# The AI2 system at SemEval-2017 Task 10 (ScienceIE): semi-supervised end-to-end entity and relation extraction

**Waleed Ammar Matthew E. Peters Chandra Bhagavatula Russell Power**
Allen Institute for Artificial Intelligence, Seattle, WA, USA
{waleeda,matthewp,chandrab,russellp}@allenai.org

## Abstract

This paper describes our submission for the ScienceIE shared task (SemEval-2017 Task 10) on entity and relation extraction from scientific papers. Our model is based on the end-to-end relation extraction model of Miwa and Bansal (2016) with several enhancements such as semi-supervised learning via neural language models, character-level encoding, gazetteers extracted from existing knowledge bases, and model ensembles. Our official submission ranked first in end-to-end entity and relation extraction (scenario 1), and second in the relation-only extraction (scenario 3).

## 1 Task overview

The ScienceIE shared task (Augenstein et al., 2017) focuses on information extraction from scientific papers. In the end-to-end evaluation scenario, participants were provided with a paragraph and asked to extract typed entities (`Task`, `Material` or `Process`) and relations (`Hyponym-of` or `Synonym-of`) in different scientific domains.

**Running example.** Consider the following input sentence: *"Here, we consider a radical pair in which the first electron spin is devoid of hyperfine interactions, while the second electron spin interacts isotropically with one spin-1 nucleus, e.g. nitrogen."* The provided human labeled entity annotations for this sentence include: *"electron spin"* of type `Process`, *"spin-1 nucleus"* of type `Material`, and *"nitrogen"* of type `Material`. The only positive relation annotation labeled for this sentence is: `Hyponym-of`(*"nitrogen"*, *"spin-1 nucleus"*). We will use this example throughout the paper to illustrate various parts of our system (see Fig. 1).

## 2 System description

In this section, we describe the components in our system.

**Text preprocessing.** We process the input text using spaCy[1] which provides sentence segmentation, tokenization, part-of-speech (POS) tagging and labeled dependency parsing.

**Label encoding.** We use the `BILOU` tagging scheme to encode the annotations for each of the three entity types. For a given entity type, each token in a sentence is labeled with: `O` if it does not belong to any entity, `U` if it belongs to a single-token entity, `B` if it is the beginning of a multi-word entity, `L` if it is the end of a multi-word entity, and `I` if it is inside a multi-word entity. In order to allow the same token to participate in multiple entities of different types, each token is assigned three labels (one for each entity type).[2]

The labeled data specify two kinds of relations: a directional relation (`Hyponym-of`) and an undirectional relation (`Synonym-of`). In our system, we automatically convert `Hyponym-of(e2,e1)` into a new label `Hypernym-of(e1,e2)` when `e2` follows `e1` in the sentence. We generate the label `No-relation(e1,e2)` as needed in order to have a label for each pair `(e1, e2)` where `e1` precedes `e2` in a sentence. This encoding allows us to only consider entity pairs in the order in which they appear in the sentence. In other

[1] https://spacy.io/

[2] In a small number of cases, the human labels specified at the character level did not coincide with our tokenization. In these cases we used the human annotations to override the tokenization. When two entity annotations of the same entity type overlap, we only keep the longer one.

