Knowledge Enhanced Contextual Word Representations

Matthew E. Peters[1], Mark Neumann[1], Robert L. Logan IV[2], Roy Schwartz[1,3],
Vidur Joshi[1], Sameer Singh[2], and Noah A. Smith[1,3]

[1]Allen Institute for Artificial Intelligence, Seattle, WA, USA
[2]University of California, Irvine, CA, USA
[3]Paul G. Allen School of Computer Science & Engineering, University of Washington
{matthewp,markn,roys,noah}@allenai.org
{rlogan,sameer}@uci.edu

Abstract

Contextual word representations, typically trained on unstructured, unlabeled text, do not contain any explicit grounding to real world entities and are often unable to remember facts about those entities. We propose a general method to embed multiple knowledge bases (KBs) into large scale models, and thereby enhance their representations with structured, human-curated knowledge. For each KB, we first use an integrated entity linker to retrieve relevant entity embeddings, then update contextual word representations via a form of word-to-entity attention. In contrast to previous approaches, the entity linkers and self-supervised language modeling objective are jointly trained end-to-end in a multitask setting that combines a small amount of entity linking supervision with a large amount of raw text. After integrating WordNet and a subset of Wikipedia into BERT, the knowledge enhanced BERT (KnowBert) demonstrates improved perplexity, ability to recall facts as measured in a probing task and downstream performance on relationship extraction, entity typing, and word sense disambiguation. KnowBert's runtime is comparable to BERT's and it scales to large KBs.

1 Introduction

Large pretrained models such as ELMo (Peters et al., 2018), GPT (Radford et al., 2018), and BERT (Devlin et al., 2019) have significantly improved the state of the art for a wide range of NLP tasks. These models are trained on large amounts of raw text using self-supervised objectives. However, they do not contain any explicit grounding to real world entities and as a result have difficulty recovering factual knowledge (Logan et al., 2019).

Knowledge bases (KBs) provide a rich source of high quality, human-curated knowledge that can be used to ground these models. In addition, they often include complementary information to that found in raw text, and can encode factual knowledge that is difficult to learn from selectional preferences either due to infrequent mentions of commonsense knowledge or long range dependencies.

We present a general method to insert multiple KBs into a large pretrained model with a Knowledge Attention and Recontextualization (KAR) mechanism. The key idea is to explicitly model *entity spans* in the input text and use an entity linker to retrieve relevant entity embeddings from a KB to form knowledge enhanced entity-span representations. Then, the model recontextualizes the entity-span representations with word-to-entity attention to allow long range interactions between contextual word representations and all entity spans in the context. The entire KAR is inserted between two layers in the middle of a pretrained model such as BERT.

In contrast to previous approaches that integrate external knowledge into task-specific models with task supervision (e.g., Yang and Mitchell, 2017; Chen et al., 2018), our approach learns the entity linkers with self-supervision on unlabeled data. This results in general purpose knowledge enhanced representations that can be applied to a wide range of downstream tasks.

Our approach has several other benefits. First, it leaves the top layers of the original model unchanged so we may retain the output loss layers and fine-tune on unlabeled corpora while training the KAR. This also allows us to simply swap out BERT for KnowBert in any downstream application. Second, by taking advantage of the existing high capacity layers in the original model, the KAR is lightweight, adding minimal additional parameters and runtime. Finally, it is easy to incorporate additional KBs by simply inserting them at other locations.

KnowBert is agnostic to the form of the

KB, subject to a small set of requirements (see Sec. 3.2). We experiment with integrating both WordNet (Miller, 1995) and Wikipedia, thus explicitly adding word sense knowledge and facts about named entities (including those unseen at training time). However, the method could be extended to commonsense KBs such as ConceptNet (Speer et al., 2017) or domain specific ones (e.g., UMLS; Bodenreider, 2004).

We evaluate KnowBert with a mix of intrinsic and extrinsic tasks. Despite being based on the smaller BERT$_{\text{BASE}}$ model, the experiments demonstrate improved masked language model perplexity and ability to recall facts over BERT$_{\text{LARGE}}$. The extrinsic evaluations demonstrate improvements for challenging relationship extraction, entity typing and word sense disambiguation datasets, and often outperform other contemporaneous attempts to incorporate external knowledge into BERT.

2 Related Work

Pretrained word representations Initial work learning word vectors focused on static word embeddings using multi-task learning objectives (Collobert and Weston, 2008) or corpus level co-occurence statistics (Mikolov et al., 2013a; Pennington et al., 2014). Recently the field has shifted toward learning context-sensitive embeddings (Dai and Le, 2015; McCann et al., 2017; Peters et al., 2018; Devlin et al., 2019). We build upon these by incorporating structured knowledge into these models.

Entity embeddings Entity embedding methods produce continuous vector representations from external knowledge sources. Knowledge graph-based methods optimize the score of observed triples in a knowledge graph. These methods broadly fall into two categories: translational distance models (Bordes et al., 2013; Wang et al., 2014b; Lin et al., 2015; Xiao et al., 2016) which use a distance-based scoring function, and linear models (Nickel et al., 2011; Yang et al., 2014; Trouillon et al., 2016; Dettmers et al., 2018) which use a similarity-based scoring function. We experiment with TuckER (Balazevic et al., 2019) embeddings, a recent linear model which generalizes many of the aforecited models. Other methods combine entity metadata with the graph (Xie et al., 2016), use entity contexts (Chen et al., 2014; Ganea and Hofmann, 2017), or a combination of

contexts and the KB (Wang et al., 2014a; Gupta et al., 2017). Our approach is agnostic to the details of the entity embedding method and as a result is able to use any of these methods.

