# LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention

**Ikuya Yamada**[1,2] ikuya@ousia.jp
**Akari Asai**[3] akari@cs.washington.edu
**Hiroyuki Shindo**[4,2] shindo@is.naist.jp
**Hideaki Takeda**[5] takeda@nii.ac.jp
**Yuji Matsumoto**[2] matsu@is.naist.jp

[1]Studio Ousia [2]RIKEN AIP [3]University of Washington
[4]Nara Institute of Science and Technology [5]National Institute of Informatics

## Abstract

Entity representations are useful in natural language tasks involving entities. In this paper, we propose new pretrained contextualized representations of words and entities based on the bidirectional transformer (Vaswani et al., 2017). The proposed model treats words and entities in a given text as independent tokens, and outputs contextualized representations of them. Our model is trained using a new pretraining task based on the masked language model of BERT (Devlin et al., 2019). The task involves predicting randomly masked words and entities in a large entity-annotated corpus retrieved from Wikipedia. We also propose an *entity-aware* self-attention mechanism that is an extension of the self-attention mechanism of the transformer, and considers the types of tokens (words or entities) when computing attention scores. The proposed model achieves impressive empirical performance on a wide range of entity-related tasks. In particular, it obtains state-of-the-art results on five well-known datasets: Open Entity (entity typing), TACRED (relation classification), CoNLL-2003 (named entity recognition), ReCoRD (cloze-style question answering), and SQuAD 1.1 (extractive question answering). Our source code and pretrained representations are available at https://github.com/studio-ousia/luke.

## 1 Introduction

Many natural language tasks involve entities, e.g., relation classification, entity typing, named entity recognition (NER), and question answering (QA). Key to solving such entity-related tasks is a model to learn the effective representations of entities. Conventional entity representations assign each entity a fixed embedding vector that stores information regarding the entity in a knowledge base (KB) (Bordes et al., 2013; Trouillon et al., 2016; Yamada et al., 2016, 2017). Although these models capture the rich information in the KB, they require entity linking to represent entities in a text, and cannot represent entities that do not exist in the KB.

By contrast, contextualized word representations (CWRs) based on the transformer (Vaswani et al., 2017), such as BERT (Devlin et al., 2019), and RoBERTa (Liu et al., 2020), provide effective general-purpose word representations trained with unsupervised pretraining tasks based on language modeling. Many recent studies have solved entity-related tasks using the contextualized representations of entities computed based on CWRs (Zhang et al., 2019; Peters et al., 2019; Joshi et al., 2020). However, the architecture of CWRs is not well suited to representing entities for the following two reasons: (1) Because CWRs do not output the span-level representations of entities, they typically need to learn how to compute such representations based on a downstream dataset that is typically small. (2) Many entity-related tasks, e.g., relation classification and QA, involve reasoning about the relationships between entities. Although the transformer can capture the complex relationships between words by relating them to each other multiple times using the self-attention mechanism (Clark et al., 2019; Reif et al., 2019), it is difficult to perform such reasoning between entities because many entities are split into multiple tokens in the model. Furthermore, the word-based pretraining task of CWRs is not suitable for learning the representations of entities because predicting a masked word given other words in the entity, e.g., predicting *"Rings"* given *"The Lord of the [MASK]"*, is clearly easier than predicting the entire entity.

In this paper, we propose new pretrained contextualized representations of words and entities by developing **LUKE** (**L**anguage **U**nderstanding with **K**nowledge-based **E**mbeddings). LUKE is based on a transformer (Vaswani et al., 2017) trained using a large amount of entity-annotated corpus



*Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6442–6454,
November 16–20, 2020. 

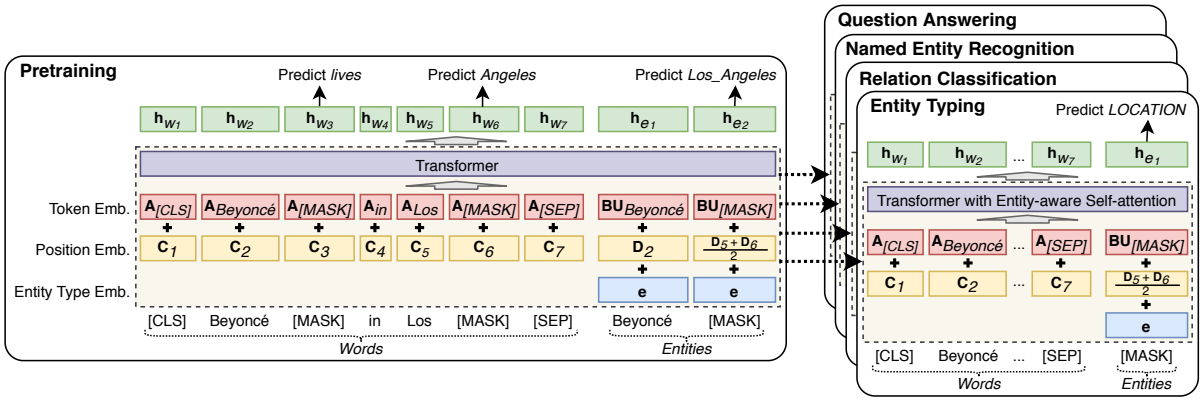


Figure 1: Architecture of LUKE using the input sentence "*Beyoncé lives in Los Angeles.*" LUKE outputs contextualized representation for each word and entity in the text. The model is trained to predict randomly masked words (e.g., *lives* and *Angeles* in the figure) and entities (e.g., *Los_Angeles* in the figure). Downstream tasks are solved using its output representations with linear classifiers.

obtained from Wikipedia. An important difference between LUKE and existing CWRs is that it treats not only words, but also entities as independent tokens, and computes intermediate and output representations for all tokens using the transformer (see Figure 1). Since entities are treated as tokens, LUKE can directly model the relationships between entities.

