Dynamic Entity Representations in Neural Language Models

Yangfeng Ji* **Chenhao Tan*** **Sebastian Martschat†** **Yejin Choi*** **Noah A. Smith***
*Paul G. Allen School of Computer Science & Engineering, University of Washington
†Department of Computational Linguistics, Heidelberg University
{yangfeng,chenhao,yejin,nasmith}@cs.washington.edu
martschat@cl.uni-heidelberg.de

Abstract

Understanding a long document requires tracking how entities are introduced and evolve over time. We present a new type of language model, EntityNLM, that can explicitly model entities, dynamically update their representations, and contextually generate their mentions. Our model is generative and flexible; it can model an arbitrary number of entities in context while generating each entity mention at an arbitrary length. In addition, it can be used for several different tasks such as language modeling, coreference resolution, and entity prediction. Experimental results with all these tasks demonstrate that our model consistently outperforms strong baselines and prior work.

1 Introduction

Understanding a narrative requires keeping track of its participants over a long-term context. As a story unfolds, the information a reader associates with each character in a story increases, and expectations about what will happen next change accordingly. At present, models of natural language do not explicitly track entities; indeed, in today's language models, entities are no more than the words used to mention them.

In this paper, we endow a generative language model with the ability to build up a dynamic representation of each entity mentioned in the text. Our language model defines a probability distribution over the whole text, with a distinct generative story for entity mentions. It explicitly groups those mentions that corefer and associates with each entity a continuous representation that is updated by every contextualized mention of the entity, and that in turn affects the text that follows.

[*John*]$_1$ wanted to go to [*the coffee shop*]$_2$ in [*downtown Copenhagen*]$_3$. [*He*]$_1$ was told that [*it*]$_2$ sold [*the best beans*]$_4$.

Figure 1: EntityNLM explicitly tracks entities in a text, including coreferring relationships between entities like [*John*]$_1$ and [*He*]$_1$. As a language model, it is designed to predict that a coreferent of [*the coffee shop*]$_2$ is likely to follow "*told that*," that the referring expression will be "*it*", and that "*sold the best beans*" is likely to come next, by using entity information encoded in the dynamic distributed representation.

Our method builds on recent advances in representation learning, creating local probability distributions from neural networks. It can be understood as a recurrent neural network language model, augmented with random variables for entity mentions that capture coreference, and with dynamic representations of entities. We estimate the model's parameters from data that is annotated with entity mentions and coreference.

Because our model is generative, it can be queried in different ways. Marginalizing everything except the words, it can play the role of a language model. In §5.1, we find that it outperforms both a strong n-gram language model and a strong recurrent neural network language model on the English test set of the CoNLL 2012 shared task on coreference evaluation (Pradhan et al., 2012). The model can also identify entity mentions and coreference relationships among them. In §5.2, we show that it can easily be used to add a performance boost to a strong coreference resolution system, by reranking a list of k-best candidate outputs. On the CoNLL 2012 shared task test set, the reranked outputs are significantly better than the original top choices from the same system. Fi-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1831–1840
Copenhagen, Denmark, September 7–11, 2017.

nally, the model can perform entity cloze tasks. As presented in §5.3, it achieves state-of-the-art performance on the InScript corpus (Modi et al., 2017).

2 Model

A language model defines a distribution over sequences of word tokens; let X_t denote the random variable for the tth word in the sequence, x_t denote the value of X_t and $\mathbf{x}_t$ the distributed representation (embedding) of this word. Our starting point for language modeling is a recurrent neural network (Mikolov et al., 2010), which defines

$$p(X_t \mid \text{history}) = \text{softmax}\left(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}\right) \quad (1)$$

$$\mathbf{h}_{t-1} = \text{LSTM}(\mathbf{h}_{t-2}, \mathbf{x}_{t-1}) \quad (2)$$

where $\mathbf{W}_h$ and $\mathbf{b}$ are parameters of the model (along with word embeddings $\mathbf{x}_t$), LSTM is the widely used recurrent function known as "long short-term memory" (Hochreiter and Schmidhuber, 1997), and $\mathbf{h}_t$ is a LSTM hidden state encoding the history of the sequence up to the tth word.

Great success has been reported for this model (Zaremba et al., 2015), which posits nothing explicitly about the words appearing in the text sequence. Its generative story is simple: the value of each X_t is randomly chosen conditioned on the vector $\mathbf{h}_{t-1}$ encoding its history.