Entity-aware language models Some previous work has focused on adding KBs to generative language models (LMs) (Ahn et al., 2017; Yang et al., 2017; Logan et al., 2019) or building entity-centric LMs (Ji et al., 2017). However, these methods introduce latent variables that require full annotation for training, or marginalization. In contrast, we adopt a method that allows training with large amounts of unannotated text.

Task-specific KB architectures Other work has focused on integrating KBs into neural architectures for specific downstream tasks (Yang and Mitchell, 2017; Sun et al., 2018; Chen et al., 2018; Bauer et al., 2018; Mihaylov and Frank, 2018; Wang and Jiang, 2019; Yang et al., 2019). Our approach instead uses KBs to learn more generally transferable representations that can be used to improve a variety of downstream tasks.

3 KnowBert

KnowBert incorporates knowledge bases into BERT using the Knowledge Attention and Recontextualization component (KAR). We start by describing the BERT and KB components. We then move to introducing KAR. Finally, we describe the training procedure, including the multitask training regime for jointly training KnowBert and an entity linker.

3.1 Pretrained BERT

We describe KnowBert as an extension to (and candidate replacement for) BERT, although the method is general and can be applied to any deep pretrained model including left-to-right and right-to-left LMs such as ELMo and GPT. Formally, BERT accepts as input a sequence of N Word-Piece tokens (Sennrich et al., 2016; Wu et al., 2016), $(x_1, \ldots, x_N)$, and computes L layers of D-dimensional contextual representations $\mathbf{H}_i \in \mathbb{R}^{N \times D}$ by successively applying non-linear functions $\mathbf{H}_i = \mathrm{F}_i(\mathbf{H}_{i-1})$. The non-linear function is a multi-headed self-attention layer followed by a position-wise multilayer perceptron (MLP)

(Vaswani et al., 2017):

$$F_i(\mathbf{H}_{i-1}) =$$
$$\text{TransformerBlock}(\mathbf{H}_{i-1}) =$$
$$\text{MLP}(\text{MultiHeadAttn}(\mathbf{H}_{i-1}, \mathbf{H}_{i-1}, \mathbf{H}_{i-1})).$$

The multi-headed self-attention uses $\mathbf{H}_{i-1}$ as the query, key, and value to allow each vector to attend to every other vector.

BERT is trained to minimize an objective function that combines both next-sentence prediction (NSP) and masked LM log-likelihood (MLM):

$$\mathcal{L}_{\text{BERT}} = \mathcal{L}_{\text{NSP}} + \mathcal{L}_{\text{MLM}}.$$

Given two inputs $\mathbf{x}_A$ and $\mathbf{x}_B$, the next-sentence prediction task is binary classification to predict whether $\mathbf{x}_B$ is the next sentence following $\mathbf{x}_A$. The masked LM objective randomly replaces a percentage of input word pieces with a special [MASK] token and computes the negative log-likelihood of the missing token with a linear layer and softmax over all possible word pieces.

3.2 Knowledge Bases

The key contribution of this paper is a method to incorporate knowledge bases (KB) into a pre-trained BERT model. To encompass as wide a selection of prior knowledge as possible, we adopt a broad definition for a KB in the most general sense as fixed collection of K entity nodes, e_k, from which it is possible to compute entity embeddings, $\mathbf{e}_k \in \mathbb{R}^E$. This includes KBs with a typical (subj, rel, obj) graph structure, KBs that contain only entity metadata without a graph, and those that combine both a graph and entity metadata, as long as there is some method for embedding the entities in a low dimensional vector space. We also do not make any assumption that the entities are typed. As we show in Sec. 4.1 this flexibility is beneficial, where we compute entity embeddings from WordNet using both the graph and synset definitions, but link directly to Wikipedia pages without a graph by using embeddings computed from the entity description.

We also assume that the KB is accompanied by an entity candidate selector that takes as input some text and returns a list of C potential entity links, each consisting of the start and end indices of the potential mention span and M_m candidate entities in the KG:

$$\mathcal{C} = \{\langle(\text{start}_m, \text{end}_m), (e_{m,1}, \dots, e_{m,M_m})\rangle \mid$$
$$m \in 1 \dots C, e_k \in 1 \dots K\}.$$

In practice, these are often implemented using precomputed dictionaries (e.g., CrossWikis; Spitkovsky and Chang, 2012), KB specific rules (e.g., a WordNet lemmatizer), or other heuristics (e.g., string match; Mihaylov and Frank, 2018). Ling et al. (2015) showed that incorporating candidate priors into entity linkers can be a powerful signal, so we optionally allow for the candidate selector to return an associated prior probability for each entity candidate. In some cases, it is beneficial to over-generate potential candidates and add a special NULL entity to each candidate list, thereby allowing the linker to discriminate between actual links and false positive candidates. In this work, the entity candidate selectors are fixed but their output is passed to a learned context dependent entity linker to disambiguate the candidate mentions.

Finally, by restricting the number of candidate entities to a fixed small number (we use 30), KnowBert's runtime is independent of the size the KB, as it only considers a small subset of all possible entities for any given text. As the candidate selection is rule-based and fixed, it is fast and in our implementation is performed asynchronously on CPU. The only overhead for scaling up the size of the KB is the memory footprint to store the entity embeddings.

3.3 KAR

The Knowledge Attention and Recontextualization component (KAR) is the heart of KnowBert. The KAR accepts as input the contextual representations at a particular layer, $\mathbf{H}_i$, and computes knowledge enhanced representations $\mathbf{H}'_i = \text{KAR}(\mathbf{H}_i, \mathcal{C})$. This is fed into the next pretrained layer, $\mathbf{H}_{i+1} = \text{TransformerBlock}(\mathbf{H}'_i)$, and the remainder of BERT is run as usual.

