E-BERT: Efficient-Yet-Effective Entity Embeddings for BERT

Nina Poerner[*†] **and Ulli Waltinger**[†] **and Hinrich Schütze**[*]
[*]Center for Information and Language Processing, LMU Munich, Germany
[†]Corporate Technology Machine Intelligence (MIC-DE), Siemens AG Munich, Germany
poerner@cis.uni-muenchen.de | inquiries@cislmu.org

Abstract

We present a novel way of injecting factual knowledge about entities into the pretrained BERT model (Devlin et al., 2019): We align Wikipedia2Vec entity vectors (Yamada et al., 2016) with BERT's native wordpiece vector space and use the aligned entity vectors as if they were wordpiece vectors. The resulting entity-enhanced version of BERT (called **E-BERT**) is similar in spirit to ERNIE (Zhang et al., 2019) and KnowBert (Peters et al., 2019), but it requires no expensive further pretraining of the BERT encoder. We evaluate E-BERT on unsupervised question answering (QA), supervised relation classification (RC) and entity linking (EL). On all three tasks, E-BERT outperforms BERT and other baselines. We also show quantitatively that the original BERT model is overly reliant on the surface form of entity names (e.g., guessing that someone with an Italian-sounding name speaks Italian), and that E-BERT mitigates this problem.

1 Introduction

BERT (Devlin et al., 2019) and its successors (e.g., Yang et al. (2019); Liu et al. (2019); Wang et al. (2019b)) continue to achieve state of the art performance on various NLP tasks. Recently, there has been interest in enhancing BERT with factual knowledge about entities (Zhang et al., 2019; Peters et al., 2019). To this end, we introduce **E-BERT**: We align Wikipedia2Vec entity vectors (Yamada et al., 2016) with BERT's wordpiece vector space (Section 3.1) and feed the aligned vectors into BERT as if they were wordpiece vectors (Section 3.2). Importantly, we do not make any changes to the BERT encoder itself, and we do no additional pretraining. This stands in contrast to previous entity-enhanced versions of BERT, such as ERNIE or KnowBert, which require additional encoder pretraining.

In Section 4, we evaluate our approach on LAMA (Petroni et al., 2019), a recent unsupervised QA benchmark for pretrained Language Models (LMs). We set a new state of the art on LAMA, with improvements over original BERT, ERNIE and KnowBert. We also find that the original BERT model is overly reliant on the surface form of entity names, e.g., it predicts that a person with an Italian-sounding name speaks Italian, regardless of whether this is factually correct. To quantify this effect, we create **LAMA-UHN** (UnHelpfulNames), a subset of LAMA where questions with overly helpful entity names were deleted (Section 4.4).

In Section 5, we show how to apply E-BERT to two entity-centric downstream tasks: relation classification (Section 5.1) and entity linking (Section 5.2). On the former task, we feed aligned entity vectors as inputs, on the latter, they serve as inputs *and* outputs. In both cases, E-BERT outperforms original BERT and other baselines.

Summary of contributions.

- Introduction of E-BERT: Feeding entity vectors into BERT without additional encoder pretraining. (Section 3)
- Evaluation on the LAMA unsupervised QA benchmark: E-BERT outperforms BERT, ERNIE and KnowBert. (Section 4)
- LAMA-UHN: A harder version of the LAMA benchmark with less informative entity names. (Section 4.4)
- Evaluation on supervised relation classification (Section 5.1) and entity linking (Section 5.2).
- Upon publication, we will release LAMA-UHN as well as E-BERT$_{\texttt{BASE}}$ and E-BERT$_{\texttt{LARGE}}$.[1]

[1] https://github.com/anonymized

2 Related work

2.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a Transformer (Vaswani et al., 2017) that was pretrained as a masked LM (MLM) on unlabeled text. At its base, BERT segments text into wordpieces from a vocabulary $\mathbb{L}_{\text{WP}}$. Wordpieces are embedded into real-valued vectors by a lookup function (denoted $\mathcal{E}_{\text{BERT}} : \mathbb{L}_{\text{WP}} \rightarrow \mathbb{R}^{d_{\text{BERT}}}$). The wordpiece vectors are combined with position and segment embeddings and then fed into a stack of Transformer layers (the encoder, denoted $\mathcal{F}_{\text{BERT}}$). During pretraining, some wordpieces are replaced by a special *[MASK]* token. The output of BERT is fed into a final feed-forward net (the MLM head, denoted $\mathcal{F}_{\text{MLM}}$), to predict the identity of the masked wordpieces. After pretraining, the MLM head is usually replaced by a task-specific layer, and the entire model is finetuned on supervised data.

2.2 Entity-enhanced BERT

This paper adds to recent work on entity-enhanced BERT models, most notably ERNIE (Zhang et al., 2019) and KnowBert (Peters et al., 2019). ERNIE and KnowBert are based on the design principle that *BERT be adapted to entity vectors*: They introduce new encoder layers to feed pretrained entity vectors into the Transformer, and they require additional pretraining to integrate the new parameters. In contrast, E-BERT's design principle is that *entity vectors be adapted to BERT*, which makes our approach more efficient (see Section 3.3).

Two other knowledge-enhanced MLMs are KEPLER (Wang et al., 2019c) and K-Adapter (Wang et al., 2020), which are based on Roberta (Liu et al., 2019) rather than BERT. Their factual knowledge does not stem from entity vectors – instead, they are trained in a multi-task setting on relation classification and knowledge base completion.

2.3 Wikipedia2Vec

Wikipedia2Vec (Yamada et al., 2016) embeds words and entities (Wikipedia URLs) into a common space. Given a vocabulary of words $\mathbb{L}_{\text{Word}}$ and a vocabulary of entities $\mathbb{L}_{\text{Ent}}$, it learns a lookup embedding function $\mathcal{E}_{\text{Wikipedia}} : \mathbb{L}_{\text{Word}} \cup \mathbb{L}_{\text{Ent}} \rightarrow \mathbb{R}^{d_{\text{Wikipedia}}}$. The Wikipedia2Vec loss has three components: (1) skipgram Word2Vec (Mikolov et al., 2013a) operating on $\mathbb{L}_{\text{Word}}$, (2) a graph loss operating on the Wikipedia hyperlink graph, whose vertices are $\mathbb{L}_{\text{Ent}}$ and (3) a version of Word2Vec where words are predicted from entities. Loss (3) ensures that entities and words are embedded into the same space.

2.4 Vector space alignment

Our vector space alignment strategy is inspired by cross-lingual word vector alignment (e.g., Mikolov et al. (2013b); Smith et al. (2017)). A related method was recently applied by Wang et al. (2019a) to map cross-lingual word vectors into the multilingual BERT wordpiece vector space.

2.5 Unsupervised QA

QA has typically been tackled as a supervised problem (e.g., Das et al. (2017); Sun et al. (2018)). Recently, there has been interest in using unsupervised LMs such as GPT-2 or BERT for this task (Radford et al., 2019; Petroni et al., 2019). Davison et al. (2019) mine unsupervised commonsense knowledge from BERT, and Jiang et al. (2019) show the importance of using good prompts for unsupervised QA. None of this prior work differentiates quantitatively between factual knowledge of LMs and their ability to reason about the surface form of entity names.

