Semi-Supervised Event Extraction with Paraphrase Clusters

James Ferguson, Colin Lockard, Daniel S. Weld, and Hannaneh Hajishirzi
University of Washington
{jfferg, lockardc, weld, hannaneh}@cs.washington.edu

Abstract

Supervised event extraction systems are limited in their accuracy due to the lack of available training data. We present a method for self-training event extraction systems by bootstrapping additional training data. This is done by taking advantage of the occurrence of multiple mentions of the same event instances across newswire articles from multiple sources. If our system can make a high-confidence extraction of some mentions in such a cluster, it can then acquire diverse training examples by adding the other mentions as well. Our experiments show significant performance improvements on multiple event extractors over ACE 2005 and TAC-KBP 2015 datasets.

1 Introduction

Event extraction is a challenging task, which aims to discover event triggers in a sentence and classify them by type. Training an event extraction system requires a large dataset of annotated event triggers and their types in a sentence. Unfortunately, because of the large amount of different event types, each with its own set of annotation rules, such manual annotation is both time-consuming and expensive. As a result, popular event datasets, such as ACE (Walker et al., 2006) and TAC-KBP (Mitamura et al., 2015), are small (*e.g.*, the median number of positive examples per subtype is only 65 and 86, respectively) and biased towards frequent event types, such as Attack.

When an event occurs, there are often multiple parallel descriptions of that event (Figure 1) available somewhere on the Web due to the large number of different news sources. Some descriptions are simple, explaining in basic language the event that occurred. These are often easier for existing extraction systems to identify. Meanwhile, other descriptions might use more complex language that falls outside the scope of typical event extractors, but which, if identified, could serve as valuable training data for said systems.

1) **LSU** *fires* head coach **Les Miles** after 12 seasons.
2) **Les Miles** is *out* at **LSU** after 12 seasons in Baton Rouge.
3) On Sunday morning, **LSU** athletic director Joe Alleva told **Les Miles** that the coach would *no longer represent* Louisiana State.

Figure 1: Example of a cluster of paraphrases. Shared entities are bolded, and the triggers are italicized. Some, such as the first sentence, are very simple. Others, like the third sentence are more difficult.

We automatically generate labeled training data for event trigger identification leveraging this wealth of event descriptions[1]. Specifically, we first group together paraphrases of event mentions. We then use the simple examples in each cluster to assign a label to the entire cluster. This simplifies the task of extracting events from difficult examples; rather than having to identify whether an event occurs, and which word serves as a trigger for that event, our system needs only to identify the most likely trigger for the given event. Finally, we combine the new examples with the original training set and retrain the event extractor.

Our experiments show that this data can be used with limited amounts of gold data to achieve significant improvement over both standard and neural event extraction systems. In particular, it achieves 1.1 and 1.3 point F1 improvements over a state-of-the-art system in trigger identification on TAC-KBP and ACE data respectively. Moreover, we show how the benefit of our method varies as a function of the amount of fully-supervised training data and the number of additional heuristically-labeled examples.

[1]The generated data and our code can be found at https://github.com/jferguson144/NewsCluster

Proceedings of NAACL-HLT 2018, pages 359–364
New Orleans, Louisiana, June 1 - 6, 2018.

2 Approach

Our goal is to automatically add high quality labeled examples, which can then be used as additional training data to improve the performance of any event extraction model. Our data generation process has three steps. The first is to identify clusters of news articles all describing the same event. The second step is to run a baseline system over the sentences in these clusters to identify events found in each cluster. Finally, once we have identified an event in one article in a cluster, our system scans through the other articles in that cluster choosing the most likely trigger in each article for the given event type.