In this section, we describe the KAR's key components: mention-span representations, retrieval of relevant entity embeddings using an entity linker, update of mention-span embeddings with retrieved information, and recontextualization of entity-span embeddings with word-to-entity-span attention. We describe the KAR for a single KB, but extension to multiple KBs at different layers is straightforward. See Fig. 1 for an overview.

Mention-span representations The KAR starts with the KB entity candidate selector that provides a list of candidate mentions which it uses to compute mention-span representations. $\mathbf{H}_i$ is first pro-
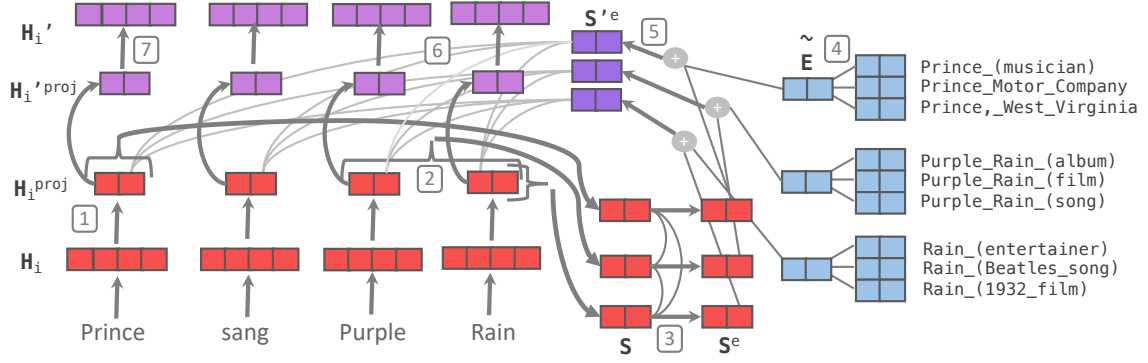
Figure 1: The Knowledge Attention and Recontextualization (KAR) component. BERT word piece representations ($\mathbf{H}_i$) are first projected to $\mathbf{H}_i^{\text{proj}}$ (1), then pooled over candidate mentions spans (2) to compute $\mathbf{S}$, and contextualized into $\mathbf{S}^e$ using mention-span self-attention (3). An integrated entity linker computes weighted average entity embeddings $\tilde{\mathbf{E}}$ (4), which are used to enhance the span representations with knowledge from the KB (5), computing $\mathbf{S}'^e$. Finally, the BERT word piece representations are recontextualized with word-to-entity-span attention (6) and projected back to the BERT dimension (7) resulting in $\mathbf{H}'_i$.

jected to the entity dimension (E, typically 200 or 300, see Sec. 4.1) with a linear projection,

$$\mathbf{H}_i^{\text{proj}} = \mathbf{H}_i \mathbf{W}_1^{\text{proj}} + \mathbf{b}_1^{\text{proj}}. \qquad (1)$$

Then, the KAR computes C mention-span representations $\mathbf{s}_m \in \mathbb{R}^E$, one for each candidate mention, by pooling over all word pieces in a mention-span using the self-attentive span pooling from Lee et al. (2017). The mention-spans are stacked into a matrix $\mathbf{S} \in \mathbb{R}^{C \times E}$.

Entity linker The entity linker is responsible for performing entity disambiguation for each potential mention from among the available candidates. It first runs mention-span self-attention to compute

$$\mathbf{S}^e = \text{TransformerBlock}(\mathbf{S}). \qquad (2)$$

The span self-attention is identical to the typical transformer layer, exception that the self-attention is between mention-span vectors instead of word piece vectors. This allows KnowBert to incorporate global information into each linking decision so that it can take advantage of entity-entity co-occurrence and resolve which of several overlapping candidate mentions should be linked.[1] Following Kolitsas et al. (2018), $\mathbf{S}^e$ is used to score each of the candidate entities while incorporating the candidate entity prior from the KB. Each candidate span m has an associated mention-span

vector $\mathbf{s}_m^e$ (computed via Eq. 2), M_m candidate entities with embeddings $\mathbf{e}_{mk}$ (from the KB), and prior probabilities p_{mk}. We compute M_m scores using the prior and dot product between the entity-span vectors and entity embeddings,

$$\psi_{mk} = \text{MLP}(p_{mk}, \mathbf{s}_m^e \cdot \mathbf{e}_{mk}), \qquad (3)$$

with a two-layer MLP (100 hidden dimensions).

If entity linking (EL) supervision is available, we can compute a loss with the gold entity e_{mg}. The exact form of the loss depends on the KB, and we use both log-likelihood,

$$\mathcal{L}_{\text{EL}} = -\sum_m \log \left(\frac{\exp(\psi_{mg})}{\sum_k \exp(\psi_{mk})} \right), \qquad (4)$$

and max-margin,

$$\mathcal{L}_{\text{EL}} = \max(0, \gamma - \psi_{mg}) + \sum_{e_{mk} \neq e_{mg}} \max(0, \gamma + \psi_{mk}), \qquad (5)$$

formulations (see Sec. 4.1 for details).

Knowledge enhanced entity-span representations KnowBert next injects the KB entity information into the mention-span representations computed from BERT vectors ($\mathbf{s}_m^e$) to form entity-span representations. For a given span m, we first disregard all candidate entities with score ψ below a fixed threshold, and softmax normalize the remaining scores:

$$\tilde{\psi}_{mk} = \begin{cases} \dfrac{\exp(\psi_{mk})}{\sum\limits_{\psi_{mk} \geq \delta} \exp(\psi_{mk})}, & \psi_{mk} \geq \delta \\ 0, & \psi_{mk} < \delta. \end{cases}$$

---

[1] We found a small transformer layer with four attention heads and a 1024 feed-forward hidden dimension was sufficient, significantly smaller than each of the layers in BERT. Early experiments demonstrated the effectiveness of this layer with improved entity linking performance.