2.1 Additional random variables and representations for entities

To introduce our model, we associate with each word an additional set of random variables. At position t,

- R_t is a binary random variable that indicates whether x_t belongs to an entity mention ($R_t = 1$) or not ($R_t = 0$). Though not explored here, this is easily generalized to a categorial variable for the *type* of the entity (e.g., person, organization, etc.).
- $L_t \in \{1, \ldots, \ell_{max}\}$ is a categorical random variable if $R_t = 1$, which indicates the number of remaining words in this mention, including the current word (i.e., $L_t = 1$ for the last word in any mention). ℓ_{max} is a predefined maximum length fixed to be 25, which is an empirical value derived from the training corpora used in the experiments. If $R_t = 0$, then $L_t = 1$. We denote the value of L_t by ℓ_t.
- $E_t \in \mathcal{E}_t$ is the index of the entity referred to, if $R_t = 1$. The set $\mathcal{E}_t$ consists of $\{1, \ldots, 1 + \max_{t'<t} e_{t'}\}$, i.e., the indices of all previously mentioned entities plus an additional value for a new entity. Thus $\mathcal{E}_t$ starts as $\{1\}$ and grows monotonically with t, allowing for an arbitrary number of entities to be mentioned. We denote the value of E_t by e_t. If $R_t = 0$, then E_t is fixed to a special value $\emptyset$.

The values of these random variables for our running example are shown in Figure 2.

In addition to using symbolic variables to encode mentions and coreference relationships, we maintain a vector representation of each entity that evolves over time. For the ith entity, let $\mathbf{e}_{i,t}$ be its representation at time t. These vectors are different from word vectors ($\mathbf{x}_t$), in that they are not parameters of the model. They are similar to history representations ($\mathbf{h}_t$), in that they are derived through parameterized functions of the random variables' values, which we will describe in the next subsections.

2.2 Generative story

The generative story for the word (and other variables) at timestep t is as follows; forward-referenced equations are in the detailed discussion that follows.

1. If $\ell_{t-1} = 1$ (i.e., x_t is *not* continuing an already-started entity mention):
   - Choose r_t (Equation 3).
   - If $r_t = 0$, set $\ell_t = 1$ and $e_t = \emptyset$; then go to step 3. Otherwise:
     - If there is no embedding for the new candidate entity with index $1 + \max_{t'<t} e_{t'}$, create one following §2.4.
     - Select the entity e_t from $\{1, \ldots, 1 + \max_{t'<t} e_{t'}\}$ (Equation 4).
     - Set $\mathbf{e}_{current} = \mathbf{e}_{e_t,t-1}$, which is the entity embedding of e_t before timestep t.
     - Select the length of the mention, ℓ_t (Equation 5).
2. Otherwise,
   - Set $\ell_t = \ell_{t-1} - 1$, $r_t = r_{t-1}$, $e_t = e_{t-1}$.
3. Sample x_t from the word distribution given the LSTM hidden state $\mathbf{h}_{t-1}$ and the current

$X_{1:12}$:	John	wanted	to	go	to	the	coffee	shop	in	downtown	Copenhagen	.
$R_{1:12}$:	1	0	0	0	0	1	1	1	0	1	1	0
$E_{1:12}$:	1	ø	ø	ø	ø	2	2	2	ø	3	3	ø
$L_{1:12}$:	1	1	1	1	1	3	2	1	1	2	1	1

$X_{13:22}$:	He	was	told	that	it	sold	the	best	beans	.
$R_{13:22}$:	1	0	0	0	1	0	1	1	1	.
$E_{13:22}$:	1	ø	ø	ø	2	ø	4	4	4	ø
$L_{13:22}$:	1	1	1	1	1	1	3	2	1	0

Figure 2: The random variable values in ENTITYNLM for the running example in Figure 1.

(or most recent) entity embedding $\mathbf{e}_{current}$ (Equation 6). (If $r_t = 0$, then $\mathbf{e}_{current}$ still represents the most recently mentioned entity.)

4. Advance the RNN, i.e., feed it the word vector $\mathbf{x}_t$ to compute $\mathbf{h}_t$ (Equation 2).

5. If $r_t = 1$, update $\mathbf{e}_{e_t,t}$ using $\mathbf{e}_{e_t,t-1}$ and $\mathbf{h}_t$, then set $\mathbf{e}_{current} = \mathbf{e}_{e_t,t}$. Details of the entity updating are given in §2.4.

6. For every entity $e_\iota \in \mathcal{E}_t \setminus \{e_t\}$, set $\mathbf{e}_{\iota,t} = \mathbf{e}_{\iota,t-1}$ (i.e., no changes to other entities' representations).

Note that at any given time step t, $\mathbf{e}_{current}$ will always contain the most recent vector representation of the most recently mentioned entity.