3 E-BERT

3.1 Aligning entity and wordpiece vectors

Conceptually, we want to transform the vectors of the entity vector space $\mathcal{E}_{\text{Wikipedia}}[\mathbb{L}_{\text{Ent}}]$ in such a way that they look to BERT like vectors from its native wordpiece vector space $\mathcal{E}_{\text{BERT}}[\mathbb{L}_{\text{WP}}]$. We model the transformation as an unconstrained linear mapping $\mathbf{W} \in \mathbb{R}^{d_{\text{BERT}} \times d_{\text{Wikipedia}}}$. Since $\mathbb{L}_{\text{WP}}$ does not contain any entities (i.e., $\mathbb{L}_{\text{WP}} \cap \mathbb{L}_{\text{Ent}} = \{\}$), we fit the mapping on $\mathbb{L}_{\text{WP}} \cap \mathbb{L}_{\text{Word}}$:

$$\sum_{x \in \mathbb{L}_{\text{WP}} \cap \mathbb{L}_{\text{Word}}} ||\mathbf{W}\mathcal{E}_{\text{Wikipedia}}(x) - \mathcal{E}_{\text{BERT}}(x)||_2^2$$

Since Wikipedia2Vec embeds $\mathbb{L}_{\text{Word}}$ and $\mathbb{L}_{\text{Ent}}$ into the same space (see Section 2.3), $\mathbf{W}$ can be applied to $\mathbb{L}_{\text{Ent}}$ as well. We define the E-BERT embedding function as:

$$\mathcal{E}_{\text{E-BERT}} : \mathbb{L}_{\text{Ent}} \rightarrow \mathbb{R}^{d_{\text{BERT}}}$$
$$\mathcal{E}_{\text{E-BERT}}(a) = \mathbf{W}\mathcal{E}_{\text{Wikipedia}}(a)$$

3.2 Using aligned entity vectors

We explore two strategies for feeding the aligned entity vectors into the BERT encoder:

E-BERT-concat. E-BERT-concat combines entity IDs and wordpieces by string concatenation, with the slash symbol as separator (Schick and Schütze, 2019). For example, the wordpiece-tokenized input

The native language of Jean Mara ##is is [MASK] .[2]

becomes

The native language of **Jean_Marais** */ Jean Mara ##is is [MASK] .*

The entity ID (**bold**) is embedded by $\mathcal{E}_{\text{E-BERT}}$ and all wordpieces (*italics*) are embedded by $\mathcal{E}_{\text{BERT}}$ (see Figure 1). After the embedding operation, the sequence of vectors is combined with position and segment embeddings and fed into $\mathcal{F}_{\text{BERT}}$, just like any normal sequence of wordpiece vectors.

E-BERT-concat is comparable to ERNIE or KnowBert, which also represent entities as a combination of surface form (wordpieces) and entity vectors. But in contrast to ERNIE and KnowBERT, **we do not change or further pretrain the BERT encoder itself**.

E-BERT-replace. For ablation purposes, we define another variant of E-BERT that substitutes the entity surface form with the entity vector. With E-BERT-replace, our example becomes:

The native language of **Jean_Marais** *is [MASK] .*

3.3 Implementation

We train cased Wikipedia2Vec on a recent Wikipedia dump (2019-09-02), setting $d_{\text{Wikipedia}} = d_{\text{BERT}}$. We ignore Wikipedia pages with fewer than 5 links (Wikipedia2Vec's default), with the exception of entities needed for the downstream entity linking experiments (see Section 5.2). This results in an entity vocabulary of size $|\mathbb{L}_{\text{Ent}}| = 2.7\text{M}$.[3]

Computational cost. Training Wikipedia2Vec took us ~6 hours on 32 CPUs, and the cost of fitting the linear transformation $\mathbf{W}$ is negligible. We did not require a GPU. For comparison, KnowBert W+W was pretrained for 1.25M steps on up to four Titan RTX GPUs, and ERNIE took one epoch on the English Wikipedia. (ERNIE's pretraining hardware was not disclosed, but it seems likely that a GPU was involved.)

[2]For readability, we omit the special tokens *[CLS]* and *[SEP]* from all examples.

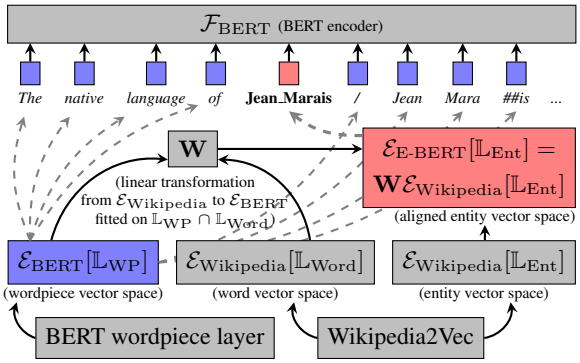

Figure 1: Schematic depiction of E-BERT-concat.

4 Unsupervised QA

4.1 Data

The LAMA (LAnguage Model Analysis) benchmark (Petroni et al., 2019) probes for "factual and commonsense knowledge" of pretrained LMs. In this paper, we use LAMA-Google-RE and LAMA-T-REx (Elsahar et al., 2018), which are aimed at factual knowledge. Contrary to most previous work on QA, LAMA tests LMs without supervised fine-tuning. Petroni et al. (2019) claim that BERT's performance on LAMA is comparable with a knowledge base (KB) automatically extracted from text, and speculate that BERT and similar models "might become a viable alternative" to such KBs.

The LAMA task follows this schema: Given a KB triple (**sub**, **rel**, **obj**), the object is elicited with a relation-specific cloze-style question, e.g., (**Jean_Marais**, **native-language**, **French**) becomes: "The native language of Jean Marais is [MASK]."[4] The model predicts a probability distribution over a limited vocabulary $\mathbb{L}_{\text{LAMA}} \subset \mathbb{L}_{\text{WP}}$ to replace [MASK], which is evaluated against the surface form of the object (here: *French*).

4.2 Baselines

Our primary baselines are cased $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$[5] as evaluated in Petroni et al. (2019).

[3]Due to the link threshold and some Wikidata-Wikipedia mismatches, we lack entity vectors for 6% of LAMA questions and 10% of FewRel sentences (RC experiment, see Section 5.1). In these cases, we fall back onto using wordpieces only, i.e., onto standard BERT behavior.

[4]LAMA provides oracle entity IDs, however, they are not used by the BERT baseline. For a fair evaluation, we ignore them too and instead use the Wikidata query API (https://query.wikidata.org) to infer entity IDs from surface forms. See Appendix for details.

[5]https://github.com/huggingface/transformers

	original BERT	E-BERT-replace	E-BERT-concat	ERNIE	Know-Bert
Jean Marais	French	French	French	french	french
Daniel Ceccaldi	Italian	French	French	french	italian
Orane Demazis	Albanian	French	French	french	french
Sylvia Lopez	Spanish	French	Spanish	spanish	spanish
Annick Alane	English	French	French	english	english

Table 1: Native language (LAMA-T-REx:P103) of French-speaking actors according to different models. Model size is BASE.

We also test ERNIE (Zhang et al., 2019)[6] and KnowBert W+W (Peters et al., 2019),[7] two entity-enhanced $\text{BERT}_{\text{BASE}}$-type models.[8] E-BERT, ERNIE and KnowBert have entity vocabularies of size 2.7M, 5M and 470K, respectively. As this might put KnowBert at a disadvantage, Table 3 also reports performance on the subset of questions whose gold subject is known to KnowBert.

4.3 Evaluation measure

We use the same evaluation measure as Petroni et al. (2019): For a given k, we count a question as 1 if the correct answer is among the top-k predictions and as 0 otherwise. Petroni et al. (2019) call this measure Precision@k (P@k). Since this is not in line with the typical use of the term "precision" in information retrieval (Manning et al., 2008, p. 161), we call the evaluation measure Hits@k. Like Petroni et al. (2019), we first average within relations and then across relations.

4.4 LAMA-UHN

Imagine a person who claims to know a lot of facts. During a quiz, you ask them about the native language of actor Jean Marais. They correctly answer "French." For a moment you are impressed, until you realize that Jean is a typical French name. So you ask the same question about Daniel Ceccaldi (a French actor with an Italian-sounding name). This time, the person says "Italian."

If this quiz were a QA benchmark, the person would have achieved a respectable Hits@1 score of 50%. Yet, you doubt that they really *knew* the first answer.

[6] https://github.com/thunlp/ERNIE

[7] https://github.com/allenai/kb

[8] ERNIE and KnowBert are uncased models. We therefore lowercase all questions for them and restrict predictions to the intersection of their wordpiece vocabulary with lowercased $\mathbb{L}_{\text{LAMA}}$. As a result, ERNIE and KnowBert select answers from ∼18K candidates (instead of ∼21K), which should work in their favor. We verify that all lowercased answers appear in this vocabulary, i.e., ERNIE and KnowBert are in principle able to answer all questions correctly.

