CMU-LTI at KBP 2015 Event Track

Zhengzhong Liu **Jun Araki** **Dheeru Dua** **Teruko Mitamura** **Eduard Hovy**
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
{liu, junaraki, ddua, teruko, hovy}@cs.cmu.edu

Abstract

We describe CMU LTI's participation in the KBP 2015 Event Track. We officially participated in Task 1: Event Nugget Detection track and Task 3: Event Coreference track. Our system rank high in both tracks. We found that our combined system is competitive but have room to improve. In addition, we have conducted follow up experiments by creating a simple piplined system, and We found it competitive comparing to the official submissions.

1 Introduction

CMU LTI officially participates in 2 tasks in the event track. In the event nugget detection task, we submit two separate systems. Both systems employ a Conditional Random Field (CRF) system, but differs in implementation details and selection of features. In particular, LTI1 uses a vanilla average perceptron CRF model and consider multi-type mention simply as new class, while LTI2 uses a Passive-Aggressive CRF model and make use of 5-best decoding to deal with multi-type mentions.

In the event hopper coreference task, we submit 3 runs generated by 2 systems. Both coreference systems are based on the Latent Antecedent Tree (LAT) model, there are some differences in the implementation and the choice of features as well. Our system rank competitively among the participants, our team rank second in the event nugget detection task and rank third in the hopper coreference task.

In follow up experiments, we found that some features in our hopper task are not actually implemented correctly. After fixing the problem, our system rank the first in all submissions. In addition, we've also conducted follow up experiments on the nugget and hopper joint task by combining our nugget system and coreference system as a pipeline. Our simple pipeline system is very competitive and score higher than all the submissions. LTI1 system on mention detection and coreference can be both found on an online repository[1].

2 Task 1: Event Nugget Detection

2.1 System 1 : LTI1

LTI1 deploys a discriminativly trained Conditional Random Field (CRF) model to detect mention span and event type, and a linear Support Vector Machine (SVM) model to determine mention realis status. The CRF model is trained with the structured perceptron(Collins, 2002), which is outlined in Algorithm 1. The decoding step is done using standard Viterbi algorithm. Our final system always make use of the average weight variation as described in Collins (2002).

A number of "Double tagging" mentions are annotated in the corpus, in which a mention might have one or more event types. In LTI1 we simply combine multiple labels for each mention into a single label[2]. The intuition behind is that some surface forms are usually associated with some fixed mention types, for instance, "KILL" is normally associated with "Life.Die" and "Conflict attack".

[1] https://bitbucket.org/hunterhector/cmu-script/

[2] Multi-label classification can be solved with sophisticated methods when simple concatenation creates too many classes. In the KBP event dataset, the number of extended classes is small (56 in the training data).

Algorithm 1 Structured perceptron.

Input: training examples $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$
Input: number of iterations T
Output: weight vector $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{0}$ ▷ Initialization.
for $t \leftarrow 1..T$ **do**
  for $i \leftarrow 1..N$ **do**
    $\hat{y}^{(i)} = \arg\max_{y \in \mathcal{Y}(x^{(i)})} \mathbf{w} \cdot \Phi(x^{(i)}, y)$
    if $\hat{y}^{(i)} \neq y^{(i)}$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \Phi(x^{(i)}, y^{(i)}) - \Phi(x^{(i)}, \hat{y}^{(i)})$
return $\mathbf{w}$

An event mention is normally composed by its mention trigger and the arguments. To get a list of arguments for the event mention. We run two Semantic Role Labeling system, the PropBank style Fanse Parser (Das and Smith, 2011) and the FrameNet style Semafor Parser (Tratz and Hovy, 2011). In addition, to reduce sparsity, LTI1 also employs a few external data resources, including WordNet (Fellbaum, 1998) and a Brown Clustering trained on newswire data (Sun et al., 2011). We also use the Stanford CoreNLP system to obtain lemma, part-of-speech and parsing information (Manning et al., 2014).

Using these resources, LTI1 employ regular linguistic features for mention type detection, which are summarized as followed:

1. Part-of-Speech, lemma, named entity tag of words in the 2-word window of the trigger (both side), the trigger word itself and the direct dependent words of the trigger.

2. Brown clusters, WordNet Synonym and derivative forms of the trigger.

3. Whether surrounding words match some selected WordNet senses, these senses are "Leader", "Worker", "Body Part", "Monetary System", "Possession", "Government", "Crime" and "Pathological State".

4. Closest named entity type.

5. Dependency features, including lemma, dependency type and part-of-speech of the child dependencies and head dependencies.

