CMUML Micro-Reader System for KBP 2016 Cold Start Slot Filling, Event Nugget Detection, and Event Argument Linking

Bishan Yang, Ndapandula Nakashole, Bryan Kisiel, Emmanouil A. Platanios, Abulhair Saparov, Shashank Srivastava, Derry Wijaya, Tom Mitchell
School of Computer Science
Carnegie Mellon University, Pittsburgh, PA
{bishan,bkisiel}@cs.cmu.edu

Abstract

In this paper, we present an overview of the CMUML's Micro-Reader system for three TAC KBP tasks: Cold Start Slot Filling (SF); Event Nugget Detection and Coreference (EN); and Event Argument Extraction and Linking (EAL). The Micro-Reader system is a result of the CMU NELL team's research efforts on single-document understanding using background knowledge. It is a general-purpose machine reading system that takes as input a text document and outputs span-level semantic annotations for document understanding. There are several reading components in the Micro-Reader, each having a different reading capability. For the SF task, we aggregate the outputs of different reading components to propose KB assertions that can be used to answer the SF queries. For the EN and EAL tasks, we mainly utilize the event reading component of the Micro-Reader, which performs joint inference of entities and events within a document context.

1 Introduction

In this paper, we describe the CMUML's Micro-Reader system for three TAC KBP 2016 tasks: Cold Start Slot Filling (SF); Event Nugget Detection and Coreference (EN); and Event Argument Extraction and Linking (EAL). Our Micro-Reader system contains several reading components, each performing span-level or sentence-level semantic analysis of the document. The goal is to achieve a deeper understanding of what each sentence means in the context of a document. The Micro-Reader system we developed this year has a similar architecture to the one that was used in our 2015 submission, with the difference that it includes several newly-developed reading components: a LSTM-based relation extractor, a joint entity and event extractor, and a high-coverage verb relation analyzer. Figure 1 shows the system architecture of our Micro-reader system for TAC KBP 2016. The newly-developed micro-readers are highlighted in red. In the following, we first briefly describe the micro-reader components. Then, we discuss the experiment setup and processing pipeline for each KBP task.

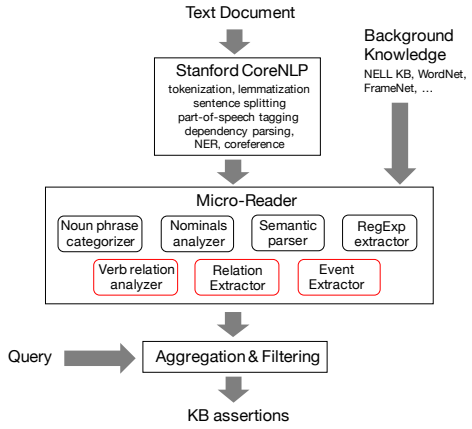

Figure 1: Overview of the NELL Micro-Reader system for the TAC KBP 2016 tasks.

2 NELL Micro-Readers

Before running our own micro-readers, we process each KBP document using the StanfordCoreNLP pipeline.[1] The *tokenize, ssplit, pos, lemma, ner,*

[1] http://nlp.stanford.edu/software/corenlp.shtml.

parse, depparse, dcoref, and *SUTime* modules were used.

Noun phrase categorizer: This reader predicts fine-grained semantic types for noun phrases. Used for type checking by other micro-readers that predict relations. This micro-reader categorizes noun phrases in context, it uses the context of a noun phrase to determine its type. The context is treated as features for a logistic regression classifier.

Nominal noun analyzer: This reader extracts relationships from nominal nouns such as "British journalist John Murray" from which the nationality and job title of John Murray are extracted. This micro reader uses background knowledge about type sequences to learn which compound noun sequences express which relations.

Verb-relation analyzer: This reader identifies verb-based relations with typed arguments. For example, in the sentence "Hemingway was born in the U.S..", the verb "born" expresses a "bornin(person, location)" relation, where the first argument is a person and the second argument is a location.

RegExp-based relation extractor This reader extracts specific relations based on specified regular expressions.

Semantic parser: We had two semantic parsers for predicting relations. One is a generative semantic parser that jointly infers the syntactic, morphological, and semantic representations of a given sentence under the guidance of background knowledge (Saparov and Mitchell, 2016). The other is our 2014 joint syntax and semantic CCG parser (Krishnamurthy and Mitchell, 2012; Krishnamurthy and Mitchell, 2014) which was trained to extract NELL(Carlson et al., 2010; Mitchell et al., 2015) relations from text using distant supervision.

