Event Detection with Minimal Supervision

Anonymous NAACL submission

Abstract

Supervised learning approaches for event trigger extraction currently train on examples that are manually annotated according to a predefined event type inventory. Annotating these examples is very labor intensive. In this paper, starting with just a set of keywords to define the event types of interest, we propose a minimally supervised approach for gathering large amounts of event trigger examples. While words are generally polysemous, we note that a set of words disambiguate one another, certain event types tend to co-occur frequently within the same document, and document headlines indicate the main focus of the article. We describe how to transform these intuitions into practical implementations for gathering event trigger examples. We demonstrate that this approach is promising. The gathered examples contain a minimal amount of noise, and a trigger model trained on the automatically gathered examples achieves performance that is close to training on manually annotated examples.

1 Introduction

We address the task of event trigger extraction from text. For instance, given the sentence "The terrorists **attacked** the cafe." We want to recognize that there is a physical *Attack* event, anchored or *triggered* by the word "attacked". Event trigger extraction[1] is an active area of research in natural language processing (NLP), and is important towards the overall task of event extraction, which also involves extracting event arguments.

Recent approaches in event trigger extraction usually employ corpus-based, supervised machine learning methods. In this approach, extraction systems first require a predefined inventory of event types. Systems also require the existence of a corpus in which each occurrence of a word has been manually annotated with the correct event type, according to the given event type inventory. This annotated corpus then serves as the training data for a machine learning algorithm. For instance, the popular Automatic Content Extraction (ACE) (Doddington et al., 2004) corpus contains 599 documents manually annotated with examples for 33 event types such as physical *Attack*, *Injury*, *Election*.

While supervised learning is used in state-of-the-art approaches for event trigger extraction, it has the drawback of requiring a large amount of manually tagged data, which is an extremely labor intensive process. This problem is severe for event trigger extraction, since data must be collected separately for each word in a language. Annotators must examine each word in a document and make 2 decisions: (i) is the word relevant to the given event type inventory? (ii) if it is relevant, then tag the word with the correct event type. To extend extraction capabilities to a new event type, one needs to manually annotate a large amount of training examples for that new event type. Re-annotation is also needed if one modifies the definitions of the existing event type inventory to support different application needs.

One approach for automatically gathering event trigger examples is to perform distant supervision (Mintz et al., 2009; Craven and Kumlien, 1999). Given a set of keywords for an event type (e.g. the words "attack", "bombing", etc. for the physical *Attack* event), extract all occurrences of the keywords from a background corpus as valid trigger examples of *Attack* to train a machine learning system. However, words are generally ambiguous or polysemous. The word "attack" has several meanings, depending on its occurrence context. It could mean a physical offensive (our event type of interest). It could also mean an intense criticism, an ap-

[1] Some researchers refer to this as *event detection* or *event trigger classification*.

proach to a problem, etc. Thus, an unconstrainted distant supervision approach would likely generate many spurious examples.

In this work, we propose to constrain the ambiguity problem by leveraging several intuitions. First, we note that while words are individually ambiguous (e.g. "attack"), certain sets of co-occurring words (e.g. "attack" and "bombing") disambiguate one another in meaning. Second, as noted by Liao and Grishman, (2011), some event types tend to co-occur frequently within the same document. For instance, a *Demonstrate* (protest) event frequently occurs with an *Injury* event within the same document. Our third intuition is that we can leveage the headline of a document. If our target event type (e.g. *Attack*) is reflected within the headline, then this improves the likelihood that keyword examples gathered from the document are valid trigger examples of our target event type.

In this paper, we implement the above intuitions and answer the following practical questions:

(i) Given a set of keywords for each event type, how do we implement the three intuitions to automate the process of gathering event trigger examples?

(ii) How accurate (or noisy) are the automatically gathered event trigger examples?

(iii) Using a state-of-the-art supervised event trigger extractor, what is its extraction performance when it is trained on our automatically gathered examples, compared with training on a manually tagged corpus (ACE)?