	Precision	Recall	F1
Span	74.36	55.72	63.62
Type	67.08	50.25	57.38
Realis	51.79	38.75	44.27
All	46.29	34.63	39.56

Table 1: Event Nugget score over 5-fold validation on training data for LTI1.

	Precision	Recall	F1
Span	82.46	50.30	62.49
Type	73.68	44.94	55.83
Realis	62.09	37.87	47.05
All	55.12	33.62	41.77

Table 2: Official Event Nugget score for LTI1.

6. Semantic role related features includes the frame name and the argument role, named entity tag, argument head word lemma and WordNet sense (selected from the above list as well) of the arguments.

The WordNet related features are selected following the intuition that certain category of words are likely to imply the existence of certain events. For example, "Leader" are normally associated with "Personnel" type. We are currently working on methods that can automatically select such senses instead of using explicit human intervention.

In the current implementation, all the raw features are simply concatenated with the mention type. However, one can further make use of the hierarchy of mention to extract features on the higher ontology. For example, when extracting features for "Personnel.End-Position", one can also add a feature about "Personnel". We didn't finish the implementation of this variation and will leave it to our future work.

The realis model is a simple SVM model trained using LIBLINEAR[3]. We use similar window features and dependency features as in mention type detection. We also add frame argument role names in to the model (i.e. Attacker). However, we didn't include most of the lexicalized features used in type detection to avoid overfitting. We design a specific

[3]https://www.csie.ntu.edu.tw/ cjlin/liblinear/

Fold	Precision	Recall	F1
1	71.68	71.63	71.66
2	64.06	64.06	64.06
3	62.07	62.07	62.07
4	72.66	72.66	72.66
5	62.21	62.21	62.21

Table 3: 5-fold validation for Realis detection on training data with gold span and types for LTI1

feature to capture whether the phrase containing the event mention is quoted (if the whole sentence is quoted, we do not fire this feature). In order to test the effect of our realis model, we evaluate its performance given gold standard mention span and types in the training data. We report the 5-fold validation result in Table 3[4].

We report our score averaged on the 5-fold validation in the training data in Table 1, and our official score in Table 2. LTI1 ranks 2nd among all the participants in terms of the final event type + realis status (All in the table) F1 score. LTI1 also rank high according the other metrics. Notably, we found that the system is very robust against across datasets. Although we have different precision, recall trade off between the training and test set, the F-score is relatively stable.

2.2 System 2 : LTI2

The event extraction task entails detecting the event mention span in the documents and classifying them into pre-defined types and subtypes as per the RichERE guidelines. In LTI2 submission we treat this as a sequence prediction task and use Conditional Random Fields for predicting the sequence of target variables in the sentence. For instance, in the sentence "If the US pressured the Jamaican government to capture and extradite him, then it was the Jamaica that arrested and imprisoned him, not the US", the event mention extradite can be classified as type(subtype) Justice(Extradite) as well as Movement(Transport-Person). In order to cater this multi-label sequence prediction task we optimized over k-Best Viterbi parses during CRF training.

[4]Because the span detection is given, the precision and recall are the same all the time.

Attributes	Prec.	Recall	F1
Span	77.00	39.53	52.24
Type	68.79	35.31	46.67
Realis	51.41	26.39	34.88
All	45.47	23.34	30.85

Table 4: Official Results on Test Set for LTI2

Algorithm 2 Passive-Aggressive style CRF

Input: training examples $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$
Input: number of iterations T
Output: weight vector $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{0}$ ▷ Initialization.
for $t \leftarrow 1..T$ **do**
  for $i \leftarrow 1..N$ **do**
    $\hat{y}_t^{(i)} \leftarrow \arg\max_{y \in \mathcal{Y}(x^{(i)}{}_t)} \mathbf{w} \cdot (x_t{}^{(i)}, w_t)$
    if $\hat{y}_t{}^{(i)} \neq y_t{}^{(i)}$ **then**
      $loss_t \leftarrow s(x, \hat{y}) - s(x, y) + sqrt(d(\hat{y}, y))$
      $\tau_t \leftarrow \frac{loss_t}{||x_t{}^2||}$
      $w_{t+1} \leftarrow w_t + \tau_t y_t x_t$
return $\mathbf{w}$

Algorithm 3 Merging Top-n sequences to handle multiple-tagging

Input: 5-Best-Viterbi-Parses $(y_j, score_j), 1 \leq j \leq 5$
Input: Threshold, ϵ and Scaling Factor p
Output: Top-n Predicted Sequence

