NewsReader at SemEval-2018 Task 5: Counting events by reasoning over event-centric-knowledge-graphs

Piek Vossen
VU University Amsterdam / De Boelelaan 1105, 1081HV Amsterdam, Netherlands
piek.vossen@vu.nl

Abstract

In this paper, we describe the participation of the NewsReader system in the SemEval-2018 Task 5 on *Counting Events and Participants in the Long Tail*. NewsReader is a generic unsupervised text processing system that detects events with participants, time and place to generate Event Centric Knowledge Graphs (ECKGs). We minimally adapted these ECKGs to establish a baseline performance for the task. We first use the ECKGs to establish which documents report on the same incident and what event mentions are coreferential. Next, we aggregate ECKGs across coreferential mentions and use the aggregated knowledge to answer the questions of the task. Our participation tests the quality of NewsReader to create ECKGs, as well as the potential of ECKGs to establish event identity and reason over the result to answer the task queries.

1 Introduction

This paper describes the NewsReader system participating in the SemEval 2018 Task 5 *Counting Events and Participants in the Long Tail* (Postma et al., 2018). Task 5 requires detection of certain events (cases of gun violence, dismissal of employees, and burning fire incidents) with participants, as well as extraction of their location and time in a set of documents and reasoning over their identity, in order to answer queries over the data set. A typical query in the task is *How many people were killed in 2016 in Columbus, MS?*. Participants were given a collection of news articles in CoNLL format to distill the answer to the queries. The task consists of 3 subtasks: subtask 1 asks for all documents from the data set that report on a single incident that fits the question constraints; subtask 2 is to provide the number of incidents (zero or more) and the documents that report on these incidents given the constraints of a query; subtask 3 asks for the number of affected (injured, dead, or fired) people in the incidents that match the query constraints, and the supporting documents for these incidents. In addition to answering the subtask queries, participants were asked to mark the (cross-document) coreferential event mentions in the CoNLL file according to a specific event schema for gun violence.

Since we also organized the task, we decided to participate out-of-competition. Our system is a version of the NewsReader system (Vossen et al., 2016) as it was delivered at the end of the project, with as little adaptation as possible to the processing of the text to answer the queries of the task. The generic NewsReader system created the semantic output by applying a deep reading approach to the text, and the tasks were addressed by loading that output and reasoning over the results.

We participated in all three subtasks by first resolving the event coreference (or identity) and next answering the questions for each task using event representations that are the results of resolving the coreference. Our approach consists of three steps: 1. the event mentions in the input documents are represented as Event-Centric Knowledge Graphs (ECKGs) using the NewsReader system as is. 2. the ECKGs of all documents are compared to each other to decide which documents refer to the same incident, resulting in an incident-document index and in cross-document event-coreference relations. 3. the constraints of each question (its event type, time, participant names, and location) are matched with the stored ECKGs, resulting in a number of incidents and source documents for each question.

Our approach is fully unsupervised and follows compositional semantic principles to 1) define the semantics of events and participants, 2) establish their identity, and 3) reason over the results.

The remainder of this paper is structured as follows. In Section 2, we briefly describe the NewsReader system and the preprocessing steps. In Section 3, we explain how we establish event identity and cross-document coreference starting from the NewsReader output. The aggregated event representations are used to answer the queries for the tasks, which we describe in Section 4.

Finally in Section 5, we discuss the results. For further details on the NewsReader system, we point to the NewsReader website and its Github repository.[1] The specific wrapper for this task that takes the Newsreader output as a starting point is available in a separate Github.[2]

2 The NewsReader system

NewsReader processes text by applying a wide range of NLP modules, among which named entity recognition, classification and disambiguation (NERCD), semantic role labeling (SRL), word sense disambiguation (WSD), and temporal expressions detection and normalization (TIMEX). The NLP modules store their output as separate layers in the Natural language processing Annotation Format (NAF) (Fokkens et al., 2014). For example, events are detected by the SRL system as PropBank predicates (Kingsbury and Palmer, 2002), while FrameNet frames (Baker, 2008) and Wordnet synsets (Fellbaum, 1998) are attached to these predicates on the basis of the WSD output. Similarly, NERCD will annotate the text with entities and entity classes and it will annotate some of them with DBpedia URIs. From these entities, we derive participant names and locations for the predicates in the SRL output, while TIMEX anchors these predicates to dates.