Then the weighted entity embedding is

$$\tilde{\mathbf{e}}_m = \sum_k \tilde{\psi}_{mk} \mathbf{e}_{mk}.$$

If all entity linking scores are below the threshold δ, we substitute a special `NULL` embedding for $\tilde{\mathbf{e}}_m$. Finally, the entity-span representations are updated with the weighted entity embeddings

$$\mathbf{s}'^e_m = \mathbf{s}^e_m + \tilde{\mathbf{e}}_m, \qquad (6)$$

which are packed into a matrix $\mathbf{S}'^e \in \mathbb{R}^{C \times E}$.

Recontextualization   After updating the entity-span representations with the weighted entity vectors, KnowBert uses them to recontextualize the word piece representations. This is accomplished using a modified transformer layer that substitutes the multi-headed self-attention with a multi-headed attention between the projected word piece representations and knowledge enhanced entity-span vectors. As introduced by Vaswani et al. (2017), the contextual embeddings $\mathbf{H}_i$ are used for the query, key, and value in multi-headed self-attention. The word-to-entity-span attention in KnowBert substitutes $\mathbf{H}^{\text{proj}}_i$ for the query, and $\mathbf{S}'^e$ for both the key and value:

$$\mathbf{H}'^{\text{proj}}_i = \text{MLP}(\text{MultiHeadAttn}(\mathbf{H}^{\text{proj}}_i, \mathbf{S}'^e, \mathbf{S}'^e)).$$

This allows each word piece to attend to all entity-spans in the context, so that it can propagate entity information over long contexts. After the multi-headed word-to-entity-span attention, we run a position-wise MLP analogous to the standard transformer layer.[2]

Finally, $\mathbf{H}'^{\text{proj}}_i$ is projected back to the BERT dimension with a linear transformation and a residual connection added:

$$\mathbf{H}'_i = \mathbf{H}'^{\text{proj}}_i \mathbf{W}^{\text{proj}}_2 + \mathbf{b}^{\text{proj}}_2 + \mathbf{H}_i \qquad (7)$$

Alignment of BERT and entity vectors   As KnowBert does not place any restrictions on the entity embeddings, it is essential to align them with the pretrained BERT contextual representations. To encourage this alignment we initialize $\mathbf{W}^{\text{proj}}_2$ as the matrix inverse of $\mathbf{W}^{\text{proj}}_1$ (Eq. 1). The use of dot product similarity (Eq. 3) and residual connection (Eq. 7) further aligns the entity-span representations with entity embeddings.

---

[2]As for the multi-headed entity-span self-attention, we found a small transformer layer to be sufficient, with four attention heads and 1024 hidden units in the MLP.

Algorithm 1: KnowBert training method

Input: Pretrained BERT and J KBs
Output: KnowBert
for $j = 1 \dots J$ **do**
  Compute entity embeddings for KB$_j$
  if EL supervision available **then**
    Freeze all network parameters except those in (Eq. 1–3)
    Train to convergence using (Eq. 4) or (Eq. 5)
  end
  Initialize $\mathbf{W}^{\text{proj}}_2$ as $(\mathbf{W}^{\text{proj}}_1)^{-1}$
  Unfreeze all parameters except entity embeddings
  Minimize $\mathcal{L}_{\text{KnowBert}} = \mathcal{L}_{\text{BERT}} + \sum_{i=1}^{j} \mathcal{L}_{\text{EL}_i}$
end

3.4 Training Procedure

Our training regime incrementally pretrains increasingly larger portions of KnowBert before fine-tuning all trainable parameters in a multitask setting with any available EL supervision. It is similar in spirit to the "chain-thaw" approach in Felbo et al. (2017), and is summarized in Alg. 1.

We assume access to a pretrained BERT model and one or more KBs with their entity candidate selectors. To add the first KB, we begin by pretraining entity embeddings (if not already provided from another source), then freeze them in all subsequent training, including task-specific fine-tuning. If EL supervision is available, it is used to pretrain the KB specific EL parameters, while freezing the remainder of the network. Finally, the entire network is fine-tuned to convergence by minimizing

$$\mathcal{L}_{\text{KnowBert}} = \mathcal{L}_{\text{BERT}} + \mathcal{L}_{\text{EL}}.$$

We apply gradient updates to homogeneous batches randomly sampled from either the unlabeled corpus or EL supervision.

To add a second KB, we repeat the process, inserting it in any layer above the first one. When adding a KB, the BERT layers above it will experience large gradients early in training, as they are subject to the randomly initialized parameters associated with the new KB. They are thus expected to move further from their pretrained values before convergence compared to parameters below the KB. By adding KBs from bottom to top, we

System	PPL	Wikidata MRR	# params. masked LM	# params. KAR	# params. entity embed.	Fwd. / Bwd. time
BERT$_{\text{BASE}}$	5.5	0.09	110	0	0	0.25
BERT$_{\text{LARGE}}$	4.5	0.11	336	0	0	0.75
KnowBert-Wiki	4.3	0.26	110	2.4	141	0.27
KnowBert-WordNet	4.1	0.22	110	4.9	265	0.31
KnowBert-W+W	**3.5**	**0.31**	110	7.3	406	0.33

Table 1: Comparison of masked LM perplexity, Wikidata probing MRR, and number of parameters (in millions) in the masked LM (word piece embeddings, transformer layers, and output layers), KAR, and entity embeddings for BERT and KnowBert. The table also includes the total time to run one forward and backward pass (in seconds) on a TITAN Xp GPU (12 GB RAM) for a batch of 32 sentence pairs with total length 80 word pieces. Due to memory constraints, the BERT$_{\text{LARGE}}$ batch is accumulated over two smaller batches.

minimize disruption of the network and decrease the likelihood that training will fail. See Sec. 4.1 for details of where each KB was added.

