Leveraging Knowledge Bases in LSTMs for Improving Machine Reading

Bishan Yang
Machine Learning Department
Carnegie Mellon University
bishan@cs.cmu.edu

Tom Mitchell
Machine Learning Department
Carnegie Mellon University
tom.mitchell@cs.cmu.edu

Abstract

This paper focuses on how to take advantage of external knowledge bases (KBs) to improve recurrent neural networks for machine reading. Traditional methods that exploit knowledge from KBs encode knowledge as discrete indicator features. Not only do these features generalize poorly, but they require task-specific feature engineering to achieve good performance. We propose KBLSTM, a novel neural model that leverages continuous representations of KBs to enhance the learning of recurrent neural networks for machine reading. To effectively integrate background knowledge with information from the currently processed text, our model employs an attention mechanism with a sentinel to adaptively decide whether to attend to background knowledge and which information from KBs is useful. Experimental results show that our model achieves accuracies that surpass the previous state-of-the-art results for both entity extraction and event extraction on the widely used ACE2005 dataset.

1 Introduction

Recurrent neural networks (RNNs), a neural architecture that can operate over text sequentially, have shown great success in addressing a wide range of natural language processing problems, such as parsing (Dyer et al., 2015), named entity recognition (Lample et al., 2016), and semantic role labeling (Zhou and Xu, 2015)). These neural networks are typically trained end-to-end where the input is only text or a sequence of words and a lot of background knowledge is disregarded.

The importance of background knowledge in natural language understanding has long been recognized (Minsky, 1988; Fillmore, 1976). Earlier NLP systems mostly exploited restricted linguistic knowledge such as manually-encoded morphological and syntactic patterns. With the advanced development of knowledge base construction, large amounts of semantic knowledge become available, ranging from manually annotated semantic networks like WordNet [1] to semi-automatically or automatically constructed knowledge graphs like DBPedia [2] and NELL [3]. While traditional approaches have exploited the use of these knowledge bases (KBs) in NLP tasks (Ratinov and Roth, 2009; Rahman and Ng, 2011; Nakashole and Mitchell, 2015), they require a lot of task-specific engineering to achieve good performance.

One way to leverage KBs in recurrent neural networks is by augmenting the dense representations of the networks with the symbolic features derived from KBs. This is not ideal as the symbolic features have poor generalization ability. In addition, they can be highly sparse, e.g., using WordNet synsets can easily produce millions of indicator features, leading to high computational cost. Furthermore, the usefulness of knowledge features varies across contexts, as general KBs involve polysemy, e.g., "Clinton" can refer to a person or a town. Incorporating KBs irrespective of the textual context could mislead the machine reading process.

Can we train a recurrent neural network that learns to adaptively leverage knowledge from KBs to improve machine reading? In this paper, we propose KBLSTM, an extension to bidirec-

[1] https://wordnet.princeton.edu
[2] http://wiki.dbpedia.org/
[3] http://rtw.ml.cmu.edu/rtw/kbbrowser/

tional Long Short-Term Memory neural networks (BiLSTMs) (Hochreiter and Schmidhuber, 1997; Graves et al., 2005) that is capable of leveraging symbolic knowledge from KBs as it processes each word in the text. At each time step, the model retrieves KB concepts that are potentially related to the current word. Then, an attention mechanism is employed to dynamically model their semantic relevance to the reading context. Furthermore, we introduce a sentinel component in BiLSTMs that allows flexibility in deciding whether to attend to background knowledge or not. This is crucial because in some cases the text context should override the context-independent background knowledge available in general KBs.

In this work, we leverage two general, readily available knowledge bases: WordNet (WordNet, 2010) and NELL (Mitchell et al., 2015). WordNet is a manually created lexical database that organizes a large number of English words into sets of synonyms (i.e. synsets) and records conceptual relations (e.g., hypernym, part_of) among them. NELL is an automatically constructed web-based knowledge base that stores beliefs about entities. It is organized based on an ontology of hundreds of semantic categories (e.g., person, fruit, sport) and relations (e.g., personPlaysInstrument). We learn distributed representations (i.e., embeddings) of WordNet and NELL concepts using knowledge graph embedding methods. We then integrate these learned embeddings with the state vectors of the BiLSTM network to enable knowledge-aware predictions.