In a second step, NewsReader derives so-called Event-Centric-Knowledge-Graphs (ECKGs) by combining the results of the NLP module. The ECKGs follow the Simple Event Model (SEM) (Van Hage et al., 2011), which represents events as instances through URIs with relations to their participants, location, and time. The same event instance and participants can be mentioned several times throughout a text and across different documents. Identity across mentions is then modeled through the Grounded Annotation Framework (GAF) (Fokkens et al., 2013), by giving event instances the same unique URI in SEM and pointing to the different mentions in the source text via a *denotedBy* relation to mention URIs based on their offsets. Assigning unique URIs to coreferential event mentions results in ECKGs with all the knowledge and information aggregated across mentions in the form of RDF properties for this subject URI.

Figure 1 shows two examples of ECKGs with the event URIs 094fe5921b642e30a00cd52ece7b0157#ev1 and 60ad5103290ae7aa16e39d3cd2695496#ev1. The ECKGs are derived from two mentions in two different documents. The triple representations capture the following properties for each event: subclass relations with WordNet synsets and FrameNet frames, *denotedBy* pointers to the offset positions in the original texts, the words or labels used to mention the event, PropBank roles filled by DBpedia URIs, or unresolved phrases that are not entities and finally, the date to which the events are anchored.

In this output of NewsReader, we did not apply any event coreference and we represent each mention as a separate event instance or ECKG. The WordNet synsets and FrameNet frames are associated through the WSD modules in NewsReader. We used the UKB (Agirre and Soroa, 2009) and IMS ((Zhong and Ng, 2010) to score the WordNet synsets for each predicate. Next, we take the highest scoring synsets and use the Predicate Matrix (Carreras et al., 2014) to obtain the associated FrameNet frames. The interpretation of the predicates as events for the task is thus derived from the SRL output in combination with the WSD output and the Predicate Matrix association.

We call the above output of NewsReader the **raw-ECKGs**. In the next sections, we describe how we post-process these to derive so-called **task-ECKGs** with only the information relevant for the task. We finally reason over these task-ECKGs to answer the queries. Both the raw-ECKGS and the task-ECKGs are available in the Github repository, including the scripts to extract the latter from the former.

3 Event coreference

As a first step for the task, we read the raw-ECKGs and filter out only those events that are relevant for the task: see section 3.2 for details. Next, we establish event identity across the different event

[1]www.newsreader-project.eu and https://github.com/newsreader

[2]https://github.com/cltl/nwr-semeval2018-5

```
094fe5921b642e30a00cd52ece7b0157#ev1
   a                  wn:eng-30-00069879-v , wn:eng-00069879-v ,
                      fn:Cause_harm , fn:Experience_bodily_harm ;
   gaf:denotedBy      094fe5921b642e30a00cd52ece7b0157#char=11,18;
   skos:prefLabel     "injure" ;
   pb:A1              dbpedia:East_Palo_Alto,_California ;
   time:inDateTime    date:20130505 .

60ad5103290ae7aa16e39d3cd2695496#ev1
   a                  wn:eng-30-00069879-v , wn:eng-30-07950786-n ,
                      wn:eng-02738701-v , wn:eng-01882814-v ,
                      fn:Cause_harm , fn:Path_shape , fn:Travel ;
   gaf:denotedBy      60ad5103290ae7aa16e39d3cd2695496#char=9,16;
   skos:prefLabel     "wound"  ;
   pb:A1              semeval2018-5:non-entities/person ;
   pb:A2              dbpedia:East_Palo_Alto,_California ;
   time:inDateTime    date:20130505 .
```