(iv) If we are allowed a minimal amount of human effort, how do we utilize it to potentially improve the accuracy of the trigger examples that we gather?

(v) How does the set of keywords that we are given affect performance?

In this paper, we address the above issues and report our findings. We evaluate on the benchmark ACE event corpus. We show that starting with just a set of keywords for each event type, we could automatically gather highly accurate event trigger examples from a background corpus. Our trigger extraction system that trains on these automatically gathered examples reaches performance that is competitive with training on the manually annotated ACE corpus. While our approach has only been tested on English, it is language agnostic and applicable to other languages. The event trigger examples we gathered will be released on github.com.

2 A Hybrid Approach for Minimally Supervised Event Extraction

To perform trigger extraction for a target event type, a machine learning system has to be provided guidance that defines the event type. This guidance could be in the form of a large set of annotated examples, as in the ACE corpus. An alternative light weight supervision is by providing a set of keywords that define the event type. As an example, if the event type of interest is physical *Attack*, then the learner could be provided keywords such as "attack", "war", "bombing", etc. In this section, we describe how we use such keywords to automatically gather event trigger examples of the target event type by leveraging the following 3 intuitions:

- Words disambiguate one another in meaning.
- Event types within the same document disambiguate one another.
- The headline of a document reflects the event focus of the document.

Specifically, given a set of keywords $\mathcal{K}_e$ for an event type *e*, our primary aim is to identify documents $\mathcal{D}_e$ that contain e. Then, occurrences of $\mathcal{K}_e$ in $\mathcal{D}_e$ will be valid trigger examples of *e*. In the rest of this section, we describe the three intuitions that constrain and filter for the relevant documents.

2.1 Intuition 1: Collective Disambiguation of Words

Most words are polysemous and take on different meanings depending on the context in which they appear. For instance, some of the meanings of the word "attack" are a physical offensive, an intense criticism, or an approach to a problem. Hence, randomly sampling individual occurrences of "attack" to serve as trigger examples for the *Attack* event would generate many spurious examples.

To tackle this, one might envision employing a word sense disambiguation (WSD) system. However, there are a few issues with this approach. First, sense disambiguation is a hard problem, and state-of-the-art WSD systems generally achieve accuracies between mid-60s to low-70s, depending on the particular corpus on which they are evaluated (Kågebäck and Salomonsson, 2016; Tag, 2015). Second, even if we have a highly accurate WSD system that disambiguates to an existing sense inventory (such as WordNet (Miller, 1990)), one would still need to identify the WordNet senses that are relevant to the target event type.

An alternative simpler approach is to note that words disambiguate one another. For instance, assume that we want to identify documents that are relevant for extracting *Attack* trigger examples, and we are provided with a few keywords such as “attack”, “war”, “bombing”, etc. Although the word “attack” has several meanings, if it appears in the same document as the word “bombing”, then it is likely that occurrences of “attack” in that document carry the physical offensive meaning and are thus valid trigger examples of the physical *Attack* event. We leverage this intuition by only considering documents that contain at least *n* different lemmas[2] of the target event type.

In our experiments, we set $n = 3$ for event types that have at least 3 distinct lemmas as keywords. For event types with fewer than 3 distinct lemmas as keywords, we lower n as necessary. For example, the event type *Divorce* has the keywords “divorce” and “divorced”. Since this translates to just the single lemma “divorce”, we set $n = 1$ for this event type.

2.2 Intuition 2: Collective Disambiguation of Events

In the preceding section, we propose that keywords of the same event type could disambiguate one another if they occur within the same document. We now extend this idea to keywords across different event types. As noted by Liao and Grishman, (2011), some event types tend to co-occur frequently within the same document. For instance, assume that *Demonstrate* and *Injury* events frequently co-occur within the same document (and as shown later, our experiments verify this). Thus, while *Demonstrate* could consists of sense ambiguous keywords such as “demonstrate”[3], if it appears in the same document as keywords belonging to the *Injury* event, then occurrences of “demonstrate” within that document are more likely to have the *protest* meaning, and thus be valid trigger examples of *Demonstrate*.

