On Zero-Shot Transfer Learning for Event Extraction

by

Shaheer Haroon

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science
Computer Science
New York University

May, 2019

Abstract

Event extraction normally requires large amounts of annotated data for each event type. Each event consist of trigger words and arguments that fulfill certain roles. This limits the ability to add new event types to an existing ontology or when building a new one because of the massive effort involved for manually annotating a corpus. Recent methods have proposed using zero-shot transfer learning to minimize the amount of annotated data required for a classifier to predict new event types. The zero-shot classifier relies on several components, including a preexisting event ontology to be successful. Our goal was to explore factors including choice of role names, event type names, and definitions of event mention and event type structures that could influence the results of a zero-shot classifier. We found that the use of paradigmatic role names and characteristic event type names in an event ontology especially have significant impact on the success of the classifier. As a result, there is still a decent amount of effort required when adding new event types to an ontology in order to promote the success of a zero-shot approach.

Contents

Abstract 3

Acknowledgments 4

0 Introduction 6

1 Background Information 10

2 Approach Details 14

3 Results & Discussion 21

4 Conclusion 31

References 34

Acknowledgments

I appreciate all the support from all my family and friends. I am also grateful to my second reader, Kyunghyun Cho. I would especially like to thank my advisor, Ralph Grishman for being incredibly understanding and patient with me throughout this entire process.

This work was supported in part by DARPA/I2O and U.S. Army Research Office Contract No. W911NF-18-C-0003 under the World Modelers program. The views, opinions, and/or findings contained in this thesis are those of the author and his advisor and should not be interpreted as representing the official views or policies, either expressed or implied, of the Department of Defense or the U.S. Government. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

0 Introduction

Event extraction is a subset of a wider field in natural language processing, known as information extraction. The focus of event extraction is to find and categorize certain events within sentences. The process of identifying if an event occurred within a sentence is to find indicative words known as triggers that signal some type of event happening, often accompanied by several argument words that fulfill certain roles. For example consider the following sentence: ”In December, a shipment of North Korean missiles travelling to Yemen was briefly stopped in the Arabian Sea.” There is a Transport event triggered by the word ”travelling”. ”Yemen” plays the role of the ”destination” and ”missiles” fulfills the role of the ”artifact”. The diagram below provides a visualization of the event mention and event type structures for this particular event.

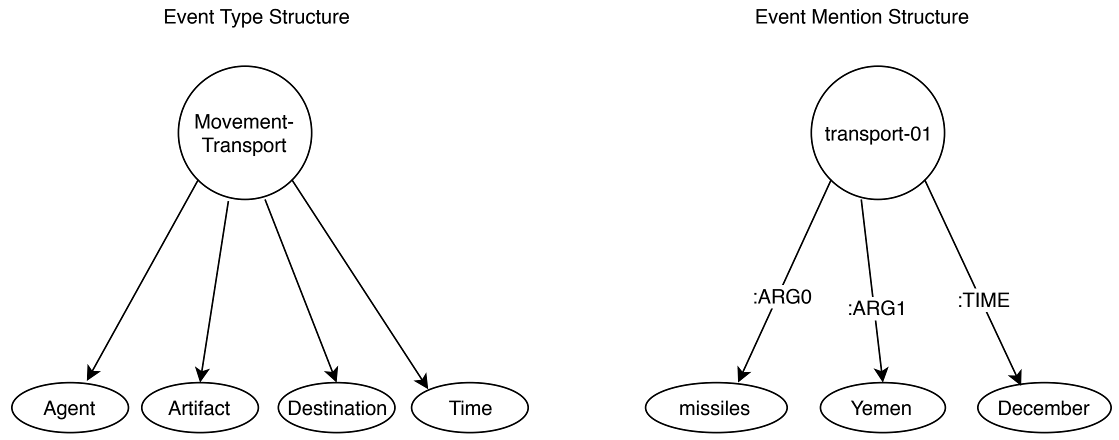

Figure 1: Example of Event Mention and Event Type Structures

Historically, event extraction involves having a pre-existing ontology of event types that are linked to candidate triggers in a sentence.[6] Supervised learning with features specific to event types in the ontology were used to classify triggers and their arguments as signalling an event type. One of the challenges with this approach is it requires mass annotation effort for each event type in order for a classifier to become proficient in identifying even a single event type. In order to make event extraction effective in real world scenarios, Huang et al. suggested applying zero-shot transfer learning to the problem.[5] The goal of this approach is to build a shared semantic space of event types in a way that allows new test samples to be classified based on its proximity to existing event types in the space.[37] Effectively, this means zero new training samples are needed to classify an unseen event type, hence the name zero-shot classifier. In order to build the shared semantic space, a structured representation of each possible event type is required, which includes a name for the event type and labels for possible roles related to the event. An example would be an Attack event with roles: Attacker, Target, Instrument, Place, Time. The proposal by Huang et al. is to build a representation of the event mention structure that captures a candidate trigger and its arguments and compare its similarity to existing event type structures in the shared semantic space to find the best fit.

