Entity and Event Extraction from Scratch Using Minimal Training Data

Laura Wendlandt
University of Michigan
wenlaura@umich.edu

Steve Wilson
University of Michigan
steverw@umich.edu

Oana Ignat
University of Michigan
oignat@umich.edu

Charles Welch
University of Michigan
cfwelch@umich.edu

Li Zhang
University of Michigan
zharry@umich.edu

Mingzhe Wang
Princeton University
mzwang@umich.edu

Jia Deng
Princeton University
jiadeng@cs.princeton.edu

Rada Mihalcea
University of Michigan
mihalcea@umich.edu

November 2018

1 Overview

Understanding current world events in real-time involves sifting through news articles, tweets, photos, and videos from many different perspectives. The goal of the DARPA-funded AIDA project[1] is to automate much of this process, building a knowledge base that can be queried to strategically generate hypotheses about different aspects of an event. We are participating in this project as a TA1 team, and we are building the first step of the overall system. Given raw multimodal input (e.g., text, images, video), our goal is to generate a knowledge graph with entities, events, and relations.

Figure 1 shows an overview of our pipeline. The first stage is pre-processing. This involves translating all the raw documents, as well as transcribing and translating audio and video data. All the translated information is input to our main processing module that extracts entities, events, and relations. Entities are extracted from both text and video data. In the final, output generation stage of the pipeline, we build a graph from all of the entities, events, and relations.

[1] https://www.darpa.mil/program/active-interpretation-of-disparate-alternatives

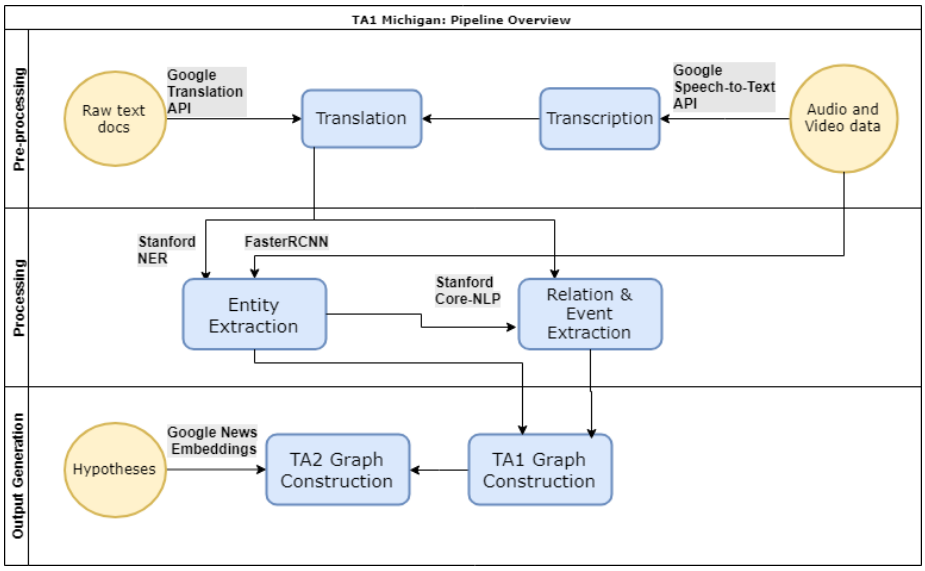

Figure 1: Overview of the system pipeline.

This graph is then used by other teams on the AIDA project to generate hypotheses, which are passed back to us and used to update the confidence scores in our graph.

2 Pre-processing

We take in raw text, audio, and video as input. These files must be pre-processed before we can extract entities, events, and relations.

2.1 Transcription

Since the video files are mainly about news, speech and interview, the knowledge and information inside speech is essential to fully understand the video content. We first got the transcription by speech-to-text.

We extracted the pure audio tracks from video files and sent them into the Google Speech-to-Text API[2] which supports the transcription for English, Russian and Ukrainian. These transcripts were then combined with normal textual documents and processed in the same way to identify textual knowledge elements. In Section 5, we describe how to identify visual knowledge elements from video frames and textual knowledge elements in transcription.

[2]https://cloud.google.com/speech-to-text/

2.2 Translation and alignment of non-English text

For each raw document, we detect the language of the text and translate non-English (Russian and Ukrainian) text into English. Each file is originally provided in the .ltf format, and we begin by converting the files to .rsd files, a more suitable input format for translation as it removes unnecessary information.