[2]A lemma is the base or canonical form of a word. As an example, the words “attack”, “attacks”, “attacking”, and “attacked” have the same lemma “attack”.

[3]Example meanings of “demonstrate” are: give an exhibition, provide evidence for, protest, etc.

Event Type	Top Associated Event Types
Attack	injure, execute, die, demonstrate, arrest-jail
Meet	phone-write, transfer-money, demonstrate, start-position, end-position
Transfer-Ownership	start-org, merge-org, declare-bankruptcy, end-org, fine

Table 1: Top associated event types using PMI scores.

To leverage this intuition, we first apply the inter-word approach described in Section 2.1 to identify the plausible event types present in each document. Then, we calculate the pointwise mutual information (PMI) between pairs of event types as a measure of their association. The PMI between a pair of event types *a* and *b* is defined as:

$$PMI(a, b) = log \frac{p(a, b)}{p(a)p(b)} \qquad (1)$$

To calculate $p(a, b)$, $p(a)$, and $p(b)$, we note the number of documents that a and b co-occur in, the number of documents that *a* occurs in, and the number of documents that *b* occurs in.

In our experiments, when we calculate PMI of *Demonstrate* with other event types over the English Gigaword (LDC2011T07) documents, the top ranking event types with the highest PMI are: *Injure*, *Arrest-Jail*, *End-Org*, *Elect*, and *End-Position*. This makes intuitive sense. When there is a rally or demonstration event, it frequently leads to injuries and arrests. And some of the frequent reasons for a demonstration are rallying for an upcoming election, or to express unhappiness over closing of companies or layoffs.

For each of the ACE event types, we have examined its top-5 ranking event types and note that the vast majority of them makes intuitive sense. Due to a lack of space, we show in Table 1 the top-5 ranking event types only for the 3 most frequent event types (*Attack, Meet, Transfer-Ownership*) in the ACE corpus: The fact that we obtained intuitive association between event types validates the accuracy of the inter-word disambiguation approach (Section 2.1) at identifying event types in each document.

In our work, when gathering trigger examples for an event type such as *Demonstrate*, we only consider documents that also contain one of its top-5 associated event types, according to PMI scores.

2.3 Intuition 3: The Title Says It All: Leveraging News Headlines

The headline of an article often indicates the main focus event of the article. For example, if we could determine that a headline such as "Militant group claims responsibility for bombings" is relevant for an *Attack* event, then occurrences of the word "attack" within the document are highly likely to be valid trigger examples of *Attack*, instead of representing other senses such as intense criticism, etc.

We implement this intuition using two approaches. First, we simply check whether the headline contains any of the keywords for our target event type. If it does, we declare the headline to be relevant for the event type.

However, some keywords are very ambiguous (e.g. the keyword "action" for *Attack*) and would be unhelpful towards checking for headline relevance. If we assume that we are allowed a minimal amount of human supervision, we could ask an annotator to select a subset of salient keywords (e.g. "bombing" for *Attack*) from the initial set of event keywords. Then, we check whether the headline contains any of these salient keywords.

2.4 Automatic Gathering of Event Trigger Examples

Given a set of keywords $\mathcal{K}_e$ for an event type e, we applied the three intuitions above in succession, to select a set of relevant documents $\mathcal{D}_e$:

- Inter-**W**ord disambiguation: Using the approach described in Section 2.1, we automatically identify the event types present in each Gigaword document, and select the documents that contain e.

- Inter-**E**vent disambiguation: As mentioned in Section 2.2, we use PMI scores to determine the top-5 event types $\{e_i^{'}\}$, for $i = 1 \ldots 5$, that are highly associated with e. We use this information to further prune the set of documents by only selecting documents that also contain any of these top-5 event types.

- Document **H**eadline filtering: As mentioned in Section 2.3, we further prune the set of documents by selecting documents whose headlines contain any keyword $k \in \mathcal{K}_e$.

The above process generates a large list of documents $\mathcal{D}_e$. We rank these in descending order by the number of keyword occurrences. The aim is to promote documents whose main topic is about event type e, thereby increasing the reliability of the event trigger examples.