Qualitative inspection of BERT's answers to LAMA suggests that the model often behaves less like a KB and more like the person just described. In Table 1 for instance, BERT predicts native languages that are plausible for people's names, even when there is no factual basis for these predictions. This kind of name-based reasoning is a useful strategy for getting a high score on LAMA, as the correct answer and the best name-based guess tend to coincide (e.g., people with Italian-sounding names frequently speak Italian). Hence, LAMA in its current form cannot differentiate whether a model is good at reasoning about (the surface form of) entity names, good at memorizing facts, or both. To quantify the effect, we create LAMA-UHN (UnHelpful Names), a subset of LAMA where overly helpful entity names are heuristically deleted:

Heuristic 1 (string match filter). We first delete all KB triples (questions) where the correct answer (e.g., *Apple*) is a case-insensitive substring of the subject entity name (e.g., *Apple Watch*). This affects 12% of all triples, and up to 81% for individual relations (see Table 2, top).

Heuristic 2 (person name filter). Entity names can be revealing in ways that are more subtle than string matches. As illustrated by our *Jean Marais* example, a person's name can be a useful prior for guessing their native language and by extension, their nationality, place of birth, etc. We therefore use cloze-style questions to elicit name associations inherent in BERT, and delete triples that correlate with them.

The heuristic is best explained via an example. Consider again (**Jean_Marais**, **native-language**, **French**). We whitespace-tokenize the subject's surface form *Jean Marais* into *Jean* and *Marais*. If BERT considers either name to be a common French name, then a correct answer is insufficient evidence for factual knowledge about the entity **Jean_Marais**. On the other hand, if neither *Jean* nor *Marais* are considered French, but a correct answer is given regardless, we consider it sufficient evidence of factual knowledge.

We query BERT with "[X] is a common name in the following language: [MASK]." for [X] = *Jean* and [X] = *Marais*. (Depending on the relation, we replace "language" with "city" or "country".) If *French* is among the top-3 answers for either question, we delete the original triple. We apply this heuristic to T-REx:P19 (place of birth),

Heuristic		Relation	% deleted	Example of a deleted question	
1	string match filter	T-REx:P176 (manufacturer)	81%	Fiat Multipla is produced by [MASK:Fiat].	
		T-REx:P138 (named after)	75%	Christmas Island is named after [MASK:Christmas].	
		T-REx:P1001 (applies to jurisdiction)	73%	Australian Senate is a legal term in [MASK:Australia].	
2	person name filter	T-REx:P1412 (language used)	63%	Fulvio Tomizza used to communicate in [MASK:Italian].	(1,1)
		T-REx:P103 (native language)	58%	The native language of Tommy Nilsson is [MASK:Swedish].	(-,1)
		T-REx:P27 (nationality)	56%	Harumi Inoue is a [MASK:Japan] citizen.	(1,-)

Table 2: Statistics and examples of LAMA questions with helpful entity names, which were deleted from LAMA-UHN. We show the top-3 most strongly affected relations per heuristic. Numbers in brackets indicate which part(s) of the person name triggered the person name filter, e.g., (-,1) means that the correct answer was ranked first for the person's last name, but was not in the top-3 for their first name.

	Model size	BASE					LARGE			
	Dataset \ Model	original BERT	E-BERT-replace	E-BERT-concat	ERNIE	Know-Bert	original BERT	E-BERT-replace	E-BERT-concat	K-Adapter
All questions	0 (original LAMA)	29.2	29.1	**36.2**	30.4	31.7	30.6	28.5	34.2	27.6
	1 (string match filter)	22.3	29.2	**32.6**	25.5	25.6	24.6	28.6	30.8	-
	2 (LAMA-UHN)	20.2	28.2	**31.1**	24.7	24.6	23.0	27.8	29.5	21.7
Questions w/ KnowBert subject only	0 (original LAMA)	32.0	28.5	**35.8**	30.4	32.0	33.1	28.2	34.9	-
	1 (string match filter)	24.8	28.6	**32.0**	25.7	25.9	27.0	28.3	31.5	-
	2 (LAMA-UHN)	22.8	27.7	**30.6**	24.9	25.1	25.5	27.4	**30.6**	-

Table 3: Mean Hits@1 on LAMA-Google-RE and LAMA-T-REx combined. 0: original LAMA dataset (Petroni et al., 2019), 1: after string match filter, 2: after string match filter and person name filter (LAMA-UHN). "Questions w/ KnowBert subject only": Evaluating on questions whose gold subject is in the KnowBert entity vocabulary. Results for K-Adapter are calculated from Wang et al. (2020, Table 5). See Appendix for individual relations.

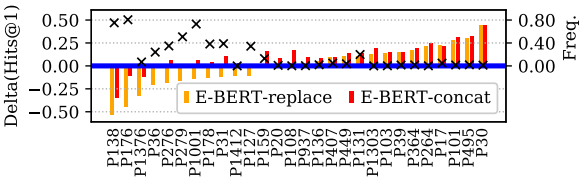

Figure 2: Left y-axis (bars): delta in mean Hits@1 relative to BERT on individual LAMA relations. Right y-axis (crosses): frequency of questions where the answer is a substring of the subject entity name (i.e., questions that would be deleted by the string match filter). Model size: BASE. Due to space constraints, we only show relations with max absolute delta ≥ 0.075.

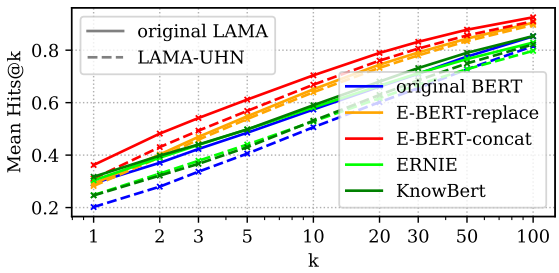

Figure 3: Mean Hits@k for different k. Model size: BASE. The x-axis is on a logarithmic scale.

T-REx:P20 (place of death), T-REx:P27 (nationality), T-REx:P103 (native language), T-REx:P1412 (language used), Google-RE:place-of-death and Google-RE:place-of-birth. See Table 2 (bottom) for examples and statistics.

4.5 Results and discussion

Table 3 shows mean Hits@1 on the original LAMA dataset (0), after applying the string match filter (1), and after applying both filters (2, LAMA-UHN). E-BERT-concat$_{\text{BASE}}$ sets a new state of the art on LAMA, with major gains over original BERT.

To understand why, compare the performances of BERT and E-BERT-replace on LAMA-UHN: While BERT drops by about 8% between original LAMA and LAMA-UHN, E-BERT-replace drops by less than 1%. This suggests that BERT's performance on original LAMA is partly due to the exploitation of helpful entity names, while that of E-BERT-replace is more strongly due to factual knowledge. Since E-BERT-concat has access to entity names *and* entity vectors, it can leverage and combine these complementary sources of information. Interestingly, ERNIE and KnowBert have access to both sources too, but they do not achieve the same performance as E-BERT-concat.

For a more in-depth analysis, Figure 2 shows Delta(Hits@1) w.r.t. BERT (bars, left axis) on individual LAMA relations, along with the frequency of questions whose correct answer is a substring of the subject name (crosses, right axis). The losses of E-BERT-replace are almost exclusively on re-

lations with a high frequency of "easy" substring answers, while its gains are on relations where such answers are rare. E-BERT-concat mitigates most of the losses of E-BERT-replace while keeping most of its gains.

Figure 3 shows that the gains of E-BERT-concat over BERT, KnowBert and ERNIE in terms of mean Hits@k are especially big for $k > 1$. This means that while E-BERT-concat is moderately better than the baselines at giving the correct answer, it is a lot better at "almost giving the correct answer". Petroni et al. (2019) speculate that even when factual knowledge is not salient enough for a top-1 answer, it may still be useful when finetuning on a downstream task.

5 Downstream tasks

We now demonstrate how to use E-BERT on two downstream tasks: relation classification (RC) and entity linking (EL). In both experiments, we keep the embedding layer ($\mathcal{E}_{\text{BERT}}$ and/or $\mathcal{E}_{\text{E-BERT}}$) fixed but finetune all other encoder parameters. We use the $\text{BERT}_{\texttt{BASE}}$ architecture throughout.