As there are many steps and complex features used to build the zero-shot classifier, we attempted to experiment with certain parts of the pipeline process to better understand the value each step contributed to the effectiveness of the overall approach and to try simpler alternatives at certain steps to improve transparency of effective techniques. One of our research goals was to investigate factors of an ontology that can influence the results of building a shared semantic space to do zero-shot classification of event types. We reasoned that the choice of role labels in an ontology and the word choice for event type names can have a significant influence on the success of using a shared semantic space to classify unseen events. For example, we might suspect the classifier to do reasonably well on Arrest-Jail events because the word ”arrest” is a common trigger for many events of that type. On the other hand, we suspect the classifier to have a harder time doing well on event types such as Start-Org. We also wanted to experiment with a simpler and more transparent MaxEnt model as opposed to the neural network architecture proposed in the paper to see the effectiveness of individual features in being able to identify good candidate triggers. Since we are not using a CNN to generate a dense vector representation for event structures, we wanted to see the effectiveness of using simpler alternatives compared to the relatively complex event mention and event type structures when performing the classification on unseen types.

To test our hypothesis, we restricted ourselves to the Automatic Content Extraction (ACE) 2005 event ontology consisting of 33 event types. Given a corpus, we start by producing an Abstract Meaning Representation (AMR) parse of each sentence. Using the AMR structure, we identify candidate triggers by linking word senses to words found in OntoNotes and FrameNet. Next, we train our MaxEnt model with negative training examples to learn to identify good candidate triggers. Finally, we use our zero-shot classifier to test our hypotheses. A detailed breakdown of each step is outlined in the following sections.

1
Background Information

1.0.1 Automatic Content Extraction (ACE)

The ACE program was founded with the intent of developing technologies that would further the field of automatic information extraction. There are three main research objectives, detecting and classifying entities, relations, and events. Annotators provided tagged broadcast transcripts, news wire, and newspaper text in several languages with the appropriate entities, relations, and events. We leveraged the annotations to generate training, development, and test sets for experiments. We limited ourselves to only identifying English events as that is the current scope of our research.

ACE also provided detailed annotation guidelines for event extraction, where in-depth descriptions for 33 event types are outlined. The guidelines define an event to be a specific occurrence involving participants and something that can be described as a change of state. Although 33 events is nowhere near sufficient to capture all possible types of events, it does provide an acceptable starting point for testing event extraction strategies. Each of the 33 event types are sub types of eight broader event types: Life, Movement, Transaction, Business, Conflict, Contact, Personnel, and Justice. Each of the sub types provide labels for roles related to that sub type. For example, the Arrest-Jail event has the following roles: person, agent, crime, time, and place. In our experiments, we investigate the importance of choosing good role labels and names for event types.

1.0.2 OntoNotes

OntoNotes provides a corpus annotated with rich structural and semantic information.[4] For the purposes of our experiments, we utilized OntoNotes to match candidate triggers to verb and noun senses found in the corpus as a filter. This helped to reduce the sheer number of candidate triggers by only considering ones that we have good annotations for.

1.0.3 FrameNet

FrameNet is a project that aims to capture semantic frames.[1] A semantic frame can be thought of as a structure including a trigger word and participants for an event. FrameNet includes hundreds of semantic frames and provides lexical units that tend to be triggers for these semantic frames.[8] We utilize this list of lexical units in our process of identifying candidate triggers. Therefore, we consider all noun and verb candidate triggers that can be mapped to a sense in OntoNotes or to a lexical unit in FrameNet.

1.0.4 Abstract Meaning Representation (AMR)

Abstract meaning representation is a way of representing a sentence in a tree like way, where the nodes represent concepts that align with words in a sentence and the edges represent relationships between nodes.[29] This representation is quite useful to the task of event extraction as the AMR relations tend to capture the role of candidate trigger arguments. A sample AMR parse tree is given below. Common types of relations include ARG0, ARG1, ARG2, ARG3, time, and location. Additionally, the AMR parse of a sentence provides us with the sense of verbs and nouns used in a sentence. The word senses can be used to gain more information about the meaning of relations (i.e. ARG0, ARG1, etc.) using OntoNotes and be used as a starting point for the candidate trigger search process.

