Nested Named Entity Recognition Revisited

Arzoo Katiyar and **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY, 14853, USA
{arzoo, cardie}@cs.cornell.edu

Abstract

We propose a novel recurrent neural network-based approach to simultaneously handle nested named entity recognition and nested entity mention detection. The model learns a hypergraph representation for nested entities using features extracted from a recurrent neural network. In evaluations on three standard data sets, we show that our approach significantly outperforms existing state-of-the-art methods, which are feature-based. The approach is also efficient: it operates linearly in the number of tokens and the number of possible output labels at any token. Finally, we present an extension of our model that jointly learns the head of each entity mention.

1 Introduction

Named entity recognition (or named entity detection) is the task of identifying text spans associated with proper names and classifying them according to their semantic class such as person, organization, etc. It is related to the task of *mention detection* (or entity mention recognition) in which text spans referring to named, nominal or prominal entities are identified and classified according to their semantic class (Florian et al., 2004). Both named entity recognition and entity mention detection are fundamental components in information extraction systems: several downstream tasks such as relation extraction (Mintz et al., 2009), coreference resolution (Chang et al., 2013) and fine-grained opinion mining (Choi et al., 2006) rely on both.

Many approaches have been successfully employed for the tasks of named entity recognition and mention detection, including linear-chain conditional random fields (Lafferty et al., 2001) and semi-Markov conditional random fields (Sarawagi and Cohen, 2005). However, most such methods suffer from an inability to handle *nested* named entities, *nested* entity mentions, or both. As a result, the downstream tasks necessarily ignore these nested entities along with any semantic relations among them. Consider, for example, the excerpts below:

(S1) Employing the [EBV - transformed [human B cell line]$_{\text{CELL\_LINE}}$]$_{\text{CELL\_LINE}}$ SKW6.4, we demonstrate . . .

(S2) . . . [the burial site of [Sheikh Abbad]$_{\text{PERSON}}$]$_{\text{LOCATION}}$ is located . . .

S1 shows a nested named entity from the GENIA dataset (Ohta et al., 2002): "human B cell line" and "EBV - transformed human B cell line" are both considered named entities of type CELL_LINE where the former is embedded inside the latter. S2, derived from the ACE corpora[1], shows a PERSON named entity ("Sheikh Abbad") nested in an entity mention of type LOCATION ("the burial site of Sheikh Abbad"). Most existing methods for named entity recognition and entity mention detection would miss the nested entity in each sentence.

Unfortunately, nested entities can be fairly common: 17% of the entities in the GENIA corpus are embedded within another entity; in the ACE corpora, 30% of sentences contain nested named entities or entity mentions, thus warranting the development of efficient models to effectively handle these linguistic phenomena.

Feature-based methods are the most common among those proposed for handling nested named entity and entity mention recognition. Alex et al.

[1] https://catalog.ldc.upenn.edu/LDC2005T09 (ACE2004) and https://catalog.ldc.upenn.edu/LDC2006T06 (ACE2005)

Proceedings of NAACL-HLT 2018, pages 861–871
New Orleans, Louisiana, June 1 - 6, 2018.

(2007), for example, proposed a cascaded CRF model but it does not identify nested named entities of the same type. Finkel and Manning (2009) proposed building a constituency parser with constituents for each named entity in a sentence. Their approach is expensive, i.e., time complexity is cubic in the number of words in the sentence. Lu and Roth (2015) later proposed a mention hypergraph model for nested entity detection with linear time complexity. And recently, Muis and Lu (2017) introduced a multigraph representation based on mention separators for this task. All of these models depend on manually crafted features. In addition, they cannot be directly applied to extend current state-of-the-art recurrent neural network-based models — for flat named entity recognition (Lample et al., 2016) or the joint extraction of entities and relations (Katiyar and Cardie, 2016) — to handle nested entities.

In this paper, we propose a recurrent neural network-based model for nested named entity and nested entity mention recognition. We present a modification to the standard LSTM-based sequence labeling model (Sutskever et al., 2014) that handles both problems and operates linearly in the number of tokens and the number of possible output labels at any token. The proposed neural network approach additionally jointly models entity mention *head*[2] information, a subtask found to be useful for many information extraction applications. Our model significantly outperforms the previously mentioned hypergraph model of Lu and Roth (2015) and Muis and Lu (2017) on entity mention recognition for the ACE2004 and ACE2005 corpora. It also outperforms their model on *joint* extraction of nested entity mentions and their heads. Finally, we evaluate our approach on nested named entity recognition using the GENIA dataset and show that our model outperforms the previous state-of-the-art parser-based approach of Finkel and Manning (2009).