Cluster Articles In order to identify groups of articles describing the same event instance, we use an approach inspired by the NewsSpike idea introduced in Zhang et al. (2015). The main intuition is that rare entities that are mentioned a lot on a single date are more indicative that two articles are covering the same event. We assign a score, S, to each pair of articles, (a_i, a_j) appearing on the same day, for whether or not they cover the same event, as follows:

$$S(a_i, a_j) = \sum_{e \in E_{a_i} \cap E_{a_j}} \frac{\text{count}(e, \text{date}_{a_i,a_j})}{\text{count}(e, \text{corpus})}, \quad (1)$$

where E_a is the list of named entities for the article a, and count is the number of times the entity appears on the given date, or in the whole corpus. This follows from the intuition above by reducing the weight given to common entities. For example, *United States* appears 367k times in the corpus, so it is not uncommon for it to appear hundreds of times on a single day, and articles mentioning it could be covering completely different topics. Meanwhile *Les Miles* appears only 1.6k times in the corpus, so when there are hundreds of mentions involving *Les Miles* on a single day, it is much more likely that he participated in some event. Accumulating these counts over all shared entities between two articles thus indicates whether the articles are covering the same event. We then group all articles that cover the same event according to this score into clusters.

Label Clusters Then, given clusters of articles, we run a baseline extractor which was trained on what limited amount of fully-supervised training data is available. The hope is that one or more of a cluster's sentences will use language similar enough to our training data that the extractor can make an accurate prediction. Our system keeps any cluster in which the baseline system identifies at least some threshold, θ_{event}, of event mentions for a single event type, and labels those clusters with the identified type.

Assign Triggers After labeling, the event clusters are comprised of articles in which at least one sentence should contain event mentions of the labeled type. Because most current event extraction systems require labeled event triggers for sentences, we identify those sentences and the event triggers therein so that we can run the baseline systems. For each sentence we identify the most likely trigger by checking the similarity of the word embeddings to the canonical vector for that event. This vector is computed as the average of the embeddings of the event triggers, v_t, in the gold training data: $v_{event} = \frac{1}{|T_{event}|} \sum_{t \in T_{event}} v_t$, where T_{event} is the set of triggers for this event in the gold training data. If the maximum similarity is greater than some threshold, θ_{sim}, the sentence and the corresponding trigger are added to the training data.

Event Trigger Identification Systems Event extraction tasks such as ACE and TAC-KBP have frequently been approached with supervised machine learning systems based on hand-crafted features, such as the system adapted from Li et al. (2013) which we make use of here. Recently, state-of-the-art results have been obtained with neural-network-based systems (Nguyen et al., 2016; Chen et al., 2015; Feng et al., 2016). Here, we make use of two systems whose implementations are publicly available and show that adding additional data would improve their performance.

The first system is the joint recurrent neural net (JRNN) introduced by Nguyen et al. (2016). This model uses a bi-directional GRU layer to encode the input sentence. It then concatenates that with the vectors of words in a window around the current word, and passes the concatenated vectors into a feed-forward network to predict trigger types for each token. Because we are only classifying triggers, and not arguments, we don't include the memory vectors/matrices, which primarily help improve argument prediction, or the argument role prediction steps of that model.

The second is a conditional random field (CRF) model with the trigger features introduced by Li et al. (2013). These include lexical features, such

as tokens, part-of-speech tags, and lemmas, syntactic features, such as dependency types and arcs associated with each token, and entity features, including unigrams/bigrams normalized by entity types, and the nearest entity in the sentence. In particular, we use the Evento system from Ferguson et al. (2017).

3 Experimental Setup

Labeled Datasets We make use of two labeled datasets: ACE-2005 and TAC-KBP 2015. For the ACE data, we use the same train/development/test split as has been previously used in (Li et al., 2013), consisting of 529 training documents, 30 development documents, and a test set consisting of 40 newswire articles containing 672 sentences. For the TAC-KBP 2015 dataset, we use the official train/test split as previously used in Peng et al. (2016) consisting of 158 training documents and 202 test documents. ACE contains 33 event types, and TAC-KBP contains 38 event types.