5.1 Relation classification

In relation classification (RC), a model learns to predict the directed relation of entities a_{sub} and a_{obj} from text. For instance, given the sentence

> *Taylor was later part of the ensemble cast in MGM 's classic World War II drama " Battleground " (1949) .*

with surface forms *Battleground* and *World War II* referring to a_{sub} = **Battleground_(film)** and a_{obj} = **Word_War_II**, the model should predict the relation **primary-topic-of-work**. We have three ways of embedding this example:

original BERT (wordpieces): *[...] classic World War II drama " Battle ##ground " (1949) .*

E-BERT-concat: *[...] classic* **World_War_II** */ World War II drama "* **Battleground_(film)** */ Battle ##ground " (1949) .*

E-BERT-replace: *[...] classic* **World_War_II** *drama "* **Battleground_(film)** *" (1949) .*

As before, entity IDs (**bold**) are embedded by $\mathcal{E}_{\text{E-BERT}}$ and wordpieces (*italics*) by $\mathcal{E}_{\text{BERT}}$.

Baselines: Wikipedia2Vec-BERT. To assess the impact of vector space alignment, we train two additional models (Wikipedia2Vec-BERT-concat and Wikipedia2Vec-BERT-replace) that feed non-aligned Wikipedia2Vec vectors directly into BERT (i.e., they use $\mathcal{E}_{\text{Wikipedia}}$ instead of $\mathcal{E}_{\text{E-BERT}}$ to embed entity IDs).

	dev set			test set		
	P	R	F1	P	R	F1
original BERT	85.88	85.81	85.75	85.57	85.51	85.45
E-BERT-concat	**88.35**	**88.29**	**88.19**	**88.51**	**88.46**	**88.38**
E-BERT-replace	87.24	87.15	87.09	87.34	87.33	87.22
Wikipedia2Vec-BERT-concat	85.96	85.71	85.69	85.94	85.93	85.84
Wikipedia2Vec-BERT-replace	77.25	77.11	77.07	77.63	77.52	77.45
ERNIE (Zhang et al., 2019)	-	-	-	**88.49**	**88.44**	**88.32**

Table 4: RC macro precision, recall and F1 (%).

Data. We evaluate on a preprocessed dataset from Zhang et al. (2019), which is a subset of the FewRel corpus (Sun et al., 2018) (see Appendix for details). We use the FewRel oracle entity IDs, which are also used by ERNIE. Our entity coverage is lower than ERNIE's (90% vs. 96%), which should put us at a disadvantage. See Appendix for details on data and preprocessing.

Modeling and hyperparameters. We adopt the setup and hyperparameters of Zhang et al. (2019): We use the # and $ tokens to mark subject and object spans in the input, and we feed the last contextualized vector of the *[CLS]* token into a randomly initialized softmax classifier. Like Zhang et al. (2019), we use the default AdamW optimizer (Loshchilov and Hutter, 2018) with a linear learning rate scheduler (10% warmup) and a batch size of 32. We tune the number of training epochs and the peak learning rate on the same parameter ranges as Zhang et al. (2019). See Appendix for more details and an expected maximum performance plot.

Results and discussion. E-BERT-concat performs better than original BERT and slightly better than ERNIE (Table 4). Recall that ERNIE required additional encoder pretraining to achieve this result. Interestingly, E-BERT-replace (which is entity-only) beats original BERT (which is surface-form-only), i.e., aligned entity vectors seem to be more useful than entity names for this task. The drop in F1 from E-BERT to Wikipedia2Vec-BERT shows the importance of vector space alignment.

5.2 Entity linking

Entity linking (EL) is the task of detecting entity spans in a text and linking them to the underlying entity ID. While there are recent advances in fully end-to-end EL (Broscheit, 2019), the task is typically broken down into three steps: (1) detecting spans that are potential entity spans, (2) generating sets of candidate entities for these spans, (3) selecting the correct candidate for each span.

For steps (1) and (2), we use KnowBert's candidate generator (Peters et al., 2019), which is based on a precomputed span-entity co-occurrence table (Hoffart et al., 2011). Given an input sentence, the generator finds all spans that occur in the table, and annotates each with a set of candidates $A = \{a_1 \dots a_N\}$ and prior probabilities $\{p(a_1) \dots p(a_N)\}$. Note that the candidates and priors are span- but not context-specific, and that the generator may over-generate. For step (3), our model must therefore learn to (a) reject over-generated spans and (b) disambiguate candidates based on context.

Modeling. Recall that BERT was pretrained as a masked LM (MLM). Given a wordpiece-tokenized input X with $x_i = \textit{[MASK]}$, it has learned to predict a probability distribution over $\mathbb{L}_{\text{WP}}$ to fill the masked position:

$$\forall w \in \mathbb{L}_{\text{WP}} \quad (1)$$
$$p(w|X) \propto \exp(\mathbf{e}_w \cdot \mathcal{F}_{\text{MLM}}(\mathbf{h}_i) + b_w)$$

where $\mathbf{h}_i$ is the contextualized embedding of *[MASK]*, b_w is a learned bias and $\mathbf{e}_w = \mathcal{E}_{\text{BERT}}(w)$. Since $\mathcal{E}_{\text{E-BERT}}[\mathbb{L}_{\text{Ent}}]$ is aligned with $\mathcal{E}_{\text{BERT}}[\mathbb{L}_{\text{WP}}]$, the pretrained MLM should have a good initialization for predicting entities from context as well.

Based on this intuition, our **E-BERT-MLM** model repurposes the MLM for the entity selection step. Given a wordpiece-tokenized span $s_1 \dots s_{T_s}$ with left context $l_1 \dots l_{T_l}$, right context $r_1 \dots r_{T_r}$, candidates A and priors $p(a)$, we define:

$$X = l_1 \dots l_{T_l} \textit{ [E-MASK] / } s_1 \dots s_{T_s} * r_1 \dots r_{T_r}$$

All tokens in X except *[E-MASK]* are embedded by $\mathcal{E}_{\text{BERT}}$. *[E-MASK]* is embedded as $\frac{1}{|A|}\sum_{a \in A} \mathcal{E}_{\text{E-BERT}}(a)$, to inform the encoder about its options for the current span. (See Table 5 for an ablation with the standard *[MASK]* token.)

The output probability distribution for *[E-MASK]* is not defined over $\mathbb{L}_{\text{WP}}$ but over $A \cup \{\epsilon\}$, where ϵ stands for rejected spans (see below):

$$\forall a \in A \cup \{\epsilon\} \quad (2)$$
$$p(a|X) \propto \exp(\mathbf{e}_a \cdot \mathcal{F}_{\text{MLM}}(\mathbf{h}_{T_l+1}) + b_a)$$

For all $a \in A$, $\mathbf{e}_a = \mathcal{E}_{\text{E-BERT}}(a)$ and $b_a = \log(p(a))$.[9] The null-entity ϵ has parameters $\mathbf{e}_\epsilon, b_\epsilon$ that are trained from scratch.

[9] To understand why we set $b_a = \log(p(a))$, assume that the priors are implicitly generated as $p(a) = \exp(b_a)/Z$, with $Z = \sum_{a'} \exp(b_{a'})$. It follows that $b_a = \log(p(a)) + \log(Z)$. Since $\log(Z)$ is the same for all a', and the softmax function is invariant to constant offsets, we can drop $\log(Z)$ from Eq. 2.

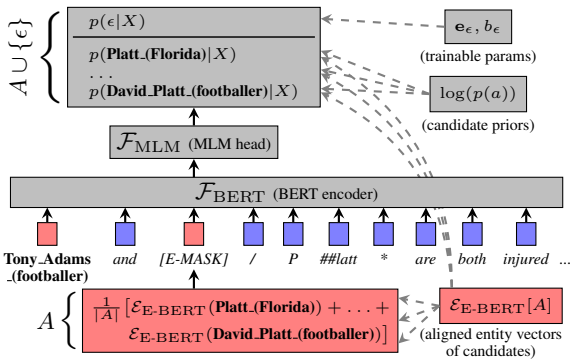

Figure 4: Schematic depiction of E-BERT-MLM. Blue: $\mathcal{E}_{\text{BERT}}$ wordpiece vectors. Red: $\mathcal{E}_{\text{E-BERT}}$ entity vectors. The candidates A and their priors $p(a)$ are from the candidate generator. Assume that the entity **Tony_Adams_(footballer)** was decoded in a previous iteration (see "Iterative refinement").