2 Related Work

Several methods have been proposed for named entity recognition in the existing literature as summarized by Nadeau and Sekine (2007) in their survey paper. Early techniques in the supervised domain have been based on hidden markov models (e.g., Zhou and Su (2002)) or, later, conditional random fields (CRFs) (e.g., McDonald and Pereira (2005)).

Many fewer approaches, however, have addressed the problem of nested entities. Alex et al. (2007) presented several techniques based on CRFs for nested named entity recognition for the GENIA dataset. They obtained their best results from a cascaded approach, where they applied CRFs in a specific order on the entity types, such that each CRF utilizes the output derived from previous CRFs. Their approach could not identify nested entities of the same type. Finkel and Manning (2009) proposed a CRF-based constituency parser for nested named entities such that each named entity is a constituent in the parse tree. Their model achieved state-of-the-art results on the GENIA dataset. However, the time complexity of their model is $O(n^3)$, where n is the number of tokens in the sentence, making inference slow. As a result, we do not adopt their parse tree-based representation of nested entities and propose instead a linear time directed hypergraph-based model similar to that of Lu and Roth (2015). Directed hypergraphs were also introduced for parsing by Klein and Manning (2001).

While most previous efforts for nested entity recognition were limited to named entities, Lu and Roth (2015) addressed the problem of nested entity mention detection where mentions can either be named, nominal or pronominal. Their hypergraph-based approach is able to represent the potentially exponentially many combinations of nested mentions of different types. They adopted a CRF-like log-linear approach to learn these mention hypergraphs and employed several hand-crafted features defined over the input sentence and the output hypergraph structure. Our approach also learns a similar hypergraph representation with differences in the types of nodes and edges in the hypergraph. It does not depend on any manually crafted features. Also, our model learns the hypergraph greedily and significantly outperforms their approach.

Recently, Muis and Lu (2017) introduced the notion of mention separators for nested entity mention detection. In contrast to the hypergraph representation that we and Lu and Roth (2015) adopt, they learn a multigraph representation and are able to perform exact inference on their structure. It is an interesting orthogonal possible approach for nested entity mention detection. How-

[2]This involves identifying the headword of a named entity or entity mention.

ever, we will show that our model also outperforms their approach on all tasks.

Recently, recurrent neural networks (RNNs) have been widely applied to several sequence labeling tasks achieving state-of-the-art results. Lample et al. (2016) proposed neural models based on long short term memory networks (LSTMs) and CRFs for named entity recognition and another transition-based approach inspired by shift-reduce parsers. Both models achieve performance comparable to a state-of-the-art model (Luo et al., 2015), but neither handles nested named entities.

3 Encoding Scheme

Figure 1 shows the desired sequence tagging output for each of three overlapping PER entities ("his", "his fellow pilot" and "his fellow pilot David Williams") according to the standard BILOU tag scheme. Our approach relies on the fact that we can (1) represent these three tag sequences in the single hypergraph structure of Figure 2 and then (2) design an LSTM-based neural network that produces the correct nested entity hypergraph for a given input sentence. In the paragraphs just below we provide a general description of hypergraphs and our task-specific use of them. Sections 3.1 and 3.2 describe the hypergraph construction process; Section 4 presents the LSTM-based sequence tagging method for automating hypergraph construction.

We express our structured prediction problem such that it corresponds to building a hypergraph that encodes the token-level gold labels for all entities in the input sentence.[3] In particular, we represent the problem as a directed hypergraph. For those new to this formalism, directed hypergraphs are very much like standard directed graphs except that nodes are connected by hyperarcs that connect a *set* of tail nodes to a *set* of head nodes. To better explain our desired output structure, we further distinguish between two types of hyperarcs — *normal edges* (or arcs) that connect a single tail node to a single head node, and *hyperarcs* that contain more than one node either as the head or as the tail. The former are shown as straight lines in Figure 2; the latter as curved edges.

In our encoding of nested entities, a hyperarc is introduced when two or more entity mentions requiring different label types are present at the same position. In Figure 2, for example, the nodes "O" (corresponding to the input token "that") and the nodes "U_PER" and "B_PER" (corresponding to the input token "his") are connected by a hyperarc because three entity mentions start at this time step from the tail "O" node (two of which share the "B_PER" tag).[4]

3.1 Hypergraph Construction

Let us first discuss how the problem of nested entity recognition can be expressed as finding a hypergraph. Our goal is to represent the BILOU tag sequences associated with "his", "his fellow pilot" and "his fellow pilot David Williams" as the single hypergraph structure of Figure 2. This is accomplished by collapsing the shared states (labels) in the output entity label sequences into a single state as shown in Figure 2: e.g., the three "O" labels for "that" become a single "O"; the two "B_PER" labels at "his" are collapsed into one "B_PER" node that joins "U_PER", the latter of which represents the entity mention "his". Thus at any time step, the representation size is bounded by the number of possible output states instead of the potentially exponential number of output sequences. We then also adjust the directed edges such that they have the same type of head node and the same type of tail node as before in Figure 1.