A generative model with a similar hierarchical structure was used by Haghighi and Klein (2010) for coreference resolution. Our approach differs in two important ways. First, our model defines a joint distribution over all of the text, not just the entity mentions. Second, we use representation learning rather than Bayesian nonparametrics, allowing natural integration with the language model.

2.3 Probability distributions

The generative story above referenced several parametric distributions defined based on vector representations of histories and entities. These are defined as follows.

For $r \in \{0, 1\}$,

$$p(R_t = r \mid \mathbf{h}_{t-1}) \propto \exp(\mathbf{h}_{t-1}^\top \mathbf{W}_r \mathbf{r}), \quad (3)$$

where $\mathbf{r}$ is the parameterized embedding associated with r, which paves the way for exploring entity type representations in future work; $\mathbf{W}_r$ is a parameter matrix for the bilinear score for $\mathbf{h}_{t-1}$ and $\mathbf{r}$.

To give the possibility of predicting a new entity, we need an entity embedding beforehand with index $(1 + \max_{t'<t} e_{t'})$, which is randomly sampled from Equation 7. Then, for every $e \in \{1, \ldots, 1 + \max_{t'<t} e_{t'}\}$:

$$\begin{aligned} &p(E_t = e \mid R_t = 1, \mathbf{h}_{t-1}) \\ &\quad \propto \exp(\mathbf{h}_{t-1}^\top \mathbf{W}_{entity} \mathbf{e}_{e,t-1} + \mathbf{w}_{dist}^\top \mathbf{f}(e)), \end{aligned} \quad (4)$$

where $\mathbf{e}_{e,t-1}$ is the embedding of entity e at time step $t-1$ and $\mathbf{W}_{entity}$ is the weight matrix for predicting entities using their continuous representations. The score above is normalized over values $\{1, \ldots, 1 + \max_{t'<t} e_{t'}\}$. $\mathbf{f}(e)$ represents a vector of distance features associated with e and the mentions of the existing entities. Hence two information sources are used to predict the next entity: (i) contextual information $\mathbf{h}_{t-1}$, and (ii) distance features $\mathbf{f}(e)$ from the current mention to the closest mention from each previously mentioned entity. $\mathbf{f}(e) = \mathbf{0}$ if e is a new entity. This term can also be extended to include other surface-form features for coreference resolution (Martschat and Strube, 2015; Clark and Manning, 2016b).

For the chosen entity e_t from Equation 4, the distribution over its mention length is drawn according to

$$\begin{aligned} &p(L_t = \ell \mid \mathbf{h}_{t-1}, \mathbf{e}_{e_t,t-1}) \\ &\qquad \propto \exp(\mathbf{W}_{length,\ell}^\top [\mathbf{h}_{t-1}; \mathbf{e}_{e_t,t-1}]), \end{aligned} \quad (5)$$

where $\mathbf{e}_{e_t,t-1}$ is the most recent embedding of the entity e_t, not updated with $\mathbf{h}_t$. The intuition is that $\mathbf{e}_{e_t,t-1}$ will help contextual information $\mathbf{h}_{t-1}$ to select the residual length of entity e_t. $\mathbf{W}_{length}$ is the weight matrix for length prediction, with $\ell_{max} = 25$ rows.

Finally, the probability of a word x as the next token is jointly modeled by $\mathbf{h}_{t-1}$ and the vector representation of the most recently mentioned entity $\mathbf{e}_{current}$:

$$\begin{aligned} &p(X_t = x \mid \mathbf{h}_{t-1}, \mathbf{e}_{current}) \\ &\qquad \propto \text{CFSM}(\mathbf{h}_{t-1} + \mathbf{W}_e \mathbf{e}_{current}), \end{aligned} \quad (6)$$

where $\mathbf{W}_e$ is a transformation matrix to adjust the dimensionality of $\mathbf{e}_{current}$. CFSM is a class factorized softmax function (Goodman, 2001; Baltescu and Blunsom, 2015). It uses a two-step prediction with predefined word classes instead of direct prediction on the whole vocabulary, and reduces the time complexity to the log of vocabulary size.