LUKE is trained using a new pretraining task, a straightforward extension of BERT's masked language model (MLM) (Devlin et al., 2019). The task involves randomly masking entities by replacing them with `[MASK]` entities, and trains the model by predicting the originals of these masked entities. We use RoBERTa as base pre-trained model, and conduct pretraining of the model by simultaneously optimizing the objectives of the MLM and our proposed task. When applied to downstream tasks, the resulting model can compute representations of arbitrary entities in the text using `[MASK]` entities as inputs. Furthermore, if entity annotation is available in the task, the model can compute entity representations based on the rich entity-centric information encoded in the corresponding entity embeddings.

Another key contribution of this paper is that it extends the transformer using our *entity-aware* self-attention mechanism. Unlike existing CWRs, our model needs to deal with two types of tokens, i.e., words and entities. Therefore, we assume that it is beneficial to enable the mechanism to easily determine the types of tokens. To this end, we enhance the self-attention mechanism by adopting different query mechanisms based on the attending token and the token attended to.

We validate the effectiveness of our proposed model by conducting extensive experiments on five standard entity-related tasks: entity typing, relation classification, NER, cloze-style QA, and extractive QA. Our model outperforms all baseline models, including RoBERTa, in all experiments, and obtains state-of-the-art results on five tasks: entity typing on the Open Entity dataset (Choi et al., 2018), relation classification on the TACRED dataset (Zhang et al., 2017), NER on the CoNLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003), cloze-style QA on the ReCoRD dataset (Zhang et al., 2018a), and extractive QA on the SQuAD 1.1 dataset (Rajpurkar et al., 2016). We publicize our source code and pretrained representations at `https://github.com/studio-ousia/luke`.

The main contributions of this paper are summarized as follows:

- We propose LUKE, a new contextualized representations specifically designed to address entity-related tasks. LUKE is trained to predict randomly masked words and entities using a large amount of entity-annotated corpus obtained from Wikipedia.
- We introduce an entity-aware self-attention mechanism, an effective extension of the original mechanism of transformer. The proposed mechanism considers the type of the tokens (words or entities) when computing attention scores.
- LUKE achieves strong empirical performance and obtains state-of-the-art results on five popular datasets: Open Entity, TACRED, CoNLL-2003, ReCoRD, and SQuAD 1.1.

## 2 Related Work

**Static Entity Representations** Conventional entity representations assign a fixed embedding to each entity in the KB. They include knowledge embeddings trained on knowledge graphs (Bordes et al., 2013; Yang et al., 2015; Trouillon et al., 2016), and embeddings trained using textual contexts or descriptions of entities retrieved from a KB (Yamada et al., 2016, 2017; Cao et al., 2017; Ganea and Hofmann, 2017). Similar to our pretraining task, **NTEE** (Yamada et al., 2017) and **RELIC** (Ling et al., 2020) use an approach that trains entity embeddings by predicting entities given their textual contexts obtained from a KB. The main drawbacks of this line of work, when representing entities in text, are that (1) they need to resolve entities in the text to corresponding KB entries to represent the entities, and (2) they cannot represent entities that do not exist in the KB.

**Contextualized Word Representations** Many recent studies have addressed entity-related tasks based on the contextualized representations of entities in text computed using the word representations of CWRs (Zhang et al., 2019; Baldini Soares et al., 2019; Peters et al., 2019; Joshi et al., 2020; Wang et al., 2019b, 2020). Representative examples of CWRs are **ELMo** (Peters et al., 2018) and **BERT** (Devlin et al., 2019), which are based on deep bidirectional long short-term memory (LSTM) and the transformer (Vaswani et al., 2017), respectively. BERT is trained using an MLM, a pretraining task that masks random words in the text and trains the model to predict the masked words. Most recent CWRs, such as **RoBERTa** (Liu et al., 2020), **XLNet** (Yang et al., 2019), **SpanBERT** (Joshi et al., 2020), **ALBERT** (Lan et al., 2020), **BART** (Lewis et al., 2020), and **T5** (Raffel et al., 2020), are based on transformer trained using a task equivalent to or similar to the MLM. Similar to our proposed pretraining task that masks entities instead of words, several recent CWRs, e.g., SpanBERT, ALBERT, BART, and T5, have extended the MLM by randomly masking word spans instead of single words.

Furthermore, various recent studies have explored methods to enhance CWRs by injecting them with *knowledge* from external sources, such as KBs. **ERNIE** (Zhang et al., 2019) and **KnowBERT** (Peters et al., 2019) use a similar idea to enhance CWRs using static entity embeddings separately learned from a KB. **WKLM** (Xiong et al., 2020) trains the model to detect whether an entity name in text is replaced by another entity name of the same type. **KEPLER** (Wang et al., 2019b) conducts pretraining based on the MLM and a knowledge-embedding objective (Bordes et al., 2013). **K-Adapter** (Wang et al., 2020) was proposed concurrently with our work, and extends CWRs using *neural adapters* that inject factual and linguistic knowledge. This line of work is related to ours because our pretraining task also enhances the model using information in the KB.

Unlike the CWRs mentioned above, LUKE uses an improved transformer architecture with an entity-aware self-attention mechanism that is designed to effectively solve entity-related tasks. LUKE also outputs entity representations by learning how to compute them during pretraining. It achieves superior empirical results to existing CWRs and knowledge-enhanced CWRs in all of our experiments.