If we look closely at Figure 2 then we realise that there is an extra "O" node in the hypergraph corresponding to the token "his" which did not appear in any entity output sequence in Figure 1: in our task-specific hypergraph construction we make sure that there is an "O" node at every timestep to model the possibility of beginning of a new entity. The need for this will become more clear in Section 4.

Note that the hypergraph representation of our model is similar to Lu and Roth (2015). Also, the expressiveness of our model is exactly the same as Lu and Roth (2015); Muis and Lu (2017). The major difference in the two approaches is in learning.

[3] We note that the complete hypergraph for the example in Figure 1 would include nodes for all possible label types at each timestep and all possible hyperarcs between them. In this work, however, we only greedily build a sub-hypergraph for the gold labels when training.

[4] In contrast, note that the nodes "L_PER" and "O" corresponding to the input token "pilot" and the node "O" corresponding to the token "David" are connected by normal edges. Hence, our hypergraph structure contains only one special kind of hyperarc which connects a single tail node to multiple head nodes. We do not have hyperarcs that connect multiple tail nodes to a single head node.

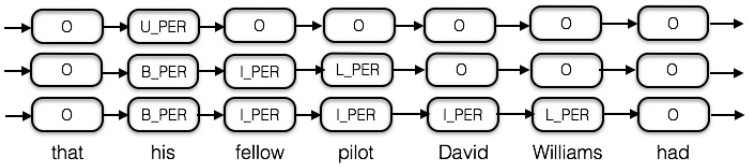

Figure 1: Nested entity mentions in an unfolded hypergraph. Each row corresponds to an entity mention sequence using the well known B_ (beginning of mention), I_ (inside a mention), L_ (last token of an entity mention), O (outside any entity mention), U_ (a single-token entity mention) tagging scheme.

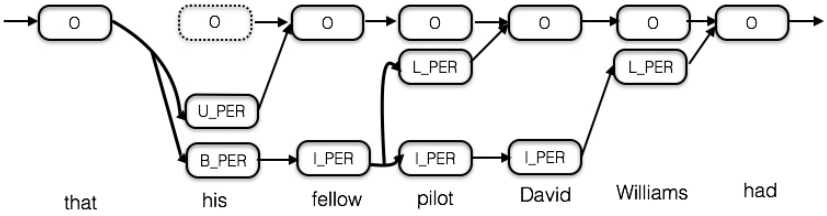

Figure 2: Directed hypergraph constructed for the example shown in Figure 1. Curved edges represent hyperarcs and straight edges are normal edges.

3.2 Edge Probability

In this section, we discuss our assignment of probabilities to all the possible edges from a tail node which helps in the greedy construction of the hypergraph. Thus at any timestep t, let g_{t-1} be the tail node and x be the current word of the sentence; then we model probability distribution over all the possible types of head nodes (different output tag types) conditioned on the tail node and the current word token. In our work we use hidden representations learned from an LSTM model as features to learn these probability distributions using a cross-entropy objective.

It is important to note that there are two types of directed edges in this hypergraph – simple edges for which there is only one head node for every tail node which can be learned as in a traditional sequence labeling task, or hyperarcs that connect more than one head node to a tail node. We learn the set of head nodes connected to a tail node by expressing it as a multi-label learning problem as described in Section 5.

3.3 Extracting Entity Mentions

As described in Section 3.2, we can assign probabilities to the different types of edges in the hypergraph and at the time of decoding we choose for each token the (normal) edge(s) with maximum probability and the hyperarcs with probability above a predefined threshold. Thus, we can extract edges at the time of decoding. Ultimately, however, we are interested in extracting nested entities from the hypergraph. For this, we construct an adjacency matrix from the edges discovered and perform depth-first search from the sentence-initial token to discover the entity mentions. This is described in detail in Section 5.1.

4 Method

We use a standard LSTM-based sequence labeling model to learn the nested entity hypergraph structure for an input sentence. Figure 3 shows part of the network structure. It is a standard bi-directional LSTM network except for a difference in the top hidden layer. When computing the representation of the top hidden layer L at any time step t, in addition to making use of the hidden unit representation from the previous time step $t-1$ and hidden unit representation from the preceding layer $L-1$, we also input the label embedding of the gold labels from the previous time step. For the token "fellow" in Figure 3, for example, we compute three different top hidden layer representations, conditioned respectively on the three labels "U_PER", "B_PER" and "O" from the previous time step $t-1$. Thus, we can model complex interactions between the input and the output. Before passing the learned hidden representation to the next time step, we average the three different top hidden layer representations. In this manner, we can model the interactions between the different overlapping labels and also it is computation-