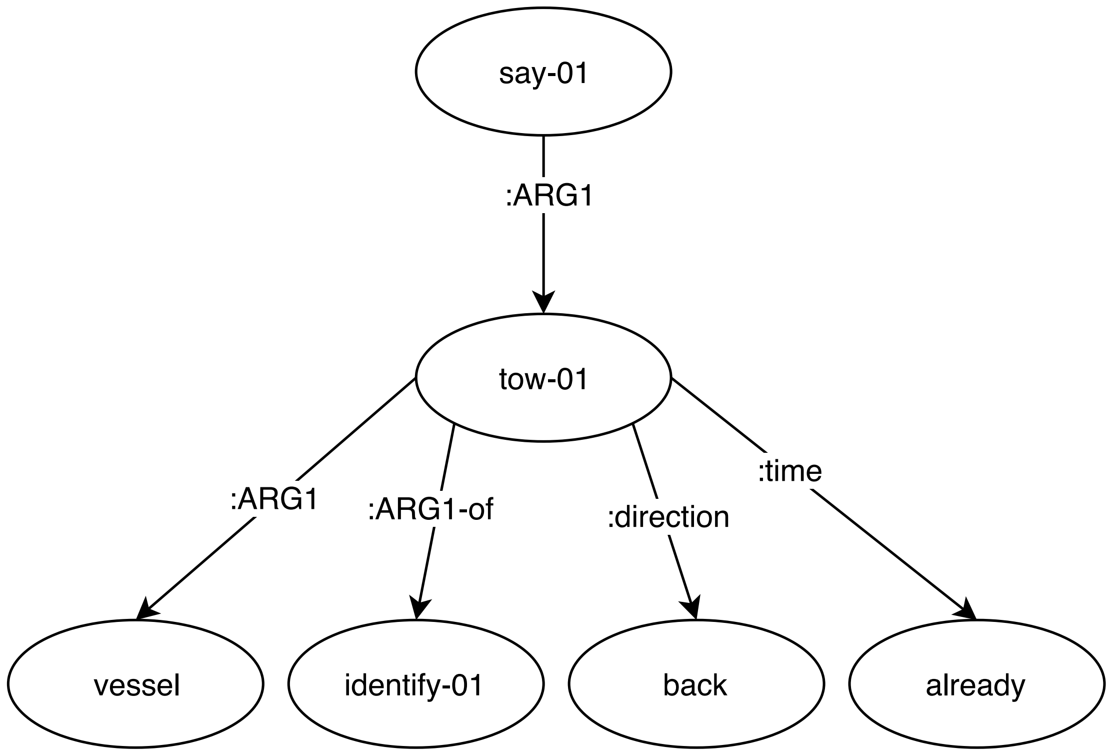

Figure 1.1: AMR Parse Tree of Sentence: "The vessel already has been towed back to an unidentified port." The word "towed" is a trigger for a "Transport" event.

1.0.5 Maximum Entropy Learners

Maximum entropy learners are useful because they are much more transparent compared to their neural network counterparts, although their performance may be slightly lower. The transparency allows us to directly see the importance of a feature in the model's decision making process. This can help us reason about what features are actually helpful or not in a model. We train a MaxEnt model to differentiate between seen types, true positive candidate triggers (an unseen type) and false positive candidate triggers (triggers not mapping to any event).

2 Approach Details

In order to test our hypotheses, there was a significant setup process involved from generating the train and test sets to getting a running version of the zero-shot classifier.

2.0.1 Data

Firstly, we generate the training, development, and test sets for our experiments. The data sets are extracted from annotated sentences from the ACE corpus. We start by selecting 150 sentences that have events only from 23 unseen event types. The unseen types were chosen to be consistent with the unseen types used in scenarios A,B,C, and D of the zero-shot paper. Among the remaining sentences, 10% were used for the development set and the remaining 90% became the training set. While the test set only contained events from unseen event types, the development and training sets could contain sentences with events from any of the 33 event types. However, we replaced the unseen event type label with ”OTHER”.

2.0.2 AMR

After generating the data sets, we used the JAMR parser to generate Abstract Meaning Representations (AMR) for all sentences. Although rarely, the parser would fail to construct a representation of a sentence, for our purposes it was usually sufficient. Furthermore, the alignment parser that maps nodes in the parse tree back to words and phrases in the sentence was sometimes unreliable and would fail to map things correctly. This would occasionally cause complications when we would try to link candidate triggers back to the actual triggers in the sentence. This would normally impact cases where the trigger word was actually a phrase and had a slight impact on performance. The AMR parses had two main uses: a first step in identifying candidate triggers using the word sense disambiguation provided by the parser and tuples for the generation of event mention structures.