## 3 LUKE

Figure 1 shows the architecture of LUKE. The model adopts a multi-layer bidirectional transformer (Vaswani et al., 2017). It treats words and entities in the document as input tokens, and computes a representation for each token. Formally, given a sequence consisting of $m$ words $w_1, w_2, ..., w_m$ and $n$ entities $e_1, e_2, ..., e_n$, our model computes $D$-dimensional word representations $\mathbf{h}_{w_1}, \mathbf{h}_{w_2}, ..., \mathbf{h}_{w_m}$, where $\mathbf{h}_w \in \mathbb{R}^D$, and entity representations $\mathbf{h}_{e_1}, \mathbf{h}_{e_2}, ..., \mathbf{h}_{e_n}$, where $\mathbf{h}_e \in \mathbb{R}^D$. The entities can be Wikipedia entities (e.g., *Beyoncé* in Figure 1) or special entities (e.g., `[MASK]`).

### 3.1 Input Representation

The input representation of a token (word or entity) is computed using the following three embeddings:

- **Token embedding** represents the corresponding token. We denote the word token embedding by $\mathbf{A} \in \mathbb{R}^{V_w \times D}$, where $V_w$ is the number of words in our vocabulary. For computational efficiency, we represent the entity token embedding by decomposing it into two small matrices, $\mathbf{B} \in \mathbb{R}^{V_e \times H}$ and $\mathbf{U} \in \mathbb{R}^{H \times D}$, where $V_e$ is the number of entities in our vocabulary. Hence, the full matrix of the entity token embedding can be computed as $\mathbf{BU}$.

- **Position embedding** represents the position of the token in a word sequence. A word and an entity appearing at the $i$-th position in the sequence are represented as $\mathbf{C}_i \in \mathbb{R}^D$ and $\mathbf{D}_i \in \mathbb{R}^D$, respectively. If an entity name contains multiple words, its position embedding is computed by averaging the embeddings of the corresponding positions, as shown in Figure 1.
- **Entity type embedding** represents that the token is an entity. The embedding is a single vector denoted by $\mathbf{e} \in \mathbb{R}^D$.

The input representation of a word and that of an entity are computed by summing the token and position embeddings, and the token, position, and entity type embeddings, respectively. Following past work (Devlin et al., 2019; Liu et al., 2020), we insert special tokens `[CLS]` and `[SEP]` into the word sequence as the first and last words, respectively.

### 3.2 Entity-aware Self-attention

The self-attention mechanism is the foundation of the transformer (Vaswani et al., 2017), and relates tokens each other based on the attention score between each pair of tokens. Given a sequence of input vectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$, where $\mathbf{x}_i \in \mathbb{R}^D$, each of the output vectors $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k$, where $\mathbf{y}_i \in \mathbb{R}^L$, is computed based on the weighted sum of the transformed input vectors. Here, each input and output vector corresponds to a token (a word or an entity) in our model; therefore, $k = m + n$. The $i$-th output vector $\mathbf{y}_i$ is computed as:

$$\mathbf{y}_i = \sum_{j=1}^{k} \alpha_{ij} \mathbf{V} \mathbf{x}_j$$
$$e_{ij} = \frac{\mathbf{K}\mathbf{x}_j^\top \mathbf{Q}\mathbf{x}_i}{\sqrt{L}}$$
$$\alpha_{ij} = \mathrm{softmax}(e_{ij})$$

where $\mathbf{Q} \in \mathbb{R}^{L \times D}$, $\mathbf{K} \in \mathbb{R}^{L \times D}$, and $\mathbf{V} \in \mathbb{R}^{L \times D}$ denote the query, key, and value matrices, respectively.

Because LUKE handles two types of tokens (i.e., words and entities), we assume that it is beneficial to use the information of target token types when computing the attention scores ($e_{ij}$). With this in mind, we enhance the mechanism by introducing an *entity-aware* query mechanism that uses a different query matrix for each possible pair of token types of $\mathbf{x}_i$ and $\mathbf{x}_j$. Formally, the attention score $e_{ij}$ is computed as follows:

$$e_{ij} = \begin{cases} \mathbf{K}\mathbf{x}_j^\top \mathbf{Q}\mathbf{x}_i, & \text{if both } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are words} \\ \mathbf{K}\mathbf{x}_j^\top \mathbf{Q}_{w2e}\mathbf{x}_i, & \text{if } \mathbf{x}_i \text{ is word and } \mathbf{x}_j \text{ is entity} \\ \mathbf{K}\mathbf{x}_j^\top \mathbf{Q}_{e2w}\mathbf{x}_i, & \text{if } \mathbf{x}_i \text{ is entity and } \mathbf{x}_j \text{ is word} \\ \mathbf{K}\mathbf{x}_j^\top \mathbf{Q}_{e2e}\mathbf{x}_i, & \text{if both } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are entities} \end{cases}$$

where $\mathbf{Q}_{w2e}$, $\mathbf{Q}_{e2w}$, $\mathbf{Q}_{e2e} \in \mathbb{R}^{L \times D}$ are query matrices. Note that the computational costs of the original mechanism and our proposed mechanism are identical except the additional cost of computing gradients and updating the parameters of the additional query matrices at the training time.

### 3.3 Pretraining Task

To pretrain LUKE, we use the conventional MLM and a new pretraining task that is an extension of the MLM to learn entity representations. In particular, we treat hyperlinks in Wikipedia as entity annotations, and train the model using a large entity-annotated corpus retrieved from Wikipedia. We randomly mask a certain percentage of the entities by replacing them with special `[MASK]` entities[1] and then train the model to predict the masked entities. Formally, the original entity corresponding to a masked entity is predicted by applying the softmax function over all entities in our vocabulary:

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{B}\mathbf{T}\mathbf{m} + \mathbf{b}_o)$$
$$\mathbf{m} = \mathrm{layer\_norm}\big(\mathrm{gelu}(\mathbf{W}_h \mathbf{h}_e + \mathbf{b}_h)\big)$$

where $\mathbf{h}_e$ is the representation corresponding to the masked entity, $\mathbf{T} \in \mathbb{R}^{H \times D}$ and $\mathbf{W}_h \in \mathbb{R}^{D \times D}$ are weight matrices, $\mathbf{b}_o \in \mathbb{R}^{V_e}$ and $\mathbf{b}_h \in \mathbb{R}^D$ are bias vectors, $\mathrm{gelu}(\cdot)$ is the gelu activation function (Hendrycks and Gimpel, 2016), and $\mathrm{layer\_norm}(\cdot)$ is the layer normalization function (Lei Ba et al., 2016). Our final loss function is the sum of MLM loss and cross-entropy loss on predicting the masked entities, where the latter is computed identically to the former.

