ from Equation 15 and action embedding representation for $a_{tr,ar}$. The reward estimator is the discriminator in GAN, and will distinguish the input is from a groudtruth instance (or in GAN, the real one, or expert one) or from an extractor (or the generated ones).

The output of the discriminator is between $(0, 1)$ range and we use a linear transform to expand it as $(-5, 5)$. Then we use this output as the reward R to optimize the extractor, or generator in the GAN scenario.

Training Details Taking "out of vocabulary" (OOV) problem into consideration, we intentionally set "UNK" (unknown) entry in the look-up dictionaries of token surface embeddings, character-based embeddings and POS embeddings. During training, we randomly mask known tokens, POS tags and characters in the training sentences with "UNK" mask, moreover, we also set an all-0 vector of randomly selected tokens in the pre-trained embeddings.

To improve performance, we expand the training dataset. The set includes documents from ACE and ERE data set. To bridge the gap among ACE, ERE and AIDA ontology, we clean up the dataset. We map the labels in ACE (e.g., `Phone-Write`, `Transport`) to ERE ones (`correspond`, `transport-art`). We also remove sentences with no trigger annotations from ACE data to ensure that ERE labels that have no counterparts in ACE (e.g., `broadcast`) will not be missed due to empty annotation. Finally, we map all ERE label to AIDA ones.

2.6.2 Event Extraction with Bi-LSTM

Trigger Labeling We develop 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., 2016; 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 obtain 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 context 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 is assigned with a feature 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 into 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.

2.6.3 Russian and Ukrainian Event Extraction

Event Extraction for Russian and Ukrainian text also follows two steps, as in (Zhang et al., 2018) and the English counter parts in Section 2.6.1 and 2.6.2. First, we apply a Bi-LSTM over the pre-trained word embeddings for each word in a sentence to generate the hidden representation denoting context. These vectors are then fed into a Softmax classifier, where each word is classified as either a trigger of a particular AIDA event type, or not.

In the second stage, we employ a CNN with Softmax classification similar to (Kim, 2014), to identify and label entities that play a role in each event mention. Here, all entities within the same sentence of an event mention are taken as candidate arguments. For each candidate argument, the portion of the sentence between the argument and its trigger word is used as input to the CNN. Similar to step one, we use pre-trained embeddings to represent each word. We use masks of length 3, 4, and 5 in the convolution layer. Then, after max-pooling, a fully connected layer generates a single vector which is classified using a Softmax classifier as a certain argument type, or not an argument. We use post processing rules to ensure that all results follow the constraints set by the AIDA ontology.

We also explore a structured transfer learning approach (Subburathinam et al.) using graph convolutional networks. It represents all entity mentions, event triggers, and contexts into a complex and structured multilingual common space, and we train a relation and event extractor from English language annotations and apply it to the Russian and Ukrainian language. In this process, we exploit event-relevant language-universal features, leveraging both symbolic (including part-of-speech and dependency path) and distributional (including type representation and contextualized representation) information.

2.6.4 Fine-grained Event Typing

For events that coarse types are not covered in the English training data, we map FrameNet frames to AIDA event types (Table 3). We tag frame elements as event arguments only if they overlap with the named entities. For Russian and Ukrainian, we use GIZA (Och, 1999; Och and Ney, 2003) to get the word alignment.

For `Government.Spy.Spy`, `Government.Vote`, `Government.Agreements`, `Disaster.AccidentCrash.AccidentCrash`, and which cannot be mapped from Framenet, we implement a dependency-based system. We detect event triggers based on fuzzy string match with keywords [3]. We further decide event arguments based on the syntactic relation between the entity end the verb keyword in the dependency tree.