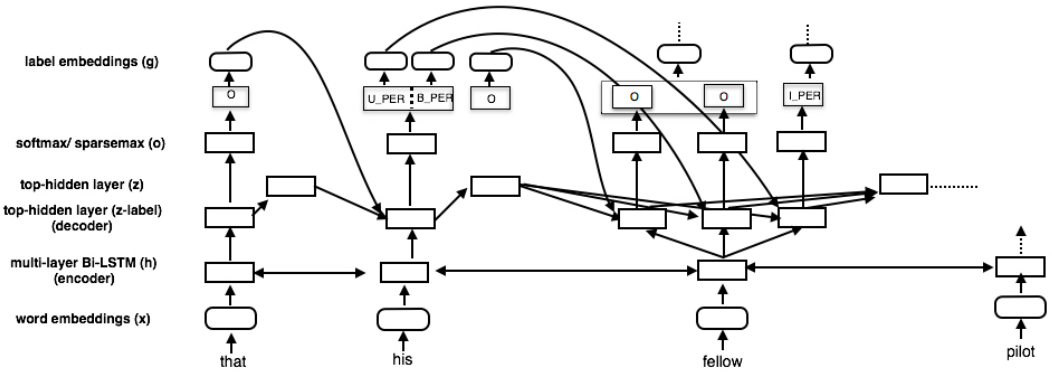

Figure 3: Dynamically computed network structure based on bi-LSTMs for nested entity mention extraction. We show part of the structure for the entity mentions in the running example in Figure 1.

ally less expensive than storing the hidden layer representations for each label sequence.

4.1 Multi-layer Bi-LSTM

We use a multi-layer bi-directional LSTM encoder, for its strength in capturing long-range dependencies between tokens, a useful property for information extraction tasks.

Using LSTMs, we can compute the hidden state $\overrightarrow{\boldsymbol{h}}_t$ in the forward direction and $\overleftarrow{\boldsymbol{h}}_t$ in the backward direction for every token, and use a linear combination of them as the token representation:

$$\overrightarrow{\boldsymbol{h}}_t^{(l)} = \text{LSTM}(\boldsymbol{x}_t, \overrightarrow{\boldsymbol{h}}_{t-1})$$
$$\overleftarrow{\boldsymbol{h}}_t^{(l)} = \text{LSTM}(\boldsymbol{x}_t, \overleftarrow{\boldsymbol{h}}_{t+1})$$
$$\boldsymbol{z}_t^{(l)} = \overrightarrow{\boldsymbol{V}}\overrightarrow{\boldsymbol{h}}_t^{(l)} + \overleftarrow{\boldsymbol{V}}\overleftarrow{\boldsymbol{h}}_t^{(l)} + \boldsymbol{b}^l$$

4.2 Top Hidden Layer

At the top hidden layer, we have a decoder-style model, with a crucial twist to accommodate the hypergraph structure, which may have multiple gold labels at the previous step. At each token t and *for each* gold label at the previous step g_{t-1}^k, our network takes the hidden representation from the previous layer $\boldsymbol{z}_t^{(L-1)}$, the hidden decoder state $\boldsymbol{h}_{t-1}^{(L)}$, as well as the gold label embedding $\boldsymbol{g}_{t-1}^k$ from the previous time step, and computes:

$$\boldsymbol{h}_t^{(L),k} = \text{LSTM}(\boldsymbol{z}_t^{(L-1)}, \boldsymbol{h}_{t-1}^{(L)}, \boldsymbol{g}_{t-1}^k)$$

Unlike the encoder LSTM, this decoder LSTM is single-directional and bifurcates when multiple gold labels are present. We use the decoder hidden states $\boldsymbol{h}_t^{(L),k}$ in the output layer for prediction, as explained in Section 4.3. However, before passing the hidden representation to the next time step we average $\boldsymbol{h}_t^{(L),k}$ over all the gold labels k:

$$\boldsymbol{h}_t^{(L)} = \frac{1}{|G_{t-1}|}\sum_k \boldsymbol{h}_t^{(L),k}$$

Thus, $\boldsymbol{h}_t^{(L)}$ summarizes the information for all the gold labels from the previous time step.

4.3 Entity Extraction

For each token t and previous gold label g_{t-1}^k, we use the decoder state $\boldsymbol{h}_t^{(L),k}$ to predict a *probability distribution* over the possible candidate labels using a linear layer followed by a normalizing transform (illustrated below with softmax). The outputs can be interpreted as conditional probabilities for the next label given the current gold label:

$$\boldsymbol{o}_t^k = \boldsymbol{U}\boldsymbol{h}_t^{(L),k} + \boldsymbol{b}$$
$$\hat{\boldsymbol{e}}_t^k = \text{softmax}(\boldsymbol{o}_t^k)$$
$$p(y_t = c | y_{t-1} = g_{t-1}^k) = (\boldsymbol{e}_t^k)_c$$

Special care is required, however, since the desired output has hyperarcs. As shown in Figure 2, there is an hyperarc between "I_PER" corresponding to the token "fellow" and the label set "L_PER" and "I_PER" corresponding to the token "pilot". Thus, in our network structure conditioned on the previous label "I_PER" in this case, we would like to predict both "L_PER" and "I_PER" as the next labels. To accommodate this, we use a multi-label training objective, as described in Section 5.