3 Event Trigger Extraction Model

To perform event trigger extraction (classification), we developed a convolution neural network (CNN) model based on the work of Chen et al. (2015). The model primarily uses word embeddings as input features and achieves competitive state-of-the-art performance for event trigger extraction. In our work, we use the word embeddings[4] trained by Baroni et al. (2014), which achieved state-of-the-art results in a variety of NLP tasks. During decoding, the model labels words in sentences with their predicted event type (if any).

3.1 Model Validation Experiments

We first conduct experiments to verify the implementation of our CNN model. Following prior work (Chen et al., 2015; Nguyen and Grishman, 2015), we use the ACE-2005 corpus with the same sets of 529 training documents, 30 development documents, and 40 test documents. We similarly use the same criteria to judge the correctness of our event trigger extractions, as follows:

- A trigger is correctly classified if its event subtype and offsets match those of a reference trigger.

We use the following model features in our verification experiments:

- Word Embeddings (WE): We represent each word in the sentence with its word embeddings vector.

- Position Embeddings (PE): For every word w_i in the sentence, this captures its relative token distance to the candidate trigger word. These distance values are represented by embedding vectors which are automatically learnt during training.

[4] http://clic.cimec.unitn.it/composes/semantic-vectors.html. The embeddings provide representation for about 300K words. It was trained on the concatenation of ukWaC, the English Wikipedia, and the British National Corpus using the Skip-gram model with 5-word context window, 10 negative samples, and 400 dimensions.

- Entity Embeddings (EE): ACE provides annotations of entity mentions and their entity types such as *Person*, *Organization*, etc. Similar to (Nguyen and Grishman, 2015), we employ the BIO annotation scheme to assign entity type labels to each token in the sentence using the heads of the entity mentions. These labels are represented by embedding vectors which are automatically learnt during training.

We tune the model parameters (batch size, size and number of CNN filters, number of epoches, etc.) on the ACE development documents. In addition, since the ratio of positive vs negative examples is relatively skewed (for instance, most words in a sentence are not triggers), we also tried different weights for the positive examples: 1, 3, or 5. Finally, we follow (Chen et al., 2015) by using the Adadelta update rule with parameters $\rho = 0.95$ and $\epsilon = 1e^{-6}$, and a dropout rate of 0.5. We did not tune the learning rate and simply set it to 0.1.

On the ACE test data, our trigger model[5] achieves a balanced 0.66 for Precision (P), Recall (R), and F1 score. Chen et al. (2015) reports the same F1 score of 0.66 using a CNN model.[6] This shows that our event extraction model performs on par to state-of-the-art systems with similar architecture.

3.2 Model Features

Similar to prior work (Nguyen and Grishman, 2015), our model used gold standard entity type information available in ACE for its entity type embeddings. However, such gold standard entity type information is not readily available in practice, such as for the English Gigaword documents from which we automatically mine event trigger examples. Hence, we removed the entity type embeddings feature from the model and re-evaluated it on the ACE test data, obtaining an F1 score of 0.65 (just a drop of 0.01 from the original F1 score of 0.66). This is in line with the observation in (Nguyen and Grishman, 2015), where the authors noted that entity type embeddings provide only a modest improvement to event trigger extraction performance.

[5]Using hyper-parameters: 200 CNN filters of sizes 2, 3, and 4. PF and EE embeddings of size 50, 20 epochs, positive weight of 5, and batch size of 100.

[6]Chen et al. (2015) also reported a F1 score of 0.69 using a dynamic multi-pooling CNN (DMCNN).

For the rest of our experiments reported in this paper, we use the word embeddings (WE) and position embeddings (PE) as features in our model.

4 Experiments

In this section, we describe our experiments on automatically mining event trigger examples from the English Gigaword corpus.

4.1 Event Keywords

Our approach starts off with a set of keywords to serve as definitions of each event type, then automatically gather event trigger examples from a background corpus. To simulate asking human users to provide initial keywords, and to fairly compare our approach to approaches trained with ACE, we use the event keywords from the ACE training data as our event keywords. We ignore keywords that occur only once in the training data, resulting in an average of 6.3 keywords for each event type.