We use the Google Cloud Translation API[3] to automatically detect the language of the text data. In comparison to other services, we found that the Google Cloud Translation API achieves qualitatively the best results. For future work, we plan to experiment with a pre-trained neural network model (e.g., [2], [7]) for translating the data.

After each text document is translated, we extract the entities, events and relations from them (discussed in following sections). Because this processing is done on translated documents, the preliminary output contains only English entities, events, and relations. However, we need entities, events, and relations in the original language of the document.

To back-align extracted elements into their original language, we translate each token back into its original language and search for its start and end offsets. Because we consider each token separately, a large portion of the words are not correctly translated, and we cannot extract the start and end offsets for them. For future evaluations, we will improve this technique.

3 Extracting Entities from Language

Once each document is translated, we extract entities and events from the text. To do this, we combine a few approaches: entities from a pre-trained named entity recognition (NER) system, entities from keyword extraction, and entities from the Wikipedia category hierarchy.

Stanford NER

We begin by using the Stanford NER system [1] to extract all entities from the translated text. Once we have extracted entities, we then label each entity with an AIDA entity category. We do this by looking at the training data and assigning each CoreNLP category to the AIDA category that it most frequently captures. If there are no training examples of a particular CoreNLP category, then the most common AIDA category, Person, is used. Our mapping can be found in Table 1. Many of the mappings are quite intuitive, though some are skewed toward the domain of international conflict (e.g. Miscellaneous maps to Geopolitical Entity).

Because of the nature of the AIDA project, we want to focus on achieving high recall for entity extraction. The results of recall using the Stanford CoreNLP system are shown for several AIDA entity categories in Table 2. Recall is fairly high for the categories that the Stanford NER system was specifically trained for (e.g., geopolitical entities, people, locations), yet recall drops very low for categories that are more domain-specific (e.g., crimes, weapons).

[3] https://cloud.google.com/translate/

CoreNLP Category	AIDA Category
Title	Person
Organization	Organization
Person	Person
Nationality	Geopolitical Entity
Miscellaneous	Geopolitical Entity
Criminal Charge	Crime
Cause of Death	Weapon
Number	Person
Country	Geopolitical Entity
City	Geopolitical Entity
Location	Geopolitical Entity
Date	Time
State or Province	Geopolitical Entity

Table 1: Our mapping between CoreNLP named entities and AIDA categories.

In order to address this issue, we consider two additional ways to extract entities that are more domain-specific, keyword extraction and entities from the Wikipedia category hierarchy.

AIDA Category	Recall
Geopolitical Entity	63%
Time	52%
Title	50%
Location	43%
Person	30%
Facility	22%
Numerical Value	17%
Organization	17%
Weapon	8.6%
Vehicle	5.9%
Commodity	5.6%
Crime	3%

Table 2: Recall on AIDA entity categories using the Stanford CoreNLP NER system.

Keyword Extraction

To extract keywords from translated text documents, we use the email keyword extractor [4]. The pipeline of the system is as follows. First, it segments and tokenizes the texts. Then, it extracts candidates by applying named entity resolution, which are pre-processed by removing punctuation, folding to lowercase, and removing numbers and leading and trailing stopwords. At this point, the system ranks words using several linguistic and centrality-based features,

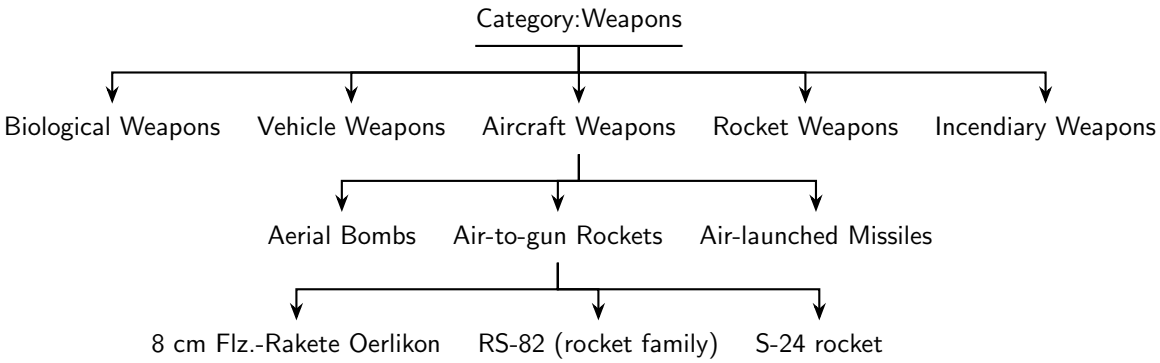

Figure 2: A subset of the Wikipedia Weapons Category.

and then collapses the top-ranked adjacent words to form keyphrases. Finally, the system applies a post-processing heuristic that constructs longer key phrases starting with the selected keywords, by collapsing adjacent words from the top ranking into phrases.