We develop a rule-based fine-grained event typing system, including verb-based rules, context-based rules, and argument-based rules. For each coarse-grained types, we collect a verb keywords list to distinguish its fine-grained types. For example, `Conflict.Attack` is `Conflict.Attack.FirearmAttack` when the trigger is *shooting*. Contextual keywords are collected similarly. For example, *funeral* in the context of `Contact.Meet` can be used to determine `Contact.FuneralVigil.Meet`. Mul-

[3] `Government.Spy.Spy`: spy;
`Government.Vote`: vote, ballot, poll, referendum, election;
`Disaster.AccidentCrash.AccidentCrash`: crash;
`Government.Agreements`: agreement

Framenet event type	AIDA event type
Conquering	Transaction.Transaction.TransferControl
Criminal_investigation	Justice.Investigate.InvestigateCrime
Sign_agreement	Government.Agreements.AcceptAgreementContractCeasefire
Inspecting	Inspection
Destroying	ArtifactExistence.DamageDestroy.Destroy
Damaging	ArtifactExistence.DamageDestroy.Damage
Quitting_a_place	Conflict.Yield.Retreat
Surrendering	Conflict.Yield.Surrender
Explosion	Disaster.FireExplosion.FireExplosion
Fire_burning	Disaster.FireExplosion.FireExplosion
Catching_fire	Disaster.FireExplosion.FireExplosion
Amalgamation	Government.Formation.MergeGPE
Intentionally_create	Government.Formation.StartGPE
Come_together	Contact.Discussion
Discussion	Contact.Discussion
Rescuing	Movement.TransportPerson.EvacuationRescue
Prevarication	Contact.Prevarication
Request	Contact.CommandOrder
Speak_on_topic	Contact.MediaStatement
Giving_in	Conflict.Yield

Table 3: Mapping from Framenet to AIDA event ontology.

tiple rules might be applied in order to determine one fine-grained type. Also, the arguments can be used to determine the fine-grained types. For example, if there is an instrument for `Life.Die` Event, then it should be `Life.Die.DeathCausedByViolentEvents`, not `Life.Die.NonviolentDeath`. Also, the fine-grained types of entities can indicate the fine-grained types, e.g., `Conflict.Attack` should be `Conflict.Attack.Bombing` when the instrument is `WEA.Bomb`.

For Russian and Ukrainian events, the main problem is the difficulty of keyword matching due to the morphology. As a result, we generate multiple-level translations using Google Translation [4], i.e., the translation of the trigger translation, the translation of the sub-sentence containing the trigger, the translation of sentence. Then we apply the English system on these translations.

2.7 Event Coreference Resolution

We apply a graph-based algorithm (Al-Badrashiny et al., 2017; Zhang et al., 2018) for our language-independent Event coreference resolution. For each event type, we view event mentions as nodes in the graph, and the undirected weighted edges between the nodes represent the coreference confidence between corresponding events. Then we apply hierarchical clustering to obtain event clusters. We train a Maximum Entropy binary classifier with the features listed in Table 4. We use the English annotations from ERE (Getman et al., 2018) as training data.

[4] https://translate.google.com/

2.8 Refinement with Human Hypotheses

A human hypothesis is a small manual knowledge graph, and we use it to refine the knowledge base constructed in TA1.a. There are three kinds of refinements, i.e., entity refinement, relation refinement, and event refinement.

The entities in the human hypothesis are linked to the background KB (LDC2019E43), and we propagate TA1.a KB by adding new entities if one entity appears in the hypothesis but not in TA1.a KB. To extract the mentions of these entities, we use the name of the entity in background KB and also its translations in other languages, and conducted string matching after stemming. For the nominal entities, such as "snipers shooting during Maidan", we extract the head word "snipers" for string matching and use "Maidan" as a constraint that "Maidan" should appear in the same document as the mention "snipers". Only when the constraint is satisfied, the mention is extracted. If the entity in the hypothesis is already extracted in TA1.a KB, we compare its fine-grained types, and trust the human hypothesis when there are conflicts. Furthermore, we use the human hypothesis to correct the entity linking results. We use the entity name and its translations to match the entities in TA1.a KB, and trust the linking results in the human hypothesis when encountering conflicts.

Relation refinement has two stages. The first

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 4: Event Coreference: Features for Maximum Entropy classifier (Zhang et al., 2018)

stage is to rerun the system using the updated entities, which will enrich the relations, especially the ones related to augmented entities. The second stage is based on a second-stage binary classifier. For each relation in the human hypothesis, we extract the co-occurrent sentences of two entities, and the binary classifier determines whether the sentence indicates a relation. If so, we trust the relation type in the human hypothesis, and the sentence serves as the justification.

Similar to relation refinement, event refinement also has two stages, with the rerunning as the first stage. In the second stage, we extract the co-occurrent sentences of event type and arguments and develop rules to decide whether the co-occurrent sentence indicates the arguments of the event. The localization of event types in each sentence is based on the triggers extracted in TA1.a KB. In the co-occurrent sentences of entity type and two arguments, if one argument is extracted by the system and the other one does not, we add the missing one to the system output.