Relation extractor: This reader extracts entity mentions and their relations based on LSTM Networks. A LSTM-CRF encoder is first trained to identify entity mention candidates. Then a dependency-path LSTM encoder is trained to predict the relations between pairs of entity mentions.

Event extractor: This reader jointly extracts entity mentions, event trigger mentions, and event role fillers using background knowledge as constraints and features. This is based on our previously published work (Yang and Mitchell, 2016).

3 Cold Start Slot Filling

3.1 Document Retrieval and Entity Matching

The KBP source documents were indexed using Lucene[2]. Now, given a query, this index was used to retrieve relevant documents. In order to identify relevant sentences in the document, we perform matching between the arguments in the query and the retrieved document.

The Entity Matcher aims to correctly map surface strings in documents to query entities, even when the strings are syntactically different from the text in the entity name. We therefore, leveraged the Freebase Annotations of the ClueWeb Corpora (FACC)[3] dataset recently released by Google. This corpus enabled us to generate synonym sets containing all surface strings that can refer to the same Freebase entity. During entity matching, we perform lookups against an indexed version of this synonym sets data. This significantly improved entity matching recall.

3.2 SF Training Data

We used the queries, answers, and assessments from past KBP years to retrieve training examples for KBP relations. Additionally, in 2014 we carried out a few rounds of internal manual evaluations of our system output on queries from 2013 and earlier years. We manually labeled each slot value produced by our system as true or false. All values that were labeled as true, that were not in the NIST training data, were used as additional training data for our 2016 system.

3.3 LSTM-based Relation Extractor

In the following, we will describe in detail our LSTM-based relation extractor for SF. It is the only micro-reader that was trained using the SF training data.

Given a sentence, we applied a sequence labeling model based on the combinations of LSTM neural networks and Conditional Random Fiels (CRFs) (Lample et al., 2016) to extract entity mention candidates and use them as input to the rela-

[2]Lucene: http://lucene.apache.org/

[3]http://lemurproject.org/clueweb09/FACC1/

tion extractor. Specifically, we trained the LSTM-CRF model using the CoNLL 2012 training data, which includes 1,940 documents and annotations for 18 types of named entities. Most of these entity types directly correspond to the slot filler types. The trained LSTM-CRF was applied to the KBP source documents to extract entity mentions. In order to reduce the recall errors in relation extraction, we over-generated entity mention candidates by using the top-K[4] best viterbi paths in CRFs instead of only the best one. The extracted entity mention candidates were then used to generate entity pairs for relation classification.

The relation classifier was a LSTM network that encodes dependency paths between entity mentions as features. And it works as follows: each word w in a dependency path between a pair of entities is mapped to a d-dimensional vector $v_w \in R^d$ through an embedding matrix $E \in R^{|V| \times d}$, where $|V|$ is the vocabulary size, and each row corresponds to a vector of a word. We initialize the word embeddings with the 300-dimensional Google News pre-trained embeddings. For the dependency relations in the path, we randomly initialize their vector embeddings, and learn them during training.

To encode the shortest path between a pair of entities we use an LSTM recurrent neural network (RNN) which is capable of learning long range dependencies. While regular RNNs can also learn long dependencies, they tend be biased towards recent inputs in the sequence. LSTMs tackle this limitation with a memory cell and an adaptive gating mechanism that controls how much of the input to give to the memory cell, and how much of the previous state to forget.

We have a path: $p = p_1, ..., p_p \in R^d$ and an associated path matrix $P \in R^{p \times d}$, where each row corresponds to the embedding vector of the word in that position.

The LSTM path encoder generates the path encoding, v_p, as follows:

$$\begin{aligned} h_i &= LSTM(v_{p_i}, h_{i-1}, c_{i-1}), i = 1, \ldots, p \quad (1) \\ v_p &= h_i : i = p \end{aligned}$$