The entity embeddings and selected candidates contain lexical information (especially in the case of WordNet), that will make the masked LM predictions significantly easier. To prevent leaking into the masked word pieces, we adopt the BERT strategy and replace all entity candidates from the selectors with a special [MASK] entity if the candidate mention span overlaps with a masked word piece.[3] This prevents KnowBert from relying on the selected candidates to predict masked word pieces.

4 Experiments

4.1 Experimental Setup

We used the English uncased BERT$_{\text{BASE}}$ model (Devlin et al., 2019) to train three versions of KnowBert: KnowBert-Wiki, KnowBert-WordNet, and KnowBert-W+W that includes both Wikipedia and WordNet.

KnowBert-Wiki The entity linker in KnowBert-Wiki borrows both the entity candidate selectors and embeddings from Ganea and Hofmann (2017). The candidate selectors and priors are a combination of CrossWikis, a large, precomputed dictionary that combines statistics from Wikipedia and a web corpus (Spitkovsky and Chang, 2012), and the YAGO dictionary (Hoffart et al., 2011). The entity embeddings use a skip-gram like objective (Mikolov et al., 2013b) to learn 300-dimensional embeddings of Wikipedia page titles directly from Wikipedia descriptions without using any explicit graph structure between nodes. As such, nodes in the KB are Wikipedia page titles, e.g., `Prince_(musician)`. Ganea and Hofmann (2017) provide pretrained embeddings for a subset of approximately 470K entities. Early experiments with embeddings derived from Wikidata relations[4] did not improve results.

We used the AIDA-CoNLL dataset (Hoffart et al., 2011) for supervision, adopting the standard splits. This dataset exhaustively annotates entity links for named entities of person, organization and location types, as well as a miscellaneous type. It does not annotate links to common nouns or other Wikipedia pages. At both train and test time, we consider all selected candidate spans and the top 30 entities, to which we add the special NULL entity to allow KnowBert to discriminate between actual links and false positive links from the selector. As such, KnowBert models both entity mention detection and disambiguation in an end-to-end manner. Eq. 5 was used as the objective.

KnowBert-WordNet Our WordNet KB combines synset metadata, lemma metadata and the relational graph. To construct the graph, we first extracted all synsets, lemmas, and their relationships from WordNet 3.0 using the `nltk` interface. After disregarding certain symmetric relationships (e.g., we kept the `hypernym` relationship, but removed the inverse `hyponym` relationship) we were left with 28 synset-synset and lemma-lemma relationships. From these, we constructed a graph where each node is either a synset or lemma, and intro-

---

[3]Following BERT, for 80% of masked word pieces all candidates are replaced with [MASK], 10% are replaced with random candidates and 10% left unmasked.

[4]https://github.com/facebookresearch/PyTorch-BigGraph

System	F_1
WN-first sense baseline	65.2
ELMo	69.2
BERT$_{BASE}$	73.1
BERT$_{LARGE}$	73.9
KnowBert-WordNet	74.9
KnowBert-W+W	**75.1**

Table 2: Fine-grained WSD F_1.

System	AIDA-A	AIDA-B
Daiber et al. (2013)	49.9	52.0
Hoffart et al. (2011)	68.8	71.9
Kolitsas et al. (2018)	**86.6**	**82.6**
KnowBert-Wiki	80.2	74.4
KnowBert-W+W	82.1	73.7

Table 3: End-to-end entity linking strong match, micro averaged F_1.

duced the special `lemma_in_synset` relationship to link synsets and lemmas. The candidate selector uses a rule-based lemmatizer without part-of-speech (POS) information.[5]

Our embeddings combine both the graph and synset glosses (definitions), as early experiments indicated improved perplexity when using both vs. just graph-based embeddings. We used TuckER (Balazevic et al., 2019) to compute 200-dimensional vectors for each synset and lemma using the relationship graph. Then, we extracted the gloss for each synset and used an off-the-shelf state-of-the-art sentence embedding method (Subramanian et al., 2018) to produce 2048-dimensional vectors. These are concatenated to the TuckER embeddings. To reduce the dimensionality for use in KnowBert, the frozen 2248-dimensional embeddings are projected to 200-dimensions with a learned linear transformation.

For supervision, we combined the SemCor word sense disambiguation (WSD) dataset (Miller et al., 1994) with all lemma example usages from WordNet[6] and link directly to synsets. The loss function is Eq. 4. At train time, we did not provide gold lemmas or POS tags, so KnowBert must learn to implicitly model coarse grained POS tags to disambiguate each word. At test time when evaluating we restricted candidate entities to just those matching the gold lemma and POS tag, consistent with the standard WSD evaluation.

Training details To control for the unlabeled corpus, we concatenated Wikipedia and the Books Corpus (Zhu et al., 2015) and followed the data preparation process in BERT with the exception of heavily biasing our dataset to shorter sequences of 128 word pieces for efficiency. Both KnowBert-

---

[5] https://spacy.io/
[6] To provide a fair evaluation on the WiC dataset which is partially based on the same source, we excluded all WiC train, development and test instances.

Wiki and KnowBert-WordNet insert the KB between layers 10 and 11 of the 12-layer BERT$_{BASE}$ model. KnowBert-W+W adds the Wikipedia KB between layers 10 and 11, with WordNet between layers 11 and 12. Earlier experiments with KnowBert-WordNet in a lower layer had worse perplexity. We generally followed the fine-tuning procedure in Devlin et al. (2019); see supplemental materials for details.