3 TA1 Visual Knowledge Extraction

We first briefly review our previous Visual Knowledge Extraction (VKE) system (Zhang et al., 2018), and then describe the new extensions and improvements that form our new system. We had designed the VKE system to complement the text KE system, by extracting entities solely by their visual appearance, and then integrating them with the knowledge extracted from text, to form a unified knowledge base. The VKE system consists of entity detection, entity recognition and entity coreference resolution, that are discussed in the rest of this section.

3.1 Entity Detection

The entity detection system consists of an ensemble of 5 object detectors, namely 3 Faster R-CNN (Ren et al., 2015) models trained on 3 different datasets, a weakly supervised CAM model (Zhou et al., 2016) for classes for which bounding box annotation is not available, and an MTCNN model (Zhang et al., 2016) specialized for face detection. Since these models are trained on various datasets with distinct but somewhat overlapping ontologies, we develop a post-processing algorithm (Zhang et al., 2018) to combine their results and map to the unified AIDA ontology. Since TAC 2018, the program ontology has changed, and so we update our class mapping, as well as the CAM model to cover more classes.

The entity detection system is trained using a typical cross-entropy loss to classify each object into a variety of types that are selected from multiple ontologies. Among these are classes which follow a hierarchy. This causes a serious issue in producing a score for a given bounding box; fine-grained entity types are in conflict with their coarse-grained counterparts resulting in artificially lower scores in several cases. For e.g. score for a bounding box containing a "man" gets distributed across "man", "person", "boy", "people". To alleviate this issue, we remove the cross-entropy responsible for generating probability distribution across classes with a binary cross-entropy which allows for independent distributions and hence the scores for each of the entity types are independent and would get a higher score as required by the query type similar to (Akiba et al., 2018).

3.2 Entity Recognition

The entity recognition (a.k.a. entity linking) module was previously based on a face recognition pipeline that can recognize a predefined set of PER (person) entities and link their face bounding box (visual equivalent of entity mention) to their KB entry. This year we expanded that system to include other entity types such as GPE (geopolitical entity) and FAC (facility). Our face recognition system (Zhang et al., 2018) is based on FaceNet (Schroff et al., 2015) which trains a metric space, which enables comparing the visual similarity of

Figure 2: Example landmark recognition results. Red circles denote local features that were matched to a reference image.

Figure 3: Example flag recognition results. Green represents correct prediction, blue represents wrong prediction, and grey represents missed detection by Faster R-CNN.

faces and determining whether a pair of faces belong to the same identity. We use a similar strategy to train other specialized models for other types of entity, besides PER (person).

For instance, for GPE (geopolitical entity), we use Google image search to collect flag images from different countries and organizations (200 classes, 100 images each). We apply our entity detection system to localize flags in those images, and tightly crop the flags to train a classifier. We train an Inception-v4 model (Szegedy et al., 2017) with 200 classes. In the test time, after entity detection we apply the classification model on the cropped region of each detected flag to classify the country and link to the KB. To evaluate, we manually labeled 69 cropped flags from a random portion of last year's evaluation data. Our model achieves 73% F1 score and 72% accuracy. Exam-

ple results are shown in Figure 3.

To recognize FAC (facility), LOC (location), and ORG (organization), we adopt the recently proposed DEep Local Feature (DELF (Noh et al., 2017)) method, which is used for large-scale instance-level image matching. The model is pre-trained on the Google Landmarks (kag) dataset which includes 15,000 landmarks across the world. We used the AIDA training data to select 500 scenario-relevant landmarks and buildings. More specifically, we collected entities that are annotated with one of the aforementioned types in the training data, or detected by the text entity extraction (Section 2.2). Then, we use each retrieved entity mention to find related images using Google image search. We use Faster-RCNN on the retrieved images to detect landmark instances and remove images without any landmark. Then we select entities that have sufficient images (100 per entity).

We use those images as prototypes for each entity, and given a test image, we use RANdom SAmple Concensus (RANSAC) (Fischler and Bolles, 1981) to align its DELF local features with each prototype. We assign the test image to the best matching entity, i.e. the prototype with the most aligned local features. This system achieves an F1 score of 0.875 on a random portion of last year's evaluation data. Example results are illustrated in Figure 2.