	AIDA-A (dev) Micro	AIDA-A (dev) Macro	AIDA-B (test) Micro	AIDA-B (test) Macro
E-BERT-MLM	**90.8**	**89.1**	**85.0**	**84.2**
w/o iterative refinement	90.6	89.0	-	-
w/ standard *[MASK]* token	90.3	88.8	-	-
Wikipedia2Vec-BERT-MLM	88.7	86.4	80.6	81.0
Wikipedia2Vec-BERT-random	88.2	86.1	80.5	81.2
Kolitsas et al. (2018)	89.4	86.6	82.4	82.6
Broscheit (2019)	86.0	-	79.3	-
KnowBert (Peters et al., 2019)	82.1	-	73.7	-
Chen et al. (2019)†	**92.6**	**93.6**	**87.5**	**87.7**

Table 5: F1 (%) on AIDA after finetuning. †Might not be comparable: Chen et al. (2019) evaluate on in-vocabulary entities only, without ensuring (or reporting) the vocabulary's coverage of the AIDA data.

Finetuning. We finetune E-BERT-MLM on the training set to minimize $\sum_{(X,\hat{a})} -\log(p(\hat{a}|X))$, where $(X, \hat{a})$ are pairs of potential spans and their gold entities. If X has no gold entity (if it was over-generated), then $\hat{a} = \epsilon$.[10]

Iterative refinement. We found it useful to iteratively refine predictions during inference, similar to techniques from non-autoregressive Machine Translation (Ghazvininejad et al., 2019). We start with a wordpiece-tokenized input, e.g.:

> *Adams and P ##latt are both injured and will miss England 's opening World Cup qualifier ...*

We make predictions for all potential spans that the candidate generator finds in the input. We gather all spans with $\text{argmax}_a[p(a|X)] \neq \epsilon$, sort them by $1 - p(\epsilon|X)$ and replace the top-k[11] non-overlapping

[10] If $\hat{a} \neq \epsilon \wedge \hat{a} \notin A$, we remove the span from the training set. We do **not** do this at test time, i.e., we evaluate on all gold standard entities.

[11] $k = \text{ceil}(\frac{j(m+n)}{J}) - m$, where $1 \leq j \leq J$ is the current iteration, m is the number of already decoded entities from

spans with the predicted entity. Our previous example might be partially decoded as:

> **Tony_Adams_(footballer)** *and P ##latt are both injured and will miss England 's opening* **1998_FIFA_World_Cup** *qualifier ...*

In the next iteration, decoded entities (**bold**) are represented by $\mathcal{E}_{\text{E-BERT}}$ in the input, while non-decoded spans continue to be represented by $\mathcal{E}_{\text{BERT}}$ (see Figure 4). We set the maximum number of iterations to $J = 3$, as there were no improvements beyond that point on the dev set.

Baselines: Wikipedia2Vec-BERT. We train two additional baseline models that combine BERT and Wikipedia2Vec without vector space alignment:

Wikipedia2Vec-BERT-MLM: BERT and its pretrained MLM head, finetuned to predict non-aligned Wikipedia2Vec vectors. In practice, this means replacing $\mathcal{E}_{\text{E-BERT}}$ with $\mathcal{E}_{\text{Wikipedia}}$ in Eq. 2. Embedding the *[E-MASK]* token with non-aligned $\mathcal{E}_{\text{Wikipedia}}$ led to a drop in dev set micro F1, therefore we report this baseline with the standard *[MASK]* token.

Wikipedia2Vec-BERT-random: Like Wikipedia2Vec-BERT-MLM, but the MLM head is replaced by a randomly initialized layer.

Data. We train and evaluate on AIDA, a news dataset annotated with Wikipedia URLs (Hoffart et al., 2011). To ensure coverage of the necessary entities, we include all gold entities and all generator candidates in the entity vocabulary $\mathbb{L}_{\text{Ent}}$, even if they fall under the Wikipedia2Vec link threshold (see Section 3.3). While this is based on the unrealistic assumption that we know the contents of the test set in advance, it is necessary for comparability with Peters et al. (2019), Kolitsas et al. (2018) and Broscheit (2019), who also design their entity vocabulary around the data. See Appendix for more details on data and preprocessing. We evaluate strong match F1, i.e., a prediction must have the same start, end and entity (URL) as the gold standard. URLs that redirect to the same Wikipedia page are considered equivalent.

Hyperparameters. We train for 10 epochs with the AdamW optimizer (Loshchilov and Hutter, 2018) and a linear learning rate scheduler (10% warmup), and we select the best epoch on the dev set. We tune peak learning rate and batch size on the dev set (see Appendix).

previous iterations, and $n = |\{X : \text{argmax}_a[p(a|X)] \neq \epsilon\}|$.

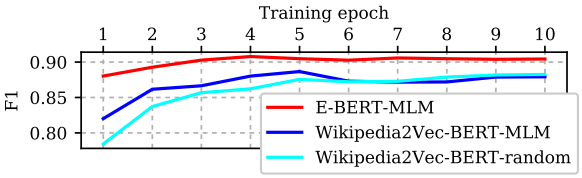

Figure 5: AIDA dev set micro F1 after every epoch.

	P	R	F1
E-BERT-MLM	21.1	61.8	31.5
w/ standard *[MASK]* token	23.3	65.2	34.3
Wikipedia2Vec-BERT-MLM	1.3	8.3	2.3
Wikipedia2Vec-BERT-random	1.3	6.8	2.2

Table 6: Unsupervised AIDA dev set micro precision / recall / F1 (%), before finetuning. Results are without iterative refinement.

Results and discussion. Table 5 shows that E-BERT-MLM is competitive with previous work on AIDA. The aligned entity vectors play a key role in this performance, as they give the model a good initialization for predicting entities from context. When we remove this initialization by using non-aligned entity vectors (Wikipedia2Vec-BERT baselines), we get worse unsupervised performance (Table 6), slower convergence during finetuning (Figure 5), and a lower final F1 (Table 5).

6 Conclusion

We introduced **E-BERT**, an efficient yet effective way of injecting factual knowledge about entities into the BERT pretrained Language Model. We showed how to align Wikipedia2Vec entity vectors with BERT's wordpiece vector space, and how to feed the aligned vectors into BERT as if they were wordpiece vectors. In doing so, we made no changes to the BERT encoder itself. This stands in contrast to other entity-enhanced versions of BERT, such as ERNIE or KnowBert, which add encoder layers and require expensive further pretraining.

We set a new state of the art on LAMA, a recent unsupervised QA benchmark. Furthermore, we presented evidence that the original BERT model sometimes relies on the surface forms of entity names (rather than "true" factual knowledge) for this task. To quantify this effect, we introduced LAMA-UHN, a subset of LAMA where questions with helpful entity names are deleted.

We also showed how to apply E-BERT to two supervised tasks: relation classification and entity linking. On both tasks, we achieve competitive results relative to BERT and other baselines.

References

Samuel Broscheit. 2019. Investigating entity knowledge in bert with simple neural end-to-end entity linking. In *CoNLL*, pages 677–685, Hong Kong, China.

Haotian Chen, Sahil Wadhwa, Xi David Li, and Andrej Zukov-Gregoric. 2019. YELM: End-to-end contextualized entity linking. *arXiv preprint arXiv:1911.03834*.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. In *ACL*, pages 358–365, Vancouver, Canada.

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. In *EMNLP-IJCNLP*, pages 1173–1178, Hong Kong, China.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, Minneapolis, USA.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A Smith. 2019. Show your work: Improved reporting of experimental results. In *EMNLP-IJCNLP*, pages 2185–2194, Hong Kong, China.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frédérique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *LREC*, pages 3448–3452, Miyazaki, Japan.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-Predict: Parallel decoding of conditional masked language models. In *EMNLP-IJCNLP*, pages 6114–6123, Hong Kong, China.

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*, pages 782–792, Edinburgh, UK.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2019. How can we know what language models know? *arXiv preprint arXiv:1911.12543*.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *CoNLL*, pages 519–529, Brussels, Belgium.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

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

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Matthew E Peters, Mark Neumann, IV Logan, L Robert, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. In *EMNLP-IJCNLP*, Hong Kong, China.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*, Hong Kong, China.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Timo Schick and Hinrich Schütze. 2019. BERTRAM: Improved word embeddings have big impact on contextualized model performance. *arXiv preprint arXiv:1910.07181*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *ICLR*, Toulon, France.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *EMNLP*, pages 4231–4242, Brussels, Belgium.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008, Long Beach, USA.