$prediction \leftarrow \{y_1\}$
$\mu \leftarrow \frac{\sum_{i=1}^{5} score_i}{5}$
$\sigma \leftarrow \sqrt{\frac{\sum_{i=1}^{5} (score_i - \mu)^2}{5}}$
for $j \leftarrow 2...5$ **do**
  $Nscore_j = [(score_j - \mu)/\sigma] * p$
  if $Nscore_j - Nscore_{j+1} \leq \epsilon$ **then**
    $prediction \leftarrow prediction \cup y_j$
  else
    return $prediction$

At the testing time we extract the 5-best Viterbi sequences and merge the results from the top-n sequences. The algorithm for choosing top-n sequence is described in Algorithm 3, where p and ϵ were set to 0.4 and 0.1 respectively after cross-validation. These top-n sequences are then merged to get multiple-taggings of triggers in prediction sequence.

The features used in training the CRF were POS tags, previous five words and next five words and their POS tags, two verbs in past and future, brown

Attributes	Prec.	Recall	F1
Span	81.7	44.36	57.52
Type	72.91	39.56	51.29
Realis	61.84	33.55	43.50
All	55.37	30.04	38.95

Table 5: Fixed Results on Test Set for LTI2 (Precision Oriented)

Attributes	Prec.	Recall	F1
Span	77.59	49.14	60.17
Type	69.61	44.08	53.98
Realis	52.71	33.38	40.87
All	47.17	29.87	36.58

Table 6: Fixed Results on Test Set for LTI2 (Recall oriented)

clusters with thirteen bits, lemmas of the event trigger in the WordNet hierarchy, parse trees from Stanford dependency parse, two events seen in the history, event arguments types extracted Semantic Role Labeling followed by Named-Entity Recognition of the arguments.

The LTI2 system could be run in two modes - high precision and high recall. The high recall system used additional features including brown clusters with 8 bits, a gazatteer of event triggers and WordNet synsets. The synsets increased recall but reduced the precision a bit.

For extracting the realis information, we used a two step approach. First, we extracted the event mention spans and types as described earlier, then, conditioned on the event information we performed yet another sequence predcition over the training instances using CRFs to find out the realis type.

Due to errors introduced by parallelizing the process, the official submission of LTI2 contains some formatting errors: results of one single document is scattered in multiple places and only the last segment of the submission are getting scored. Table 4 shows the official performance. We've fixed the problem after the official runs. We rerun the system with two variations, Table 5 and 6 show the results on the test-set using our high precision and high recall variants respectively.

3 Task 3: Event Nugget Coreference

3.1 System 1: LTI1 and LTI3

LTI1 and LTI3 use the perceptron model with latent antecedent tree method (Fernandes et al., 2012; Björkelund and Kuhn, 2014). These two systems only differs with slight training variations. The features employed can be roughly classified into 3 categories:

Trigger match: exact and fuzzy match on the trigger word, uses standard linguistic features (pos, lemma, etc.) and resources like Brown Clustering and WordNet. Information from mention type and realis type are also used;

Argument match: exact and fuzzy match on the arguments, including their string, argument role and coreference information;

Discourse features: encodes sentence and mention distances.

We train the Latent Tree model with a passive-aggressive algorithm(Koby Crammer, 2006) similar to that of Björkelund and Kuhn (2014). Our implementation is slightly difference in the Passive-Aggressive step. Our algorithm is detailed in algorithm 4, where: $\mathcal{A}$ represent the set of possible antecedents on the left; $\tilde{\mathcal{A}}$ represent the set of antecedents that are allowed by the gold standard coreference; Φ is the feature function over the tree; $\hat{y}$ represent the best decoding given current features; $\tilde{y}$ represent the current best decoding among the correct coreference structure, i.e., the latent tree. The algorithm iterativly update the weight vector in a Passive-Aggressive manner. During implementation, we found that the PA algorithm is important for the algorithm to converge well.

Algorithm 4 PA algorithm for latent trees in LTI1

Input: Training data D, number of iterations T
Output: Weight vector w

$w = \vec{0}$
for $t \leftarrow 1..T$ **do**
  for $\langle M_i, \mathcal{A}_i, \tilde{\mathcal{A}}_i \rangle \in D$ **do**
    $\hat{y}_i = \arg\max_{\mathcal{Y}(\mathcal{A})} score(y)$
    if $\neg Correct(\hat{y}_i)$ **then**
      $\tilde{y}_i = \arg\max_{\mathcal{Y}(\tilde{\mathcal{A}})} score(y)$
      $\Delta = \Phi(\tilde{y}_i) - \Phi(\hat{y}_i)$
      $\tau = \frac{\Delta * w}{||\Delta||^2}$
      $w = w + \tau\Delta$
return w