4.2 Event Types

The ACE event inventory defines a set of 33 event types, with an average of 134 examples per type. In this work, we focus on automatic mining of trigger examples for 31 of those event types, leaving out *Movement-Transport* and *End-Org*:

- *Movement-Transport*: the ACE guidelines state that either the *origin* or *destination* argument must be explicit somewhere in the document for this event type to be taggable. This requires incorporating event information that might exist beyond the event trigger sentence. We leave handling of global contexts to future work.

- *End-Org*: Due to the very small number of keywords for this event type, we did not identify any relevant Gigaword documents for *End-Org* when applying our intuitions described in Section 2.

4.3 Training on ACE Examples

We trained a trigger model using the ACE training data with word embeddings and position embeddings as features. Decoding for these 31 event types on the ACE test data, we obtained a micro-average F1 score of 0.66, as shown in Table 3 under the column "ACE F1". In that table, we also show the score breakdown per event type.

Dataset	#examples	Scores		
		R	P	F1
KW_{weh}	970	0.61	0.62	0.62
KW_{w}	1606	0.63	0.60	0.62
KW'_{weh}	928	0.58	0.65	0.61

Table 2: Statistics on the various training corpora, and summary of the test results when evaluating on the ACE test corpus. Here, R=recall, P=precision, F1=F1 score. Subscripts w, e, and h represent utilizing the intuitions of Sections 2.1, 2.2, and 2.3 respectively.

Event Type	ACE F1	KW_{weh} F1	KW'_{weh} Valid%
attack	0.73	0.74	0.87
meet	0.68	0.68	0.83
transfer-ownership	0.33	0.49	0.81
end-position	0.59	0.41	0.75
start-org	0.11	0.00	0.66
die	0.74	0.79	0.89
elect	0.86	0.49	0.84
transfer-money	0.32	0.23	0.78
start-position	0.44	0.00	0.65
sentence	0.80	0.80	0.98
marry	0.95	0.75	0.84
Less than 10 test examples			
divorce	0.84	0.89	0.89
phone-write	0.40	0.24	0.58
charge-indict	0.63	0.33	0.89
demonstrate	0.67	0.60	0.88
appeal	0.92	0.80	0.98
arrest-jail	0.77	0.91	0.94
convict	0.91	1.00	1.00
fine	0.67	0.71	0.95
trial-hearing	0.77	0.50	0.83
Less than 5 test examples			
sue	0.50	0.80	0.88
be-born	0.80	0.00	0.92
declare-bankruptcy	1.00	1.00	0.94
execute	0.00	1.00	0.99
acquit	0.00	0.00	1.00
extradite	0.00	0.00	0.96
release-parole	1.00	1.00	0.94
injure	1.00	1.00	0.97
nominate	0.00	0.00	0.90
Average	**0.66**	**0.62**	**0.88**

Table 3: Event trigger results for each event type, when evaluated on the ACE test corpus. We show the event types in descending order of their number of examples in the ACE test corpus. We show the per-event-type F1 scores when we train on the 529 ACE training documents, and when we train on the examples automatically gathered via our proposed KW_{weh} approach. We also show their respective (micro) average F1 scores. Finally, under the column "Valid%", we show the accuracy of our gathered trigger examples. When we allow minimal human effort to prune the keywords (Section 5.1), we obtained a macro average accuracy of 0.88. With no keyword pruning (KW_{weh}), we obtained a macro average accuracy of 0.84.

4.4 Training on Gigaword Examples

We now describe experiments using event trigger examples we automatically gathered from English Gigaword. For each of the 31 event types, we selected a maximum of 100 Gigaword documents to gather event trigger examples. On average, this gathered 970 trigger examples per event type.