3.3 Cross-Modal Entity Coreference

The entity coreference system is used to create a coherent knowledge graph from individually detected entities. As described in (Zhang et al., 2018), the entity coreference has two submodules: intra-modal and cross-modal. The intra-modal coreference module connects visually similar or related entities such as a person's face that appears in multiple images. The cross-modal coreference module links entities mentioned in text to entities appeared in images and video keyframes. This year, we significantly improved our cross-modal coreference module by incorporating a multi-layer attention mechanism, as described in the following.

We advanced our previous visual grounding system (Zhang et al., 2018) by introducing a novel multi-level attention mechanism. As illustrated in Figure 4, our system extracts a multi-level visual feature map for each image in a document, that is a set of grids of feature vectors, where each grid has a certain granularity, and each cell of a grid is a feature vector representing an image region. On the other hand, our network takes each sentence of the document and represents each word using a language model. Then for each word (or phrase, entity mention, etc.) we compute an attention map to every level and every location of the feature map. This way, we can choose the level that most strongly matches the semantic query, and within that level, the attention map can be used to localize the query. The multi-level aspect of attention is important, because some queries may convey low-level information such as object types (e.g. person) while other queries may refer to high-level semantics such as protest. Higher-level semantics tend to be reflected in the latter layers of convolution nets, which means the selection of an appropriate layer for query localization is important. Our cross-modal coreference system outperforms the state of the art in various visual grounding benchmarks. More details are discussed in (Akbari et al., 2019).

Furthermore, this year we added a complementary visual grounding method to our pipeline, to boost recall. Given bounding boxes and their predicted classes using the above approach we are able to perform grounding by matching the text with those of the predicted classes. We use a GloVe (Pennington et al., 2014) with thresholds set to 0.75 to obtain the relevant bounding boxes.

4 TA2

The goal of the TA2 clustering work is to identify entities and events across documents. The primary challenge is that the entities and events extracted by the TA1 modules contain few attributes. For entities, name and type are always present. About 20% of the entities contain a link to an external knowledge base (Freebase), and some entities also contain subtypes (e.g., politician). A secondary challenge is that the names are present in three languages (English, Russian and Ukranian), and extractions are noisy.

To address the sparsity of information, our approach is to enrich the information present in each entity using information from an external knowledge graph. For this purpose we use Wikidata as it contains a large number of entities, including entities that do not satisfy the notability criteria in Wikipedia. Wikipedia contains over 60 million en-

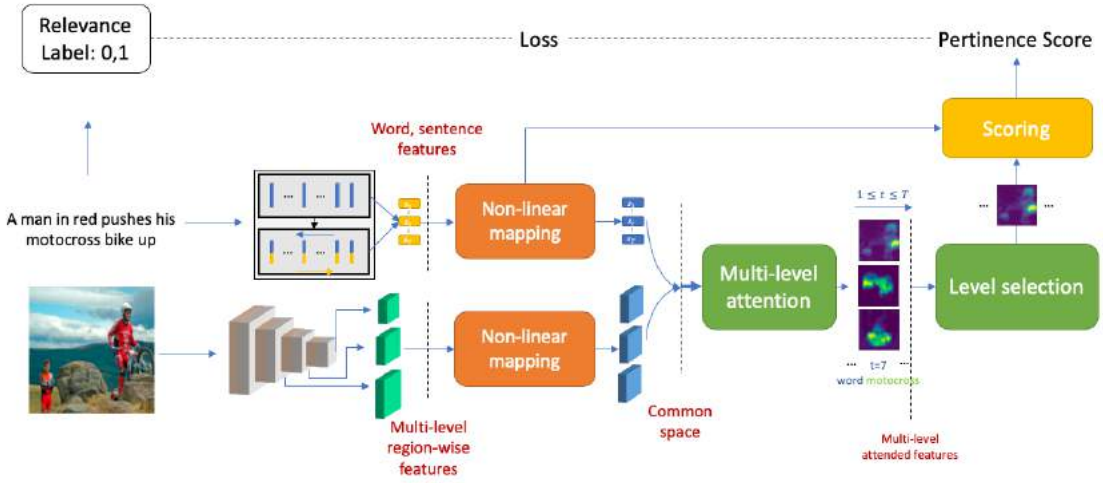

Figure 4: An overview of our cross-modal coreference (visual grounding) system.

tities, compared to the roughly 5 million entities present in the English Wikipedia. A secondary benefit of Wikidata is that it is a multi-lingual knowledge graph, containing labels and aliases for all entities in multiple languages, including Russian and Ukranian.