We evaluate the proposed model on two core information extraction tasks: entity extraction and event extraction. For entity extraction, the model needs to recognize all mentions of entities such as person, organization, location, and other things from text. For event extraction, the model is required to identify event mentions or event triggers[4] that express certain types of events, e.g., elections, attacks, and travels. Both tasks are challenging and often require the combination of background knowledge and the text context for accurate prediction. For example, in the sentence "Maigret left viewers in tears.", knowing that "Maigret" can refer to a TV show can greatly help disambiguate its meaning. However, knowledge bases may hurt performance if used blindly. For example, in the sentence "Santiago is charged with murder.", methods that rely heavily on KBs are likely to interpret "Santiago" as a location due to the popular use of Santiago as a city. Similarly for events, the same word can trigger different types of events, for example, "release" can be used to describe different events ranging from book publishing to parole. It is important for machine learning models to learn to decide which knowledge from KBs is relevant given the context.

Extensive experiments demonstrate that our KBLSTM models effectively leverage background knowledge from KBs in training BiLSTM networks for machine reading. They achieve significant improvement on both entity and event extraction compared to traditional feature-based methods and LSTM networks that disregard knowledge in KBs, resulting in new state-of-the-art results for entity extraction and event extraction on the widely used ACE2005 dataset.

2 Related Work

Essential to RNNs' success on natural language processing is the use of Long Short-Term Memory neural networks (Hochreiter and Schmidhuber, 1997) (LSTMs) or Gated Recurrent Unit (Cho et al., 2014) (GRU) as they are able to handle long-term dependencies by adaptively memorizing values for either long or short durations. Their bidirectional variants BiLSTM (Graves et al., 2005) or BiGRU further allow the incorporation of both past and future information. Such ability has been shown to be generally helpful in various NLP tasks such as named entity recognition (Huang et al., 2015; Chiu and Nichols, 2016; Ma and Hovy, 2016), semantic role labeling (Zhou and Xu, 2015), and reading comprehension (Hermann et al., 2015; Chen et al., 2016). In this work, we also employ the BiLSTM architecture.

In parallel to the development for text processing, neural networks have been successfully used to learn distributed representations of structured knowledge from large KBs (Bordes et al., 2011, 2013; Socher et al., 2013; Yang et al., 2015; Guu et al., 2015). Embedding the symbolic representations into continuous space not only makes KBs more easy to use in statistical learning approaches, but also offers strong generalization ability. Many attempts have been made on connecting distributed representations of KBs with text in the

[4] An event also consists of participants whose types depend on the event triggers. In this work, we focus on predicting event triggers and leave the prediction of event participants for future work.

context of knowledge base completion (Lao et al., 2011; Gardner et al., 2014; Toutanova et al., 2015), relation extraction (Weston et al., 2013; Chang et al., 2014; Riedel et al., 2013), and question answering (Miller et al., 2016). However, these approaches model text using shallow representations such as subject/relation/object triples or bag of words. More recently, Ahn et al. (2016) proposed a neural knowledge language model that leverages knowledge bases in RNN language models, which allows for better representations of words for language modeling. Unlike their work, we leverage knowledge bases in LSTMs and applies it to information extraction.

The architecture of our KBLSTM model draws on the development of attention mechanisms that are widely employed in tasks such as machine translation (Bahdanau et al., 2015) and image captioning (Xu et al., 2015). Attention allows the neural networks to dynamically attend to salient features of the input. With a similar motivation, we employ attention in KBLSTMs to allow for dynamic attention to the relevant knowledge given the text context. Our model is also closely related to a recent model of caption generation introduced by Lu et al. (2017), where a visual sentinel is introduced to allow the decoder to decide whether to attend to image information when generating the next word. In our model, we introduce a sentinel to control the tradeoff between background knowledge and information from the text.

3 Method

In this section, we present our KBLSTM model. We first describe several basic recurrent neural network frameworks for machine reading, including basic RNNs, LSTMs, and bidirectional LSTMs (Sec. § 3.1). We then introduce our extension to bidirectional LSTMs that allows for the incorporation of KB information at each time step of reading (Sec. § 3.2). The KB information is encoded using continuous representations (i.e., embeddings) which are learned using knowledge embedding methods (Sec. § 3.3).

3.1 RNNs, LSTMs, and Bidirectional LSTMs

RNNs are a class of neural networks that take a sequence of inputs and compute a hidden state vector at each time step based on the current input and the entire history of inputs. The hidden state vector can be computed recursively using the following equation (Elman, 1990):

$$\mathbf{h}_t = F(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t)$$

where $\mathbf{x}_t$ is the input at time step t, $\mathbf{h}_t$ is the hidden state at time step t, $\mathbf{U}$ and $\mathbf{W}$ are weight matrices, and F is a nonlinear function such as tanh or ReLu. Depending on the applications, RNNs can produce outputs based on the hidden state of each time step or just the last time step.