### 3.4 Modeling Details

Our model configuration follows RoBERTa$_{\text{LARGE}}$ (Liu et al., 2020), pretrained CWRs based on a bidirectional transformer and a variant of BERT (Devlin et al., 2019). In particular, our model is based on the bidirectional transformer with $D = 1024$

[1] Note that LUKE uses two different `[MASK]` tokens: the `[MASK]` word for MLM and the `[MASK]` entity for our proposed pretraining task.

hidden dimensions, 24 hidden layers, $L = 64$ attention head dimensions, and 16 self-attention heads. The number of dimensions of the entity token embedding is set to $H = 256$. The total number of parameters is approximately 483 M, consisting of 355 M in RoBERTa and 128 M in our entity embeddings. The input text is tokenized into words using RoBERTa's tokenizer with the vocabulary consisting of $V_w = 50K$ words. For computational efficiency, our entity vocabulary does not include all entities but only the $V_e = 500K$ entities most frequently appearing in our entity annotations. The entity vocabulary also includes two special entities, i.e., `[MASK]` and `[UNK]`.

The model is trained via iterations over Wikipedia pages in a random order for 200K steps. To reduce training time, we initialize the parameters that LUKE have in common with RoBERTa (parameters in the transformer and the embeddings for words) using RoBERTa. Following past work (Devlin et al., 2019; Liu et al., 2020), we mask 15% of all words and entities at random. If an entity does not exist in the vocabulary, we replace it with the `[UNK]` entity. We perform pretraining using the original self-attention mechanism rather than our entity-aware self-attention mechanism because we want an ablation study of our mechanism but can not afford to run pretraining twice. Query matrices of our self-attention mechanism ($\mathbf{Q}_{w2e}$, $\mathbf{Q}_{e2w}$, and $\mathbf{Q}_{e2e}$) are learned using downstream datasets. Further details of our pretraining are described in Appendix A.

## 4 Experiments

We conduct extensive experiments using five entity-related tasks: entity typing, relation classification, NER, cloze-style QA, and extractive QA. We use similar model architectures for all tasks based on a simple linear classifier on top of the representations of words, entities, or both. Unless otherwise specified, we create the input word sequence by inserting tokens of `[CLS]` and `[SEP]` into the original word sequence as the first and the last tokens, respectively. The input entity sequence is built using `[MASK]` entities, special entities introduced for the task, or Wikipedia entities. The token embedding of a task-specific special entity is initialized using that of the `[MASK]` entity, and the query matrices of our entity-aware self-attention mechanism ($\mathbf{Q}_{w2e}$, $\mathbf{Q}_{e2w}$, and $\mathbf{Q}_{e2e}$) are initialized using the original query matrix $\mathbf{Q}$.

| Name | Prec. | Rec. | F1 |
|---|---|---|---|
| UFET (Zhang et al., 2019) | 77.4 | 60.6 | 68.0 |
| BERT (Zhang et al., 2019) | 76.4 | 71.0 | 73.6 |
| ERNIE (Zhang et al., 2019) | 78.4 | 72.9 | 75.6 |
| KEPLER (Wang et al., 2019b) | 77.2 | 74.2 | 75.7 |
| KnowBERT (Peters et al., 2019) | 78.6 | 73.7 | 76.1 |
| K-Adapter (Wang et al., 2020) | 79.3 | 75.8 | 77.5 |
| RoBERTa (Wang et al., 2020) | 77.6 | 75.0 | 76.2 |
| LUKE | **79.9** | **76.6** | **78.2** |

Table 1: Results of entity typing on the Open Entity dataset.

Because we use RoBERTa as the base model in our pretraining, we use it as our primary baseline for all tasks. We omit a description of the baseline models in each section if they are described in Section 2. Further details of our experiments are available in Appendix B.

### 4.1 Entity Typing

We first conduct experiments on entity typing, which is the task of predicting the types of an entity in the given sentence. Following Zhang et al. (2019), we use the Open Entity dataset (Choi et al., 2018), and consider only nine general entity types. Following Wang et al. (2020), we report loose micro-precision, recall, and F1, and employ the micro-F1 as the primary metric.

**Model** We represent the target entity using the `[MASK]` entity, and enter words and the entity in each sentence into the model. We then classify the entity using a linear classifier based on the corresponding entity representation. We treat the task as multi-label classification, and train the model using binary cross-entropy loss averaged over all entity types.

**Baselines** **UFET** (Choi et al., 2018) is a conventional model that computes context representations using the bidirectional LSTM. We also use BERT, RoBERTa, ERNIE, KnowBERT, KEPLER, and K-Adapter as baselines.