5 Training

We train our model using two different multi-label learning objectives. The idea is to represent the

	ACE2004			ACE2005		
Method	P	R	F1	P	R	F1
MH-F (Lu and Roth, 2015)	70.0	59.2	63.8	70.0	56.9	62.8
Muis and Lu (2017)	72.7	58.0	64.5	69.1	58.1	63.1
LSTM-flat	70.3	48.4	57.3	62.4	49.4	55.1
LSTM-output_layer	72.0	63.3	67.4	66.3	68.2	67.2
Our model (softmax)	72.2	65.2	68.5	70.1	67.9	69.0
Our model (sparsemax)	**73.6**	**71.8**	**72.7**	**70.6**	**70.4**	**70.5**

Table 1: Performance on ACE2004 and ACE2005 test set on mention extraction and classification.

gold labels as a distribution over all possible labels, encoded as a vector $\boldsymbol{e}$. Hence, for simple edges, the distribution has a probability of 1 for the unique gold label ($\boldsymbol{e}_g = 1$), and 0 everywhere else. For hyperarcs, we distribute the probability mass uniformly over all the gold labels in the gold label set ($\boldsymbol{e}_g^k = \frac{1}{|G|}$ for all k). Thus, for the example described earlier in Section 4.3, both the labels "L_PER" and "I_PER" receive a probability of 0.5 in the gold label distribution $\boldsymbol{e}_t^k$, conditioned on the label "I_PER" from the previous time step.

Softmax. Our first training method uses softmax to estimate the predicted probabilities, and the KL-divergence multi-label loss between the true distribution $\boldsymbol{e}_t^k$ and the predicted distribution $\hat{\boldsymbol{e}}_t^k = \text{softmax}(\boldsymbol{o}_t^k)$:

$$\ell_{t(\text{softmax})}^k = -\sum_c \left(\boldsymbol{e}_t^k\right)_c \log\left(\hat{\boldsymbol{e}}_t^k\right)_c$$

Sparsemax. Our second training method makes use of *sparsemax*, recently introduced by Martins and Astudillo (2016) as a sparse drop-in replacement to softmax, as well as a loss function. Unlike softmax, which always outputs a nonzero probability for any output, sparsemax outputs zero probability for most of the unlikely classes, leading to good empirical results on multi-label tasks. For our problem, there are only a few nested entities at any timestep in the gold labels thus using a training objective that learns a sparse distribution is more appropriate. Sparsemax can be used to filter part of the output space as in the case for multi-label problems thus leaving non-zero probability on the desired output labels.

Formally, sparsemax returns the euclidean projection of its input $\boldsymbol{o}$ onto the probability simplex:

$$\hat{\boldsymbol{e}} = \text{sparsemax}(\boldsymbol{o}) := \underset{\hat{\boldsymbol{e}} \in \Delta}{\text{argmin}} \|\boldsymbol{o} - \hat{\boldsymbol{e}}\|^2$$

The corresponding loss, a sparse version of the KL divergence, is (up to a constant):

$$\ell_{t(\text{sparsemax})}^k = -2\boldsymbol{e}_t^{k\top}\boldsymbol{o}_t^k + \sum_{c:(\hat{\boldsymbol{e}}_t^k)_c \neq 0} \left((\boldsymbol{o}_t^k)_c^2 - \tau^2\right)$$

This function is convex and differentiable, and the quantity τ is a biproduct of the simplex projection, as described in Martins and Astudillo (2016).

For either choice of probability estimation, the total loss of a training sample is the sum of losses for each token and for each previous gold label:

$$\mathcal{L} = \sum_t \sum_{k \in G_{t-1}} \ell_t^k.$$

5.1 Decoding

At the time of inference, we greedily decode our hypergraph from left-to-right to find the most likely sub-hypergraph. During training, at each timestep the most likely label set is learned conditioned on a gold label from the previous timestep. However, gold labels are not available at test time. Thus, we use the predicted labels from the previous time step as an input to the current time step to find the most likely label set. We use a hard threshold T to determine the predicted label set $P_t^k = \{c : \left(\hat{\boldsymbol{e}}_t^k\right)_c > T\}$

We can get the most likely label set P_t^c for any predicted label at the previous time step $c \in P_{t-1}^k$ using the above decoding strategy. We now combine these inferences to find the most likely entity mention sequences. We construct an adjacency matrix $\boldsymbol{A}$ for each time step, such that $\boldsymbol{A}[\hat{\boldsymbol{e}}_{t-1}^c][\hat{\boldsymbol{e}}_t^k]$ += 1 for every c in the predicted label set P_t^k at timestep t conditioned on $\hat{\boldsymbol{e}}_t^k$ and for every k in predicted labels P_{t-1} at time step $t-1$. This can be viewed as a directed hypergraph with several connected components. We then perform a depth-first search on this directed hypergraph to find all the entity mentions in the sentence.