A Long Short-Term Memory network (Hochreiter and Schmidhuber, 1997) (LSTM) is a variant of RNNs which was design to better handle cases where the output at time t depends on much earlier inputs. It has a memory cell and three gating units: an input gate that controls what information to add to the current memory, a forget gate which controls what information to remove from the previous memory, and an output gate which controls what information to output from the current memory. Each gate is implemented as a logistic function σ that takes as input the previous hidden state and the current input, and outputs values between 0 and 1. Multiplication with these values controls the flow of information into or out of the memory. In equations, the updates at each time step t are:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{h}_{t-1} + \mathbf{U}_i\mathbf{x}_t) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{h}_{t-1} + \mathbf{U}_f\mathbf{x}_t) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{h}_{t-1} + \mathbf{U}_o\mathbf{x}_t) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot tanh(\mathbf{W}_c\mathbf{h}_{t-1} + \mathbf{U}_c\mathbf{x}_t) \\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{c}_t)
\end{aligned}$$

where $\mathbf{i}_t$ is the input gate, $\mathbf{f}_t$ is the forget gate, $\mathbf{o}_t$ is the output gate, $\mathbf{c}_t$ is the memory cell, and $\mathbf{h}_t$ is the hidden state. $\odot$ denotes element-wise multiplication. $\mathbf{W}_i, \mathbf{U}_i, \mathbf{W}_f, \mathbf{U}_f, \mathbf{W}_o, \mathbf{U}_o, \mathbf{W}_c, \mathbf{U}_c$ are weight matrices to be learned.

Bidirectional LSTMs (Graves et al., 2005) (BiLSTMs) are essentially a combination of two LSTMs in two directions: one operates in the forward direction and the other operates in the backward direction. This leads to two hidden states $\overrightarrow{\mathbf{h}_t}$ and $\overleftarrow{\mathbf{h}_t}$ at time step t, which can be viewed as a summary of the past and the future respectively. Their concatenation $\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$ provides a whole summary of the information about the input around time step t. Such property is attractive in NLP tasks, since information of both previous words and future words can be helpful for interpreting the meaning of the current word.

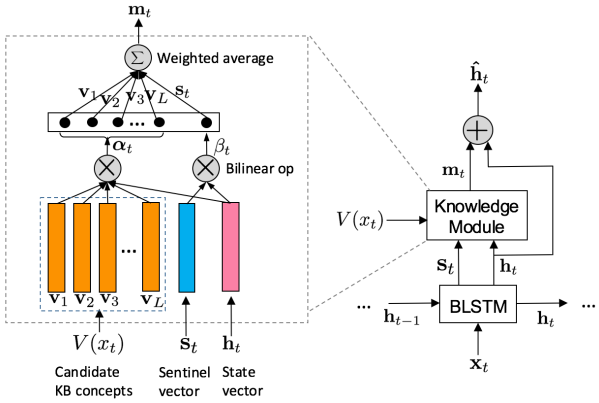


Figure 1: Architecture of the KBLSTM model. As each time step t, the knowledge module retrieves a set of candidate KB concepts $V(x_t)$ that are related to the current input x_t, and then computes a knowledge state vector $\mathbf{m}_t$ that integrates the embeddings of the candidate KB concepts $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_L$ and the current context vector $\mathbf{s}_t$. See Section § 3.2 for details.

3.2 Knowledge-aware Bidirectional LSTMs

Our model (referred to as KBLSTM) extends BiLSTMs to allow flexibility in incorporating symbolic knowledge from KBs. Instead of encoding knowledge as discrete features, we encode it using continuous representations. Concretely, we learn embeddings of concepts in KBs using a knowledge graph embedding method. (We will describe the details in Section § 3.3). The KBLSTM model then retrieves the embeddings of candidate concepts that are related to the current word being processed and integrates them into its state vector to make knowledge-aware predictions. Figure 1 depicts the architecture of our model.

The core of our model is the knowledge module, which is responsible for transferring background knowledge into the BiLSTMs. The knowledge at time step t consists of candidate KB concepts $V(x_t)$ for input x_t. (We will describe how to obtain the candidate KB concepts from NELL and WordNet in Section § 3.3). Each candidate KB concept $i \in V(x_t)$ is associated with a vector embedding $\mathbf{v}_i$. We compute an attention weight α_{ti} for concept vector $\mathbf{v}_i$ via a bilinear operator, which reflects how relevant or important concept i is to the current reading context $\mathbf{h}_t$:

$$\alpha_{ti} \propto \exp(\mathbf{v}_i^T \mathbf{W}_v \mathbf{h}_t) \tag{1}$$

where $\mathbf{W}_v$ is a parameter matrix to be learned.