4.2 Intrinsic Evaluation

Perplexity Table 1 compares masked LM perplexity for KnowBert with BERT$_{BASE}$ and BERT$_{LARGE}$. To rule out minor differences due to our data preparation, the BERT models are fine-tuned on our training data before being evaluated. Overall, KnowBert improves the masked LM perplexity, with all KnowBert models outperforming BERT$_{LARGE}$, despite being derived from BERT$_{BASE}$.

Factual recall To test KnowBert's ability to recall facts from the KBs, we extracted 90K tuples from Wikidata (Vrandečić and Krötzsch, 2014) for 17 different relationships such as `companyFoundedBy`. Each tuple was written into natural language such as "Adidas was founded by Adolf Dassler" and used to construct two test instances, one that masks out the subject and one that masks the object. Then, we evaluated whether a model could recover the masked entity by computing the mean reciprocal rank (MRR) of the masked word pieces. Table 1 displays a summary of the results (see supplementary material for results across all relationship types). Overall, KnowBert-Wiki is significantly better at recalling facts than both BERT$_{BASE}$ and BERT$_{LARGE}$, with KnowBert-W+W better still.

Speed KnowBert is almost as fast as BERT$_{BASE}$ (8% slower for KnowBert-Wiki, 32% for KnowBert-W+W) despite adding a large number

System	LM	P	R	F_1
Zhang et al. (2018)	—	69.9	63.3	66.4
Alt et al. (2019)	GPT	70.1	65.0	67.4
Shi and Lin (2019)	BERT$_{BASE}$	73.3	63.1	67.8
Zhang et al. (2019)	BERT$_{BASE}$	70.0	66.1	68.0
Soares et al. (2019)	BERT$_{LARGE}$	—	—	70.1
Soares et al. (2019)	BERT$_{LARGE}$†	—	—	**71.5**
KnowBert-W+W	BERT$_{BASE}$	71.6	71.4	**71.5**

Table 4: Single model test set results on the TACRED relationship extraction dataset. † with MTB pretraining.

System	LM	F_1
Wang et al. (2016)	—	88.0
Wang et al. (2019b)	BERT$_{BASE}$	89.0
Soares et al. (2019)	BERT$_{LARGE}$	89.2
Soares et al. (2019)	BERT$_{LARGE}$†	**89.5**
KnowBert-W+W	BERT$_{BASE}$	89.1

Table 5: Test set F_1 for SemEval 2010 Task 8 relationship extraction. † with MTB pretraining.

of (frozen) parameters in the entity embeddings (Table 1). KnowBert is much faster than BERT$_{LARGE}$. By taking advantage of the already high capacity model, the number of trainable parameters added by KnowBert is a fraction of the total parameters in BERT. The faster speed is partially due to the entity parameter efficiency in KnowBert as only as small fraction of parameters in the entity embeddings are used for any given input due to the sparse linker. Our candidate generators consider the top 30 candidates and produce approximately $O(\text{number tokens})$ candidate spans. For a typical 25 token sentence, approximately 2M entity embedding parameters are actually used. In contrast, BERT$_{LARGE}$ uses the majority of its 336M parameters for each input.

Integrated EL It is also possible to evaluate the performance of the integrated entity linkers inside KnowBert using diagnostic probes without any further fine-tuning. As these were trained in a multitask setting primarily with raw text, we do not *a priori* expect high performance as they must balance specializing for the entity linking task and learning general purpose representations suitable for language modeling.

Table 2 displays fine-grained WSD F_1 using the evaluation framework from Navigli et al. (2017) and the ALL dataset (combing SemEval 2007, 2013, 2015 and Senseval 2 and 3). By linking to nodes in our WordNet graph and restricting to gold lemmas at test time we can recast the WSD task under our general entity linking framework. The ELMo and BERT baselines use a nearest neighbor approach trained on the SemCor dataset, similar to the evaluation in Melamud et al. (2016), which has previously been shown to be competitive with task-specific architectures (Raganato et al., 2017). As can be seen, KnowBert provides competitive performance, and KnowBert-W+W is able to

match the performance of KnowBert-WordNet despite incorporating both Wikipedia and WordNet.

Table 3 reports end-to-end entity linking performance for the AIDA-A and AIDA-B datasets. Here, KnowBert's performance lags behind the current state-of-the-art model from Kolitsas et al. (2018), but still provides strong performance compared to other established systems such as AIDA (Hoffart et al., 2011) and DBpedia Spotlight (Daiber et al., 2013). We believe this is due to the selective annotation in the AIDA data that only annotates named entities. The CrossWikis-based candidate selector used in KnowBert generates candidate mentions for all entities including common nouns from which KnowBert may be learning to extract information, at the detriment of specializing to maximize linking performance for AIDA.

4.3 Downstream Tasks

This section evaluates KnowBert on downstream tasks to validate that the addition of knowledge improves performance on tasks expected to benefit from it. Given the overall superior performance of KnowBert-W+W on the intrinsic evaluations, we focus on it exclusively for evaluation in this section. The main results are included in this section; see the supplementary material for full details.

The baselines we compare against are BERT$_{BASE}$, BERT$_{LARGE}$, the pre-BERT state of the art, and two contemporaneous papers that add similar types of knowledge to BERT. ERNIE (Zhang et al., 2019) uses TAGME (Ferragina and Scaiella, 2010) to link entities to Wikidata, retrieves the associated entity embeddings, and fuses them into BERT$_{BASE}$ by fine-tuning. Soares et al. (2019) learns relationship representations by fine-tuning BERT$_{LARGE}$ with large scale "matching the blanks" (MTB) pretraining using entity linked text.