Our entity enrichment approach leverages the links to the external knowledge graph. Wikidata entities contain Freebase identifiers, so our algorithm first maps the Freebase identifiers to Wikidata indentifiers to link TA1 entities to Wikidata entities. Then the algorithm forms entity clusters by collecting into a cluster all entities that link to the same Wikidata entity. This initial set of clusters contain only those TA1 entities for which an external link is present.

The second step of the algorithm the enriched set of entity labels to add to clusters entities whose labels match a label for an entity already present in the cluster. To compare labels we use Jaccard similarity with a 0.4 threshold. Our experiments revealed that a high threshold produces good results as it reduces false positives. The expectation is that after label enrichment the likelihood of a close match is increased. When a TA1 entity cannot be added to an existing cluster, it forms a new cluster.

5 HypoGator: Alternative Hypotheses Generation and Ranking

HypoGator is the University of Florida hypothesis Generation system. It uses a search-score-rank approach to find alternative answers to complex queries over the automatically extracted TA2 knowledge graph. In a nutshell, HypoGator decomposes a complex graph query into subqueries of simple subgraph patterns. For each subquery its entry points are matched into the TA2 knowledge graph and their local context generates candidate answers. Candidates are scored and ranked using multiple features that are indicative of coherence and relevance. A join algorithm combines the answers from each atomic sub-query and re-scores the final set of answers using features that encourage answer cohesion. Figure 5 shows the architecture of the proposed system. In this section we describe the core components of HypoGator.

5.1 Query Processing

A statement of information need is a subgraph pattern with event/relation types and entities as nodes, event/relation argument roles as edges and a set of grounded entities known as entry points. We classify an information need as simple if each entry points is used as the argument of only one event/relation. In contrast, a complex information need has entry points that are shared by multiple events/relations developing into a star-like structure. HypoGator's query processing module first scan an information need and decomposes a complex information needs into multiple simple information needs that we refer to as atomic. The decomposition algorithm first finds all connected components in the information need and for each component, the algorithm visits the neighbors of its entry points and traverse each of them until a different entry point or a terminal node is found. The resulting subgraphs are added to the atomic query list. Figure 6 shows the statement of infor-

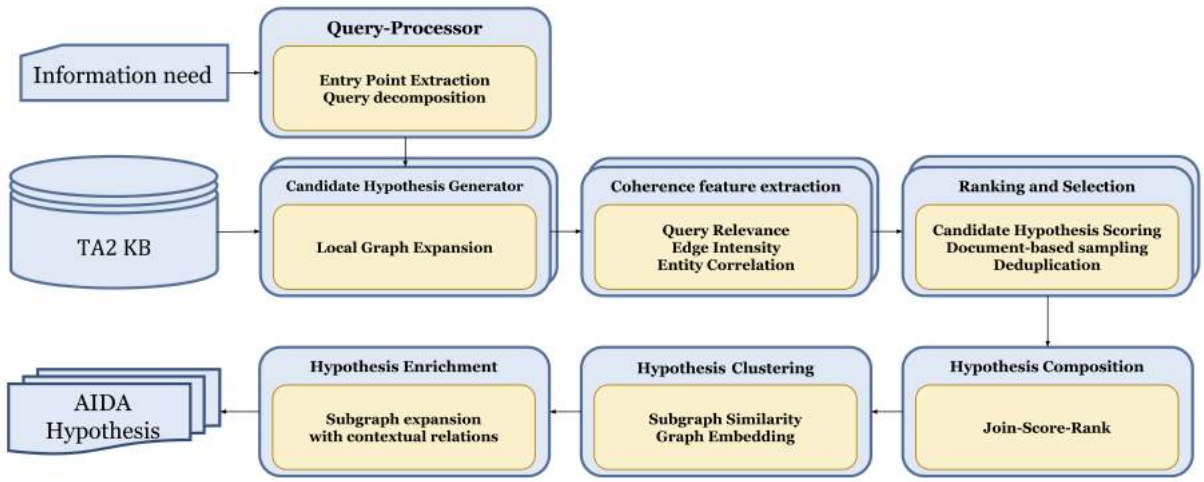

Figure 5: HypoGator System Architecture

mation need *E102* with entry points *Odessa (a.k.a. Odesa)* and *Trade Unions House (a.k.a. Trade Unions Building)* in the center and atomic queries derived from it using the decomposition algorithm round it.