5.2 Modeling Entity Heads for ACE datasets

The ACE datasets also have annotations for mention heads along with the entity mentions. For example, a sentence with the entity mention "the U.S. embassy" also contains an annotation for its head word which is "embassy" in this case. Thus, we modify our model to also extract the head of the entity mentions for ACE dataset. We jointly model the entity mentions and their heads. To do this, we propose a simple extension to our model by only changing the output label sequence. We introduce new labels starting with "H" to indicate that the current token in the entity mention is part of its head. Thus, we only change the output label sequence for the entity mentions to include the head label: We train with the label sequence "B_ORG I_ORG H_ORG" instead of "B_ORG I_ORG L_ORG". Also, for all our entity sequences we predict the "O" tag at the end, hence we can still extract the entity mentions. At decoding time, we output the sequence of words with the "H" tag as the head words for a mention.

6 Experiments

We evaluate our model on two tasks – nested entity mention detection for the ACE corpora and nested named entity recognition for the GENIA dataset.

6.1 ACE Experiments

6.1.1 Data

We perform experiments on the English section of the ACE2004 and ACE2005 corpora. There are 7 main entity types — Person (PER), Organization (ORG), Geographical Entities (GPE), Location (LOC), Facility (FAC), Weapon (WEA) and Vehicle (VEH). For each entity type, there are annotations for the entity mention and mention heads.

6.1.2 Evaluation Metrics

We use a strict evaluation metric similar to Lu and Roth (2015): an entity mention is considered correct if both the mention span and the mention type are exactly correct. Similarly, for the task of joint extraction of entity mentions and mention heads, the mention span, head span and the entity type should all exactly match the gold label.

6.1.3 Baselines and Previous Models

We compare our model with the feature-based model (MH-F) on hypergraph structure (Lu and Roth, 2015) on both entity mention detection as well as the joint mention and mention heads extraction. We also compare with Muis and Lu (2017) on entity mention detection only as their model cannot detect head phrases of the entity mentions. Lu and Roth (2015) compare their approach with CRF-based approaches such as a linear-chain CRF, semi-markov CRF and a cascaded approach (Alex et al., 2007) and show that their model outperforms them. Hence, we do not include those results in our paper.

We also implement several LSTM-based baselines for comparison. Our first baseline is a standard sequence labeling LSTM model (LSTM-flat). A sequence model is not capable of handling the nested mentions, so we remove the embedded entity mention and keep the mention longer in length. Our second baseline is a hypergraph model (LSTM-output_layer) except that the dependencies are only modeled at the output layer and hence there are no connections to the top-hidden layer from the label embeddings from the previous timestep; instead, these connections are limited to the output layer.

6.1.4 Hyperparameters and Training Details

We use Adadelta (Zeiler, 2012) for training our models. We initialize our word vectors with 300-dimensional word2vec (Mikolov et al., 2013) word embeddings. These word embeddings are tuned during training. We regularize our network using dropout (Srivastava et al., 2014), with the dropout rate tuned on the development set. There are 3 hidden layers in our network and the dimensionality of hidden units is 100 in all our experiments. And we set the threshold T as 0.3.

6.1.5 Results

We show the performance of our approaches in Table 1 compared to the previous state-of-the-art system (Lu and Roth, 2015; Muis and Lu, 2017) on both the ACE2004 and ACE2005 datasets. We find that our LSTM-flat baseline that ignores embedded entity mentions during training performs worse than Lu and Roth (2015); however, our other neural network-based approaches all outperform the previous feature-based approach. Among the neural network-based models, we find that our models that construct a hypergraph perform better than the LSTM-flat models. Also, our approach that models dependencies between the input and the output by passing the prediction from the pre-

	ACE2004			ACE2005		
Method	P	R	F1	P	R	F1
MH-F (Lu and Roth, 2015)	74.4	50.0	59.8	63.4	53.8	58.3
Our model(softmax)	68.2	60.5	64.2	67.5	62.3	64.8
Our model(sparsemax)	**72.3**	**66.8**	**69.7**	**70.6**	**69.8**	**70.2**

Table 2: Performance on ACE2004 and ACE2005 test set on joint entity mention and its head prediction. Muis and Lu (2017) do not predict head of the nested entity mentions.

vious timestep as shown in Figure 3 performs better than the LSTM-output_layer model which only models dependencies at the output layer. Also, as expected, the sparsemax method that produces a sparse probability distribution performs better than the softmax approach for modeling hyperedges. In summary, our sparsemax model is the best performing model.