System	Accuracy
ELMo†	57.7
$\text{BERT}_{\text{BASE}}$†	65.4
$\text{BERT}_{\text{LARGE}}$†	65.5
$\text{BERT}_{\text{LARGE}}$††	69.5
KnowBert-W+W	**70.9**

Table 6: Test set results for the WiC dataset (v1.0).
†Pilehvar and Camacho-Collados (2019)
††Wang et al. (2019a)

Relation extraction Our first task is relation extraction using the TACRED (Zhang et al., 2017) and SemEval 2010 Task 8 (Hendrickx et al., 2009) datasets. Systems are given a sentence with marked a subject and object, and asked to predict which of several different relations (or no relation) holds. Following Soares et al. (2019), our KnowBert model uses special entity tokens `[E1]`, `[/E1]`, `[E2]`, `[/E2]` to mark the location of the subject and object in the input sentence, then concatenates the contextual word representations for `[E1]` and `[E2]` to predict the relationship. For TACRED, we also encode the subject and object types with special tokens and concatenate them to the end of the sentence.

For TACRED (Table 4), KnowBert-W+W significantly outperforms the comparable $\text{BERT}_{\text{BASE}}$ systems including ERNIE by 3.5%, improves over $\text{BERT}_{\text{LARGE}}$ by 1.4%, and is able to match the performance of the relationship specific MTB pretraining in Soares et al. (2019). For SemEval 2010 Task 8 (Table 5), KnowBert-W+W F_1 falls between the entity aware $\text{BERT}_{\text{BASE}}$ model from Wang et al. (2019b), and the $\text{BERT}_{\text{LARGE}}$ model from Soares et al. (2019).

Words in Context (WiC) WiC (Pilehvar and Camacho-Collados, 2019) is a challenging task that presents systems with two sentences both containing a word with the same lemma and asks them to determine if they are from the same sense or not. It is designed to test the quality of contextual word representations. We follow standard practice and concatenate both sentences with a `[SEP]` token and fine-tune the `[CLS]` embedding. As shown in Table 6, KnowBert-W+W sets a new state of the art for this task, improving over $\text{BERT}_{\text{LARGE}}$ by 1.4% and reducing the relative gap to 80% human performance by 13.3%.

System	P	R	F_1
UFET	68.8	53.3	60.1
$\text{BERT}_{\text{BASE}}$	76.4	71.0	73.6
ERNIE	78.4	72.9	75.6
KnowBert-W+W	78.6	73.7	**76.1**

Table 7: Test set results for entity typing using the nine general types from (Choi et al., 2018).

Entity typing We also evaluated KnowBert-W+W using the entity typing dataset from Choi et al. (2018). To directly compare to ERNIE, we adopted the evaluation protocol in Zhang et al. (2019) which considers the nine general entity types.[7] Our model marks the location of a target span with the special `[E]` and `[/E]` tokens and uses the representation of the `[E]` token to predict the type. As shown in Table 7, KnowBert-W+W shows an improvement of 0.6% F_1 over ERNIE and 2.5% over $\text{BERT}_{\text{BASE}}$.

5 Conclusion

We have presented an efficient and general method to insert prior knowledge into a deep neural model. Intrinsic evaluations demonstrate that the addition of WordNet and Wikipedia to BERT improves the quality of the masked LM and significantly improves its ability to recall facts. Downstream evaluations demonstrate improvements for relationship extraction, entity typing and word sense disambiguation datasets. Future work will involve incorporating a diverse set of domain specific KBs for specialized NLP applications.

Acknowledgements

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2017. A neural knowledge language model. arXiv:1608.00318.

Christoph Alt, Marc Hübner, and Leonhard Hennig. 2019. Improving relation extraction by pre-trained language representations. In *AKBC*.

---

[7]Data obtained from `https://github.com/thunlp/ERNIE`

Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. In *EMNLP*.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *EMNLP*.

Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32 Database issue:D267–70.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *ACL*.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *ACL*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *NeurIPS*.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *I-SEMANTICS*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *EMNLP*.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *EMNLP*.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *EMNLP*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *HLT-NAACL*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL 2018*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *CoNLL*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328.

Robert L Logan, Nelson F. Liu, Matthew E. Peters, Matthew Ph Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *ACL*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. In *Advances in Neural Information Processing Systems*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *CoNLL*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. arXiv:1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *HLT*.

Roberto Navigli, José Camacho-Collados, and Alessandro Raganato. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *EACL*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

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

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Mohammad Taher Pilehvar and José Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *NAACL-HLT*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *EMNLP*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Peng Shi and Jimmy Lin. 2019. Simple BERT models for relation extraction and semantic role labeling.

Livio B. Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *ACL*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *AAAI*.

Valentin I. Spitkovsky and Angel X. Chang. 2012. A cross-lingual dictionary for English Wikipedia concepts. In *LREC*.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *ICLR*.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan R. Salakhutdinov, and William W. Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *EMNLP*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. arXiv:1905.00537.

Chao Wang and Hui Jiang. 2019. Explicit utilization of general knowledge in machine reading comprehension. In *ACL*.

Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. 2019b. Extracting multiple-relations in one-pass with pre-trained transformers. In *ACL*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *ACL*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014a. Knowledge graph and text jointly embedding. In *EMNLP*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. From one point to a manifold: knowledge graph embedding for precise link prediction. In *AAAI*.

Ruobing Xie, Zhiyuan Liu, J. J. Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. 2019. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *ACL*.

Bishan Yang and Tom Michael Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *ACL*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. arXiv:1412.6575.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *EMNLP*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *EMNLP*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *ACL*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan R. Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *ICCV*.