For our approach, we use a collection of news articles scraped from the web. These articles were scraped following the approach described in Zhang and Weld (2013). The process involves collecting article titles from RSS news seeds, and then querying the Bing news search with these titles to collect additional articles. This process was repeated on a daily basis between January 2013 and February 2015, resulting in approximately 70 million sentences from 8 million articles. Although the seed titles were collected during that two year period, the search results include articles from prior years with similar titles, so the articles range from 1970 to 2015.

Evaluation We report the micro-averaged F1 scores over all events. A trigger is considered correctly labeled if both its offsets and event type match those of a reference trigger.

Implementation details For creating the automatically-generated data, we set thresholds θ_{event} and θ_{sim} to 2 and 0.4 respectively, which were selected according to validation data. We use CoreNLP (Manning et al., 2014) for named entity recognition, and we use a pre-trained Word2Vec model (Mikolov et al., 2013) for the vector representations.

For the JRNN model, we follow the parameter settings of (Nguyen et al., 2016) and use a context window of 2 for context words, and a feed-forward neural network with one hidden layer for trigger prediction with hidden layer size of 300. Finally, for training, We apply the stochastic gradient descent algorithm with mini-batches of size 50 and the AdaDelta update rule (Zeiler, 2012) with L_2 regularization. For the CRF model, we maximize the conditional log likelihood of the training data with a loss function via softmax-margin (Gimpel and Smith, 2010). We optimize using AdaGrad (Duchi et al., 2011) with L_2 regularization.

		ACE			TAC-KBP		
		P	R	F1	P	R	F1
CRF	0%	62.9	70.0	66.3	53.5	52.3	52.9
	10%	64.5	69.8	67.0	**59.9**	49.3	**54.1***
	20%	**65.1**	**70.2**	**67.6***	59.3	49.2	53.8
	30%	**65.1**	69.9	67.4	58.1	**49.4**	53.4
JRNN	0%	65.7	72.9	69.1	**68.8**	49.2	57.3
	10%	67.4	72.7	69.9	65.4	52.1	58.0
	20%	**67.6**	**73.5**	**70.4***	65.3	52.8	**58.4***
	30%	67.5	73.3	70.3	64.7	**52.9**	58.2
HNN		84.6	64.9	73.4	-	-	-
SSED		-	-	-	69.9	48.8	57.5

Table 1: Results after adding varying amounts of automatically-generated news data. Percentages indicate the amount of additional data relative to the size of the gold training data. Using a modest amount of semi-supervised data improves extractor performance on both ACE & TAC-KBP events. * indicates that the difference in F1 relative to training with just the gold data is statistically significant ($p < 0.05$).

4 Experiments

Varying Amounts of Additional Data In this section we show that the addition of automatically-generated training examples improves the performance of both systems we tested it on. We sample examples from the automatically-generated data, limiting the total number of positive examples to a specific number. In order to avoid biasing the system in favor of a specific event type, we ensure that the additional data has a uniform distribution of event types. We run 10 trials at each point, and report average results.

Table 1 reports the results of adding varying amounts of our generated data to both CRF and JRNN systems. We observe that that adding any amount of heuristically-generated data improves performance. Optimal performance, however, is achieved fairly early in both datasets. This is likely due to the domain mismatch between the gold and additional data. For reference purposes, we also include the result of using the HNN model from (Feng et al., 2016) and the SSED system from

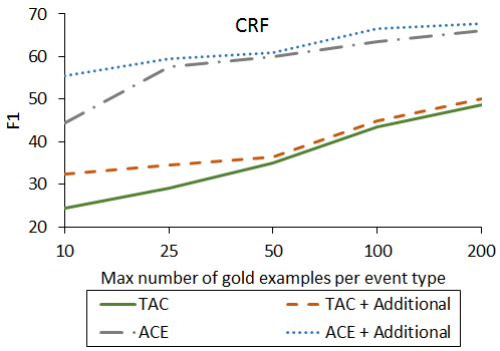

Figure 2: Adding a reasonable amount (200 examples per event) of semi-supervised data on top of limited amounts of gold training data improves performance across the board, but the gain is dramatic when the number of supervised examples is extremely small.