The LSTM encodes the word at timestep $i = t$ in the path using the word embedding vector v_{p_t}, the previous output h_{t-1}, and the previous state of the LSTM cell c_{t-1}. The output h_t is computed using the four main elements in the LSTM cell: an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t with a self-recurrent connection. The cell takes as input a d-dimensional input vector for mention word $x_t = p_i$, the previous hidden state h_{t-1}, and the memory cell c_{t-1}. It calculates the new vectors using the following equations:

$$\begin{aligned} i_t &= \sigma\left(W_{xi}x_t + U_{hi}h_{t-1} + b_i\right), \quad (2) \\ f_t &= \sigma\left(W_{xf}x_t + U_{hf}h_{t-1} + b_f\right), \\ o_t &= \sigma\left(W_{xo}x_t + U_{ho}h_{t-1} + b_o\right), \\ u_t &= \tanh\left(W_{xu}x_t + U_{hu}h_{t-1} + b_u\right), \\ c_t &= i_t \odot u_t + f_t \odot c_{t-1}, \\ h_t &= o_t \odot \tanh(c_t), \end{aligned}$$

where σ is the sigmoid function, $\odot$ is element-wise multiplication, the W and U parameters are weight matrices, and the b parameters are bias vectors.

From path encoding v_p, we compute the output of the neural network, a distribution over the relationships that can hold between a pair of entities. The output for each path is decoded by a linear layer and a *softmax* layer into probabilities over the labels. Therefore, the prediction d_r

$$d_r = (W_r \cdot v_p) \quad (3)$$

where $(z_i) = e^{z_i} / \sum_j e^{z_j}$.

We found that performing entity mention extraction and relation classification in a pipeline could result in invalid relations that have incompatible argument types. To solve this problem, we formulated an ILP-based objective that combines the LSTM-CRF entity mention extractor and the LSTM relation classifier to jointly predicts entities and relations with respect to type-consistency constraints, e.g., the "per:top_member_employee_of" is required to have a PERSON entity as the first argument and a ORG entity as the second argument.

3.4 Aggregation of different Micro-readers

At prediction time, we combine the outputs of the micro-readers [5]. Each micro-reader yielded a set of sentences potentially expressing a variety of slot

[4] we set K=5 in our experiments

[5] Some of our submissions also included the outputs of our previous years' CRF-based extractor.

Run Id	Measure	Precision	Recall	F1
CMUML1	CSSF	0.18	0.01	0.02
CMUML1	CSLDC	0.18	0.01	0.02
CMUML2	CSSF	0.12	0.01	0.02
CMUML2	CSLDC	0.13	0.02	0.03
CMUML4	CSSF	0.33	0.01	0.02
CMUML4	CSLDC	0.34	0.01	0.02

Table 1: Official evaluation scores (ALL-Micro) of various CMUML submissions for the SF task.

fillers, typically with significant redundancy. Redundant predictions were eliminated by way of identifying sentences expressing the same (relation, filler) pair, or where two filler values were deemed to be synonymous. Each filler is then assigned a confidence score based on the number of times it was found to be be expressed in the corpus.

3.5 Provenance Finder

It was necessary to locate the spans of text expressing filler values in the original source documents so that character offsets could be provided for provenance information. We again used the Apache Lucene index over source documents along with a series of heuristic string similarity metrics to identify the span of characters in the original documents that sufficiently matched the post-processed version of the text seen by the micro-readers. While not perfect, we did not find during system development that this approach ever failed to locate the correct span of text.

3.6 Submissions

We have submitted five entries (CMUML1-5) for the KBP cold start slot filling evaluation.

- CMUML1: Our main run using all of our 2016 micro readers except for the CRF-based extractor used in our previous years' submissions.
- CMUML2: This is the same as CMUML1, but with an unfiltered version of our LSTM-based relation extractor (without the ILP-based objective).
- CMUML3: This only uses the outputs of the LSTM-based relation extractor.
- CMUML4: This uses all the micro-readers except for the LSTM-based relation extractor.
- CMUML5: This combines CMUML1 with the CRF-based relation extractor used in our previous years' submissions.

Experimental results of these systems are shown in Table 1. We skip the results of CMUML3 since the micro-F1 scores are lower than 0.01. The scores of CMUML5 are similar to the scores of CMUML1.

4 Event Nuggets, Event Arguments, and Event Coreference

4.1 Event Training Data

We used the ACE2005 corpus and the "Event Argument Linking Pilot Gold Standard" corpus (LDC2016E60) as our training data. ACE2005 contains annotations of entities, values, relations, and events for 599 documents collected from newswire sources, discussion forums, web blogs, and broadcast conversations. LDC2016E60 contains annotations of entities, relations, and events (RichERE) for 91 documents collected from newswire sources and discussion forums. We manually mapped the ACE ontology to the RichERE ontology as both the EN and EAL tasks adopt the RichERE ontology. We also used the LDC released RichERE Training Annotation (LDC2016E31) as our development data, which contains 69 discussion forum threads from the TAC KBP 2015 Tri-Lingual EDL training data.