2.0.3 Candidate Trigger Identification

The next step in the pipeline is to extract the candidate triggers from the AMR parses. To do this, we consider all noun and verb word senses at nodes in the AMR parse tree that are either found in OntoNotes or are lexical units in FrameNet. The reasoning behind this approach is that if a word sense appears in OntoNotes, we can more easily leverage the resources found in OntoNotes as features later on. If the word appears as a lexical unit in FrameNet, it has increased probability of being a trigger as the lexical units in FrameNet are common trigger words for semantic frames. Using this approach we find 14985, 1690, and 851 candidate triggers for the training (2556 sentences), development (285 sentences), and test (150 sentences) sets respectively. The number of candidate triggers is quite high compared to the number of actual events annotated per sentence. Despite the high quantity of trigger words, we noticed that among all candidate triggers, only about 65% of actual triggers were captured. As a result, the actual performance in the experiments was upper bounded by this number.

2.0.4 Baseline Classifier

To compare the results of our later experiments, we also designed a baseline classifier to get a baseline of performance measures which we could compare against. In the baseline case, we do not train any model. Instead, we compute a vector representation for each event type by taking a weighted average of the word vectors for all triggers grouped by event type. Next, we compute the cosine similarity between the word vectors for each candidate trigger and each event type vector. Then we rank the most likely event types for each candidate trigger and predict the most similar type. The results for the baseline classifier are given in the results section.

2.0.5 Maximum Entropy (MaxEnt) Classifier

Next, we train a MaxEnt model to distinguish between three main categories: seen types, unseen types, false candidate triggers. We generate training data as follows, for each candidate trigger in the training set we set its label to be either the correct event type if it is a seen type, "OTHER" if the event type is unseen, and "OTHER2" if the candidate trigger is not actually a trigger. As features, we tested including the word sense of the candidate trigger, AMR relationships, and their arguments from the AMR parse tree. We found that the word sense was the dominating feature in the classifier's decision making. We reason that this has to do with fact that given a word sense, the types of potential relations and arguments a word can have are already predetermined. For example, according to OntoNotes, the word sense attack-01 can be anticipated to have the following arguments: attacker (ARG0), entity attacked (ARG1), and attribute (ARG2). So knowing the word sense already gives an idea of generic arguments that can be present in a sentence. Overall, we found the MaxEnt model was able to significantly reduce the number of false candidate triggers, while maintaining the majority of true positive candidate triggers.

After training our MaxEnt model, we would predict the appropriate label for candidate triggers in the development and test sets. We would ignore each label predicted "OTHER2", while each trigger predicted as "OTHER" would be passed onto the zero-shot classifier to attempt to predict the correct unseen type. We attempted to use several different ways of computing scores to rank event types for each candidate trigger. Usually, the method involved some composition of word embedding vectors and computing similarity measures between vector representations.

2.0.6 Event mentions and event types structures

We decided to use slightly simpler versions of the event mention and event type structures described in section 4 of the paper. Our reasoning was that we wanted to test if there would be any gains in performance as we added additional complexity to the structures.

The event mention structures were defined as a list of tuples of the form (word1, relation, word2), where each tuple is extracted from the AMR parse subtree containing our candidate trigger. For each tuple, we compute an embedding representing the tuple as the concatenation of the word vectors for 'word1' and 'word2'. We skip the multiplication by the AMR relation matrix for the purposes of simplicity, but we think it would be a good future experiment to see if any increase in performance is observed using the AMR relation matrix. Next, we add all the vector representations for the tuples to get a single vector of dimension 2D, where D is the original dimensionality of each individual word vector. This approach makes it easier to deal with dimensionality mismatch between event mention and event type structures. However, it can be seen as a bit lossy, as we condensing 2D*(number of event mention tuples) information into a vector of size only 2D. On the other hand, Huang et al. use a trained CNN to generate a dense vector representation of the event mention structure.

The event type structures are constructed similarly to the event mention structures. We build a list of tuples of the form (y,r), where y is the name of the event type and r is name of a role associated with the event type as defined in the ontology. We concatenate word vector representations of y and r to form a representation for each tuple and sum the tuple representations to form the event type structure. Similar to the event mention structures, we also skip multiplying by a tensor representing the relationship between the type and role, but think it is a good idea for a future experiment. Compared to the approach outlined in the paper, this approach makes it easier to deal with dimensionality mismatch between event mention and event type structures, but suffers from the same issue of being lossy since we condense a 2D*(number of event type tuples) matrix into a vector size 2D.