A Supplemental Material

A.1 KnowBert training details

We generally followed the fine-tuning procedure in Devlin et al. (2019), using Adam (Kingma and Ba, 2015) with weight decay 0.1 and learning rate schedule that linearly increases then decreases. Similar to ULMFiT (Howard and Ruder, 2018), we found it beneficial to vary the learning rate in different layers when fine tuning with this simple schedule: the randomly initialized newly added layers in the KAR surrounding each KB had the largest learning rate α; all BERT layers below had learning rate 0.25α; and BERT layers above had 0.5α. KnowBert-Wiki was fine-tuned for a total of 750K gradient updates, KnowBert-WordNet for 500K updates, and KnowBert-W+W for 500K updates (after pretraining KnowBert-Wiki). Fine-tuning was done on a single Titan RTX GPU with batch size of 32 using gradient accumulation for long sequences, except for the final 250K steps of KnowBert-W+W which was trained with batch size of 64 using 4 GPUs. At the end of training, masked LM perplexity continued to decrease, suggesting that further training would improve the results.

Due to computational expense, we performed very little hyperparameter tuning during the pre-training stage, and none with the full KnowBert-W+W model. All tuning was performed with partial training runs by monitoring masked LM perplexity in the early portion of training and terminating the under performing configurations. For each KnowBert-WordNet and KnowBert-Wiki we ran experiments with two learning rates (maximum learning rate of 1e-4 and 2e-4) and chose the best value after a few hundred thousand gradient updates (2e-4 for KnowBert-WordNet and 1e-4 for KnowBert-Wiki).

When training KnowBert-WordNet and KnowBert-Wiki we chose random batches from the unlabeled corpus and annotated entity linking datasets with a ratio of 4:1 (80% unlabeled, 20% labeled). For KnowBert-W+W we used a 85%/7.5%/7.5% sampling ratio.

A.2 Task fine-tuning details

This section details the fine tuning procedures and hyperparameters for each of the end tasks. All optimization was performed with the Adam optimizer with linear warmup of learning rate over the first 10% of gradient updates to a maximum value, then linear decay over the remainder of training. Gradients were clipped if their norm exceeded 1.0, and weight decay on all non-bias parameters was set to 0.01. Grid search was used for hyperparameter tuning (maximum values bolded below), using five random restarts for each hyperparameter setting for all datasets except TACRED (which used a single seed). Early stopping was performed on the development set. Batch size was 32 in all cases.

TACRED This dataset provides annotations for 106K sentences with typed subject and object spans and relationship labels across 41 different classes (plus the no-relation label). The hyperparameter search space was:

- learning rate: [**3e-5**, 5e-5]

- number epochs: [1, 2, **3**, 4, 5]

- β_2: [**0.98**, 0.999]

Maximum development micro F_1 is 71.7%.

SemEval 2010 Task 8 This dataset provides annotations for 10K sentences with untyped subject and object spans and relationship labels across 18 different classes (plus the no-relation label). As

the task does not define a standard development split, we randomly sampled 500 of the 8000 training examples for development. The hyperparameter search space was:

- learning rate: [**3e-5**, 5e-5]

- number epochs: [1, 2, **3**, 4, 5, 6, 7, 8, 9, 10]

with $\beta_2 = 0.98$. We used the provided `semeval2010_task8_scorer-v1.2.pl` script to compute F_1. The maximum development F_1 averaged across the random restarts was 89.1 $\pm$ 0.77 (maximum value was 90.5 across the seeds).

WiC WiC is a binary classification task with 7.5K annotated sentence pairs. Due to the small size of the dataset, we found it helpful to use model averaging to reduce the variance in the development accuracy across random restarts. The hyperparameter search space was:

- learning rate: [**1e-5**, 2e-5, 3e-5, 5e-5]

- number epochs: [2, 3, 4, **5**]

- $beta_2$: [0.98, **0.999**]

- weight averaging decay: [no averaging, **0.95**, 0.99]

The maximum development accuracy was 72.6.

Entity typing As described in Section 4.3, we evaluated on a subset of data corresponding to entities classified by nine different general classes: person, location, object, organization, place, entity, object, time, and event. β_2 was set to 0.98. The hyperparameter search space was:

- learning rate: [2e-5, **3e-5**]

- number epochs: [2, 3, 4, 5, 6, 7, 8, 9, **10**, 11, 12, 13, 14]

- $beta_2$: [0.98, **0.999**]

- weight averaging decay: [**no averaging**, 0.99]

The maximum development F_1 was 75.5 $\pm$ 0.38 averaged across five seeds, with maximum value of 76.0.

A.3 Wikidata probing results

Table 8 shows the results for the Wikidata probing task for all relationships.

| | BERT | | KnowBert | | |
	base	large	Wiki	Wordnet	W+W
companyFoundedBy	0.08	0.08	0.23	0.21	0.28
movieDirectedBy	0.05	0.04	0.08	0.09	0.10
movieStars	0.06	0.05	0.09	0.09	0.11
personCityOfBirth	0.18	0.20	0.33	0.35	0.40
personCountryOfBirth	0.16	0.17	0.33	0.36	0.47
personCountryOfDeath	0.18	0.18	0.30	0.30	0.44
personEducatedAt	0.11	0.10	0.37	0.29	0.43
personEmployer	0.03	0.02	0.20	0.13	0.21
personFather	0.15	0.14	0.42	0.40	0.48
personMemberOfBand	0.11	0.11	0.19	0.18	0.23
personMemberOfSportsTeam	0.01	0.03	0.05	0.08	0.13
personMother	0.11	0.11	0.28	0.24	0.32
personOccupation	0.14	0.24	0.24	0.24	0.30
personSpouse	0.05	0.06	0.12	0.08	0.18
songPerformedBy	0.07	0.05	0.10	0.10	0.13
videoGamePlatform	0.16	0.22	0.27	0.20	0.34
writtenTextAuthor	0.06	0.05	0.15	0.16	0.18
Total	0.09	0.11	0.26	0.22	0.31

Table 8: Full results on the Wikidata probing task including all relations.