Note that the candidate concepts in some cases are misleading. For example, a KB may store the fact that "Santiago" is a city but miss the fact that it can also refer to a person. Incorporating such knowledge in the sentence "Santiago is charged with murder." could be misleading. To address this issue, we introduce a knowledge sentinel that records the information of the current context and use a mixture model to allow for better tradeoff between the impact of background knowledge and information from the context. Specifically, we compute a sentinel vector $\mathbf{s}_t$ as:

$$\mathbf{b}_t = \sigma(\mathbf{W}_b \mathbf{h}_{t-1} + \mathbf{U}_b \mathbf{x}_t) \tag{2}$$

$$\mathbf{s}_t = \mathbf{b}_t \odot tanh(\mathbf{c}_t) \tag{3}$$

where $\mathbf{W}_b$ and $\mathbf{U}_b$ are weight parameters to be learned. The weight on the local context is computed as:

$$\beta_t \propto \exp(\mathbf{s}_t^T \mathbf{W}_s \mathbf{h}_t) \tag{4}$$

where $\mathbf{W}_s$ is a parameter matrix to be learned. The mixture model is defined as:

$$\mathbf{m}_t = \sum_{i \in V(x_t)} \alpha_{ti} \mathbf{v}_i + \beta_t \mathbf{s}_t \tag{5}$$

where $\sum_{i \in V(x_t)} \alpha_{ti} + \beta_t = 1$. $\mathbf{m}_t$ can be viewed as a knowledge state vector that encodes external KB information with respect to the input at time t. We combine it with the state vector $\mathbf{h}_t$ of BiLSTMs to obtain a knowledge-aware state vector $\hat{\mathbf{h}}_t$:

$$\hat{\mathbf{h}}_t = \mathbf{h}_t + \mathbf{m}_t \tag{6}$$

If $V(x_t) = \emptyset$, we set $\mathbf{m}_t = 0$. $\hat{\mathbf{h}}_t$ can be used for predictions in the same way as the original state vector $\mathbf{h}_t$ (see Section § 4 for details).

3.3 Embedding Knowledge Base Concepts

Now we describe how to learn embeddings of concepts in KBs. We consider two KBs: WordNet and NELL, which are both knowledge graphs that can be stored in the form of RDF[5] triples. Each triple consists of a subject entity, a relation, and an object entity. Examples of triples in WordNet are (*location*, *hypernym_of*, *city*), and (*door*, *has_part*, *lock*), where both the subject and object entities are synsets in WordNet. Examples of triples in NELL are (*New York*, *located_in*, *United States*)

[5] https://www.w3.org/TR/rdf11-concepts/

and (*New York*, *is_a*, *city*), where the subject entity is a noun phrase that can refer to a real-world entity and the object entity can be either a noun phrase entity or a concept category.

In this work, we refer to the synsets in WordNet and the concept categories in NELL as *KB concepts*. They are the key components of the ontologies and provide generally useful information for language understanding. As our KBLSTM model reads through each word in a sentence, it retrieves knowledge from NELL by searching for entities with the current word and collecting the related concept categories as candidate concepts; and it retrieves knowledge from WordNet by treating the synsets of the current word as candidate concepts.

We employ a knowledge graph embedding approach to learn the representations of the candidate concepts. Denote a KB triple as (e_1, r, e_2), we want to learn embeddings of the subject entity e_1, the object entity e_2, and the relation r, so that the relevance of the triple can be measured by a scoring function based on the embeddings. We employ the BILINEAR model described in (Yang et al., 2015).[6] It computes the score of a triple (e_1, r, e_2) via a bilinear function: $S_{(e_1,r,e_2)} = \mathbf{v}_{e_1}^T \mathbf{M}_r \mathbf{v}_{e_2}$, where $\mathbf{v}_{e_1}$ and $\mathbf{v}_{e_2}$ are vector embeddings for e_1 and e_2 respectively, and $\mathbf{M}_r$ is a relation-specific embedding matrix. We train the embeddings using the max-margin ranking objective:

$$\sum_{q=(e_1,r,e_2)\in\mathcal{T}} \sum_{q'=(e_1,r,e_2')\in\mathcal{T}'} \max\{0, 1 - S_q + S_{q'}\} \tag{7}$$

where $\mathcal{T}$ denotes the set of triples in the KB and $\mathcal{T}'$ denotes the "negative" triples that are not observed in the KB.