2.0.7 Zero-Shot Classifier

Given each candidate trigger predicted as "OTHER", we attempt to assign the correct event type to the trigger. This process involves using similarity measures between a representation of the candidate trigger and its arguments (event mention structure) and representations of event types (event type structure). Normally, the representation for the candidate trigger is the concatenation of the word vector for the trigger and the event mention structure as defined above for the trigger. This produces a vector of dimension 3D. The event type representation is the concatenation of the event type embedding we computed for the baseline approach and the event type structures defined above. We rank event type predictions by similarity which is computed as follows:

$$similarity = cos([V_t; V_{S_t}], [V_y; V_{S_y}]);$$

1. V_t is the word representation of the trigger t ($\in \mathbb{R}^D$)

2. V_{S_t} is the event mention structure for trigger t ($\in \mathbb{R}^{2D}$)

3. V_y is the word representation of event type name y ($\in \mathbb{R}^D$)

4. V_{S_y} is the event type structure for type y ($\in \mathbb{R}^{2D}$)

In performing zero-shot classification, the goals of our experiments were to assess the importance of paradigmatic role names, characteristic event type names, and the effectiveness of our event mention and event type structures.

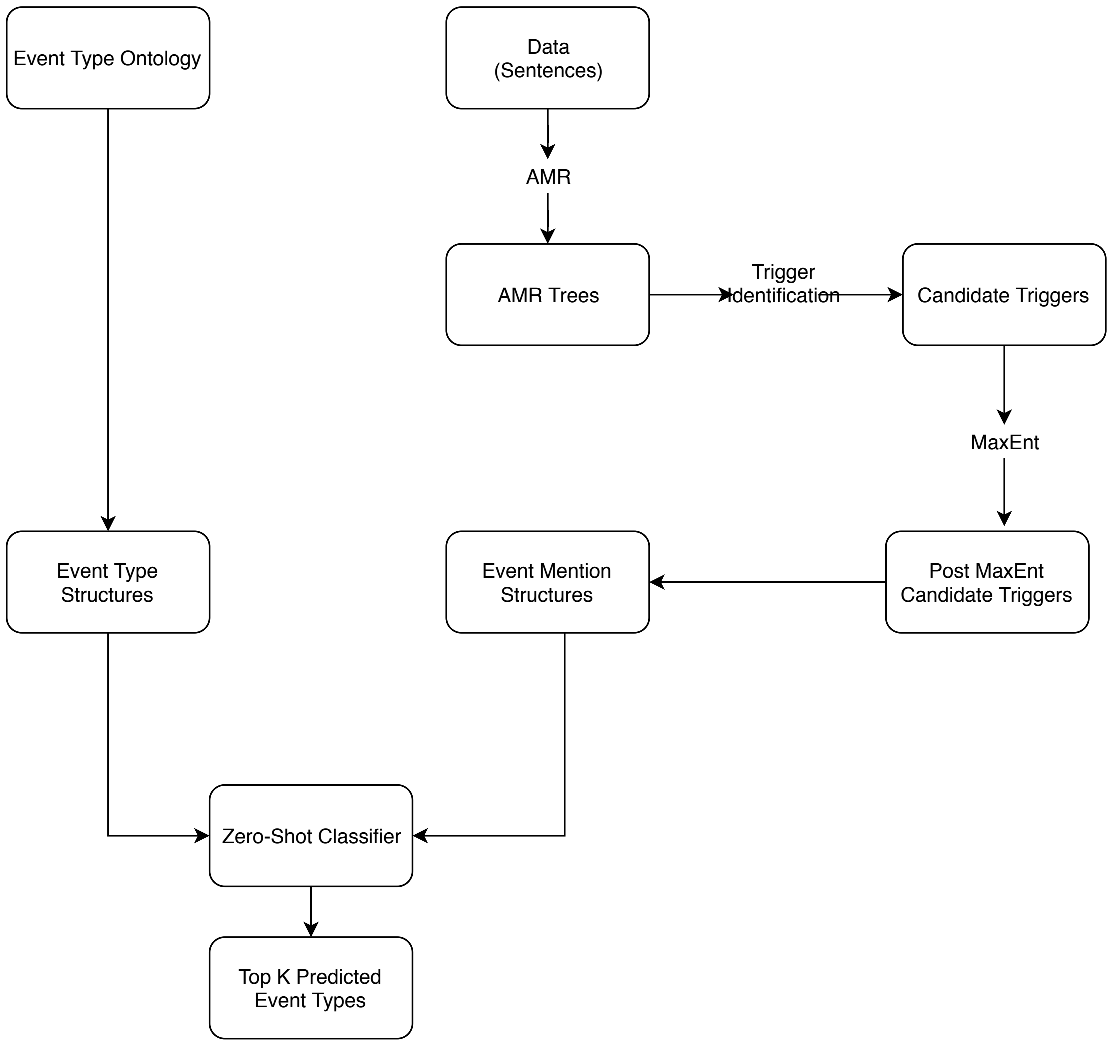

Figure 2.1: Overview of Pipeline Process

3 Results & Discussion

In this section, we provide background information on our experiments and interpretations for the presented results.