To train an event trigger extraction system, one usually employs a single multi-class model and generates positive and negative examples as follows. For each word in a document, if it is a trigger for an event type, generate a positive example and use the event type as the class label. If the word is not a valid trigger for any event type, generate it as a negative trigger example. Hence, given an inventory of L event types, one generates trigger examples belonging to $L + 1$ labels. In this training paradigm, a 'negative' example means that it is not a valid trigger for any of the event types.

Our keyword based approach described in this section generates a set of documents $\mathcal{D}_e$ for each event type e. Word occurrences of keywords $\mathcal{K}_e$ within $\mathcal{D}_e$ are then gathered as trigger training examples of e. However, we cannot simply generate the remaining words as negative examples, because in general, they might be valid trigger examples for another event type $e^{'}$. To avoid generating false negative trigger training example, we leverage the inter-word disambiguation approach of Section 2.1. We note the set of event types the approach identifies in each Gigaword document. We do not generate keyword occurrences of these event types as negative examples from that document.

Using the above paradigm, we automatically generated positive (average of 970 per event type) and negative trigger examples from Gigaword documents. We trained our **K**ey**W**ord based system KW_{weh} on these Gigaword examples, tuned hyper-parameters on the ACE development data, and obtained Recall (R), Precision (P), and F1 score (F1) of 0.61, 0.62, and 0.62 when evaluated on the ACE test data. We show these results in the row KW_{weh} of Table 2. We show the score breakdown by event type under the column KW_{weh} of Table 3. This result is competitive with the F1 score of 0.66 obtained by training on the manually annotated ACE training data (Section 4.3).

5 Analysis

5.1 Pruning Event Keywords

We notice two potential issues with the event keywords from the ACE training data. First, a small set of event keywords such as "kill" are used across more than one event type (*Die*, *Execute*, *Attack*). Second, some keywords are overly general and ambiguous, for example the keywords "action" and "force" for *Attack*. Thus, we tried an experiment where we assume a minimal amount of human effort to take two actions: (i) for keywords that are used across more than one event type, allow our annotator to prune them from some of those event types, (ii) allow our annotator to select a subset of salient keywords from the original keywords. This results in an average of 3.6 salient keywords for each event type (down from the original average of 6.3 keywords). We then use these salient keywords to apply the headline filtering approach of Section 2.3.

Incorporating the above revisions generated a slightly smaller average of 927 trigger examples per event type. Training on these and evaluating on ACE gives us an F1 score of 0.61, as shown in row KW'_{weh} of Table 2.

5.2 Accuracy of Automatically Gathered Examples

To estimate the accuracy of our automatically gathered event trigger examples, we randomly sampled a maximum of 100 examples for each event type, using the above KW'_{weh} approach. We gave 2 annotators the ACE annotation guidelines and asked them to judge whether the examples are valid for their respective event types. We then calculated the proportion of valid examples (averaged between the 2 annotators) and show these results under the column "Valid %" of Table 3. Calculating a macro average across the event types gave us an overall accuracy of 0.88.

We also calculated the judgement agreement between our annotators using 2 metrics. Over the event types in Table 3, we calculated a Cohen's Kappa of 0.55, indicating moderate agreement. However, we note that in the almost ideal case where an annotator judge all gathered examples as valid while another annotator judge *almost* all examples as valid, Cohen's Kappa score will be 0. This is counter intuitive as having one or both annotators judge all gathered examples as valid indicates the high precision of the examples. Hence, we also tabulated the simple measure of proportion of agreement between the two annotators, which came out to 92%.

We note that some event types (e.g. start-position, start-org, phone-write) suffer in a lower accuracy in their generated examples. For both start-position and start-org, low-precision keywords included incipient language like "started" and "became". These phrases are not unique to new businesses and personnel, but often occur in temporal modifications to other event descriptions. The start-position keyword "head" was considered incorrect when it indicated an ongoing state rather than a change in personnel, and findings for the start-org event type also included a number of references to start-position events. It's not unreasonable to assume that articles talking about the founding of a company might also talk about its new employees; so the three intuitions might still hold for finding the correct event type articles.