**Results** Table 1 shows the experimental results. LUKE significantly outperforms our primary baseline, RoBERTa, by 2.0 F1 points, and the previous best published model, KnowBERT, by 2.1 F1 points. Furthermore, LUKE achieves a new state of the art by outperforming K-Adapter by 0.7 F1 points.

| Name | Prec. | Rec. | F1 |
|---|---|---|---|
| BERT (Zhang et al., 2019) | 67.2 | 64.8 | 66.0 |
| C-GCN (Zhang et al., 2018b) | 69.9 | 63.3 | 66.4 |
| ERNIE (Zhang et al., 2019) | 70.0 | 66.1 | 68.0 |
| SpanBERT (Joshi et al., 2020) | 70.8 | 70.9 | 70.8 |
| MTB (Baldini Soares et al., 2019) | - | - | 71.5 |
| KnowBERT (Peters et al., 2019) | **71.6** | 71.4 | 71.5 |
| KEPLER (Wang et al., 2019b) | 70.4 | 73.0 | 71.7 |
| K-Adapter (Wang et al., 2020) | 68.9 | **75.4** | 72.0 |
| RoBERTa (Wang et al., 2020) | 70.2 | 72.4 | 71.3 |
| LUKE | 70.4 | 75.1 | **72.7** |

Table 2: Results of relation classification on the TACRED dataset.

### 4.2 Relation Classification

Relation classification determines the correct relation between *head* and *tail* entities in a sentence. We conduct experiments using TACRED dataset (Zhang et al., 2017), a large-scale relation classification dataset containing 106,264 sentences with 42 relation types. Following Wang et al. (2020), we report the micro-precision, recall, and F1, and use the micro-F1 as the primary metric.

**Model** We introduce two special entities, `[HEAD]` and `[TAIL]`, to represent the head and the tail entities, respectively, and input words and these two entities in each sentence to the model. We then solve the task using a linear classifier based on a concatenated representation of the head and tail entities. The model is trained using cross-entropy loss.

**Baselines** **C-GCN** (Zhang et al., 2018b) uses graph convolutional networks over dependency tree structures to solve the task. **MTB** (Baldini Soares et al., 2019) learns relation representations based on BERT through the *matching-the-blanks* task using a large amount of entity-annotated text. We also compare LUKE with BERT, RoBERTa, SpanBERT, ERNIE, KnowBERT, KEPLER, and K-Adapter.

**Results** The experimental results are presented in Table 2. LUKE clearly outperforms our primary baseline, RoBERTa, by 1.4 F1 points, and the previous best published models, namely MTB and KnowBERT, by 1.2 F1 points. Furthermore, it achieves a new state of the art by outperforming K-Adapter by 0.7 F1 points.

### 4.3 Named Entity Recognition

We conduct experiments on the NER task using the standard CoNLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003). Following past work, we report the span-level F1.

| Name | F1 |
|---|---|
| LSTM-CRF (Lample et al., 2016) | 91.0 |
| ELMo (Peters et al., 2018) | 92.2 |
| BERT (Devlin et al., 2019) | 92.8 |
| Akbik et al. (2018) | 93.1 |
| Baevski et al. (2019) | 93.5 |
| RoBERTa | 92.4 |
| LUKE | **94.3** |

Table 3: Results of named entity recognition on the CoNLL-2003 dataset.

**Model** Following Sohrab and Miwa (2018), we solve the task by enumerating all possible spans (or n-grams) in each sentence as entity name candidates, and classifying them into the target entity types or *non-entity* type, which indicates that the span is not an entity. For each sentence in the dataset, we enter words and the `[MASK]` entities corresponding to all possible spans. The representation of each span is computed by concatenating the word representations of the first and last words in the span, and the entity representation corresponding to the span. We classify each span using a linear classifier with its representation, and train the model using cross-entropy loss. We exclude spans longer than 16 words for computational efficiency. During the inference, we first exclude all spans classified into the *non-entity* type. To avoid selecting overlapping spans, we greedily select a span from the remaining spans based on the logit of its predicted entity type in descending order if the span does not overlap with those already selected. Following Devlin et al. (2019), we include the maximal document context in the target document.

**Baselines** **LSTM-CRF** (Lample et al., 2016) is a model based on the bidirectional LSTM with conditional random fields (CRF). **Akbik et al. (2018)** address the task using the bidirectional LSTM with CRF enhanced with character-level contextualized representations. Similarly, **Baevski et al. (2019)** use the bidirectional LSTM with CRF enhanced with CWRs based on a bidirectional transformer. We also use ELMo, BERT, and RoBERTa as baselines. To conduct a fair comparison with RoBERTa, we report its performance using the model described above with the span representation computed by concatenating the representations of the first and last words of the span.

**Results** The experimental results are shown in Table 3. LUKE outperforms RoBERTa by 1.9 F1

points. Furthermore, it achieves a new state of the art on this competitive dataset by outperforming the previous state of the art reported in Baevski et al. (2019) by 0.8 F1 points.

### 4.4 Cloze-style Question Answering

We evaluate our model on the ReCoRD dataset (Zhang et al., 2018a), a cloze-style QA dataset consisting of over 120K examples. An interesting characteristic of this dataset is that most of its questions cannot be solved without external knowledge. The following is an example question and its answer in the dataset:

> **Question:** According to claims in the suit, "Parts of 'Stairway to Heaven,' instantly recognizable to the music fans across the world, sound almost identical to significant portions of '**X**.'"
> **Answer:** Taurus

Given a question and a passage, the task is to find the entity mentioned in the passage that fits the missing entity (denoted by **X** in the question above). In this dataset, annotations of entity spans (start and end positions) in a passage are provided, and the answer is contained in the provided entity spans one or multiple times. Following past work, we evaluate the models using exact match (EM) and token-level F1 on the development and test sets.

**Model** We solve this task by assigning a relevance score to each entity in the passage and selecting the entity with the highest score as the answer. Following Liu et al. (2020), given a question $q_1, q_2, ..., q_j$, and a passage $p_1, p_2, ..., p_l$, the input word sequence is constructed as: `[CLS]` $q_1, q_2, ..., q_j$ `[SEP] [SEP]` $p_1, p_2, ..., p_l$ `[SEP]`. Further, we input `[MASK]` entities corresponding to the missing entity and all entities in the passage. We compute the relevance score of each entity in the passage using a linear classifier with the concatenated representation of the missing entity and the corresponding entity. We train the model using binary cross-entropy loss averaged over all entities in the passage, and select the entity with the highest score (logit) as the answer.