Table 3.1: Candidate Trigger Information

Setting	# of Vetted Candidates	# of Candidates	# of Events	Accuracy (%)	Retention (%)
Development	571	1590	354	56.94	76.42
Test	276	851	172	54.65	83.18

Table 3.1 gives statistics on candidate triggers and actual triggers for both the development and test sets. '# of Candidates' gives the number of the candidate triggers identified after matching words from the AMR parse trees with word senses in OntoNotes and lexical units in FrameNet. '# of Events' gives the actual number of events annotated in the data set. It is easy to see that the number of candidate triggers is roughly 5 times more than the number of actual events, so the overwhelming majority of candidates are false positives. As a result, attempting to correctly classify triggers without a mechanism of vetting out false candidates can give misleading performance metrics. Our attempt to solve this issue boiled down to training a MaxEnt classifier to distinguish between three categories as outlined in the previous section. With only the word sense and candidate trigger word as features, the model is able to significantly reduce the number of false candidates while retaining 76-83% of true positive candidate triggers (see 'Retention' column). We hypothesize including more features, such as tuples from the event mention structure for a candidate trigger will further increase the retention rate. Additionally, it seems that the current approach of identifying candidate triggers bottlenecks the performance of any classifier, as only (54.65 * 100/83.18) = 65.7% of actual test triggers were identified in the candidate trigger identification phase. This means that even if our classifier was perfect, its performance would be upper bounded by 65.7%. We think it is worth looking into ways to improve candidate trigger identification to infer more triggers, while training an intermediary model to rule out

false positives.

Table 3.2: Baseline and Zero-shot classifier Results on Dev and Test Sets For Experiment 1 (% Accuracy)

Setting	K=1	K=3	K=5
Baseline (dev)	3.11	7.36	9.92
A (dev)	23.51	33.14	44.48
B (dev)	35.13	42.21	46.18
C (dev)	35.98	43.91	45.89
D (dev)	39.66	43.63	45.61
Baseline (test)	2.91	6.4	9.3
A (test)	20.93	30.81	37.79
B (test)	20.93	30.81	36.63
C (test)	21.5	32.56	38.95
D (test)	25	33.72	41.86

Table 3.2 contains the results of Experiment 1 categorized by setting. Experiment 1 involved using the zero-shot classifier by computing cosine similarities between the concatenation of the trigger embedding and event mention structure representation with the concatenation of the event type embedding and event type structure representation. $\hat{y} = argmax_{y \in Y} cos([V_t; V_{S_t}], [V_y; V_{S_y}])$, where Y is the set of event types in the ontology and $\hat{y}$ is the predicted type for candidate trigger 't'. The baseline classifier results do not involve the zero-shot classifier, but are the results for the baseline approach described in the previous section. There are 4 different settings we tested under. Settings A, B, C, and D differ in the number of seen types used when training our MaxEnt model. Each setting has 1, 3, 5, and 10 seen types respectively. We used the same seen types as the ones reported by Huang

et al.

Some general trends are immediately apparent from the table. Firstly, the baseline classifier which solely relies on the trigger embedding and weighted average embedding representations of the event type triggers seem to be low. The results seem to be somewhat consistent with the results reported by the baseline classifier used by Huang et al. We expected that the baseline should perform better in practice, but we were somewhat surprised by the weak performance. One of the disadvantages of the baseline classifier we think impacts the performance is the large number of false candidate triggers that it considers compared to the zero-shot classifier. We also tried experimenting with euclidean distance as a similarity measure, but found that the majority of the time only a single event would be predicted. We reason that for high dimensionality spaces, most vectors are very distant from each other, so a single event type would usually dominates predictions.

Additionally, we notice that the zero-shot classifier performs reasonably well even for k = 1 on the test set, predicting approximately (100*(20-25)/56.94) = 35-43% of the remaining good candidate triggers. We also notice that the performance on the development set improves considerably as the number of seen types increases, but not for the test set. This makes sense as our development set contains both seen and unseen types, but the test set only contains unseen types. Interestingly, the performance for k=5 on the development set are very similar (and k=3 with exception of scenario A). This could indicate that even when the correct event is not predicted as the most likely, the zero-shot classifier is doing a relatively good job of ranking the most probable event in the top 3 or 5 event types regardless of being trained on more seen types, with the exception of being trained on one seen type (scenario A).