2.4 Dynamic entity representations

Before predicting the entity at step t, we need an embedding for the new candidate entity with index $e' = 1 + \max_{t'<t} e_{t'}$ if it does not exist. The new embedding is generated randomly, according to a normal distribution, then projected onto the unit ball:

$$\begin{aligned} \mathbf{u} &\sim \mathcal{N}(\mathbf{r}_1, \sigma^2\mathbf{I}); \\ \mathbf{e}_{e',t-1} &= \frac{\mathbf{u}}{\|\mathbf{u}\|_2}, \end{aligned} \tag{7}$$

where $\sigma = 0.01$. The time step $t-1$ in $\mathbf{e}_{e',t-1}$ means the current embedding contains no information from step t, although it will be updated once we have $\mathbf{h}_t$ and if $E_t = e'$. $\mathbf{r}_1$ is the parameterized embedding for $R_t = 1$, which will be jointly optimized with other parameters and is expected to encode some generic information about entities. All the initial entity embeddings are centered on the mean $\mathbf{r}_1$, which is used in Equation 3 to determine whether the next token belongs to an entity mention. Another choice would be to initialize with a zero vector, although our preliminary experiments showed this did not work as well as random initialization in Equation 7.

Assume $R_t = 1$ and $E_t = e_t$, which means x_t is part of a mention of entity e_t. Then, we need to update $\mathbf{e}_{e_t,t-1}$ based on the new information we have from $\mathbf{h}_t$. The new embedding $\mathbf{e}_{e_t,t}$ is a convex combination of the old embedding ($\mathbf{e}_{e_t,t-1}$) and current LSTM hidden state ($\mathbf{h}_t$) with the interpolation (δ_t) determined dynamically based on a bilinear function:

$$\begin{aligned} \delta_t &= \sigma(\mathbf{h}_t^\top \mathbf{W}_\delta \mathbf{e}_{e_t,t-1}); \\ \mathbf{u} &= \delta_t \mathbf{e}_{e_t,t-1} + (1-\delta_t)\mathbf{h}_t; \\ \mathbf{e}_{e_t,t} &= \frac{\mathbf{u}}{\|\mathbf{u}\|_2}, \end{aligned} \tag{8}$$

This updating scheme will be used to update e_t in *each* of all the following ℓ_t steps. The projection in the last step keeps the magnitude of the entity embedding fixed, avoiding numeric overflow. A similar updating scheme has been used by Henaff et al. (2016) for the "memory blocks" in their recurrent entity network models. The difference is that their model updates all memory blocks in each time step. Instead, our updating scheme in Equation 8 only applies to the selected entity e_t at time step t.

2.5 Training objective

The model is trained to maximize the log of the joint probability of $\mathbf{R}$, $\mathbf{E}$, $\mathbf{L}$, and $\mathbf{X}$:

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \log P(\mathbf{R}, \mathbf{E}, \mathbf{L}, \mathbf{X}; \boldsymbol{\theta}) \\ &= \sum_t \log P(R_t, E_t, L_t, X_t; \boldsymbol{\theta}), \end{aligned} \tag{9}$$

where $\boldsymbol{\theta}$ is the collection of all the parameters in this model. Based on the formulation in §2.3, Equation 9 can be decomposed as the sum of conditional log-probabilities of each random variable at each time step.

This objective requires the training data annotated as in Figure 2. We do not assume that these variables are observed at test time.

3 Implementation Details

Our model is implemented with DyNet (Neubig et al., 2017) and available at `https://github.com/jiyfeng/entitynlm`. We use AdaGrad (Duchi et al., 2011) with learning rate $\lambda = 0.1$ and ADAM (Kingma and Ba, 2014) with default learning rate $\lambda = 0.001$ as the candidate optimizers of our model. For all the parameters, we use the initialization tricks recommended by Glorot and Bengio (2010). To avoid overfitting, we also employ dropout (Srivastava et al., 2014) with the candidate rates as $\{0.2, 0.5\}$.

In addition, there are two tunable hyperparameters of ENTITYNLM: the size of word embeddings and the dimension of LSTM hidden states. For both of them, we consider the values $\{32, 48, 64, 128, 256\}$. We also experiment with the option to either use the pretrained GloVe word embeddings (Pennington et al., 2014) or randomly initialized word embeddings (then updated during training). For all experiments, the best configuration of hyperparameters and optimizers is selected based on the objective value on the development data.

4 Evaluation Tasks and Datasets

We evaluate our model in diverse use scenarios: (i) language modeling, (ii) coreference resolution,

and (iii) entity prediction. The evaluation on language modeling shows how the internal entity representation, when marginalized out, can improve the perplexity of language models. The evaluation on coreference resolution experiment shows how our new language model can improve a competitive coreference resolution system. Finally, we employ an entity cloze task to demonstrate the generative performance of our model in predicting the next entity given the previous context.

We use two datasets for the three evaluation tasks. For language modeling and coreference resolution, we use the English benchmark data from the CoNLL 2012 shared task on coreference resolution (Pradhan et al., 2012). We employ the standard training/development/test split, which includes 2,802/343/348 documents with roughly 1M/150K/150K tokens, respectively. We follow the coreference annotation in the CoNLL dataset to extract entities and ignore the singleton mentions in texts.