For WordNet, we train the concept embeddings using the preprocessed data provided by (Bordes et al., 2013), which contains 151,442 triples with 40,943 synsets and 18 relations. We use the same data splits for training, development, and testing. During training, we use AdaGrad (Duchi et al., 2011) to optimize objective 7 with an initial learning rate of 0.05 and a mini-batch size of 100. At each gradient step, we sample 10 negative object entities with respect to the positive triple. Our implementation of the BILINEAR model achieves top-10 accuracy of 91% for predicting missing object entities on the WordNet test set, which is comparable with previous work (Yang et al., 2015).

For NELL, we train the concept embeddings using a subset of the NELL data[7]. We filter noun phrases with annotation confidence less than 0.9 in order to remove erroneous labels introduced during the automatic construction of NELL (Wijaya, 2016). This results in 180,107 noun phrases and 258 concept categories in total. We randomly split 80% of the data for training, 10% for development and 10% for testing. For each training example, we enumerate all the unobserved concept categories as negative labels. Instead of treating each entity as a unit, we represent it as an average of its constituting word vectors concatenated with its head word vector. The word vectors are initialized with pre-trained paraphrastic embeddings (Wieting et al., 2015) and fine-tuned during training. Using the same optimization parameters as before, the BILINEAR model achieves 88% top-1 accuracy for predicting concept categories of given noun phrases on the test set.

4 Experiments

4.1 Entity Extraction

We first apply our model to entity extraction, a task that is typically addressed by assigning each word/token BIO labels ($Begin$, $Inside$, and $Outside$) (Ratinov and Roth, 2009) indicating the token's position within an entity mention as well as its entity type.

To allow tagging over phrases instead of words, we address entity extraction in two steps. The first step detects mention chunks, and the second step assigns entity type labels to mention chunks (including an O type indicating *other* types). In the first stage, we train a BiLSTM network with a conditional random field objective (Huang et al., 2015) using gold-standard BIO labels regardless of entity types, and only predict each token's position within an entity mention. This produces a sequence of chunks for each sentence. In the second stage, we train another supervised BiLSTM model to predict type labels for the previously extracted chunks. Each chunk is treated as a unit input to the BiLSTMs and the input vector is computed by averaging the input vectors of individual words within the chunk concatenated with its head word vector. The BiLSTMs can be trained

[6] We also experimented with TransE (Bordes et al., 2013) and NTN (Socher et al., 2013), and found that they perform significantly worse than the Bilinear method, especially on predicting the "is_a" facts in NELL.

[7] http://rtw.ml.cmu.edu/rtw/resources

with a softmax objective that minimizes the cross-entropy loss for each individual chunk. It computes the probability of the correct type for each chunk:

$$p_{y_t} = \frac{\exp(\mathbf{w}_{y_t}^T \mathbf{h}_t)}{\sum_{y'_t} \exp(\mathbf{w}_{y'_t}^T \mathbf{h}_t)} \quad (8)$$

The BiLSTMs can also be trained with a CRF objective (referred to as BiLSTM-CRF) that minimizes the negative log-likelihood for the entire sequence. It computes the probability of the correct types for a sequence of chunks:

$$p_{\mathbf{y}} = \frac{\exp(g(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp(g(\mathbf{x}, \mathbf{y}'))} \quad (9)$$

where $g(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{l} P_{t,y_t} + \sum_{t=0}^{l} A_{y_t,y_{t+1}}$, $P_{t,y_t} = \mathbf{w}_{y_t}^T \mathbf{h}_t$ is the score of assigning the t-th chunk with tag y_t and $A_{y_t,y_{t+1}}$ is the score of transitioning from tag y_t to y_{t+1}. By replacing $\mathbf{h}_t$ in Eq. 8 and Eq. 9 with the knowledge-aware state vector $\hat{\mathbf{h}}_t$ (Eq. 6), we can compute the objective for KBLSTM and KBLSTM-CRF respectively.

4.1.1 Implementation Details

We evaluate our models on the ACE2005 corpus (LDC, 2005) and the OntoNotes 5.0 corpus (Hovy et al., 2006) for entity extraction. Both datasets consist of text from a variety of sources such as newswire, broadcast conversations, and web text. We use the same data splits and task settings for ACE2005 as in Li et al. (2014) and for OntoNotes 5.0 as in Durrett and Klein (2014).