Table 3.3: Zero-Shot classifier Results on Dev and Test Sets For Experiment 2 (Role Names) (% Accuracy)

Setting	K=1	K=3	K=5
A (dev)	22.1	34.28	43.91
B (dev)	33.43	41.64	44.76
C (dev)	34.84	42.21	44.75
D (dev)	40.51	42.49	44.48
A (test)	22.09	34.88	38.37
B (test)	21.51	33.72	37.2
C (test)	21.51	34.3	37.2
D (test)	21.51	34.3	38.37

In experiment 2, we were interested in evaluating the impact of choosing paradigmatic names for event type roles. In order to test our hypothesis, we replaced the role names of the event types with synonyms and regenerated our event type structures used by the zero-shot classifier, then recomputed the results, while keeping all other factors fixed. For example, for the Arrest-Jail event type we replaced the role names (Defendant, Adjudicator, Crime, Time, Place) with (Accused, Mediator, Offense, Occasion, Location). Since word embeddings are used for the event type structures, we hypothesized changing the names of the roles will cause differences in the zero-shot classifier performance. From the results in Table 3.3, we can see that the classification accuracy went up or down by up to 3-4 percentage points under certain test settings. For example, under setting D with k = 1 on the test set, the performance dropped (25 - 21.51) = 3.49%, which is significant considering the scale of the numbers we're dealing with. Although, it is clear the classifier is still able to learn, it is clear that choosing paradigmatic role names will lead to improved performance for new event types.

Table 3.4: Simplified Zero-Shot Classifier Results on Test Set For Experiment 3 (% Accuracy)

Setting	K=1
A (test)	37.79
B (test)	37.79
C (test)	39.53
D (test)	43.02

Table 3.4 shows the results of Experiment 3, where we used simplified structures for the zero-shot classification. For this experiment, for each candidate trigger, we computed the cosine similarity between the trigger embedding and the embedding of the event type name. So for example, for the trigger 'arrest', we would compute the cosine similarity of 'arrest' and the sum of the embeddings of 'arrest' and 'jail' for the Arrest-Jail event type. The results clearly show a large improvement on the test set performance for k = 1. As a result, it makes one question the effectiveness of the event mention and event type structures used in the earlier experiments. Firstly, it's important to note that the event mention and event type structures used in our experiments differ compared to the ones used by Huang et al. Our structures are quite simplified and condensed as we pointed out in the previous section. Regardless, it does show that even simple representations can achieve relatively good performance. In order for the event mention and event type structures to be more useful, it seems that it is important not to condense the tuples representations into a single vector naively and to potentially incorporate the semantics of AMR relations and type-role relations into the event mention and type structures. However, this approach also comes

with challenges including handling dimensionality mismatch between event mention and event type structures and mapping candidate arguments in the mention structures to roles in the type structures. We hypothesize the effectiveness of this simple approach is due to being able to predict triggers of event types whose names are similar to common triggers for the said event type. For example, 'arrest' is a common trigger for the Arrest-Jail event. As a result, we hypothesized that the name of an event type also impacts the effectiveness of a zero-shot classifier. To support our hypothesis, we looked at the accuracy of our classifier across different event types. The results can be found in Table 3.5.

Table 3.5: Simplified Zero-Shot Classifier Results Per Event Type Breakdown on Test Set (% Accuracy)

Event Type	Correct	Total	(Total - |Missed Triggers|)	Trigger %	Event %
Justice-Sue	3	6	4	67	50/75
Justice-Convict	2	8	3	38	25/67
Justice-Trial-Hearing	7	13	7	54	53/100
Justice-Appeal	8	11	8	73	72/100
Justice-Charge-Indict	7	11	8	73	64/88
Justice-Release-Parole	5	12	5	42	42/100
Justice-Fine	6	9	9	100	67/67
Justice-Extradite	0	4	0	0	0/0
Justice-Pardon	1	2	1	50	50/100
Justice-Execute	1	3	2	67	33/50
Justice-Acquit	2	4	2	50	50/100
Business-Merge-Org	5	8	5	63	63/100
Business-Start-Org	0	6	2	33	0/0
Business-End-Org	0	7	3	43	0/0
Business-Declare-Bankruptcy	0	6	0	0	0/0
Life-Divorce	3	4	3	75	75/100
Life-Be-Born	6	8	6	75	75/100
Life-Marry	6	12	10	83	50/60
Life-Injure	3	6	3	50	50/100
Personnel-Nominate	3	7	3	43	43/100
Personnel-Start-Position	0	9	3	33	0/0
Conflict-Demonstrate	4	8	5	63	50/80
Contact-Phone-Write	2	8	2	25	25/100