After query decomposition, HypoGator matches entry points into the TA2 knowledge graph. At first, HypoGator matches entry points with background KB ids directly to the corresponding entity mention id in the TA2 KG. If the background KB id can not be matched or the query doesn't have one, HypoGator uses the entry point name string to match TA2 entity mention nodes using common string similarity metrics. Finally, entry points are attempted to be matched using the provenance offset.

5.2 Candidate Hypothesis Generation

To generate relevant hypothesis for an atomic query HypoGator explores the neighborhood of the entry point mention in the knowledge graph using the A* algorithm. Te results is a set of paths up to a fixed length that serve as backbone structure for candidate hypothesis. While the resulting paths are rooted at entity nodes, the generated paths visit event and relation nodes, to generate full candidate hypothesis HypoGator expands a path by adding all arguments of the visited event and relations in the path.

5.3 Coherence Feature Extraction

Our candidate generation module ensures that the generated candidate hypothesis include the entry points. However, this does not guarantee them to be fully relevant to the query at hand. Moreover, the candidates need to be pruned if they are not logically or semantically coherent. Another important factor determining the quality of a candidate hypothesis is the validity confidence of each of its knowledge elements, whether they are from the document sources (extraction confidence) or inferred (inference confidence) or TA2 clustering.

We use a variety of features to measure each hypothesis's semantic coherence, logical coherence, and degree of relevance to the query. We use an aggregation method to obtain an overall confidence score from each knowledge elements confidence. For example we use an ensemble of graph distance functions to measure the query relevance or use a set of predefined logical rules to detect logical inconsistency. The overall score for each hypothesis is computed as a linear combination of the individual scores from each of the features. We use the LDC labeled data to learn appropriate weights for each feature or use reasonable handcrafted weights for each feature.

5.4 Ranking and Selection

While we have multiple features and each of them scores the hypothesis for some important consistency or coherence property, we need a condense score that can be used to give full quantitative significance to a hypothesis and therefore use it for ranking candidates. We use a simple approach to aggregate different scores, a weighted sum of the feature values. We manually select the weights with what we believe are more salient features of a hypothesis. We look forward to include a learned version of the weights.