At each time step, our models take as input a word vector and a capitalization feature (Chiu and Nichols, 2016). We initialize the word vectors using pretrained paraphrastic embeddings (Wieting et al., 2015), as we find that they significantly outperforms randomly initialized embeddings. The word embeddings are fine-tuned during training. For the KBLSTM models, we obtain the embeddings of KB concepts from NELL and WordNet as described in Section § 3.3. These embeddings are kept fix during training.

We implement all the models using Theano on a single GPU. We update the model parameters on every training example using Adam with default settings (Kingma and Ba, 2014) and apply dropout to the input layer of the BiLSTM with a rate of 0.5. The word embedding dimension is set to 300 and the hidden vector dimension is set to 100. We train models on ACE2005 for about 5 epochs and on OntoNotes 5.0 for about 10 epochs with early stopping based on development results.

For each experiment, we report the average results over 10 random runs. We also apply the Wilcoxon rank sum test to compare our models with the baseline models.

Model	P	R	F1
BiLSTM	83.5	86.4	84.9
BiLSTM-CRF	87.3	84.7	86.0
BiLSTM-Fea	86.1	84.7	85.4
BiLSTM-Fea-CRF	87.7	86.1	86.9
KBLSTM	87.8	86.6	87.2
KBLSTM-CRF	**88.1**	**87.8**	**88.0***

Table 1: Entity extraction results on the ACE2005 test set with gold-standard mention boundaries.

4.1.2 Results

We compare our KBLSTM and KBLSTM-CRF models with the following baselines: BiLSTM, a vanilla BiLSTM network trained using the same input, and BiLSTM-Fea, a BiLSTM network that combines its hidden state vector with discrete KB features (i.e., indicators of candidate KB concepts) to produce the final state vector. We also include their variants BiLSTM-CRF and BiLSTM-Fea-CRF, which are trained using the CRF objective instead of the standard softmax objective.

We first report results on entity extraction with gold-standard boundaries for multi-word mentions. This allows us to focus on the performance of entity type prediction without considering mention boundary errors and the noise they introduce in retrieving candidate KB concepts. Table 1 shows the results.[8] We find that the CRF objective generally outperforms the softmax objective. Our KBLSTM-CRF model significantly improves over its counterpart BiLSTM-Fea-CRF. This demonstrates the effectiveness of KBLSTM architecture in leveraging KB information.

Table 2 breaks down the results of the KBLSTM-CRF and the BiLSTM-Fea-CRF using different KB settings. We find that the KBLSTM-CRF outperforms the BiLSTM-Fea-CRF in all the settings and that incorporating both KBs leads to the best performance.

Next, we evaluate our models on entity extraction with predicted mention boundaries. We first train a BiLSTM-CRF to perform mention

[8] $*$ indicates $p < 0.05$ when comparing to the BiLSTM-based models.

Model	KB	P	R	F1
BiLSTM-Fea-CRF	NELL	87.2	86.1	86.6
	WordNet	86.4	86.0	86.2
	Both	87.7	86.1	86.9
KBLSTM-CRF	NELL	87.4	87.6	87.5
	WordNet	87.1	87.4	87.3
	Both	**88.1**	**87.8**	**88.0**

Table 2: Ablation results with different KBs.

Model	P	R	F1
Li and Ji (2014)	85.2	76.9	80.8
Yang and Mitchell (2016)	83.5	80.2	81.8
Miwa and Bansal (2016)	82.9	83.9	83.4
BiLSTM	82.5	83.1	82.8
BiLSTM-CRF	84.6	82.5	83.6
BiLSTM-Fea	84.3	83.2	83.7
BiLSTM-Fea-CRF	84.7	83.5	84.1
KBLSTM	85.5	85.2	85.3
KBLSTM-CRF	**85.4**	**86.0**	**85.7***

Table 3: Entity extraction results on the ACE2005 test set.

Model	P	R	F1
Ratinov and Roth (2009)	82.0	84.9	83.4
Durrett and Klein (2014)	85.2	82.8	84.0
BiLSTM-CNN	82.5	82.4	82.5
BiLSTM-CNN+emb	85.9	86.3	86.1
BiLSTM-CNN+emb+lexicon	86.0	86.5	86.2
BiLSTM	84.9	86.3	85.6
BiLSTM-CRF	85.3	86.6	85.9
BiLSTM-Fea	85.2	86.4	85.8
BiLSTM-Fea-CRF	85.2	86.8	86.0
KBLSTM	**86.3**	86.2	86.2
KBLSTM-CRF	86.1	**86.8**	**86.4***

Table 4: Entity extraction results on the OntoNotes 5.0 test set.