(Sammons et al., 2015), which are the best reported results on the ACE-2005 and TAC-KBP 2015 corpora respectively. These systems could also benefit from our additional data since our approach is system independent.

Varying Amounts of Supervised Data In this section we evaluate how the benefit of adding semi-supervised data varies given different amounts of gold (supervised) data to start. We conjecture that semi-supervision will be more beneficial when gold data is very limited, but the conclusion isn't obvious, since semi-supervision is more likely to add noisy examples in this case. Specifically, we limit the number of positive gold examples for each event by randomly sampling the overall set. We then add in the same amount of automatically-generated data to each trial. We again run 10 trials for each size, and report the average.

The results for this experiment using the CRF model can be seen in figure 2: training with large amounts of semi-supervised data improves performance considerably when limited gold training data is available, but those gains diminish with more high-quality supervised data. We observe the same trend for the JRNN system as well.

Discussion We randomly selected 100 examples from the automatically-generated data and manually annotated them. For each example that did not contain a correctly labeled event mention, we further annotated where in the pipeline an error occurred to cause the incorrect labeling. This breakdown can be seen in table 2. As observed in the table, the errors are mainly due to the incorrect event identification or trigger assignment.

Correct		72
Incorrect	clustering	5
	event identification	13
	trigger assignment	10

Table 2: The results of manually labeling 100 examples that were automatically-generated using JRNN as the supervised system.

Incorrect clustering refers to cases in which a sentence does not cover the same topic as other sentences in its cluster. This was primarily caused by entities participating in multiple events around the same time period. For example, this occurred in sentences from the 2012 US presidential election coverage involving Barack Obama and Mitt Romney.

Incorrect event identification refers to clusters that were incorrectly labeled by the supervised system. The primary reason for these errors is due to domain mismatch between the news articles and the gold data. For example, our system identifies the token *shot* in *Bubba Watson shot a 67 on Friday* as an attack event trigger. Because the gold data does not contain examples involving sports, the baseline system mistakenly identifies a paraphrase of the above sentence as an attack event, and our system is not able to fix that mistake. However, this problem can be solved by training the baseline extractor on the same domain as the additional data.

Incorrect trigger assignment refers to errors in which a sentence is correctly identified as containing an event mention, but the wrong token is selected as a trigger. The most common source of this error is tokens that are strongly associated with multiple events. For example, *shooting* is strongly associated with both attack and die events, but only actually indicates an attack event.

Looking through the correct examples, the data collection process is able to identify uncommon triggers that do not show up in the baseline training data. For example, it correctly identifies "offload" as a trigger for Transfer-Ownership in *Barclays is to offload part of its Spanish business to Caixabank*. Despite the trigger identification step having no context awareness, the process is also able to correctly identify triggers that rely on context, such as "contributions" triggering Transfer-Money in *Chatwal made $188,000 of illegal campaign contributions to three U.S. candidates via straw donors*.

5 Related Work

A challenge in event extraction is the relatively small number of labeled training examples available. Researchers have dealt with this by framing event extraction in a way that allows them to rely heavily on systems built for dependency parsing (McClosky et al., 2011) and semantic role labeling (Peng et al., 2016). Unlike these researchers, we join a line of work that attempts to directly harvest additional training examples for use in traditional event extraction systems.

Distant supervision is one source of additional data that has been successfully applied to relation extraction tasks (Riedel et al., 2010; Hoffmann et al., 2011; Mintz et al., 2009), which align a background knowledge base to an accompanying corpus of natural language documents. For event extraction, such data sources are not as easily available since there are no pre-existing stores of tuples of attacks, movements or meetings.

Other work has generated additional data by using a pattern-based model of event mentions and bootstrapping on top of a small set of seed examples. Huang and Riloff (2012) begin with a set of nouns that are specific to certain event roles and extract patterns based on the contexts in which those words appear. Li et al. (2014) extracted additional patterns using various event inference mechanisms.