As the algorithm rank the extracted keywords by confidence, we extract top ten keywords from each document and treat them as entities.

Wikipedia

In addition to using entities from Stanford NER and keyword extraction, we extract domain-specific entities from Wikipedia. We do this using the extensive Wikipedia hierarchy system (Figure 2 shows a subset of the hierarchy) [8]. For each category of interest, we iterate through the category tree and extract a list of entities. If any of these entities are found in our AIDA text, we mark it as an entity. We do this for three categories: weapons, vehicles, and crimes.

4 Extracting Events and Relations from Language

Given a set of entities of interest, we then use the parsed language output to infer events and relations as outlined in the seedling ontology. Two approaches are considered to obtain coarse labels for the possible relationships between the entities: a graph-based and a span-based approach.

4.1 Graph-Based Approach

First, we construct a small graph, $G_S = (V_S, E_S)$ for each parsed sentence, S, where $V_S = \{w|w \in S\}$ and w is a word. The graph's edges, E_S, are defined by the set of dependency links between words (vertices) as well as adjacency links that occur between two words, w_1 and w_2. If w_2 immediately follows w_1 in S and both w_1 and w_2 belong to the same noun phrase or verb phrase as

identified by the syntactic constituency parse of S, then we add an adjacency link between them. Then, to identify relationships between entities, we consider the set of vertices corresponding to words that are part of a given entity phrase and explore all incoming and outgoing edges. If any edge links to a vertex that represents a words belonging to a different entity of interest, we determine that there is a relationship between these two entities. If the connecting edge is a dependency edge, we use the dependency label as a coarse initial label for the relationship. Otherwise, if the edge is phrase-based (i.e., the two entities are part of a larger identified phrase), we use the span of words between the entities as a *plain text label* for the relationship.

4.2 Span-Based Approach

In this alternative method, we rely on the ordering of words in the parsed sentences to naturally convey information about the relationships between entities. Given two entities that appear in the same sentence, we extract the span of words between the end of the first entity and the start of the second entity. We do not consider spans that are too long (we define this as a 15+ word span) or those that contain end-of-sentence markers, since these likely do not capture a meaningful connection between the two entities. All other spans are treated as a relationship between the two entities, with the span itself being treated as a *plain text label*.

After identifying pairs of entities and *plain text labels*, we use a nearest-neighbor matching approach to predict the relationship or event label from the ontology, as described in the next section.

4.3 Entity, Relation, and Event Matching

To assign entity types from the ontology to the extracted keywords, we use a nearest-neighbor approach based on word embedding similarity, using the GloVE word emnbeddings [5]. Since each entity or keyword is a word or a short phrase, we compute its embedding as the average of word embeddings:

$$E(ent) = \frac{1}{N} \sum_{w} E(w) \tag{1}$$

where $E(\cdot)$ is the embedding function, w is a word in the entity ent and N is the number of words in the entity. In the *entity mentions* training data, each entity is associated with a type. Treating each type as a cluster, the embedding of the cluster is the average of all embeddings of the entities in that cluster:

$$E(C) = \frac{1}{M} \sum_{ent \in C} E(ent) \tag{2}$$

Then, each keyword is assigned to the cluster, whose embedding is the closest to that of the keyword, determined by cosine similarity:

$$Category(keyword) = argmax_C(cosine(E(C), E(keyword))) \tag{3}$$

Relations and evsnts are handled similarly. However, as there was a smaller set of annotations data available for many even and relation types, we instead matched each *plain text label* with the event or relation *definition* provided in the ontology. Each *definition* and *plain text label* was embedded with GloVE vectors using equation 1. The relation or event was then assigned to the *definition* whose embedding had the highest cosine similarity to the *plain text label* embedding. The first and second argument to each event or relation were the subject and object entities involved in the relation or event, respectively.