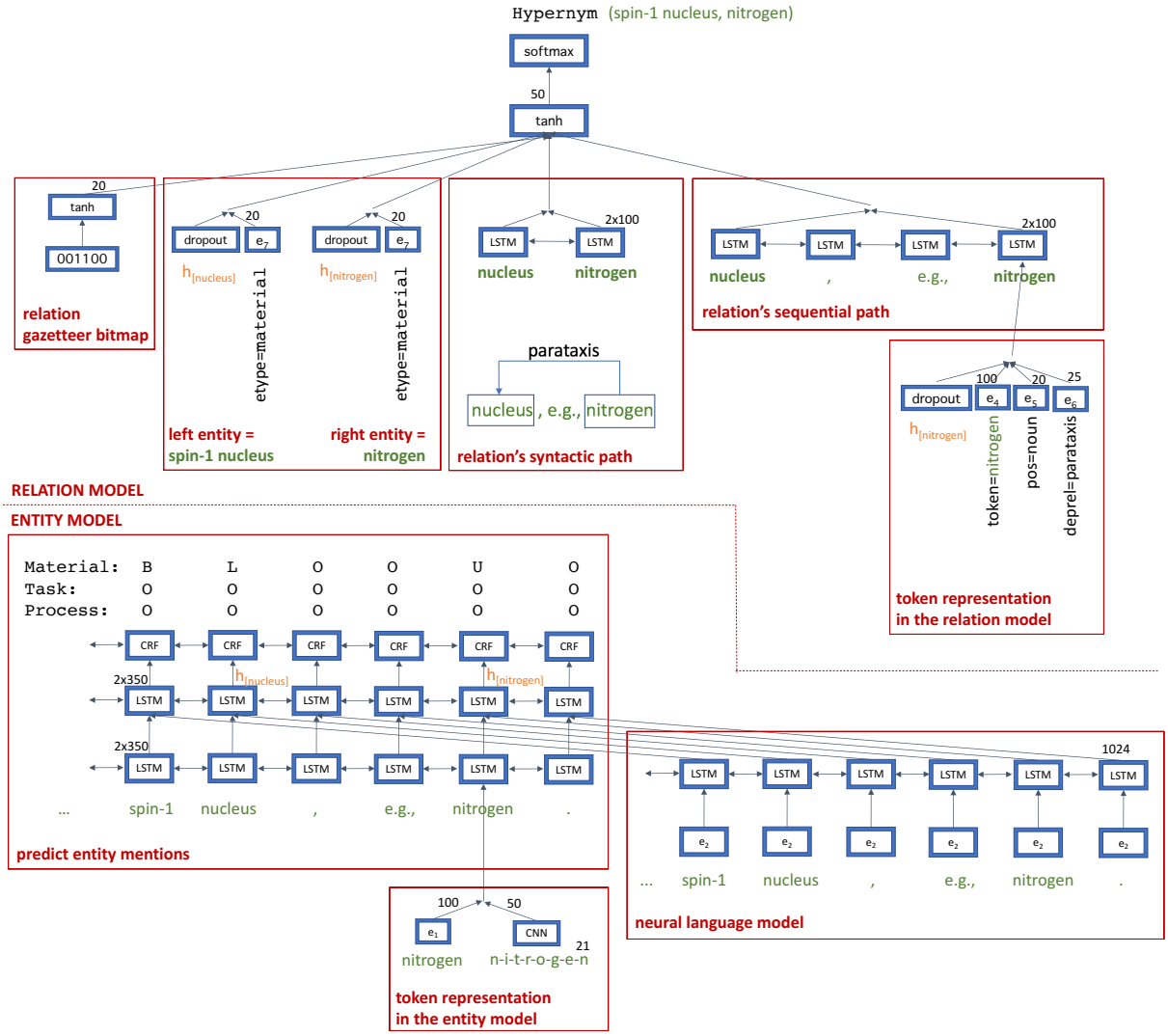


Figure 1: Schematic diagram of the end-to-end model for entity and relation extraction. **Bottom part:** the main components of the entity model (see §2.1 for details). **Top part:** the main components of the relation model (see §2.2 for details).

words, all relations (including `No-relation` are modeled as directional relations. In a post-processing step, `Hypernym-of(e1,e2)` relations are deterministically converted to `Hyponym-of(e2,e1)` relations.

## 2.1 Entity model

In this section, we describe the entity model, which is illustrated in the bottom part of Fig. 1. Given the token sequence $(t_1, t_2, \ldots, t_N)$, this model predicts a sequence of `BILOU` labels for each entity type.

**Token representation.** We first form a token representation, $\mathbf{x}_k = [\mathbf{c}_k; \mathbf{w}_k]$, by concatenating a character based representation $\mathbf{c}_k$ with a token embedding $\mathbf{w}_k$. The character representation, $\mathbf{c}_k$, is parameterized as a convolutional neutral network (CNN) with a filter width of 3 characters.[3] The token embeddings, $\mathbf{w}_k$, are initialized using pre-trained GloVe word embeddings (Pennington et al., 2014) and fine tuned during training. This component is illustrated at the bottom of Fig. 1.

**Neural language model.** In addition to using unlabeled data to learn feature representations of individual word types, we also learn feature representations of words in a particular context using neural language models. Following Józefowicz

[3]The filter width was decided based on preliminary experiments on the development set.

et al. (2016), we feed the output of the embedding layer through one or two layers of LSTMs (Hochreiter and Schmidhuber, 1997) to embed the history $(t_1, t_2, \ldots, t_k)$ into a fixed dimensional vector $\overrightarrow{\mathbf{h}}_k^{LM}$, the *forward LM embedding* of the token at position $k$. While training the parameters of the language model, a softmax layer over words in the vocabulary is used to predict the probability of token $t_{k+1}$.