4.2 Joint Event and Entity Extraction

We applied our previous work – a joint event and entity extractor (Yang and Mitchell, 2016) to the EAL task and the EN task. It has the advantage of performing joint inference over the dependencies among entities, event nuggets, and event arguments, and therefore, is able to make more globally-informed decisions than standard pipeline approaches.

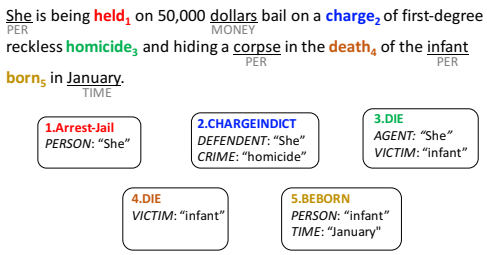

Figure 2: Example of the annotations output by the joint event and entity extractor.

Our joint event and entity extractor models three types of dependencies: (1) the dependencies between a single event and all of its arguments, (2) the co-occurrence relations between events across the document, and (3) dependencies between entity mentions. All of these dependencies are learned using graphical models. During inference, entities, event nuggets, and event arguments are simultaneously extracted while accounting for their dependencies as well as consistency constraints (e.g., a DESTINATION argument cannot be filled by a PERSON entity).

Figure 2 shows an example of the annotations output by our joint event and entity extractor. Given a text, it identifies all the entity mentions (the underlying spans) with appropriate types and all the event nuggets (the colored words) with appropriate event frames. Each event frame has an event type and a list of event arguments that are filled by a subset of the entity mentions. For example, the first event frame stores a ARREST-JAIL event and it has a PERSON argument that can be filled by the entity mention "She".

4.3 Features

Our joint event and entity extractor employs rich features constructed based on lexical and semantic resources as well as context. In the following we list two sets of features: one for the event-trigger factors and the other for the argument-role factors[6].

For event triggers/nuggets:

1. lemmas of the words in the trigger mention.
2. nominalization of the words based on Nomlex (Macleod et al., 1998).
3. context words within a window of size 2.
4. similarity features between the head word and a list of trigger seeds based on WordNet (Bronstein et al., 2015).
5. semantic frames that associate with the head word and its p-o-s tag based on FrameNet (Li et al., 2014).
6. pre-trained vector for the head word (Mikolov et al., 2013).
7. dependency edges involving the head word, both lexicalized and unlexicalized.
8. whether the head word is a pronoun.

For event arguments:

1. lemmas of the words in the entity mention.
2. lemmas of the words in the trigger mention.
3. words between the entity mention and the trigger mention.
4. the relative position of the entity mention to the trigger mention (before, after, or contain).
5. whether the entity mention and the trigger mention are in the same clause.
6. the shortest dependency paths between the entity mention and the trigger mention.

4.4 Realis Prediction

We developed a Maximum-Entropy classifier to predict the Realis values for the event nuggets (ACTUAL, GENERIC or OTHER). We used the same set of event trigger features listed above except for the WordNet-based similarity features (No. 4).

Similarly, we developed a Maximum-Entropy classifier to predict the Realis values for the event arguments (ACTUAL, GENERIC or OTHER), using the same set of event argument features listed above.

4.5 Event Argument Normalization

We used the Stanford SUTime library for time normalization. For pronoun entities, we used the Stanford Coreference system to resolve their references, except that we resolved the references of first-person nouns "I", "me", "my", "mine", and "myself" to the speaker or author of the text.

4.6 Event Coreference

We employed a standard mention-pair model combined with single-link clustering for event corefer-

[6]Please refer to Table 1 in Yang and Mitchell (2016) for a description of the whole set of features.