5 Extracting Entities from Images and Videos

In our current framework, we focus on extracting visual entities and leave identifying relations and events as future work. Our key observation is that the important knowledge elements from images and videos are usually mentioned in their corresponding textual documents (for images) or transcription, so we design a two step pipeline to identify visual knowledge elements. Given an image or a video frame, we first detect objects via pretrained object detectors, then align each object with one knowledge element in the texts or transcription and group them together.

Since the AIDA project contains many documents in specific domains, such as weapons, we hope our detector could identify objects in a large scale of categories. We choose the FasterRCNN [6] model pre-trained on the OpenImage[3] dataset which is the largest training data available for object detection containing images from about 600 categories. After applying this detector on images and video frames, we obtain several bounding boxes for each image and each bounding box has a coarse category label such as person or car.

To assign each bounding box a specific knowledge element, we matched them with the pre-extracted knowledge elements from corresponding textual documents for images or transcriptions for videos. Given a set of bounding boxes $\{b_1, ..., b_m\}$ with labels $\{x_1, ..., x_m\}$, and a set of knowledge elements $\{e_1, ..., e_n\}$ with tags $\{y_1, ..., y_n\}$ that can be either its textual description (such as MH-17 or President Obama) or its type (such as person or vehicle), we first calculate the word embedding v for image labels and knowledge element tags $u_i = w(x_i)$, $v_j = w(y_j)$ where w is the embedding function. Then we use cosine similarity between u_i and v_j as the matching score the bounding box b_i and the knowledge element e_j. For each bounding box, we select its best matched knowledge element and add it as a visual mention for this knowledge element.

6 Graph Construction

Using the extracted entities, relations, and events we combine this information into AIF formatted graphs for each document. We generate the graphs for Task 1A using these knowledge elements from the audio, video, images and text and then look at how to regenerate the graphs for Task 1B using a hypothesis graph

as input.

For each document we create a graph which links our entities, relations, and events to our system. For entities with the same IDs given by our extraction step we create a single entity node in the graph and assign it to a cluster by itself. We then add to these entities the set of text, image, and key frame justifications for that entity ID. We then take the entities and link them to extracted relations and events by adding them as arguments in the graph. We have only text justifications for events and relations so we add the set of text justifications as a compound justification for the associated event/relation.

To adjust graph generation for Task 1B we incorporate the hypothesis information by adjusting the confidence scores for our justifications. We take each hypothesis file and generate an embedding of the hypothesis by extracting the strings from the hypothesis that exist in hasName or textValue attributes (we call this set H) and taking the max value from the embeddings in each dimension. We use the Google News embeddings of the tokens in these strings after translating them all into English:

$$h_{emb} = max_{h \in H}(emb(h)) \tag{4}$$

We then adjust the confidence scores, c, of entities in our graph based on the cosine similarity of the text span, T, associated with that entity to the hypothesis embedding:

$$t_{emb} = max_{t \in T}(emb(t)) \tag{5}$$

$$c = 1.0 - \frac{t_{emb} \cdot h_{emb}}{\|t_{emb}\| \|h_{emb}\|} \tag{6}$$

The graphs contain the same knowledge elements as in Task 1A but have the updated confidence scores c.

7 Conclusion

In this project, we built a complete pipeline to extract entities, relations and events from multimedia resources including text, videos and audio. Many individual problems, such as machine translation, speech recognition, named entity recognition and object detection were tackled using off-the-shelf NLP and CV toolboxes. We also propose to extract relations and events from text using graph-based and span-based approaches and then match knowledge elements inside text and between text and videos using embedding-based approaches. Finally, the generated knowledge elements are written as a knowledge graph in the AIF format.

This pilot evaluation presented a lot of challenges in terms of organization and focus. We propose for future evaluations that each team focuses on what knows best. We, at Michigan, are currently interested in working on human action detection and recognition and the events associated with those actions.

Acknowledgments

This material is based in part upon work supported by the DARPA AIDA program under grant #FA8750-18-2-0019. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency.

References

[1] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 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, 2005.

[2] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. Opennmt: Open-source toolkit for neural machine translation. *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, 2017.

[3] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv:1811.00982*, 2018.

[4] Shibamouli Lahiri, Rada Mihalcea, and P-H Lai. Keyword extraction from emails. *Natural Language Engineering*, 23(2):295–317, 2017.

[5] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

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

[7] Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, 2017.

[8] Jakob Voss. Collaborative thesaurus tagging the wikipedia way. *arXiv preprint cs/0604036*, 2006.