In order to capture future context in the LM embeddings, we also use a *backward LM embedding* $\overleftarrow{\mathbf{h}}_k^{LM}$ which predicts the previous token $t_{k-1}$ given the following tokens $(t_k, t_{k+1}, \ldots, t_N)$. In our formulation, the forward and backward LMs are independent, without any shared parameters.

In our final system, after pre-training the forward and backward LMs separately, we remove the top layer softmax and concatenate the forward and backward LM embeddings to form bidirectional LM embeddings, i.e., $\mathbf{h}_k^{LM} = [\overrightarrow{\mathbf{h}}_k^{LM}; \overleftarrow{\mathbf{h}}_k^{LM}]$, and use it in the sequence tagging model, which is explained next. This component is illustrated at the bottom right corner of Fig. 1.

**Sequence tagging model.** We employ two layers of bidirectional LSTMs, followed by a conditional random field (CRF) layer to predict entity mentions of each type (see Fig. 1). For each token position, $k$, the hidden state of the first bidirectional LSTM layer is formed by concatenating the hidden states from the forward and backward LSTMs. We also concatenate the LM embeddings $\mathbf{h}^{LM}$ with the output from the first LSTM layer before feeding it into the second LSTM layer.

The output of the second LSTM layer $\mathbf{h}_k$ is used to predict a score for each possible tag. To predict token tags from $\mathbf{h}_k$ we use a dense layer for each entity type and compute the conditional random field (CRF) loss (Lafferty et al., 2001) using the forward-backward algorithm at training time, and using the Viterbi algorithm to find the most likely tag sequence at test time, similar to Collobert et al. (2011). In this layer, we use different parameters for each entity type in order to allow for overlapping entities of different types.

**Entity gazetteer features.** We use lists of scientific terms collected from the web[4] and several topics from freebase,[5] and add them as features in the sequence tagging model. Given an input token sequence, we found all phrases which match one or more of the gazetteers. We encode this information at the token level using a binary feature for each gazetteer (i.e., the number of binary features $= N \times$ the number of gazetteers). The binary features for each token feeds into a dense `tanh` layer which is concatenated to the output of the top LSTM layer. This component was omitted from Fig. 1 to simplify the exposition.

### 2.2 Relation model

In this subsection, we describe the relation model, which is illustrated in the top part of Fig. 1. Given a pair of entity mention spans and their entity types, this model predicts the relation between the two mentions by feeding a context-sensitive representation of the relation into a dense tanh layer, followed by a softmax layer to predict the label.

**Left and right entities.** We represent each of the left and right entities by concatenating an embedding of its entity type with a hidden representation based on the sequence tagging model.

Since many entity mentions consist of multiple tokens, we need to compress their representations to obtain a fixed size encoding. First, we use the dependency tree to identify the syntactic head of the each entity mention. If none of the words in the entity mention qualifies as a direct or indirect head of all other words in the entity mention, we use the last word in the mention, which is often the head in English. For example, 'nucleus' is the head of the entity mention "spin-1 nucleus" in our running example, because there is a direct dependency where 'nucleus' is the head of 'spin-1' with relation type 'compound'. After identifying the head word in an entity mention, we then use the input to the CRF layer from the sequence tagging model at this position, feed it into a dropout layer, and concatenate it with the entity type embedding. This component is second to the left in the top part of Fig. 1.

**Syntactic and sequential path.** We use a bidirectional LSTM layer to encode the shortest path in a dependency tree between the heads of the left and right entities. The input to the LSTM layer at each node in the dependency path concatenates four components: a context-sensitive embedding

[4]We thank Peter Turney for providing this list.

[5]The freebase topics we used are 'dissertation', 'material', 'scholarly work', 'task', 'chemical element', 'competitive space', 'invention', 'drug', 'project_focus', 'literature_subject', 'patent', 'project', 'field_of_study' and 'industry'.

of the word (the input to the CRF layer followed by a dropout layer), a context-insensitive embedding of the word, an embedding of the POS tag, and an embedding of the dependency relation between this node and its direct head.