For entity prediction, we employ the InScript corpus created by Modi et al. (2017). It consists of 10 scenarios, including grocery shopping, taking a flight, etc. It includes 910 crowdsourced simple narrative texts in total and 18 stories were ignored due to labeling problems (Modi et al., 2017). On average, each story has 12.4 sentences, 24.9 entities and 217.2 tokens. Each entity mention is labeled with its entity index. We use the same training/development/test split as in (Modi et al., 2017), which includes 619, 91, 182 texts, respectively.

Data preprocessing

For the CoNLL dataset, we lowercase all tokens and remove any token that only contains a punctuation symbol unless it is in an entity mention. We also replace numbers in the documents with the special token NUM and low-frequency word types with UNK. The vocabulary size of the CoNLL data after preprocessing is 10K. For entity mention extraction, in the CoNLL dataset, one entity mention could be embedded in another. For embedded mentions, only the enclosing entity mention is kept. We use the same preprocessed data for both language modeling and coreference resolution evaluation.

For the InScript corpus, we apply similar data preprocessing to lowercase all tokens, and we replace low-frequency word types with UNK. The vocabulary size after preprocessing is 1K.

5 Experiments

In this section, we present the experimental results on the three evaluation tasks.

5.1 Language modeling

Task description. The goal of language modeling is to compute the marginal probability:

$$P(\mathbf{X}) = \sum_{\mathbf{R},\mathbf{E},\mathbf{L}} P(\mathbf{X}, \mathbf{R}, \mathbf{E}, \mathbf{L}). \quad (10)$$

However, due to the long-range dependency in recurrent neural networks, the search space of $\mathbf{R}, \mathbf{E}, \mathbf{L}$ during inference grows exponentially. We thus use importance sampling to approximate the marginal distribution of $\mathbf{X}$. Specifically, with the samples from a proposal distribution $Q(\mathbf{R}, \mathbf{E}, \mathbf{L}|\mathbf{X})$, the approximated marginal probability is defined as

$$\begin{aligned} P(\mathbf{X}) &= \sum_{\mathbf{R},\mathbf{E},\mathbf{L}} P(\mathbf{X}, \mathbf{R}, \mathbf{E}, \mathbf{L}) \\ &= \sum_{\mathbf{R},\mathbf{E},\mathbf{L}} Q(\mathbf{R}, \mathbf{E}, \mathbf{L} \mid \mathbf{X}) \frac{P(\mathbf{X}, \mathbf{R}, \mathbf{E}, \mathbf{L})}{Q(\mathbf{R}, \mathbf{E}, \mathbf{L} \mid \mathbf{X})} \\ &\approx \frac{1}{N} \sum_{\{\mathbf{r}^{(i)}, \mathbf{e}^{(i)}, \ell^{(i)}\} \sim Q} \frac{P(\mathbf{r}^{(i)}, \mathbf{e}^{(i)}, \ell^{(i)}, \mathbf{x})}{Q(\mathbf{r}^{(i)}, \mathbf{e}^{(i)}, \ell^{(i)} \mid \mathbf{x})} \end{aligned} \quad (11)$$

A similar idea of using importance sampling for language modeling evaluation has been used by Dyer et al. (2016).

For language modeling evaluation, we train our model on the training set from the CoNLL 2012 dataset with coreference annotation. On the test data, we treat coreference structure as latent variables and use importance sampling to approximate the marginal distribution of $\mathbf{X}$. For each document, the model randomly draws $N = 100$ samples from the proposal distribution, discussed next.

Proposal distribution. For implementation of Q, we use a discriminative variant of ENTITYNLM by taking the current word x_t for predicting the entity-related variables in the same time step. Specifically, in the generative story described in §2.2, we delete step 3 (words are not generated, but rather conditioned upon), move step 4 before step 1, and replace $\mathbf{h}_{t-1}$ with $\mathbf{h}_t$ in the steps for predicting entity type R_t, entity E_t and mention length L_t. This model variant provides a

Model	Perplexity
1. 5-gram LM	138.37
2. RNNLM	134.79
3. ENTITYNLM	**131.64**

Table 1: Language modeling evaluation on the test sets of the English section in the CoNLL 2012 shared task. As mentioned in §4, the vocabulary size is 10K. ENTITYNLM does not require any coreference annotation on the test data.

conditional probability $Q(R_t, E_t, L_t \mid X_t)$ at each timestep.