Phone-write's low-precision keywords included variations on "write", "call", and "phone". Writings that are not part of a correspondence are often included as primary sources in news articles, resulting in quotations cited using language like "[...] she writes." Meanwhile, "call" has a wide range of senses, from the correct "phone call" to exhortations ("call for resignation") and naming ("called John Doe"). References to phones on occasion fail to indicate even an implicit act of communication–for example, legal proceedings involving phone companies or descriptions of cellular technology.

5.3 Model Variants

We experimented with a few other model variations. First, instead of training a single multi-class trigger classifier, we tried training individual binary models, one for each event type. This training paradigm is natural, given that our approach individually generates relevant documents for each event type. However, our experiments show that this approach performs slightly worse than the single multi-class model, lagging by about 2 F1 points.

Next, we ablated the inter-event disambiguation and headline filtering processes, depending on just the inter-words disambiguation approach of Section 2.1 to automatically generate trigger examples from Gigaword. Training a multi-class model on these examples and evaluating on ACE gives an F1 score of 0.62, as shown in the row KW_w of Table 2. This is somewhat surprising, but does demonstrate the robustness of the simple inter-words disambiguation strategy. However, we note in Table 2 that this gives us a higher average of 1606 examples per event type as shown in the row KW_w, versus the 970 examples generated by the KW_{weh} approach.

6 Related Work

Most prior work in event extraction have been on proposing features (Huang and Riloff, 2012; Ji and Grishman, 2008) or exploring model archiectures to improve extraction performance. Recently, researchers have started applying neural network models for event extraction. Chen et al. (2015) applied CNNs for event extraction, while Nguyen (2016a) proposed using recurrent neural networks.

In comparison, prior work that aims to alleviate the human effort required to train extractors for new event types have been relatively sparse. Nguyen et al. (2016b) proposed a two-stage model for event type extension. Given a new event type with a small set of seed instances, their model leverages training data from other event types to improve event type extension. In (Chen et al., 2017), the authors leverage the event records within Freebase to mine event examples from Wikipedia. They further employed FrameNet to filter noisy triggers, and reported an accuracy of 0.89 for their generated event trigger examples. However, their approach depends on employing the manually constructed resources Freebase and FrameNet, which limits applicability to event types found within Freebase. In another work (Peng et al., 2016), the authors employ the sentence examples in the ACE annotation guidelines to support minimally supervised event trigger extraction. However, their approach depends on having access to a semantic role labeling (SRL) system and a small set of sentences annotated with event triggers and event arguments. In contrast, our approach in this paper is very lightweight, requiring only event keywords.

7 Conclusion

We present an approach for automatically gathering large amounts of event trigger examples, when given only a set of keywords to define the event types of interest. We demonstrate that the examples gathered are of high quality. They contain minimal amounts of noise, and a trigger model trained on them achieves performance close to training on manually annotated examples.

References

2011. Using document level cross-event inference to improve event extraction. In *ACL-2011*, pages 789–797.

2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *NAACL-2015*, pages 314–323.

Marco Baroni, Georgiana Dinu, and German Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL-2014*, pages 238–247, Baltimore, USA.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *ACL-2017*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL-IJCNLP2-2015*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ICISMB*, pages 77–86. AAAI Press.

George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie M. Strassel, and Ralph M. Weischedel. 2004. The automatic content extraction (ace) program - tasks, data, and evaluation. In *LREC*. European Language Resources Association.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *AAAI-CAI*, AAAI'12, pages 1664–1670. AAAI Press.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL-HLT-2008*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional lstm. In *5th Workshop on Cognitive Aspects of the Lexicon (CogALex)*. Association for Computational Linguistics.

George Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACLIJCNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016a. Joint event extraction via recurrent neural networks. In *NAACL-HLT-2016*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Thien Huu Nguyen, Lisheng Fu, Kyunghyun Cho, and Ralph Grishman. 2016b. A two-stage approach for extending event detection to new types via neural networks. In *WRepL4NLP*, pages 158–165, Berlin, Germany. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolution neural networks. In *ACL-2015*, pages 365–371.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *EMNLP*.