**Baselines** **DocQA+ELMo** (Clark and Gardner, 2018) is a model based on ELMo, bidirectional attention flow (Seo et al., 2017), and self-attention mechanism. **XLNet+Verifier** (Li et al., 2019) is a model based on XLNet with rule-based answer verification, and is the winner of a recent competition based on this dataset (Ostermann et al., 2019). We also use BERT and RoBERTa as baselines.

| Name | Dev EM | Dev F1 | Test EM | Test F1 |
|---|---|---|---|---|
| DocQA+ELMo (Zhang et al., 2018a) | 44.1 | 45.4 | 45.4 | 46.7 |
| BERT (Wang et al., 2019a) | - | - | 71.3 | 72.0 |
| XLNet+Verifier (Li et al., 2019) | 80.6 | 82.1 | 81.5 | 82.7 |
| RoBERTa (Liu et al., 2020) | 89.0 | 89.5 | - | - |
| RoBERTa (ensemble) (Liu et al., 2020) | - | - | 90.0 | 90.6 |
| LUKE | **90.8** | **91.4** | **90.6** | **91.2** |

Table 4: Results of cloze-style question answering on the ReCoRD dataset. All models except RoBERTa (ensemble) are based on a single model.

**Results** The results are presented in Table 4. LUKE significantly outperforms RoBERTa, the best baseline, on the development set by 1.8 EM points and 1.9 F1 points. Furthermore, it achieves superior results to RoBERTa (ensemble) on the test set without ensembling the models.

### 4.5 Extractive Question Answering

Finally, we conduct experiments using the well-known Stanford Question Answering Dataset (SQuAD) 1.1 consisting of 100K question/answer pairs (Rajpurkar et al., 2016). Given a question and a Wikipedia passage containing the answer, the task is to predict the answer span in the passage. Following past work, we report the EM and token-level F1 on the development and test sets.

**Model** We construct the word sequence from the question and the passage in the same way as in the previous experiment. Unlike in the other experiments, we input Wikipedia entities into the model based on entity annotations automatically generated on the question and the passage using a mapping from entity names (e.g., "U.S.") to their referent entities (e.g., *United States*). The mapping is automatically created using the entity hyperlinks in Wikipedia as described in detail in Appendix C. We solve this task using the same model architecture as that of BERT and RoBERTa. In particular, we use two linear classifiers independently on top of the word representations to predict the span boundary of the answer (i.e., the start and end positions), and train the model using cross-entropy loss.

**Baselines** We compare our models with the results of recent CWRs, including BERT, RoBERTa, SpanBERT, XLNet, and ALBERT. Because the results for RoBERTa and ALBERT are reported only on the development set, we conduct a comparison

| Name | Dev EM | Dev F1 | Test EM | Test F1 |
|---|---|---|---|---|
| BERT (Devlin et al., 2019) | 84.2 | 91.1 | 85.1 | 91.8 |
| SpanBERT (Joshi et al., 2020) | - | - | 88.8 | 94.6 |
| XLNet (Yang et al., 2019) | 89.0 | 94.5 | 89.9 | 95.1 |
| ALBERT (Lan et al., 2020) | 89.3 | 94.8 | - | - |
| RoBERTa (Liu et al., 2020) | 88.9 | 94.6 | - | - |
| LUKE | **89.8** | **95.0** | **90.2** | **95.4** |

Table 5: Results of extractive question answering on the SQuAD 1.1 dataset.

| Name | CoNLL-2003 (Test F1) | SQuAD (Dev EM) | SQuAD (Dev F1) |
|---|---|---|---|
| LUKE w/o entity inputs | 92.9 | 89.2 | 94.8 |
| LUKE | **94.3** | **89.8** | **95.0** |

Table 6: Ablation study of our entity representations.

with these models using this set. To conduct a fair compassion with RoBERTa, we use the same model architecture and hyper-parameters as those of RoBERTa (Liu et al., 2020).

**Results** The experimental results are presented in Table 5. LUKE outperforms our primary baseline, RoBERTa, by 0.9 EM points and 0.4 F1 points on the development set. Furthermore, it achieves a new state of the art on this competitive dataset by outperforming XLNet by 0.3 points both in terms of EM and F1. Note that XLNet uses a more sophisticated model involving beam search than the other models considered here.

## 5 Analysis

In this section, we provide a detailed analysis of LUKE by reporting three additional experiments.

### 5.1 Effects of Entity Representations

To investigate how our entity representations influence performance on downstream tasks, we perform an ablation experiment by addressing NER on the CoNLL-2003 dataset and extractive QA on the SQuAD dataset without inputting any entities. In this setting, LUKE uses only the word sequence to compute the representation for each word. We address the tasks using the same model architectures as those for RoBERTa described in the corresponding sections. As shown in Table 6, this setting clearly degrades performance, i.e., 1.4 F1 points on the CoNLL-2003 dataset and 0.6 EM points on the SQuAD dataset, demonstrating the effectiveness of our entity representations on these two tasks.

### 5.2 Effects of Entity-aware Self-attention

We conduct an ablation study of our entity-aware self-attention mechanism by comparing the performance of LUKE using our mechanism with that using the original mechanism of the transformer. As shown in Table 7, our entity-aware self-attention mechanism consistently outperforms the original mechanism across all tasks. Furthermore, we observe significant improvements on two kinds of tasks, relation classification (TACRED) and QA (ReCoRD and SQuAD). Because these tasks involve reasoning based on relationships between entities, we consider that our mechanism enables the model (i.e., attention heads) to easily focus on capturing the relationships between entities.