Joint Modeling of Heads We report the performance of our best performing models on the joint modeling of entity mentions and its head in Table 2. We show that our sparsemax model is still the best performing model. We also find that as the total number of possible labels at any timestep increases because of the way we implemented the entity heads, the gains that we get after incorporating sparsemax are significantly higher compared to the results shown in Table 1.

6.2 GENIA Experiments

6.2.1 Data

We also evaluate our model on the GENIA dataset (Ohta et al., 2002) for nested named entity recognition. We follow the same dataset split as Finkel and Manning (2009); Lu and Roth (2015); Muis and Lu (2017). Thus, the first 90% of the sentences were used in training and the remaining 10% were used for evaluation. We also consider five entity types – DNA, RNA, protein, cell line and cell type.

6.2.2 Baselines and Previous Models

We compare our model with Finkel and Manning (2009) based on a constituency CRF-based parser and the mention hypergraph model by Lu and Roth (2015) and a recent multigraph model by Muis and Lu (2017).

6.2.3 Results

Table 3 shows the performance of our different models compared to the previous models. Interestingly, our LSTM-flat model outperforms Lu and Roth (2015). We suspect that it is because we use pretrained word embeddings[5] trained on PubMed data (Pyysalo et al., 2013) whereas Lu and Roth (2015) did not have access to them. We again find that our neural network model outperforms the previous state-of-the-art (Finkel and Manning, 2009; Muis and Lu, 2017) system. However, we see that both softmax and sparsemax models perform comparably on this dataset.

Method	P	R	F1
Finkel and Manning (2009)	75.4	65.9	70.3
MH-F (Lu and Roth, 2015)	72.5	65.2	68.7
Muis and Lu (2017)	75.4	66.8	70.8
LSTM-flat	75.5	63.5	68.9
LSTM-output_layer	78.4	67.9	72.8
Our model (softmax)	76.7	**71.1**	**73.8**
Our model (sparsemax)	**79.8**	68.2	73.6

Table 3: Performance on the GENIA dataset on nested named entity recognition.

7 Error Analysis

Consistent with existing results on the joint modeling of related tasks in NLP, we find that joint modeling of heads and their entity mentions leads to an increase in F-score by 1pt (i.e., 71.4 for the sparsemax model on the ACE2005 dataset) on the performance of the entity mentions. The precision on extracting entity mentions is 72.1 (vs. 70.6 in Table 1) for our sparsemax model for the ACE2005 dataset.

Example S1 below compares the output from a softmax vs. a sparsemax model on the joint modeling of an entity mention and its head on the ACE2005 dataset. Gold-standard annotations are shown in *red*.

(S1) [[[They]]]$_{\text{PERSON}}$ don't abandon [[[[[their]]]$_{\text{PERSON}}$ patients]]$_{\text{PERSON}}$, except

[5] Word vectors trained on PubMed data are available at http://bio.nlplab.org/#source-data.

for the high premiums of a few specialities?

Based on the gold standard, the models are required to extract "their" — an entity mention of type PER as well as its head — and "their patients", which overlaps with the previous entity mention "their" and has the head word "patients". This means that the models are required to predict a hyperedge from "O" to "H_PER; B_PER". We find that the softmax model shown in *blue* can only predict the entity mention "their" omitting completely the entity mention "their patients" whereas the sparsemax model shown in *green* can predict both nested entities. Overall then, sparsemax seems to allow the modeling of hyperedges more efficiently compared to the softmax model and performance gains are due to extracting more nested entities with the help of sparsemax model.

7.1 Limitations and Future Directions

We also manually scanned the test set predictions on ACE dataset for our sparsemax model to understand its current limitations.

Document Context. Given the following sentence

(S2) [They]$_{\text{VEHICLE}}$ roar, [they]$_{\text{VEHICLE}}$ screech.

the sparsemax model predicts both entity mentions of "they" as PER entity type. Only if the previous sentence in the corpus is accessible — "And if you ride inside that tank, it is like riding in the bowels of a dragon" — can we understand that "they" in S2 refers to the tank and hence is a VEH. Thus, our model can be improved by providing additional context for each sentence rather than making predictions on each sentence in the corpus independently.

Pronominal Entity Mention (It). Next, consider examples S3 and S4:

(S3) [It]$_{\text{FACILITY}}$ also seemed to be [some kind of monitoring station]$_{\text{FACILITY}}$.