Hai Wang, Dian Yu, Kai Sun, Janshu Chen, and Dong Yu. 2019a. Improving pre-trained multilingual models with vocabulary expansion. In *CoNLL*, pages 316–327, Hong Kong, China.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Cuihong Cao, Daxin Jiang, Ming Zhou, et al. 2020. K-Adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Liwei Peng, and Luo Si. 2019b. StructBERT: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2019c. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *arXiv preprint arXiv:1911.06136*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL*, Berlin, Germany.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *ACL*, pages 1441–1451, Florence, Italy.

E-BERT: Efficient-Yet-Effective Entity Embeddings for BERT (Appendix)

Unsupervised QA (LAMA)

Data

We downloaded the LAMA dataset from https://dl.fbaipublicfiles.com/LAMA/data.zip. We use the LAMA-T-REx and LAMA-Google-RE relations, which are aimed at factual knowledge. Table 9 shows results on indiviual relations, as well as the number of questions per relation before and after applying the LAMA-UHN heuristics.

Preprocessing

As mentioned in Section 4.1, we do not use LAMA's oracle entity IDs. Instead, we map surface forms to entity IDs via the Wikidata query API (https://query.wikidata.org). For example, to look up *Jean Marais*:

```
SELECT ?id ?str WHERE {
  ?id rdfs:label ?str .
  VALUES ?str { 'Jean Marais'@en } .
  FILTER((LANG(?str)) = 'en') .
}
```

If more than one Wikidata ID is returned, we select the lowest one. We then map Wikidata IDs to the corresponding Wikipedia URLs:

```
SELECT ?id ?wikiurl WHERE {
  VALUES ?id { wd:Q168359 } .
  ?wikiurl schema:about ?id .
  ?wikiurl schema:inLanguage 'en' .
  FILTER REGEX(str(?wikiurl),
        '.*en.wikipedia.org.*') .
}
```

Relation classification

Data

The RC dataset, which is a subset of the FewRel corpus, was compiled by Zhang et al. (2019). We downloaded it from https://cloud.tsinghua.edu.cn/f/32668247e4fd4f9789f2/. Table 7 shows dataset statistics.

Preprocessing

The dataset contains sentences with annotated subject and object entity mentions, their oracle entity IDs and their relation (which must be predicted). We use the BERT wordpiece tokenizer to tokenize the sentence and insert special wordpieces to mark the entity mentions: *#* for subjects and *$* for objects. Then, we insert the entity IDs. For example, an input to E-BERT-concat would look like this:

> *[CLS] Taylor was later part of the ensemble cast in MGM 's classic $* ***World_War_II*** */ World War II $ drama " #* ***Battleground_(film)*** */ Battle ##ground # " (1949) . [SEP]*

We use the oracle entity IDs of the dataset, which are also used by ERNIE (Zhang et al., 2019).

Hyperparameters

We tune peak learning rate and number of epochs on the dev set (selection criterion: macro F1). We do a full search over the same hyperparameter space as Zhang et al. (2019):

Learning rate: $[2 \cdot 10^{-5}, 3 \cdot 10^{-5}, \mathbf{5 \cdot 10^{-5}}]$

Number of epochs: $[3, 4, 5, 6, 7, 8, 9, \mathbf{10}]$

The best configuration for E-BERT-concat is marked in bold. Figure 6 shows expected maximum performance as a function of the number of evaluated configurations (Dodge et al., 2019).

Entity linking (AIDA)

Data

We downloaded the AIDA dataset from:

- https://allennlp.s3-us-west-2.amazonaws.com/knowbert/wiki_entity_linking/aida_train.txt
- https://allennlp.s3-us-west-2.amazonaws.com/knowbert/wiki_entity_linking/aida_dev.txt
- https://allennlp.s3-us-west-2.amazonaws.com/knowbert/wiki_entity_linking/aida_test.txt

Preprocessing

Each AIDA file contains documents with annotated entity spans (which must be predicted). The documents are already whitespace tokenized, and we further tokenize words into wordpieces with the standard BERT tokenizer. If a document is too long (length > 512), we split it into smaller chunks by (a) finding the sentence boundary that is closest to the document midpoint, (b) splitting the document, and (c) repeating this process recursively until all chunks are short enough. Table 8 shows dataset statistics.

Hyperparameters

We tune batch size and peak learning rate on the AIDA dev set (selection criterion: strong match micro F1). We do a full search over the following hyperparameter space:

Batch size: $[16, 32, 64, \mathbf{128}]$

Learning rate: $[\mathbf{2 \cdot 10^{-5}}, 3 \cdot 10^{-5}, 5 \cdot 10^{-5}]$

The best configuration for E-BERT-MLM is marked in bold. Figure 7 shows expected maximum performance as a function of the number of evaluated configurations (Dodge et al., 2019).

# relations	80		
# unique entities	54648		
	train	dev	test
# samples	8000	16000	16000
# samples per relation	100	200	200

Table 7: Relation classification dataset statistics.

# unique gold entities	5574		
# unique candidate entities	463663		
	train	dev	test
# documents	946	216	231
# documents (after chunking)	1111	276	271
# potential spans (candidate generator)	153103	38012	34936
# gold entities	18454	4778	4478

Table 8: Entity linking (AIDA) dataset statistics.

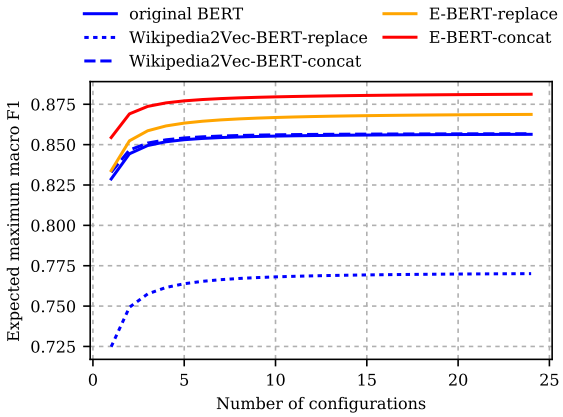

Figure 6: Relation classification: Expected maximum macro F1 (dev set) as a function of the number of hyperparameter configurations.

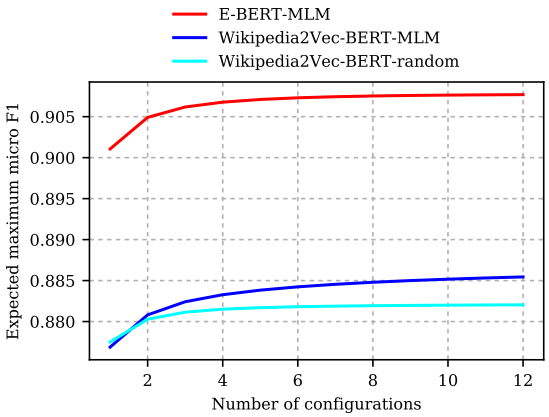

Figure 7: Entity linking: Expected maximum micro F1 (dev set) as a function of the number of hyperparameter configurations.