Figure 1: ECKG representation of events extracted by NewsReader for two different mentions in two different documents, showing the type of event, the mentions linked through the *denotedBy* property, the PropBank roles, the date and the actual words used to make reference. The denotedBy links are simplified to reduce space.

mentions: see subsections 3.3 and 3.4. For this, we assume that each document reports on a single incident and mentions within a document are coreferential. We carried out the following steps for this:

1. We build an index of all documents that report on the same incident as follows:
   (a) We determine the incident time for a document.
   (b) We determine the overall incident type for a document: killing, injuring, job firing, or fire burning.
   (c) We compare all documents with the same incident time and incident type to further match the locations and participants.
   (d) If there are sufficient matching locations and participants across documents, we store them relative to the same incident.
2. Iterating over the incident-document index, we determine the mentions of incidents and their subevents over all documents that report on the same incident. We establish coreference relations among these mentions:
   (a) All mentions of the incident as a whole, e.g. *accident*, *shooting*, *this*, receive the same URI that represents the incident.
   (b) All further subevents of an incident (*hit*, *injure*, *death*) are identified by their subevent type and the victims associated within and across documents related to the same incident. Incident subevents of the same type and with the same victims become coreferential and receive the same URI.

In the next subsections, we explain these steps in more detail.

3.1 Incident time

The document-creation-time is given by the organizers for each document but it does not necessarily correspond with the date of the incident. We therefore extract the mostly mentioned year and month throughout the document and select the most frequently mentioned date for that year/month. We experimented with selecting different proportions of the text, as we assumed that the actual incident data is most likely mentioned in the beginning and other incidents from the past may be mentioned later in the text. We tested these approaches on the trial data and found that restricting the date references to the first two sentences gave the best results. If there are no time expressions in the first two sentences, we use the document-creation-time as the incident date as a fall-back.[3]

3.2 Incident type

For each document, we classify all the predicates for the event types of the task: *shooting*, *burning*, and *dismissal of employees* and count which type is most dominant. To classify the predicates, we

[3]We also experimented with other granularities e.g. by lumping incidents by the week of the month but these did not give better results.

collected all FrameNet frames that NewsReader assigned to the trial data and ordered the frames by frequency. We manually selected the following frames for each event type:

incident fn:Attack,fn:Catastrophe,fn:Cause_harm, fn:Destroying

kill fn:Cause_to_end,fn:Death,fn:Killing

injured fn:Cause_harm,fn:Cause_impact, fn:Experience_bodily_harm, fn:Hit_target,fn:Recovery,fn:Resurrection

hit fn:Cause_impact,fn:Hit_target

shoot fn:Shoot_projectiles,fn:Use_firearm

burn fn:Absorb_heat,fn:Apply_heat, fn:Setting_fire,fn:Fire_burning,fn:Fire_going_out

dismiss fn:Firing,fn:Quitting_a_place, fn:Quitting,fn:Get_a_job,fn:Hiring,fn:Employing, fn:Being_employed

We further noticed that some task-relevant words in the trial data were not matched with WordNet synsets or FrameNet frames by our system. After analysing the output of the trial data, we manually selected 84 predicates that were sometimes missed by the system (due to upper case, part-of-speech errors, out-of-vocabulary) to ensure higher coverage. We used this word list together with the FrameNet mappings to select only those events that are relevant for the tasks and derive the dominant event type of the document.

3.3 Incident-document index

After determining the dominant date and the type of incident, we compare documents with the same incident date and the same incident type to determine which documents report on the same incident. For this we compare the locations and the participants. If there is a sufficient degree of matching, we assume that documents report on the same incident.[4] For participants, we first check the names of the entities detected by the NERC module. If there was no match, we check all other phrases with PropBank A0 or A1 role, such as *person*, *child*, *girl* that denote persons, but are not classified as entities by NERC.[5] For locations, we assume that the entity linking software[6] found a match to DBpedia. We directly compare the DBpedia URIs for locations across the documents to find a match. Eventually when documents match in terms of all properties: same incident date, same incident type, one participant and one location, we assume they report on the same incident and we store them together in an incident-document index.