In addition to encoding the dependency path between the two entities, we also found it useful to encode the sequential path (i.e., the tokens between the two heads in the sentence). We again use a bidirectional LSTM layer and use the same four components to represent the LSTM input at each position in the sequential path. The hidden states at both ends of the syntactic and sequential path are then concatenated (simple concatenation) with the left and right entity representations. This component is illustrated in the right most three boxes in the top part of Fig. 1.

**Relation gazetteer features.** We use two publicly available knowledge bases (Wikipedia and freebase) to derive gazetteer-like features in the relation model. We also encode three features as implicit gazetteers to indicate whether one of the two entities is an acronym, a suffix, or an exact copy of the other. For a given entity pair, we compute input binary features which indicate whether the entity pair matches each gazetteer. The input binary features feed into a dense layer as illustrated in the left-most box in the top part of Fig. 1.

### 2.3 Training

Although we combine the entity model and the relation model at test time, each model is trained separately.[6] Joint training adds some practical complexities, but may result in better results.

**Hyperparameters.** We use performance on development set to guide our selection of hyperparameters. The numbers on the arrows in Fig. 1 correspond to the size of the hidden layers (or the number of filters in the CNN module) in the best performing single model.

We initialize the word embeddings using GloVe (Pennington et al., 2014). For the pre-trained language models, we use the single best forward model from Józefowicz et al. (2016) with two LSTM layers, and a backward LM with one LSTM layer with 2048 hidden units and a 512 dimension projection. These models have test set perplexity of 30.0 and 47.7 on the 1B Word Benchmark (Chelba et al., 2014), respectively.

We use the Adam optimizer (Kingma and Ba, 2014) with learning rate of 0.001 and 0.0003 and gradient norms clipped at 5.0 and 1.0 for the entity and relation models, respectively. We use early stopping by monitoring development set performance.

| **Model** | $F_1$ |
|---|---|
| Our best model without language model | 49.9 |
| Our best model with language model | 54.1 |
| Our 15-model ensemble | 55.2 |

Table 1: Development set entity only $F_1$ comparison.

| **Team** | **End-to-end** | **Entities** | **Relations** |
|---|---|---|---|
| Ours | **0.43** | 0.55 | **0.28** |
| Team_24 | 0.42 | **0.56** | - |
| Team_21 | 0.38 | 0.50 | 0.21 |
| Team_19 | 0.37 | 0.51 | 0.19 |
| Team_14 | 0.33 | 0.47 | 0.20 |

Table 2: Final test set $F_1$ for top five teams in Scenario 1, end-to-end extraction.

**Ensembles.** While tuning the hyperparameters of the models, we save the models with the best results on a development set and use them to create an ensemble. The entity model ensemble averages the label predictions at each position, while the relation model ensemble only predicts a positive relation if 50% of the individual models predict the same relation. Our final submission uses an ensemble of 15 entity models and 8 relation models.

**Differences between Scenario 1 and Scenario 3.** While training the relation model in Scenario 1, we use both the gold entities and the entities predicted by the entity model to generate candidate relations. In Scenario 3, only gold entities are used to generate candidate relations for training.

To make predictions on the test set, only entities predicted by the entity model were used to generate candidate relations in Scenario 1. In Scenario 3, only gold entities were used to generate candidate relations for the test set.

[6] All parameters (including word and character embeddings) in each model are trained. None of the parameters are fixed.

## 3 Results

We compare three variants of our entity extraction model on the development set in Table 1. Adding a bidirectional LM to our sequence tagging model amounts to an improvement of 4.2 $F_1$, while using an ensemble of 15 models amounts to a further improvement of 1.1 $F_1$.

In Scenario 1, our submission ranked first with $F_1$ of 0.43%, second in the entity only subtask (0.55% $F_1$) and first in the relation only subtask for end-to-end extraction (0.28% $F_1$), as shown in Table 2. In Scenario 3, our submission ranked second with 0.54% $F_1$.

## Acknowledgements

We thank the shared task organizers for their efforts and the anonymous reviewers for their helpful comments.

## References

Isabelle Augenstein, Mrinal Kanti Das, Sebastian Riedel, Lakshmi Nair Vikraman, and Andrew McCallum. 2017. SemEval 2017 Task 10: ScienceIE - Extracting Keyphrases and Relations from Scientific Publications. In *Proceedings of the International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Vancouver, Canada.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2014. One billion word benchmark for measuring progress in statistical language modeling. *CoRR* abs/1312.3005.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. In *JMLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9:1735–1780.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR* abs/1602.02410.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.