Model size:		BASE					LARGE			
Relation (dataset)	Model	original BERT	E-BERT replace	E-BERT-concat	ERNIE	Know-Bert	original BERT	E-BERT-replace	E-BERT-concat	number of questions
T-REx:P17	(0, original LAMA)	31.3	53.7	52.4	55.3	23.7	36.5	43.3	42.8	930
T-REx:P17	(1)	31.0	55.0	53.3	55.5	23.2	36.2	44.5	43.3	885
T-REx:P17	(2, LAMA-UHN)	31.0	55.0	53.3	55.5	23.2	36.2	44.5	43.3	885
T-REx:P19	(0, original LAMA)	21.1	26.4	28.1	28.7	23.3	22.2	24.6	25.3	944
T-REx:P19	(1)	20.6	26.5	27.5	28.2	22.9	21.8	24.5	24.8	933
T-REx:P19	(2, LAMA-UHN)	9.8	20.3	18.7	19.4	12.2	11.7	18.1	15.5	728
T-REx:P20	(0, original LAMA)	27.9	29.7	35.8	16.6	31.1	31.7	37.1	33.5	953
T-REx:P20	(1)	28.2	29.9	36.0	16.5	31.0	32.0	37.2	33.8	944
T-REx:P20	(2, LAMA-UHN)	15.5	21.5	23.3	8.4	20.0	18.9	27.3	22.6	656
T-REx:P27	(0, original LAMA)	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.1	966
T-REx:P27	(1)	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.1	945
T-REx:P27	(2, LAMA-UHN)	0.0	0.0	0.2	0.0	0.1	0.0	0.0	0.2	423
T-REx:P30	(0, original LAMA)	25.4	69.9	69.8	66.8	24.0	28.0	75.0	60.4	975
T-REx:P30	(1)	25.1	70.3	69.9	66.6	23.9	27.5	75.0	60.3	963
T-REx:P30	(2, LAMA-UHN)	25.1	70.3	69.9	66.6	23.9	27.5	75.0	60.3	963
T-REx:P31	(0, original LAMA)	36.7	25.5	46.9	43.7	18.7	30.2	12.3	16.1	922
T-REx:P31	(1)	21.1	28.4	35.8	30.3	12.4	16.3	9.9	9.8	564
T-REx:P31	(2, LAMA-UHN)	21.1	28.4	35.8	30.3	12.4	16.3	9.9	9.8	564
T-REx:P36	(0, original LAMA)	62.2	42.1	61.6	57.3	62.2	67.0	44.7	66.0	703
T-REx:P36	(1)	51.5	41.9	53.9	45.9	51.7	57.5	43.8	58.8	534
T-REx:P36	(2, LAMA-UHN)	51.5	41.9	53.9	45.9	51.7	57.5	43.8	58.8	534
T-REx:P37	(0, original LAMA)	54.6	51.2	56.5	60.2	53.1	61.5	54.3	62.7	966
T-REx:P37	(1)	52.9	51.6	55.5	59.4	51.9	60.5	54.2	62.1	924
T-REx:P37	(2, LAMA-UHN)	52.9	51.6	55.5	59.4	51.9	60.5	54.2	62.1	924
T-REx:P39	(0, original LAMA)	8.0	22.9	22.5	17.0	17.2	4.7	8.1	8.6	892
T-REx:P39	(1)	7.5	23.0	22.3	17.1	16.5	4.6	8.1	8.5	878
T-REx:P39	(2, LAMA-UHN)	7.5	23.0	22.3	17.1	16.5	4.6	8.1	8.5	878
T-REx:P47	(0, original LAMA)	13.7	8.9	10.8	9.8	14.0	18.2	15.1	15.9	922
T-REx:P47	(1)	13.6	9.1	10.7	9.6	13.9	18.6	15.2	15.9	904
T-REx:P47	(2, LAMA-UHN)	13.6	9.1	10.7	9.6	13.9	18.6	15.2	15.9	904
T-REx:P101	(0, original LAMA)	9.9	37.8	40.8	16.7	12.2	11.5	37.8	36.1	696
T-REx:P101	(1)	9.5	38.2	40.9	16.1	11.4	10.8	38.0	35.8	685
T-REx:P101	(2, LAMA-UHN)	9.5	38.2	40.9	16.1	11.4	10.8	38.0	35.8	685
T-REx:P103	(0, original LAMA)	72.2	85.8	86.8	85.5	73.4	78.2	84.4	84.9	977
T-REx:P103	(1)	72.1	85.7	86.8	85.4	73.3	78.2	84.4	84.9	975
T-REx:P103	(2, LAMA-UHN)	45.8	81.9	74.7	83.6	72.2	58.6	81.2	71.1	415
T-REx:P106	(0, original LAMA)	0.6	6.5	5.4	8.4	1.6	0.6	4.3	2.1	958
T-REx:P106	(1)	0.6	6.5	5.4	8.4	1.6	0.6	4.3	2.1	958
T-REx:P106	(2, LAMA-UHN)	0.6	6.5	5.4	8.4	1.6	0.6	4.3	2.1	958
T-REx:P108	(0, original LAMA)	6.8	9.9	23.2	14.1	10.7	1.6	11.7	15.9	383
T-REx:P108	(1)	6.5	9.9	23.0	13.9	10.5	1.3	11.8	16.0	382
T-REx:P108	(2, LAMA-UHN)	6.5	9.9	23.0	13.9	10.5	1.3	11.8	16.0	382
T-REx:P127	(0, original LAMA)	34.8	24.0	34.9	36.2	31.4	34.8	25.3	35.8	687
T-REx:P127	(1)	14.2	19.7	23.5	17.1	15.5	14.6	21.1	24.6	451
T-REx:P127	(2, LAMA-UHN)	14.2	19.7	23.5	17.1	15.5	14.6	21.1	24.6	451

Table 9: Mean Hits@1 and number of questions per LAMA relation. 0: original LAMA dataset, 1: after applying heuristic 1 (string match filter), 2: after applying both heuristics (LAMA-UHN).

Model size:		BASE					LARGE			
Relation (dataset)	Model	original BERT	E-BERT replace	E-BERT-concat	ERNIE	Know-Bert	original BERT	E-BERT-replace	E-BERT-concat	number of questions
T-REx:P131	(0, original LAMA)	23.3	33.4	36.4	37.3	27.7	26.3	31.4	37.2	881
T-REx:P131	(1)	16.7	32.0	33.9	32.7	21.5	20.1	31.0	33.4	706
T-REx:P131	(2, LAMA-UHN)	16.7	32.0	33.9	32.7	21.5	20.1	31.0	33.4	706
T-REx:P136	(0, original LAMA)	0.8	5.2	9.1	0.6	0.6	1.3	6.9	13.1	931
T-REx:P136	(1)	0.2	5.1	8.7	0.2	0.1	0.2	6.9	12.2	913
T-REx:P136	(2, LAMA-UHN)	0.2	5.1	8.7	0.2	0.1	0.2	6.9	12.2	913
T-REx:P138	(0, original LAMA)	61.6	8.8	26.5	0.2	63.7	45.1	2.6	24.0	645
T-REx:P138	(1)	5.0	10.0	8.8	0.0	6.9	4.4	4.4	6.2	160
T-REx:P138	(2, LAMA-UHN)	5.0	10.0	8.8	0.0	6.9	4.4	4.4	6.2	160
T-REx:P140	(0, original LAMA)	0.6	0.6	1.1	0.0	0.8	0.6	1.1	0.6	473
T-REx:P140	(1)	0.4	0.6	0.9	0.0	0.6	0.4	0.9	0.4	467
T-REx:P140	(2, LAMA-UHN)	0.4	0.6	0.9	0.0	0.6	0.4	0.9	0.4	467
T-REx:P159	(0, original LAMA)	32.4	30.3	48.3	41.8	36.8	34.7	22.3	45.2	967
T-REx:P159	(1)	23.1	31.6	41.9	34.4	28.7	25.6	20.9	37.8	843
T-REx:P159	(2, LAMA-UHN)	23.1	31.6	41.9	34.4	28.7	25.6	20.9	37.8	843
T-REx:P176	(0, original LAMA)	85.6	41.6	74.6	81.8	90.0	87.5	36.6	81.3	982
T-REx:P176	(1)	31.4	42.9	51.8	26.2	51.3	40.8	44.5	57.1	191
T-REx:P176	(2, LAMA-UHN)	31.4	42.9	51.8	26.2	51.3	40.8	44.5	57.1	191
T-REx:P178	(0, original LAMA)	62.8	49.8	66.6	60.1	70.3	70.8	51.2	69.4	592
T-REx:P178	(1)	40.7	42.6	51.6	36.9	52.2	53.6	51.1	57.7	366
T-REx:P178	(2, LAMA-UHN)	40.7	42.6	51.6	36.9	52.2	53.6	51.1	57.7	366
T-REx:P190	(0, original LAMA)	2.4	2.9	2.5	2.6	2.8	2.3	2.3	2.8	995
T-REx:P190	(1)	1.5	2.4	1.6	1.6	2.0	1.7	1.9	2.3	981
T-REx:P190	(2, LAMA-UHN)	1.5	2.4	1.6	1.6	2.0	1.7	1.9	2.3	981
T-REx:P264	(0, original LAMA)	9.6	30.5	33.6	13.3	21.2	8.2	23.1	15.6	429
T-REx:P264	(1)	9.6	30.6	33.4	13.3	21.3	8.2	23.1	15.7	428
T-REx:P264	(2, LAMA-UHN)	9.6	30.6	33.4	13.3	21.3	8.2	23.1	15.7	428
T-REx:P276	(0, original LAMA)	41.5	23.8	47.7	48.4	43.3	43.8	23.1	51.8	959
T-REx:P276	(1)	19.8	26.1	31.7	27.0	20.6	23.4	25.0	36.0	625
T-REx:P276	(2, LAMA-UHN)	19.8	26.1	31.7	27.0	20.6	23.4	25.0	36.0	625
T-REx:P279	(0, original LAMA)	30.7	14.7	30.7	29.4	31.6	33.5	15.5	29.8	963
T-REx:P279	(1)	3.8	8.6	8.0	4.6	5.3	6.8	8.6	10.1	474
T-REx:P279	(2, LAMA-UHN)	3.8	8.6	8.0	4.6	5.3	6.8	8.6	10.1	474
T-REx:P361	(0, original LAMA)	23.6	19.6	23.0	25.8	26.6	27.4	22.3	25.4	932
T-REx:P361	(1)	12.6	17.9	17.7	13.7	15.3	18.5	20.2	22.0	633
T-REx:P361	(2, LAMA-UHN)	12.6	17.9	17.7	13.7	15.3	18.5	20.2	22.0	633
T-REx:P364	(0, original LAMA)	44.5	61.7	64.0	48.0	40.9	51.1	60.6	61.3	856
T-REx:P364	(1)	43.5	61.7	63.5	47.4	40.0	50.7	60.5	61.2	841
T-REx:P364	(2, LAMA-UHN)	43.5	61.7	63.5	47.4	40.0	50.7	60.5	61.2	841
T-REx:P407	(0, original LAMA)	59.2	68.0	68.8	53.8	60.1	62.1	57.9	56.3	877
T-REx:P407	(1)	57.6	69.5	67.9	53.1	58.6	61.0	59.0	55.2	834
T-REx:P407	(2, LAMA-UHN)	57.6	69.5	67.9	53.1	58.6	61.0	59.0	55.2	834
T-REx:P413	(0, original LAMA)	0.5	0.1	0.0	0.0	41.7	4.1	14.0	7.0	952
T-REx:P413	(1)	0.5	0.1	0.0	0.0	41.7	4.1	14.0	7.0	952
T-REx:P413	(2, LAMA-UHN)	0.5	0.1	0.0	0.0	41.7	4.1	14.0	7.0	952