3.4 Coreference across incident mentions

The second step in this process establishes the event coreference relations across all event mentions in all the documents related to the same incident. All references to the incident as a whole will receive a unique URI that identifies that incident. For example, if a document has *shooting* as the dominant incident type, then we consider all references to *shooting* as a mention of the incident as a whole. We consider abstract incident mentions such as *catastrophe*, *accident*, and even pronouns such as *it* and *this*, as coreferential with the incident as a whole. Next, we extract all references to subevents, e.g. *hit*, *injured*, *death* as separate event instances relative to the incident to which they are associated. Subevents are separated by their subtype in combination with the participants or victims. Each subevent receives a URI that is composed of the incident URI, the subevent type, and the participant string. An example of such a subevent URI is shown in Figure 2. It starts with a document reference[7] followed by #incident, #INJURED and the words that make up all linked participant phrases: 6-year+old+girl+child+little+girl+person +young+girl. These participant phrases are aggregated across various mentions. In the case of phrases such as this one, it is difficult to reason over the participant identity; how many participants are injured? In this approach, we assume that the URI and therefore their identity was resolved by the generic processing of NewsReader. No specific matching strategy was implemented to establish coreference across participants. We see in Figure 2 the resulting list of distinct participant URIs based on their surface forms. In case of entity names, it is more likely to match participants across mentions directly through their URIs or their first name or surname. For example in Figure 2, the second ECKG shows

[4]In our experiments, matching a single participant and a single location was sufficient.

[5]In NewsReader, these phrases are typed as non-entities because they can refer to generic or role instantiations, e.g. *victim*, *mother*

[6]mendes2011dbpedia

[7]We arbitrarily take the name of the first document used in comparison just to get a unique URI

an incident reference with the participant name *Tewalt* that needs to match a participant in another document with the same name to find a match.

3.5 Aggregating properties

Establishing coreference through the same URI, results in further aggregation of all properties that were initially expressed for separate event mentions. We also normalised the properties by lumping all PropBank A0 and A1 roles to *sem:hasActor* for participants and *sem:hasPlace* for location. Figure 2 shows two examples of **task-ECKG** resulting after aggregating data from mentions from the **raw-ECKGs**. The first ECKG shows a subevent of the type INJURED based on the raw-ECKGs shown in Figure 1. The second ECKG shows an event at the incident level, aggregated over various mentions in the same document and their corresponding properties. We also include all subevents for the incident as links. There are subevents both for being injured and for death although the participants detected are linked only to injured subevents. This has consequences for answering subtask 3 questions for number of people injured or died.

After aggregating the properties, we store these task-ECKGs to an output file named after the incident date inside a subfolder that corresponds with each incident event type: (*BURN*, *DISMISS* and *SHOOT*). For example for the test data, the software created a BURN subfolder with 6 ECKGs files in TRiG-RDF format: 20071026.trig, 20071022.trig, 20151027.trig, 20151024.trig, 20070207.trig, 20070201.trig, each representing the incident date. Each of these TRiG files contains all incidents of the same type that are associated with the same date. For SHOOT and DISMISS the number of RDF files with incidents on the same date is far larger: 1,367 and 43 respectively.

In addition to storing the task-ECKGs, we also read the CoNLL file and annotate its tokens with numeric event identifiers by taking the checksum of each URI in the ECKGs and assign the checksum to each *denotedBy* match with a token. The annotated CoNLL file is submitted for the task.

4 Counting incidents and victims

Given the ECKG representations of incident instances and their subevents, it is straightforward to answer the task questions. To achieve this, we process all the questions and match their constraints with the knowledge on the task-ECKGs: the incident type, the date, the participant name, and location (if any). This results in a number of matching incidents and their associated source documents. For subtask 3, we additionally extract the victims for the specified subevents *killing* and *injuring*.