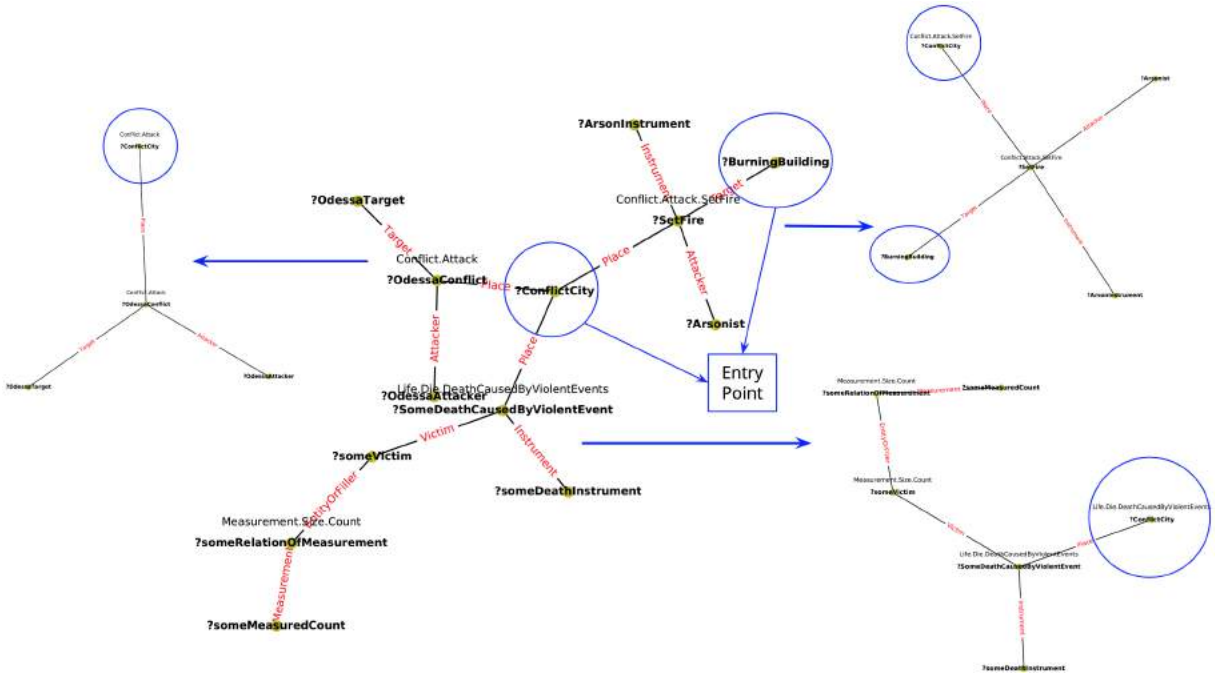

Figure 6: Query Decomposition E102

5.5 Hypotheses Clustering

Due to the nature of AIDAs data e.g. multiple documents about the same hypothesis, 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, TA2s 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.

5.6 Hypotheses Composition

After all candidate hypothesis are scored, deduplicated and ranked for each atomic part, we use a join algorithm to combine atomic hypothesis. The algorithm generates all possible combinations of candidate atomic answers. We set the value of **K** in the clustering step such that the cross product of atomic hypothesis is feasible. Every resulting combinations is deemed as a full hypothesis. Full hypothesis are re scored using the same same set of features, in addition a new score is given based on how far each connected component is with respect to others in the same hypothesis. The top resulting hypotheses are selected as final candidates and further deduplicated by clustering. The representatives of each cluster are returned as final hypotheses.

5.7 Hypothesis Enrichment

Based on development observation, we found that the hypothesis generated by HypoGator matched totally or partially the core elements found in the statement of information need, but lack a mechanism to add context to the answer. E.g. an information need may ask about the dead of a politician, and hypo gator may be able to find the correct *life.die* event but it would not contain contextual information such as who did the killer work for, or the details of the weapon used in the assault. Therefore, we increase the size of the hypothesis by including entity neighbors from the background knowledge base. Since the random addition of neighbors may induce noise into the hypothesis, we only add context to certain entities of interest e.g., the entities of type PER. the enrichment method also allows to select what kind of contextual relations we are relevant e.g., the relations of type *AFFILIATION, RESPONSIBILITY, DELIBERATENESS, etc.* Context expansion, increases the reach of our hypotheses while maintaining the core set of matched elements intact.

6 Hypothesis Generation at ISI

TA2 generates a full Knowledge Graph (KG). Given such a KG and a file containing the Statements of Information Need (SIN) 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 an XML file containing the SIN to SPARQL and perform a query on the full KG generated by TA2.

Convert the full KG to data in an internal format. The data in this internal format only contains information about the knowledge elements that are within a user-specified number of hops from the Entry Points and that will be used for reasoning, such as the arguments of Events and Relations. Each of these arguments has a confidence value.

Generate ontological constraints. Based on the LDCOntology, we generate two types of ontological constraints: (a) domain constraints, and (b) cardinality constraints. For each type of Event or Relation, each argument can accept only specific types of objects. Such constraints qualify as domain constraints. At a practical level, sufficient reasoning done by TA1 and TA2 already minimizes the number of domain constraints to be added by TA3. The cardinality constraints, often implemented as mutual exclusion constraints, can be added for two different reasons. First, 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. Second, each Entity can serve the same role only in a limited number of Events or Relations. For example, each Person Entity can serve the argument Person in only one Life.Born Event.

Represent the new data as weighted constraints. In our approach, assertions in the KG are not automatically deemed to be true in a hypothesis. This is because they have certain confidence values associated with them that have to be reasoned about jointly with the ontological constraints. 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 an instance of the Weighted Constraint Satisfaction Problem (WCSP) defined on the weighted constraints. We use the Toulbar2 solver to solve the WCSP instance. This solver finds an optimal assignment of values to all Boolean variables that minimizes the total weight. This optimal assignment is the top hypothesis base of the original problem. To generate the k^{th} hypothesis base, for $1 \leq k \leq K$, we construct a new WCSP instance that includes all the weighted constraints described above as well as additional constraints that disallow the previously generated $k-1$ hypothesis bases. Each hypothesis base is consistent with the ontological constraints.