Table 10: Mean Hits@1 and number of questions per LAMA relation (cont'd). 0: original LAMA dataset, 1: after applying heuristic 1 (string match filter), 2: after applying both heuristics (LAMA-UHN).

Relation (dataset	Model	original BERT	E-BERT replace	E-BERT-concat	ERNIE	Know-Bert	original BERT	E-BERT-replace	E-BERT-concat	number of questions
	Model size:	BASE					LARGE			
T-REx:P449	(0, original LAMA)	20.9	30.9	34.7	33.8	57.0	24.0	32.5	28.6	881
T-REx:P449	(1)	18.8	31.1	33.4	32.0	56.0	21.8	32.9	27.5	848
T-REx:P449	(2, LAMA-UHN)	18.8	31.1	33.4	32.0	56.0	21.8	32.9	27.5	848
T-REx:P463	(0, original LAMA)	67.1	61.8	68.9	43.1	35.6	61.3	52.0	66.7	225
T-REx:P463	(1)	67.1	61.8	68.9	43.1	35.6	61.3	52.0	66.7	225
T-REx:P463	(2, LAMA-UHN)	67.1	61.8	68.9	43.1	35.6	61.3	52.0	66.7	225
T-REx:P495	(0, original LAMA)	16.5	46.3	48.3	1.0	30.8	29.7	56.7	46.9	909
T-REx:P495	(1)	15.0	46.0	47.5	0.9	29.6	28.5	56.6	46.2	892
T-REx:P495	(2, LAMA-UHN)	15.0	46.0	47.5	0.9	29.6	28.5	56.6	46.2	892
T-REx:P527	(0, original LAMA)	11.1	7.4	11.9	5.4	12.9	10.5	8.9	12.9	976
T-REx:P527	(1)	5.7	7.6	8.7	0.5	3.0	4.2	8.7	6.3	804
T-REx:P527	(2, LAMA-UHN)	5.7	7.6	8.7	0.5	3.0	4.2	8.7	6.3	804
T-REx:P530	(0, original LAMA)	2.8	1.8	2.0	2.3	2.8	2.7	2.3	2.8	996
T-REx:P530	(1)	2.8	1.8	2.0	2.3	2.8	2.7	2.3	2.8	996
T-REx:P530	(2, LAMA-UHN)	2.8	1.8	2.0	2.3	2.8	2.7	2.3	2.8	996
T-REx:P740	(0, original LAMA)	7.6	10.5	14.7	0.0	10.4	6.0	13.1	10.4	936
T-REx:P740	(1)	5.9	10.3	13.5	0.0	9.0	5.2	12.7	9.5	910
T-REx:P740	(2, LAMA-UHN)	5.9	10.3	13.5	0.0	9.0	5.2	12.7	9.5	910
T-REx:P937	(0, original LAMA)	29.8	33.0	38.8	40.0	32.3	24.9	28.3	34.5	954
T-REx:P937	(1)	29.9	32.9	38.7	39.9	32.2	24.8	28.2	34.4	950
T-REx:P937	(2, LAMA-UHN)	29.9	32.9	38.7	39.9	32.2	24.8	28.2	34.4	950
T-REx:P1001	(0, original LAMA)	70.5	56.9	76.0	75.7	73.0	73.3	49.5	78.0	701
T-REx:P1001	(1)	38.1	67.7	66.7	65.6	43.4	40.7	60.3	66.7	189
T-REx:P1001	(2, LAMA-UHN)	38.1	67.7	66.7	65.6	43.4	40.7	60.3	66.7	189
T-REx:P1303	(0, original LAMA)	7.6	20.3	26.6	5.3	9.1	12.5	29.7	33.2	949
T-REx:P1303	(1)	7.6	20.3	26.6	5.3	9.1	12.5	29.7	33.2	949
T-REx:P1303	(2, LAMA-UHN)	7.6	20.3	26.6	5.3	9.1	12.5	29.7	33.2	949
T-REx:P1376	(0, original LAMA)	73.9	41.5	62.0	71.8	75.2	82.1	47.4	70.1	234
T-REx:P1376	(1)	74.8	42.2	62.8	73.4	75.2	83.5	48.6	72.0	218
T-REx:P1376	(2, LAMA-UHN)	74.8	42.2	62.8	73.4	75.2	83.5	48.6	72.0	218
T-REx:P1412	(0, original LAMA)	65.0	54.0	67.8	73.1	69.2	63.6	49.3	61.2	969
T-REx:P1412	(1)	65.0	54.0	67.8	73.1	69.2	63.6	49.3	61.2	969
T-REx:P1412	(2, LAMA-UHN)	37.7	42.9	47.4	69.2	65.7	51.5	43.5	54.8	361
Google-RE:date_of_birth	(0)	1.6	1.5	1.9	1.9	2.4	1.5	1.5	1.3	1825
Google-RE:date_of_birth	(1)	1.6	1.5	1.9	1.9	2.4	1.5	1.5	1.3	1825
Google-RE:date_of_birth	(2)	1.6	1.5	1.9	1.9	2.4	1.5	1.5	1.3	1825
Google-RE:place_of_birth	(0)	14.9	16.2	16.9	17.7	17.4	16.1	14.8	16.6	2937
Google-RE:place_of_birth	(1)	14.9	16.2	16.8	17.7	17.4	16.0	14.8	16.6	2934
Google-RE:place_of_birth	(2)	5.9	9.4	8.2	10.3	9.4	7.2	8.5	7.9	2451
Google-RE:place_of_death	(0)	13.1	12.8	14.9	6.4	13.4	14.0	17.0	14.9	766
Google-RE:place_of_death	(1)	13.1	12.8	14.9	6.4	13.4	14.0	17.0	14.9	766
Google-RE:place_of_death	(2)	6.6	7.5	7.8	2.0	7.5	7.6	11.8	8.9	655

Table 11: Mean Hits@1 and number of questions per LAMA relation (cont'd). 0: original LAMA dataset, 1: after applying heuristic 1 (string match filter), 2: after applying both heuristics (LAMA-UHN).