Baselines. We compare the language modeling performance with two competitive baselines: 5-gram language model implemented in KenLM (Heafield et al., 2013) and RNNLM with LSTM units implemented in DyNet (Neubig et al., 2017). For RNNLM, we use the same hyperparameters described in §3 and grid search on the development data to find the best configuration.

Results. The results of ENTITYNLM and the baselines on both development and test data are reported in Table 1. For ENTITYNLM, we use the value of $2^{-\frac{1}{T}\sum_{t=1}^{T}\log P(X_t,R_t,E_t,L_t)}$ on the development set with coreference annotation to select the best model configuration and report the best number. On the test data, we are able to calculate perplexity by marginalizing all other random variables using Equation 11. To compute the perplexity numbers on the test data, our model only takes account of log probabilities on word prediction. The difference is that coreference information is only used for training ENTITYNLM and not for test. All three models reported in Table 1 share the same vocabulary, therefore the numbers on the test data are directly comparable. As shown in Table 1, ENTITYNLM outperforms both the 5-gram language model and the RNNLM on the test data. Better performance of ENTITYNLM on language modeling can be expected, if we also use the marginalization method defined in Equation 11 on the development data to select the best configuration. However, we plan to use the same experimental setup for all experiments, instead of customizing our model for each individual task.

5.2 Coreference reranking

Task description. We show how ENTITYLM, which allows an efficient computation of the probability $P(\mathbf{R}, \mathbf{E}, \mathbf{L}, \mathbf{X})$, can be used as a coreference reranker to improve a competitive coreference resolution system due to Martschat and Strube (2015). This task is analogous to the reranking approach used in machine translation (Shen et al., 2004). The specific formulation is as follows:

$$\arg \max_{\{\mathbf{r}^{(i)},\mathbf{e}^{(i)},\mathbf{l}^{(i)}\}\in\mathcal{K}} P(\mathbf{r}^{(i)}, \mathbf{e}^{(i)}, \mathbf{l}^{(i)}, \mathbf{x}) \quad (12)$$

where $\mathcal{K}$ is the k-best list for a given document. In our experiments, $k = 100$. To the best of our knowledge, the problem of obtaining k-best outputs of a coreference resolution system has not been studied before.

Approximate k-best decoding. We rerank the output of a system that predicts an antecedent for each mention by relying on pairwise scores for mention pairs. This is the dominant approach for coreference resolution (Martschat and Strube, 2015; Clark and Manning, 2016a). The predictions induce an antecedent tree, which represents antecedent decisions for all mentions in the document. Coreference chains are obtained by transitive closure over the antecedent decisions encoded in the tree. A mention also can have an empty mention as antecedent, which denotes that the mention is non-anaphoric.

For extending Martschat and Strube's greedy decoding approach to k-best inference, we cannot simply take the k highest scoring trees according to the sum of edge scores, because different trees may represent the same coreference chain. Instead, we use an heuristic that creates an approximate k-best list on candidate antecedent trees. The idea is to generate trees from the original system output by considering suboptimal antecedent choices that lead to different coreference chains. For each mention pair (m_j, m_i), we compute the difference of its score to the score of the optimal antecedent choice for m_j. We then sort pairs in ascending order according to this difference and iterate through the list of pairs. For each pair (m_j, m_i), we create a tree $t_{j,i}$ by replacing the antecedent of m_j in the original system output with m_i. If this yields a tree that encodes different coreference chains from all chains encoded by trees in the k-best list, we add $t_{i,j}$ to the k-best list. In the case that we cannot generate a given number of trees (particularly for a short document with a large k), we pad the list with the last item added to the list.

Evaluation measures. For coreference resolution evaluation, we employ the CoNLL scorer (Pradhan et al., 2014). It computes three commonly used evaluation measures MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and $CEAF_e$ (Luo, 2005). We report the F_1 score of each evaluation measure and their average as the CoNLL score.

Competing systems. We employed CORT[1] (Martschat and Strube, 2015) as our baseline coreference resolution system. Here, we compare with the original (one best) outputs of CORT's latent ranking model, which is the best-performing model implemented in CORT. We consider two rerankers based on ENTITYNLM. The first reranking method only uses the log probability for ENTITYNLM to sort the candidate list (Equation 12). The second method uses a linear combination of both log probabilities from ENTITYNLM and the scores from CORT, where the coefficients were found via grid search with the CoNLL score on the development set.

Results. The reranked results on the CoNLL 2012 test set are reported in Table 2. The numbers of the baseline are higher than the results reported in Martschat and Strube (2015) since the feature set of CORT was subsequently extended. Lines 2 and 3 in Table 2 present the reranked best results. As shown in this table, both reranked results give more than 1% of CoNLL score improvement on the test set over CORT, which are significant based on an approximate randomization test[2].