The work most similar to ours is that of Liao and Grishman (2010, 2011) to identify articles from a corpus which described the same event instances found in training examples. These articles are then used in self-training an ACE-trained system after being filtered to select passages with consistent roles and triggers. Their method provides a 2.7 point boost to F1, but their baseline system results are much lower than ours (54.1 vs 69.1) and it is unclear what improvement their method would have on a state-of-the-art extractor. In addition, their system attempts to identify relevant articles that describe event instances already present in their training data, while we attempt to find clusters of sentences describing a common event, at least one of which we can confidently label.

The use of parallel news streams to acquire event extraction training data in an unsupervised fashion was explored in (Zhang et al., 2015), whose clustering methods we have adapted here. Unlike Zhang *et al.*, we have a defined event ontology for which we are acquiring data, rather than attempting to learn event types from the data. Furthermore, we use an extractor trained on fully-supervised examples to filter clusters, in contrast to Zhang *et al.*, whose method is completely unsupervised, which allows us to relax some of the assumptions made by Zhang *et al.* and consider "spikes" of individual entities as opposed to pairs.

6 Conclusion

We present a method for self-training event extraction systems by taking advantage of parallel mentions of the same event instance in newswire text. By examining clusters of sentences which produce at least two extractions of the same event type and assigning a trigger label to each sentence via word embedding similarity, we add diverse training examples to our dataset. Our experiments show a 1.3 point F1 increase in trigger labeling for a state-of-the-art baseline system on ACE, and a 1.1 point increase on TAC-KBP. For future research, this work can be applied to arbitrary event extraction models to improve performance, or make up for a lack of training data. The code and data are publicly available at our github repository.

Acknowledgements

This research was supported in part by ONR grants N00014-11-1-0294 and N00014-15-1-2774, DARPA contract FA8750-13-2- 0019, NSF (IIS-1616112, IIS-1420667, IIS-1703166), ARO grant W911NF13-1-0246 and the WRF/T.J. Cable Professorship. We would like to thank Stephen Soderland for many helpful discussions and our anonymous reviewers for their comments.

References

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. In *Journal of Machine Learning Research*.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *ACL*.

James Ferguson, Colin Lockard, Natalie Hawkins, Stephen Soderland, Hannaneh Hajishirzi, and Daniel S. Weld. 2017. University of washington tac-kbp 2016 system description. In *TAC-KBP*.

Kevin Gimpel and Noah A. Smith. 2010. Softmaxmargin crfs: Training log-linear models with cost functions. In *HLT-NAACL*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*.

Ruihong Huang and Ellen Riloff. 2012. Bootstrapped training of event extraction classifiers. In *EACL*.

Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2014. Employing event inference to improve semi-supervised chinese event extraction. In *COLING*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Shasha Liao and Ralph Grishman. 2010. Filtered ranking for bootstrapping in event extraction. In *COLING*.

Shasha Liao and Ralph Grishman. 2011. Can document selection help semi-supervised learning? a case study on event extraction. In *ACL*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event extraction as dependency parsing. In *ACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL/IJCNLP*.

Teruko Mitamura, Yukari Yamakawa, Susan Holm, Zhiyi Song, Ann Bies, Seth Kulick, and Stephanie Strassel. 2015. Event nugget annotation: Processes and issues.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *HLT-NAACL*.

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

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML/PKDD*.

Mark Sammons, Haoruo Peng, Yangqiu Song, Shyam Upadhyay, Chen-Tse Tsai, Pavankumar Reddy, Subhro Roy, and Dan Roth. 2015. Illinois ccg tac 2015 event nugget, entity discovery and linking, and slot filler validation systems. In *TAC*.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. *ACE 2005 Multilingual Training Corpus LDC2006T06*. Linguistic Data Consortium, Philadelphia.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Congle Zhang, Stephen Soderland, and Daniel S. Weld. 2015. Exploiting parallel news streams for unsupervised event extraction. *TACL*, 3:117–129.

Congle Zhang and Daniel S. Weld. 2013. Harvesting parallel news streams to generate paraphrases of event relations. In *EMNLP*.