Filter edges relevant to the SIN and construct hypotheses. We generate the k^{th} final hypothesis from the k^{th} hypothesis base by retaining only those edges that are relevant to the SIN. This relevance filtering mimics the popular *Word2Vec* model in Natural Language Processing. For each Event or Relation, we calculate similarity scores between its type and the Event types specified in the SIN. We then retain only the M Events and Relations with the highest similarity scores.

Score the hypotheses. The total score of a hypothesis is the average of its relevance score and its consistency score. In turn, the relevance score of a hypothesis is the average similarity score over all Events and Relations in it, as derived from the *Word2Vec* model in the previous step. The consistency score of a hypothesis is e^{-w}, where w is the total weight of the optimal assignment to the WCSP instance that defines its base. This score uses the transformation rule that defines weighted constraints from confidence values.

Acknowledgments

This work was supported by the U.S. DARPA AIDA Program No. FA8750-18-2-0014. 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

Google landmark recognition challenge.

Hassan Akbari, Svebor Karaman, Surabhi Bhargava, Brian Chen, Carl Vondrick, and Shih-Fu Chang. 2019. Multi-level multimodal common semantic space for image-phrase grounding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12476–12486.

Takuya Akiba, Tommi Kerola, Yusuke Niitani, Toru Ogawa, Shotaro Sano, and Shuji Suzuki. 2018. Pfdet: 2nd place solution to open images challenge 2018 object detection track. *arXiv preprint arXiv:1809.00778*.

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*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

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

Xiaocheng Feng, Bing Qin, and Ting Liu. 2018. A language-independent neural network for event detection. *Science China Information Sciences*, 61(9):092106.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Jeremy Getman, Joe Ellis, Stephanie Strassel, Zhiyi Song, and Jennifer Tracey. 2018. Laying the groundwork for knowledge base population: Nine years of linguistic resources for tac kbp. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

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

Heng Ji, Ralph Grishman, Dayne Freitag, Matthias Blume, John Wang, Shahram Khadivi, Richard Zens, and Hermann Ney. 2009. Name extraction and translation for distillation. *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

M. V. Kopotev and L. A. Ianda. 2006. Natsionalny korpos russkogo iazyka.

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.

Ying Lin, Liyuan Liu, Heng Ji, Dong Yu, and Jiawei Han. 2019. Reliability-aware dynamic feature composition for name tagging. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 165–174, Florence, Italy. Association for Computational Linguistics.

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

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

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 Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Joakim Nivre et al. 2019. Universal dependencies 2.4. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. 2017. Large-scale image retrieval with attentive deep local features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3456–3465.

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 71–76. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with Abstract Meaning Representation. In *Proc. the 2015 Conference of the North American Chapter of the Association for Computational Linguistics, Human Language Technologies (NAACL-HLT 2015)*.

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*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

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.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Ananya Subburathinam, Di Lu, Heng Ji, Jonathan May, Shih-Fu Chang, Avirup Sil, and Clare Voss. Cross-lingual structure transfer for relation and event extraction.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

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*. ACM.

Theresa Wilson. 2008. Fine-grained subjectivity and sentiment analysis: recognizing the intensity, polarity, and attitudes of private states.

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

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, Heng Ji, and Avirup Sil. 2019. Joint entity and event extraction with generative adversarial imitation learning. *Data Intelligence*, 1(2):99–120.

Tongtao Zhang, Ananya Subburathinam, Ge Shi, Lifu Huang, Di Lu, Xiaoman Pan, Manling Li, Boliang Zhang, Qingyun Wang, Spencer Whitehead, et al. 2018. Gaia-a multi-media multi-lingual knowledge extraction and hypothesis generation system. In *Proceedings of TAC KBP 2018, the 25th International Conference on Computational Linguistics: Technical Papers.*

Ruiqi Zhong, Steven Shao, and Kathleen McKeown. 2019. Fine-grained sentiment analysis with faithful attention. *ArXiv*, abs/1908.06870.

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.