To obtain the answers, we first check the type of event in the query (*killing*, *injuring*, *fire_burning*, and *job_firing*) and match it with the subfolders of the adapted-ECKGs. We only consider the ECKGs files for a matching type. For *shooting* events that are differentiated into *killing* and *injuring* incidents, we additionally filter the ECKGs for the occurrence of the corresponding subevents. From each subfolder, we only load the ECKGs with matching dates. If there is no date constraint, we load all ECKGs. In case of a date constraint, we check if the constraint specifies a day or only a year or month. If no specific day is specified, we check if the ECKG files start with the corresponding year and/or month. Else if a specific day is asked for, we match the full date with the incident time.

We load all ECKG files that match the above constraints and consider each incident and its properties for further matching with other constraints on location or participant names. In the case of location, we first directly match the DBpedia URI for each incident against the sem:hasPlace properties. If there is no match, we expand the location in the query to DBPedia URIs that are related using spatial properties: north, east, west, south, northeast, southeast, northwest, and southwest of the specified location. For participants, we first check all participants of the incident that are classified as an entity of type PERSON by the NERCD module. We check the beginning of the name in case of a first name constraint and the ending of the name in case of a surname constraint.

After selecting incidents that match the query constraints, we derive the answers. In the case of subtask 1, we only provide the document ids for matched incidents and the numeric answer 1 (if any document was recovered). In the case of subtask 2, we count the unique number of incidents within the selected ECKGs and the associated documents of their mentions. If there are none, the answer is zero. If there is one or more, we provide the number of incidents and the associated document identifiers for their mentions. In the case of

```
 <094fe5921b642e30a00cd52ece7b0157#incident#INJURED#
   6-year+old+girl+child+little+girl+person+young+girl>
  a                 nwrontology:INJURED ;
  gaf:denotedBy     094fe5921b642e30a00cd52ece7b0157#char=11,18,
                    60ad5103290ae7aa16e39d3cd2695496#char=20,28, etc...;
  sem:hasActor      se2018-5:non-entity/6-year+old+girl ,
                    se2018-5:non-entity/person, se2018-5:non-entity/child ,
                    se2018-5:non-entity/young+girl, se2018-5:non-entity/little+girl ;
  sem:hasPlace      dbpedia:East_Palo_Alto,_California, dbpedia:Richmond,_California ;
  time:inDateTime   time:20130505 ;
  skos:prefLabel    "injury" , "injure" , "shoot" , "wound" .

<8554b200f12a9e9f6fed68f6795ada07#incident>
  a                 nwrontology:SHOOT ;
  gaf:denotedBy     8554b200f12a9e9f6fed68f6795ada07#char=2311,2318, etc. ;
  sem:hasActor      se2018-5:non-entity/kuna+man, se2018-5:entity/Tewalt ;
  sem:hasSubEvent   <8554b200f12a9e9f6fed68f6795ada07#incident#DEAD#>,
                    <#incident#INJURED#kuna+man>, <#incident#INJURED#Tewalt+kuna+man>,
                    <#incident#HIT#Tewalt>, <#incident#INJURED#Tewalt> ;
 sem:hasSubType     nwrontology:INJURED , nwrontology:DEAD , nwrontology:HIT ;
 time:inDateTime    time:20161228
 skos:prefLabel     "accident", "shooting", "incident", "gun", "start", "leave",
                    "hit", "it", "use", "discharge", "handling", "handle" .
```

Figure 2: task-ECKG representation of an *injure* event resulting from establishing event-coreference and aggregating the event properties across different mentions. The denotedBy and subevent links are adapted to save space.

subtask 3, we additionally extract all victims from the ECKGs as being *injured* or *killed*. If the victims in the ECKG are entities of the type PERSON, we count the unique list of names. If there are no entities of the type PERSON associated with the subevents, we simple count the unique strings of the non-entities associated with *injuring* or *death*. The victim count is then used to answer subtask 3 queries numerically.