Argument Summary Score			
System	5%	50%	95%
CMUML1	2.5	3.1	3.6
CMUML2	2.9	3.5	4.0
CMUML3	2.5	3.1	3.6
Linking Summary Score			
System	5%	50%	95%
CMUML1	1.4	1.8	2.2
CMUML2	1.6	2.0	2.4
CMUML3	1.5	1.8	2.2

Table 2: Official evaluation scores of various CMUML submissions for the EAL task.

ence. To train the mention-pair model, we generate positive and negative examples (pairs of event nuggets) from the training data, and used the following features to train a logistic regression model:

1. head word string match; 2. head POS tags; 3. event types; 4. realis types; 5. cosine similarity between the head word embeddings of the two event nuggets (we use the pre-trained 300-dimensional word embeddings from word2vec[7]); 6. similarity between the words in the two event nuggets (based on term frequency (TF) vectors); 7. similarity between the context words (a window of three words before and after each event nugget); 8. similarity between the words in the argument mentions.

4.7 EAL submissions

We have submitted three entries (CMUML1-3) for the KBP Event Argument Extraction and Linking Task. All of them ran the same joint event and entity extractor, with minor differences on the post-processing of realis values and corpus-level event coreference.

- CMUML1: Our main run using the event extractor described above.
- CMUML2: This is the same as CMUML1, but with a rule-based realis value predictor for event arguments: we created a dictionary of modal words in English and classified an argument to be OTHER if it links to a modal word via the dependency parse tree.

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

CMUML1			
	Precision	Recall	F1
Plain	71.44	18.11	28.89
Mention_Type	61.0	15.48	24.70
Realis_Status	51.76	13.12	20.93
Mention+Realis	44.16	11.19	17.86
Coref	CoNLL F1: 15.21		
CMUML2			
	Precision	Recall	F1
Plain	72.19	17.91	28.70
Mention_Type	61.81	15.34	24.58
Realis_Status	52.18	12.95	20.75
Mention+Realis	44.71	11.10	17.78
Coref	CoNLL F1: 14.59		
CMUML3			
	Precision	Recall	F1
Plain	71.27	18.37	29.21
Mention_Type	60.44	15.58	24.77
Realis_Status	51.54	13.29	21.13
Mention+Realis	43.60	11.24	17.87
Coref	CoNLL F1: 15.19		

Table 3: Official evaluation scores of various CMUML submissions for the EN task.

- CMUML3: This is the same as CMUML1, but with a rule-based cross-document event coreference resolver: we considered two event nuggets to be coreferent if they share the same head word and appear in two documents that have similar headlines.

Experimental results of these systems are shown in Table 2.

4.8 EN submissions

We have submitted two entries (CMUML1-2) for the KBP Event Nugget Detection and Coreference Task. All of them ran the same joint event and entity extractor, with minor differences on the event coreference methods.

- CMUML1: Our main run using the event extractor described above.
- CMUML2: This uses a simplified version of the event extractor where the entity extraction component was removed and the joint inference is over event nuggets and event arguments.

- CMUML3: This is the same as CMUML1, but augmented with a rule-based component that aims to improve recall: in addition to the system predicted coreferent mentions, we identified all the other mentions that have the same head word and the same subject or object within the same document and considered them to be coreferent.

Experimental results of these systems are shown in Table 3.

5 Conclusion

In this paper, we presented an overview of the CMUML system for the KBP 2016 English Cold Start Slot Filling (SF); Event Argument Extraction and Linking (EAL); and Event Detection and Coreference. The system used a combination of micro-readers that analyze different semantic aspects of text. For future submissions, we hope to further improve the performance of our existing micro-readers and explore more effective ways of integrating different micro-readers.

Acknowledgments

This work was supported in part by DARPA (award number FA87501320005), and Google. Any opinions, findings, conclusions and recommendations expressed in this papers are the authors' and do not necessarily reflect those of the sponsors.

References

Ofer Bronstein, Ido Dagan, Qi Li, Heng Ji, and Anette Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? In *ACL Volume 2: Short Papers*, pages 372–376. Association for Computational Linguistics.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI 2010*.

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of EMNLP 2012*.

Jayant Krishnamurthy and Tom M Mitchell. 2014. Joint syntactic and semantic parsing with combinatory categorial grammar. In *Proceedings of ACL 2014*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 1846–1851. Association for Computational Linguistics.

Catherine Macleod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. 1998. Nomlex: A lexicon of nominalizations. In *Proceedings of EURALEX*, volume 98, pages 187–193. Citeseer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NIPS)*, pages 3111–3119.

Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Tanti Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, (AAAI)*, pages 2302–2310.

Abulhair Saparov and Tom M Mitchell. 2016. A probabilistic generative grammar for semantic parsing. *arXiv preprint arXiv:1606.06361*.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *NAACL*, pages 289–299.