Additional experiments also found that increasing k from 100 to 500 had a minor effect. That is because the diversity of each k-best list is limited by (i) the number of entity mentions in the document, (ii) the performance of the baseline coreference resolution system, and possibly (iii) the approximate nature of our k-best inference procedure. We suspect that a stronger baseline system (such as that of Clark and Manning, 2016a) could give greater improvements, if it can be adapted to provide k-best lists. Future work might incorporate the techniques embedded in such systems into ENTITYNLM.

$[I]_1$ was about to ride $[my]_1$ $[bicycle]_2$ to the $[park]_3$ one day when $[I]_1$ noticed that the front $[tire]_4$ was flat . $[I]_1$ realized that $[I]_1$ would have to repair $[it]_4$. $[I]_1$ went into $[my]_1$ $[garage]_5$ to get some $[tools]_5$. The first thing $[I]_1$ did was remove the xxxx

---

Figure 3: A short story on bicycles from the InScript corpus (Modi et al., 2017). The entity prediction task requires predicting xxxx given the preceding text either by choosing a previously mentioned entity or deciding that this is a "new entity". In this example, the ground-truth prediction is $[tire]_4$. For training, ENTITYNLM attempts to predict every entity. While, for testing, it predicts a maximum of 30 entities after the first three sentences, which is consistent with the experimental setup suggested by Modi et al. (2017).

5.3 Entity prediction

Task description. Based on Modi et al. (2017), we introduce a novel entity prediction task that tries to predict the next entity given the preceding text. For a given text as in Figure 3, this task makes a forward prediction based on only the left context. This is different from coreference resolution, where both left and right contexts from a given entity mention are used in decoding. It is also different from language modeling, since this task only requires predicting entities. Since ENTITYNLM is generative, it can be directly applied to this task. To predict entities in test data, R_t is always given and ENTITYNLM only needs to predict E_t when $R_t = 1$.

Baselines and human prediction. We introduce two baselines in this task: (i) the **always-new** baseline that always predicts "new entity"; (ii) a linear classification model using **shallow features** from Modi et al. (2017), including the recency of an entity's last mention and the frequency. We also compare with the model proposed by Modi et al. (2017). Their work assumes that the model has prior knowledge of all the participant types, which are specific to each scenario and fine-grained, e.g., rider in the bicycle narrative, and predicts participant types for new entities. This assumption is unrealistic for pure generative models like ours.

[1] https://github.com/smartschat/cort, we used version 0.2.4.5.

[2] https://github.com/smartschat/art

Model	CoNLL	MUC			B^3			$CEAF_e$		
		P	R	F_1	P	R	F_1	P	R	F_1
1. Baseline: CORT's one best	62.93	77.15	68.67	72.66	66.00	54.92	59.95	60.07	52.76	56.18
2. Rerank: ENTITYNLM	**64.00**	77.90	69.45	73.44	66.84	56.12	61.01	61.73	53.90	57.55
3. Rerank: ENTITYNLM + CORT	**64.04**	77.93	69.49	73.47	67.08	55.99	61.04	61.76	53.98	57.61

Table 2: Coreference resolution scores on the CoNLL 2012 test set. CORT is the best-performing model of Martschat and Strube (2015) with greedy decoding.

	Accuracy (%)
1. Baseline: always-new	31.08
2. Baseline: shallow features	45.34
3. Modi et al. (2017)	62.65
4. ENTITYNLM	**74.23**
5. *Human prediction*	77.35

Table 3: Entity prediction accuracy on the test set of the InScript corpus.

Therefore, we remove this assumption and adapt their prediction results to our formulation by mapping all the predicted entities that have not been mentioned to "new entity". We also compare to the adapted **human prediction** used in the InScript corpus. For each entity slot, Modi et al. (2017) acquired 20 human predictions, and the majority vote was selected. More details about human predictions are discussed in (Modi et al., 2017).

Results. Table 3 shows the prediction accuracies. ENTITYNLM (line 4) significantly outperforms both baselines (line 1 and 2) and prior work (line 3) ($p \ll 0.01$, paired t-test). The comparison between line 4 and 5 shows our model is even close to the human prediction performance.