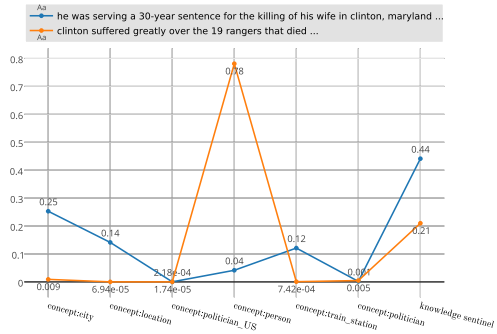
chunking using the same setting as described in Section 4.1.1. We then treat the predicted chunks as units for entity type labeling. Table 3 reports the full entity extraction results on the ACE2005 test set. We compare our models with the state-of-the-art feature-based linear models Li et al. (2014), Yang and Mitchell (2016), and the recently proposed sequence- and tree-structured LSTMs (Miwa and Bansal, 2016). Interestingly, we find that using BiLSTM-CRF without any KB information already gives strong performance compared to previous work. The KBLSTM-CRF model demonstrates the best performance among all the models and achieves the new state-of-the-art performance on the ACE2005 dataset.

We also report the entity extraction results on the OntoNotes 5.0 test set in Table 4. We compare our models with the existing feature-based models Ratinov and Roth (2009) and Durrett and Klein (2014), which both employ heavy feature engineering to bring in external knowledge. BiLSTM-CNN (Chiu and Nichols, 2016) employs a hybrid BiLSTM and Convolutional neural network (CNN) architecture and incorporates rich lexicon features derived from SENNA and DBPedia. Our KBLSTM-CRF model shows competitive results compared to their results. It also presents significant improvements compared to the BiLSTM and BiLSTM-Fea models. Note that the benefit of adding KB information is smaller on OntoNotes compared to ACE2005. We believe that there are two main reasons. One is that NELL has a lower coverage of entity mentions in OntoNotes than in ACE2005 (57% vs. 65%). Second, OntoNotes has a significantly larger amount of training data, which allows for more accurate models without much help from external resources.

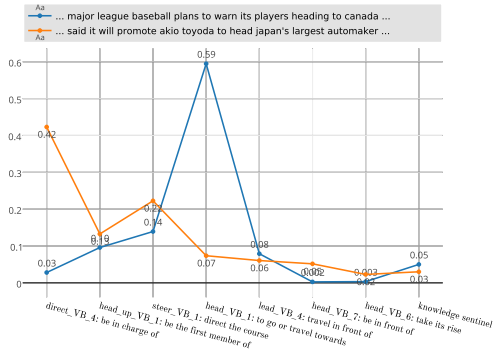
4.2 Event Extraction

We now apply our model to the task of event extraction. Event extraction is concerned with detecting *event triggers*, i.e., a word that expresses the occurrence of a pre-defined type of event. Event triggers are mostly verbs and eventive nouns but can occasionally be adjectives and other content words. Therefore, the task is typically addressed as a classification problem where the goal is to label each word in a sentence with an event type or an O type if it does not express any of the defined events. It is straightforward to apply the BiLSTM architecture to event extraction. Similarly to the models for entity extraction, we can train the BiLSTM network with both the softmax objective and the CRF objective.

We evaluate our models on the portion ACE2005 corpus that has event annotations. We use the same data split and experimental setting as in Li et al. (2013). The training procedure is the same as in Section 4.1.1, and we train all the models for about 5 epochs. For the KBLSTM models, we integrate the learned embeddings of WordNet synsets during training.



(a) The X-axis represents relevant NELL concepts for the entity mention **clinton**. The Y-axis represents the concept weights and the knowledge sentinel weight.



(b) The X-axis represents relevant WordNet concepts for the event trigger **head**. The Y-axis represents the concept weights and the knowledge sentinel weight.

Figure 2: Visualization of the attention weights for KB features learned by KBLSTM-CRF. Higher weights imply higher importance.

Model	P	R	F1
JOINTBEAM	74.0	56.7	64.2
JOINTEVENTENTITY	**75.1**	63.3	68.7
DMCNN	71.8	63.8	69.0
JRNN	66.0	73.0	69.3
BiLSTM	71.3	59.3	64.7
BiLSTM-CRF	64.2	66.6	65.4
BiLSTM-Fea	68.4	62.7	65.5
BiLSTM-Fea-CRF	65.5	66.7	66.1
KBLSTM	70.1	67.3	68.7
KBLSTM-CRF	71.6	67.8	**69.7***

Table 5: event extraction results on the ACE2005 test set.