5 Results and discussion

Our system answers the task queries using ECKGs in which event and participant identity is established by the unsupervised NewsReader system. Next, we reason over the properties of the ECKGs in relation to the query constraints. We did minimal adaptations for the task and the performance heavily relies on the quality of the ECKGs. The adaptations mainly involved detecting the relevant event types among all events detected by NewsReader, reasoning over the incident date and matching the location in the query with locations in the ECKGs using spatial relations from DBpedia. Furthermore, we relied on the one-document-one-incident heuristic. We expect that the system can be improved considerably by: 1) improving the incident-document index using state-of-the-art clustering techniques ((Li et al., 2005; Nicholls and Bright, 2018; Wei et al., 2018)), 2) improving the detection of predicates and the associated event type on the basis of WSD, 3) improve the detection and reasoning over locations, 4) establishing coreference relations and identity of the participants of the events.

Except for subtask 1, our system ranked 2nd in all tasks. This suggests that it is relatively stable and can be used to obtain detailed interpretations such as the victim counts for subtask 3. Also for the event coreference, our system ranks 2nd. Given the low performance on the document-incident clustering in subtask 1, where we have an F1 of 23.82, we can expect that this performance can be substantial higher if we use a state-of-the-art document-incident clustering technique. Currently, we used a very simple semantic comparison over the event properties and do not use most of the textual data in the documents. We also noticed that the NewsReader raw-ECKG output is noisy with respect to the event participants and the location detection. There is room for improvement to better associate frames to events, interpret locations in the documents and the victims and their names.

Acknowledgments

This research was funded by the Netherlands Organization for Scientific Research (NWO) via the Spinoza grant awarded to Piek Vossen in the project *Understanding Language by Machines*.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–41. Association for Computational Linguistics.

Collin Baker. 2008. Framenet, present and future. In *The First International Conference on Global Interoperability for Language Resources*, pages 12–17.

Xavier Carreras, Lluís Padró, Lei Zhang, Achim Rettinger, Zhixing Li, Esteban García-Cuesta, Željko Agić, Bozo Bekavac, Blaz Fortuna, and Tadej Štajner. 2014. Xlike project language analysis services. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 9–12.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Antske Fokkens, Marieke van Erp, Piek Vossen, Sara Tonelli, Willem Robert van Hage, Luciano Serafini, Rachele Sprugnoli, and Jesper Hoeksema. 2013. Gaf: A grounded annotation framework for events. In *Proceedings of the 1st workshop on Events: Definition, Detection, Coreference, and Representation, NAACL2013*, Atlanta, GA, USA.

Antske Fokkens, Aitor Soroa, Zuhaitz Beloki, German Rigau, Willem Robert van Hage, and Piek Vossen. 2014. NAF: the NLP Annotation Format. Technical report, Vrije Universiteit Amsterdam.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*, pages 1989–1993. Citeseer.

Zhiwei Li, Bin Wang, Mingjing Li, and Wei-Ying Ma. 2005. A probabilistic model for retrospective news event detection. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 106–113. ACM.

Tom Nicholls and Jonathan Bright. 2018. Understanding news story chains using information retrieval and network clustering techniques. *arXiv preprint arXiv:1801.07988*.

Marten Postma, Filip Ilievski, and Piek Vossen. 2018. Semeval-2018 task 5: Counting events and participants in the long tail. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*. Association for Computational Linguistics.

Willem Robert Van Hage, Véronique Malaisé, Roxane Segers, Laura Hollink, and Guus Schreiber. 2011. Design and use of the simple event model (sem). *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2):128–136.

Piek Vossen, Rodrigo Agerri, Itziar Aldabe, Agata Cybulska, Marieke van Erp, Antske Fokkens, Egoitz Laparra, Anne-Lyse Minard, Alessio Palmero Aprosio, and German Riga. 2016. Newsreader: using knowledge resources in a cross-lingual reading machine to generate more knowledge from massive streams of news. *Knowledge-Based Systems*.

Yifang Wei, Lisa Singh, David Buttler, and Brian Gallagher. 2018. Using semantic graphs to detect overlapping target events and story lines from newspaper articles. *International Journal of Data Science and Analytics*, 5(1):41–60.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 system demonstrations*, pages 78–83. Association for Computational Linguistics.