6 Related Work

Rich-context language models. The originally proposed recurrent neural network language models only capture information within sentences. To extend the capacity of RNNLMs, various researchers have incorporated information beyond sentence boundaries. Previous work focuses on contextual information from previous sentences (Ji et al., 2016a) or discourse relations between adjacent sentences (Ji et al., 2016b), showing improvements to language modeling and related tasks like coherence evaluation and discourse relation prediction. In this work, ENTITYNLM adds explicit entity information to the language model, which is another way of adding a memory network for language modeling. Unlike the work by Tran et al. (2016), where memory blocks are used to store general contextual information for language modeling, ENTITYNLM assigns each memory block specifically to only one entity.

Entity-related models. Two recent approaches to modeling entities in text are closely related to our model. The first is the "reference-aware" language models proposed by Yang et al. (2016), where the referred entities are from either a predefined item list, an external database, or the context from the same document. Yang et al. (2016) present three models, one for each case. For modeling a document with entities, they use coreference links to recover entity clusters, though they only model entity mentions as containing a single word (an inappropriate assumption, in our view). Their entity updating method takes the latest hidden state (similar to $\mathbf{h}_t$ when $R_t = 1$ in our model) as the new representation of the current entity; no long-term history of the entity is maintained, just the current local context. In addition, their language model evaluation assumes that entity information is provided at test time (Yang, personal communication), which makes a direct comparison with our model impossible. Our entity updating scheme is similar to the "dynamic memory" method used by Henaff et al. (2016). Our entity representations are dynamically allocated and updated only when an entity appears up, while the EntNet from Henaff et al. (2016) does not model entities and their relationships explicitly. In their model, entity memory blocks are pre-allocated and updated simultaneously in each timestep. So there is no dedicated memory block for every entity and no distinction between entity mentions and non-mention words. As a consequence, it is not clear how to use their model for coreference reranking and entity prediction.

Coreference resolution. The hierarchical structure of our entity generation model is inspired by

Haghighi and Klein (2010). They implemented this idea as a probabillistic graphical model with the distance-dependent Chinese Restaurant Process (Pitman, 1995) for entity assignment, while our model is built on a recurrent neural network architecture. The reranking method considered in our coreference resolution evaluation could also be extended with samples from additional coreference resolution systems, to produce more variety (Ng, 2005). The benefit of such a system comes, we believe, from the explicit tracking of each entity throughout the text, providing entity-specific representations. In previous work, such information has been added as features (Luo et al., 2004; Björkelund and Kuhn, 2014) or by computing distributed entity representations (Wiseman et al., 2016; Clark and Manning, 2016b). Our approach complements these previous methods.

Entity prediction. The entity prediction task discussed in §5.3 is based on work by Modi et al. (2017). The main difference is that we do not assume that all entities belong to a previously known set of entity types specified for each narrative scenario. This task is also closely related to the "narrative cloze" task of Chambers and Jurafsky (2008) and the "story cloze test" of Mostafazadeh et al. (2016). Those studies aim to understand relationships between events, while our task focuses on predicting upcoming entity mentions.

7 Conclusion

We have presented a neural language model, ENTITYNLM, that defines a distribution over texts and the mentioned entities. It provides vector representations for the entities and updates them dynamically in context. The dynamic representations are further used to help generate specific entity mentions and the following text. This model outperforms strong baselines and prior work on three tasks: language modeling, coreference resolution, and entity prediction.

Acknowledgments

We thank anonymous reviewers for the helpful feedback on this work. We also thank the members of Noah's ARK and XLab at University of Washington for their valuable comments, particularly Eunsol Choi for pointing out the InScript corpus. This research was supported in part by a University of Washington Innovation Award, Samsung GRO, NSF grant IIS-1524371, the DARPA CwC program through ARO (W911NF-15-1-0543), and gifts by Google and Facebook.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *LREC Workshop on Linguistic Coreference*.

Paul Baltescu and Phil Blunsom. 2015. Pragmatic neural language modelling in machine translation. In *NAACL*.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *ACL*.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *ACL*.

Kevin Clark and Christopher D. Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *EMNLP*.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *ACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *EMNLP*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *ICASSP*.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *NAACL*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *ACL*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. arXiv:1612.03969.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2016a. Document context language models. In *ICLR (workshop track)*.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016b. A latent variable recurrent neural network for discourse-driven language models. In *NAACL-HLT*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *HLT-EMNLP*.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *ACL*.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. 2017. Modeling semantic expectation: Using script knowledge for referent prediction. *Transactions of the Association of Computational Linguistics*, 5:31–44.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *NAACL*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. arXiv:1701.03980.

Vincent Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *ACL*.

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

Jim Pitman. 1995. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102(2):145–158.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *ACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *EMNLP-CoNLL*.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *NAACL*.

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(1):1929–1958.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *NAACL-HLT*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *MUC*.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *NAACL*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. arXiv:1611.01628.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. *ICLR*.