4.2.1 Results

We compare our models with the prior state-of-the-art approaches for event extraction, including neural and non-neural ones: JOINTBEAM refers to the joint beam search approach with local and global features (Li et al., 2013); JOINTENTITYEVENT refers to the graphical model for joint entity and event extraction (Yang and Mitchell, 2016); DMCNN is the dynamic multi-pooling CNNs in Chen et al. (2015); and JRNN is an RNN model with memory introduced by Nguyen et al. (2016). The first block in Table 5 shows the results of the feature-based linear models (taken from Yang and Mitchell (2016)). The second block shows the previously reported results for the neural models. Note that they both make use of gold-standard entity annotations. The third block shows the results of our models. We can see that our KBLSTM models significantly outperform the BiLSTM and BiLSTM-Fea models, which again confirms their effectiveness in leveraging KB information. The KBLSTM-CRF model achieves the best performance and outperforms the previous state-of-the-art methods without having access to any gold-standard entities.

4.3 Model Analysis

In order to better understand our model, we visualize the learned attention weights $\boldsymbol{\alpha}$ for KB concepts and the sentinel weight β that measures the tradeoff between knowledge and context. Figure 2a visualizes these weights for the entity mention "clinton". In the first sentence, "clinton" refers to a LOCATION while in the second sentence, "clinton" refers to a PERSON. Our model learns to attend to different word senses for 'clinton' (KB concepts associated with 'clinton') in different sentences. Note that the weight on the knowledge sentinel is higher in the first sentence. This is because the local text alone is indicative of the entity type for "clinton": the word "in" is more likely to be followed by a location than a person. We find that BiLSTM-Fea-CRF models often make wrong predictions on examples like this due to its inflexibility in modeling knowledge relevance with respect to context. Figure 2b shows the learned weights for the event trigger word "head" in two sentences, one expresses a TRAVEL event while the other expresses a STARTPOSITION event. Again, we find that our model is capable of attending to relevant WordNet synsets and more accurately disambiguate event types.

5 Conclusion

In this paper, we introduce the KBLSTM network architecture as one approach to incorporating background KBs for improving recurrent neural networks for machine reading. This architecture employs an adaptive attention mechanism with a sentinel that allows for learning an appropriate tradeoff for blending knowledge from the KBs with information from the currently processed text, as well as selecting among relevant KB concepts for each word (e.g., to select the correct semantic categories for "clinton" as a town or person in figure 2a). Experimental results show that our model achieves state-of-the-art performance on standard benchmarks for both entity extraction and event extraction.

We see many additional opportunities to integrate background knowledge with training of neural network models for language processing. Though our model is evaluated on entity extraction and event extraction, it can be useful for other machine reading tasks. Our model can also be extended to integrate knowledge from a richer set of KBs in order to capture the diverse variety and depth of background knowledge required for accurate and deep language understanding.

Acknowledgments

This research was supported in part by DARPA under contract number FA8750-13-2-0005, and by NSF grants IIS-1065251 and IIS-1247489. We also gratefully acknowledge the support of the Microsoft Azure for Research program and the AWS Cloud Credits for Research program. In addition, we would like to thank anonymous reviewers for their helpful comments.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*. pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 167–176.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics* 4:357–370.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics* 2:477–490.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

Charles J Fillmore. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences* 280(1):20–32.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*. Springer, pages 799–804.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*. pages 1693–1701.

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

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*. pages 57–60.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations (ICLR)* .

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 (NAACL)*.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 529–539.

LDC. 2005. The ace 2005 evaluation plan. In *NIST*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 402–412.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1846–1851.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 73–82.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Marvin Minsky. 1988. *Society of mind*. Simon and Schuster.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)* .

Ndapandula Nakashole and Tom M Mitchell. 2015. A knowledge-intensive model for prepositional phrase attachment. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 365–375.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*. pages 300–309.

Altaf Rahman and Vincent Ng. 2011. Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 814–824.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)*. pages 147–155.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of HLT-NAACL*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems (NIPS)*. pages 926–934.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Association for Computational Linguistics (ACL)*.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *Proceedings of the International Conference on Learning Representations (ICLR)* .

Derry Tanti Wijaya. 2016. *VerbKB: A Knowledge Base of Verbs for Natural Language Understanding*. Ph.D. thesis, Carnegie Mellon University.

WordNet. 2010. About wordnet. http://wordnet.princeton.edu.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference for Machine Learning (ICML)*.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*. pages 289–299.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *International Conference on Learning Representations (ICLR)* .

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.