Table 3.5 showcases the accuracy per event type of the simplified zero-shot classifier for scenario D. ’Correct’ is the number of events correctly predicted for the event type. ’Total’ is the total number of event of that type in the test set. ’(Total - |Missed Triggers|)’ is the number of candidate triggers that map to an actual trigger. This is the maximum number of events the zero-shot classifier could have gotten correct. ’Trigger %’ is the number of triggers for the event type that were correctly identified in the candidate trigger identification phase and ’Event %’ is the event classification accuracy considering all triggers and the accuracy taking into account missed triggers during the identification process, after the ’/’.

From the results, we can see that the candidate trigger identification process misses several true triggers for the majority of event types. In some cases, no triggers for a type are identified (e.g. Justice-Extradite). This shows that the candidate trigger identification is in fact a bottleneck in event extraction. Some of the drawbacks in the current process may be due to the fact that certain words do not appear as senses in OntoNotes or as lexical units in FrameNet, for example the word, extradite. We can from the results see that not a single Extradite event trigger made it past the candidate trigger process. Furthermore, we noticed it is less likely for multiple word triggers to be correctly identified, which may imply some deficiency in the alignments proposed by the JAMR parser.

Another important point to note is that the zero-shot classifier does very well on certain event types, while poorly on others. Upon further inspection, we can see that the classifier performs very well on event types where the event type is indicative of common triggers for the event (e.g. Arrest-Jail), while poorly on event types where that is not the case like Start-Position or Start-Org. Since Justice event types tend to be named based on common triggers for that event type category, the zero-shot classifier does quite well on all Justice events. As a result, we infer that new event types should be named while keeping common indicative verbs or nouns in mind to maximize the success of zero-shot classification methods.

4 Conclusion

To summarize, we found the following choices to be of crucial importance when designing a zero-shot classifier: distance metric, choice of role names for event types, names of event types, and structures representing event mentions and event structure types. Cosine similarity seemed to be a good choice for a similarity measure, while euclidean distance was not. The choice of role names also influenced the performance of the zero-shot classifier by merely swapping out some words for similar synonyms. Therefore choosing the most characteristic names for roles will be advantageous when adding new events to an ontology. Event types with names that include words similar to triggers for the event also are more easily predicted than event types with less reflective names. As for leveraging event mention and event type structures, we found that simplifying these structures too much by condensing the information into a single vector representation was too lossy when done naively and could even lead to reduced performance for the classifier. Additionally, we found that the candidate trigger identification phase creates somewhat of a bottleneck by limiting the triggers considered during classification. Finally, we found that training a classifier with very simple features could do a relatively good job at significantly reducing the number of false positive candidates while retaining the majority of true positives. In conclusion, although zero-shot learning can be effective even with no training examples, a fair amount of effort is required when adding new events to an event ontology to ensure names of events and roles are chosen in a way to maximize the classifier performance.

References

[1] Baker, C. F., Fillmore, C. J., and Lowe, J. B. The Berkeley FrameNet project. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1* (Stroudsburg, PA, USA, 1998), COLING '98, Association for Computational Linguistics, pp. 86–90.

[2] Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse* (Sofia, Bulgaria, August 2013), Association for Computational Linguistics, pp. 178–186.

[3] Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M., and Mikolov, T. Devise: A deep visual-semantic embedding model. In *NIPS* (2013), C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., pp. 2121–2129.

[4] Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers* (Stroudsburg, PA, USA, 2006), NAACL-Short '06, Association for Computational Linguistics, pp. 57–60.

[5] Huang, L., Ji, H., Cho, K., Dagan, I., Riedel, S., and Voss, C. Zero-shot transfer learning for event extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 2160–2170.

[6] Ji, H., and Grishman, R. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT* (Columbus, Ohio, June 2008), Association for Computational Linguistics, pp. 254–262.

[7] Kodirov, E., Xiang, T., Fu, Z., and Gong, S. Unsupervised domain adaptation for zero-shot learning. In *2015 IEEE International Conference on Computer Vision (ICCV)* (Dec 2015), pp. 2452–2460.

[8] Liu, S., Chen, Y., He, S., Liu, K., and Zhao, J. Leveraging framenet to improve automatic event detection. In *ACL* (2016).

[9] Wang, C., Xue, N., and Pradhan, S. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Denver, Colorado, May–June 2015), Association for Computational Linguistics, pp. 366–375.