### 5.3 Effects of Extra Pretraining

As mentioned in Section 3.4, LUKE is based on RoBERTa with pretraining for 200K steps using our Wikipedia corpus. Because past studies (Liu et al., 2020; Lan et al., 2020) suggest that simply increasing the number of training steps of CWRs tends to improve performance on downstream tasks, the superior experimental results of LUKE compared with those of RoBERTa may be obtained because of its greater number of pretraining steps. To investigate this, we train another model based on RoBERTa with extra pretraining based on the MLM using the Wikipedia corpus for 200K training steps. The detailed configuration used in the pretraining is available in Appendix A.

We evaluate the performance of this model on the CoNLL-2003 and SQuAD datasets using the same model architectures as those for RoBERTa described in the corresponding sections. As shown in Table 8, the model achieves similar performance to the original RoBERTa on both datasets, which indicates that the superior performance of LUKE is not owing to its longer pretraining.

## 6 Conclusions

In this paper, we propose LUKE, new pretrained contextualized representations of words and entities based on the transformer. LUKE outputs the contextualized representations of words and entities using an improved transformer architecture with using a novel entity-aware self-attention mechanism. The experimental results prove its effectiveness on various entity-related tasks. Future work involves applying LUKE to domain-specific tasks, such as those in biomedical and legal domains.

| Name | Open Entity (Test F1) | TACRED (Test F1) | CoNLL-2003 (Test F1) | ReCoRD (Dev EM) | ReCoRD (Dev F1) | SQuAD (Dev EM) | SQuAD (Dev F1) |
|---|---|---|---|---|---|---|---|
| Original Attention | 77.9 | 72.2 | 94.1 | 90.1 | 90.7 | 89.2 | 94.7 |
| Entity-aware Attention | **78.2** | **72.7** | **94.3** | **90.8** | **91.4** | **89.8** | **95.0** |

Table 7: Ablation study of our entity-aware self-attention mechanism.

| Name | CoNLL-2003 (Test F1) | SQuAD (Dev EM) | SQuAD (Dev F1) |
|---|---|---|---|
| RoBERTa w/ extra training | 92.5 | 89.1 | 94.7 |
| RoBERTa | 92.4 | 88.9 | 94.6 |
| LUKE | **94.3** | **89.8** | **95.0** |

Table 8: Results of RoBERTa additionally trained using our Wikipedia corpus.

## References


Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven Pretraining of Self-attention Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5360–5369.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the Blanks: Distributional Similarity for Relation Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*, pages 2787–2795.

Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juanzi Li. 2017. Bridge Text and Knowledge by Learning Multi-Prototype Entity Mention Embedding. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1623–1633.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-Fine Entity Typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 87–96.

Christopher Clark and Matt Gardner. 2018. Simple and Effective Multi-Paragraph Reading Comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep Joint Entity Disambiguation with Local Neural Attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415v3*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer Normalization. *arXiv preprint arXiv:1607.06450v1*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Xiepeng Li, Zhexi Zhang, Wei Zhu, Zheng Li, Yuan Ni, Peng Gao, Junchi Yan, and Guotong Xie. 2019. Pingan Smart Health and SJTU at COIN - Shared Task: utilizing Pre-trained Language Models and Common-sense Knowledge in Machine Reading Tasks. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 93–98.

Jeffrey Ling, Nicholas FitzGerald, Zifei Shan, Livio Baldini Soares, Thibault Févry, David Weiss, and Tom Kwiatkowski. 2020. Learning Cross-Context Entity Representations from Text. *arXiv preprint arXiv:2001.03765v1*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692v1*.

Simon Ostermann, Sheng Zhang, Michael Roth, and Peter Clark. 2019. Commonsense Inference in Natural Language Processing (COIN) - Shared Task Report. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 66–74.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge Enhanced Contextual Word Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 43–54.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and Measuring the Geometry of BERT. In *Advances in Neural Information Processing Systems 32*, pages 8594–8603.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2017. Bidirectional Attention Flow for Machine Comprehension. In *International Conference on Learning Representations*.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep Exhaustive Model for Nested Named Entity Recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 2071–2080.

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 *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *Advances in Neural Information Processing Systems 32*, pages 3266–3280.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Cuihong Cao, Daxin Jiang, Ming Zhou, and others. 2020. K-Adapter: Infusing Knowledge into Pre-trained Models with Adapters. *arXiv preprint arXiv:2002.01808v3*.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2019b. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *arXiv preprint arXiv:1911.06136v1*.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2020. Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model. In *International Conference on Learning Representations*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning Distributed Representations of Texts and Entities from Knowledge Base. *Transactions of the Association for Computational Linguistics*, 5:397–411.

Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the International Conference on Learning Representations*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237v1*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018a. ReCoRD: Bridging the Gap between Human and Machine Commonsense Reading Comprehension. *arXiv preprint arXiv:1810.12885v1*.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018b. Graph Convolution over Pruned Dependency Trees Improves Relation Extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware Attention and Supervised Data Improve Slot Filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

## Appendix for "LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention"

## A Details of Pretraining

As input corpus for pretraining, we use the December 2018 version of Wikipedia, comprising approximately 3.5 billion words and 11 million entity annotations. We generate input sequences by splitting the content of each page into sequences comprising $\leq 512$ words and their entity annotations (i.e., hyperlinks). We optimize the model using AdamW with learning rate warmup and linear decay of the learning rate. To stabilize training, we update only those parameters that are randomly initialized (i.e., fix the parameters that are initialized using RoBERTa) in the first 100K steps, and update all parameters in the remaining 100K steps. We run the pretraining on NVIDIA's PyTorch Docker container 19.02 hosted on a server with two Intel Xeon Platinum 8168 CPUs and 16 NVIDIA Tesla V100 GPUs. The training takes approximately 30 days. The detailed hyper-parameters are shown in Table 9.