(S4) It does not matter to [these people]$_{\text{PERSON}}$ that crime has skyrocketed . . .

In the example sentences, "It" refers to a facility and an event, respectively. Our model does not distinguish between the two cases and always predicts the token "It" as a non-entity. We found this true for all occurrences of the token "It" in our test set. The incorporation of coreference information can potentially overcome this limitation.

Inconsistency in Gold-standard Annotations. We also identified potential inconsistencies in the gold-standard annotations.

(S5) . . . results may affect what happens to [both of these teams]$_{\text{ORG}}$, but in just . . .

For S5, the gold-standard annotation for "both of these teams" is an ORG entity mention with the token "teams" as its head word. Our sparsemax model identifies the entity mention correctly but instead predicts the token "both" as the head. It also identifies "these teams" as another nested entity mention with the head word "teams". In contrast, however, we also found entity mentions such as "all of the victims that get a little money" for which the gold-standard has "all" annotated as its head and another nested mention "the victims that get a little money" with "victims" as the head. We recognize this as an inconsistency in the gold-standard annotation.

8 Conclusion and Future Work

In this paper, we present a novel recurrent network-based model for nested named entity recognition and nested entity mention detection. We propose a hypergraph representation for this problem and learn the structure using an LSTM network in a greedy manner. We show that our model significantly outperforms a feature based mention hypergraph model (Lu and Roth, 2015) and a recent multigraph model (Muis and Lu, 2017) on the ACE dataset. Our model also outperforms the constituency parser-based approach of Finkel and Manning (2009) on the GENIA dataset.

In future work, it would be interesting to learn global dependencies between the output labels for such a hypergraph structure and training the model globally. We can also experiment with different representations such as the one in Finkel and Manning (2009) and use the recent advances in neural network approaches (Vinyals et al., 2015) to learn the constituency parse tree efficiently.

Acknowledgments

We thank Wei Lu for help with the datasets. We also thank Jack Hessel, Vlad Niculae and the reviewers for their helpful comments and feedback. This work was supported in part by NSF grant SES-1741441 and DARPA DEFT Grant FA8750-13-2-0015. The views and conclusions contained herein are those of the authors and should not be

interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, DARPA or the U.S. Government.

References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, BioNLP '07, pages 65–72. http://dl.acm.org/citation.cfm?id=1572392.1572404.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 601–612. http://www.aclweb.org/anthology/D13-1057.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sydney, Australia, pages 431–439. http://www.aclweb.org/anthology/W/W06/W06-1651.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, pages 141–150. http://www.aclweb.org/anthology/D/D09/D09-1015.

R Florian, H Hassan, A Ittycheriah, H Jing, N Kambhatla, X Luo, N Nicolov, and S Roukos. 2004. A statistical model for multilingual entity detection and tracking. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*. Association for Computational Linguistics, Boston, Massachusetts, USA, pages 1–8.

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1087.pdf.

Dan Klein and Christopher D. Manning. 2001. Parsing and hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT-2001), 17-19 October 2001, Beijing, China*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '01, pages 282–289. http://dl.acm.org/citation.cfm?id=645530.655813.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 260–270. http://www.aclweb.org/anthology/N16-1030.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 857–867. http://aclweb.org/anthology/D15-1102.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 879–888. https://aclweb.org/anthology/D/D15/D15-1104.

André F. T. Martins and Ramón F. Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. JMLR.org, ICML'16, pages 1614–1623. http://dl.acm.org/citation.cfm?id=3045390.3045561.

Ryan McDonald and Fernando Pereira. 2005. Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics* 6(1):S6. https://doi.org/10.1186/1471-2105-6-S1-S6.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, Suntec, Singapore, pages 1003–1011. http://www.aclweb.org/anthology/P/P09/P09-1113.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2598–2608. https://www.aclweb.org/anthology/D17-1275.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes* 30(1):3–26. Publisher: John Benjamins Publishing Company.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The genia corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the Second International Conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, HLT '02, pages 82–86. http://dl.acm.org/citation.cfm?id=1289189.1289260.

S Pyysalo, F Ginter, H Moen, T Salakoski, and S Ananiadou. 2013. *Distributional Semantics Resources for Biomedical Text Processing*, pages 39–44.

Sunita Sarawagi and William W Cohen. 2005. Semi-markov conditional random fields for information extraction. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, MIT Press, pages 1185–1192. http://papers.nips.cc/paper/2648-semi-markov-conditional-random-fields-for-information-extraction.pdf.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, NIPS'14, pages 3104–3112. http://dl.acm.org/citation.cfm?id=2969033.2969173.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 2773–2781. http://dl.acm.org/citation.cfm?id=2969442.2969550.

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

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '02, pages 473–480. https://doi.org/10.3115/1073083.1073163.