	B^3	CEAF-E	MUC	BLANC	Aver.
LTI1	82.27	75.14	60.90	71.56	72.47
LTI2	79.72	71.49	49.38	68.33	67.23
LTI3	82.63	76.32	58.86	71.08	72.22

Table 7: Official results of LTI1-3 on Hopper Coreference task[6]

We didn't submit the averaged perceptron version for LTI1 due to a system bug. In later experiments, we observe that the performance are better and consistent with the averaged version. Our averaged score on the development set reports **77.25** over 5-fold validation, better than **71.71** with the vanilla version. And our final evaluation result is now **76.78**, which rank highest in all the submissions. Our official results for the final system are summarized in Table 7. Using the fixed model, we also run a simple end-to-end pipeline by putting the two steps together. We obtained **64.08** (official) and **38.24** (updated) averaged coreference score for task 2. In terms of the new updated scores, we rank at the 3rd place among all official submissions.

3.2 System 2: LTI2

We formalize event nugget coreference as a problem of document-level structure prediction. Given input document x with n gold standard event nuggets $\{en_j\}_{j=1}^{n}$, the goal of our algorithms is to predict an event graph y whose edges represent coreference links between the event nuggets. To solve the problem, we employs latent antecedent trees (Fernandes et al., 2012; Björkelund and Kuhn, 2014) with structured perceptron (Collins, 2002), as shown in Algorithm 1.

LTI2 also employs latent antecedent trees, trained with averaged perceptron (Collins, 2002). It uses a handful of features, which are string match, part-of-speech combinations, and word embedding similarities. The motivation for the first two features are relatively obvious. As for the first feature, we assume that two event nuggets with the same surface form are likely to corefer to the same event. With respect to the second feature, we assume that some particular pairs of part-of-speeches such as (verb, verb) and (noun, noun) will be a relatively strong indicator for event nugget coreference. The string-match feature can be effective to some extent, but in general it is weak since event coreference can happen frequently with different lexical types such as paraphrases. To complement the weakness, we also devised the word embedding features to help the model resolve such event coreference. Word embeddings have been shown to capture lexico-semantic regularities; semantically similar words are close to each other in the embedding space (Agirre et al., 2009; Mikolov et al., 2013). Our assumption for the word embedding features is that if two lexically different event nuggets corefer, their semantics should be still similar, and thus their corresponding word embeddings should be close to each other in the embedding space. For word embeddings, we use the pre-trained 300-dimensional word vectors from Google News dataset (around 100 billion words) using word2vec tool[7], and apply cosine similarity as a numeric feature value to indicate how likely two event nuggets corefer. Since event types are also given with the gold standard event nuggets in Task 3, we further added a constraint that coreferential event nuggets should share the same even type.

4 Conclusion

In this paper we briefly report CMU LTI's participation in the event nugget track. Due to time constraint, we could not participate the joint mention detection and coreference track. In follow-up experiments, we found that our final pipelined system is competitive. However, We are interested in finding better methods to combine these two systems and make use of the interactions of the two stages. The LTI1 coreference and detection system is currently available at an online repository[8].

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of NAACL-HLT 2009*, pages 19–27.

[6]We report the updated results using the newest Scorer (V1.7). The new score is slightly different from the official score (by around 0.02 points in this case). See Liu et al. (2015) for details.

[7]https://code.google.com/p/word2vec/

[8]https://bitbucket.org/hunterhector/cmu-script/

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of ACL 2014*, pages 47–57.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP 2002*, pages 1–8.

Dipanjan Das and NA Smith. 2011. Semi-Supervised Frame-Semantic Parsing for Unknown Predicates. In *HLT '11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, volume 1, pages 1435–1444.

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

Eraldo Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of EMNLP/CoNLL 2012*, pages 41–48.

Joseph Keshet Shai Shalev-Shwartz Yoram Singer Koby Crammer, Ofer Dekel. 2006. Online passive-aggressive algorithms. In *Journal of Machine Learning Research 7*, page 551585.

Zhengzhong Liu, Teruko Mitamura, and Eduard Hovy. 2015. Overview of TAC KBP 2015 Event Nugget Track. In *Proceedings of Text Analysis Conference 2015*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of 52nd Annual Meeting of the ACL: System Demonstrations*, pages 55–60.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT 2013*, pages 746–751.

A Sun, R Grishman, and S Sekine. 2011. Semi-supervised relation extraction with large-scale word clustering. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 521–529.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, number 2010, pages 1257–1268.