| Name | Value |
|---|---|
| Maximum word length | 512 |
| Batch size | 2048 |
| Peak learning rate | 1e-5 |
| Peak learning rate (first 100K steps) | 5e-4 |
| Learning rate decay | linear |
| Warmup steps | 2500 |
| Mask probability for words | 15% |
| Mask probability for entities | 15% |
| Dropout | 0.1 |
| Weight decay | 0.01 |
| Gradient clipping | none |
| Adam $\beta_1$ | 0.9 |
| Adam $\beta_2$ | 0.999 |
| Adam $\epsilon$ | 1e-6 |

Table 9: Hyper-parameters used to pretrain LUKE.

| Name | Value |
|---|---|
| Maximum word length | 512 |
| Batch size | 2048 |
| Peak learning rate | 1e-5 |
| Learning rate decay | linear |
| Warmup steps | 2500 |
| Mask probability for words | 15% |
| Dropout | 0.1 |
| Weight decay | 0.01 |
| Gradient clipping | none |
| Adam $\beta_1$ | 0.9 |
| Adam $\beta_2$ | 0.999 |
| Adam $\epsilon$ | 1e-6 |

Table 10: Hyper-parameters used for the extra pretraining of RoBERTa on our Wikipedia corpus.

Table 10 shows the hyper-parameters used for the extra pretraining of RoBERTa on our Wikipedia corpus described in Section 5. As shown in the Table, we use the same hyper-parameters as the ones used to train LUKE. We train the model for 200K steps and update all parameters throughout the training.

## B Details of Experiments

We conduct the experiments using NVIDIA's PyTorch Docker container 19.02 hosted on a server with two Intel Xeon E5-2698 v4 CPUs and eight V100 GPUs. For each dataset, excluding SQuAD, we conduct hyper-parameter tuning using grid search based on the performance on the development set. We evaluate performance using EM on the ReCoRD dataset, and F1 on the other datasets. Because our computational resources are limited, we use the following constrained search space:

- learning rate: 1e-5, 2e-5, 3e-5

| Name | Open Entity | TACRED | CoNLL-2003 | ReCoRD | SQuAD |
|---|---|---|---|---|---|
| Learning rate | 1e-5 | 1e-5 | 1e-5 | 1e-5 | 15e-6 |
| Batch size | 4 | 32 | 8 | 32 | 48 |
| Training epochs | 3 | 5 | 5 | 2 | 2 |
| Training time | 10min | 190min | 203min | 92min | 42min |
| Number of GPUs | 1 | 1 | 1 | 8 | 8 |
| Dev score | 78.5 F1 | 72.0 F1 | 97.1 F1 | 90.8 EM/91.4 F1 | 89.8 EM/95.0 F1 |

Table 11: Hyper-parameters and other details of our experiments.

| Name | Value |
|---|---|
| Maximum word length | 512 |
| Learning rate decay | linear |
| Warmup ratio | 0.06 |
| Dropout | 0.1 |
| Weight decay | 0.01 |
| Gradient clipping | none |
| Adam $\beta_1$ | 0.9 |
| Adam $\beta_2$ | 0.98 |
| Adam $\epsilon$ | 1e-6 |

Table 12: Common hyper-parameters used in our experiments.

- batch size: 4, 8, 16, 32, 64
- number of training epochs: 2, 3, 5

We do not tune the hyper-parameters of the SQuAD dataset, and use the ones described in Liu et al. (2020). The hyper-parameters and other details, including the training time, number of GPUs used, and the best score on the development set, are shown in Table 11. For the other hyper-parameters, we simply follow Liu et al. (2020) (see Table 12). We optimize the model using AdamW with learning rate warmup and linear decay of the learning rate. We also use early stopping based on performance on the development set. The details of the datasets used in our experiments are provided below.

### B.1 Open Entity

The Open Entity dataset used in Zhang et al. (2019) consists of training, development, and test sets, where each set contains 1,998 examples with labels of nine general entity types. The dataset is downloaded from the website for Zhang et al. (2019).[2] We compute the reported results using our code based on that of Zhang et al. (2019).

### B.2 TACRED

The TACRED dataset contains 68,124 training examples, 22,631 development examples, and 15,509 test examples with labels of their relation types. The total number of relation types is 42. The dataset is obtained from the LDC website.[3] We compute the reported results using our code based on that of Zhang et al. (2019).

### B.3 CoNLL-2003

The CoNLL-2003 dataset comprises training, development, and test sets, containing 14,987, 3,466, and 3,684 sentences, respectively. Each sentence contains annotations of four entity types, namely *person*, *location*, *organization*, and *miscellaneous*. The dataset is downloaded from the relevant website.[4] The reported results are computed using the `conlleval` script obtained from the website.

### B.4 ReCoRD

The ReCoRD dataset consists of 100,730 training, 10,000 development, and 10,000 test questions created based on 80,121 unique news articles. The dataset is obtained from the relevant website.[5] We compute the performance on the development set using the official evaluation script downloaded from the website. Performance on the test set is obtained by submitting our model to the leaderboard.

### B.5 SQuAD 1.1

The SQuAD 1.1 dataset contains 87,599 training, 10,570 development, and 9,533 test questions created based on 536 Wikipedia articles. The dataset is downloaded from the relevant website.[6] We compute performance on the development set using the official evaluation script downloaded from the website. Performance on the test set is obtained by submitting our model to the leaderboard.

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

[3] https://catalog.ldc.upenn.edu/LDC2018T24

[4] https://www.clips.uantwerpen.be/conll2003/ner

[5] https://sheng-z.github.io/ReCoRD-explorer

[6] https://rajpurkar.github.io/SQuAD-explorer

## C Adding Entity Annotations to SQuAD dataset

For each question–passage pair in the SQuAD dataset, we first create a mapping from the entity names (e.g., "U.S.") to their referent Wikipedia entities (e.g., *United States*) using the entity hyperlinks on the source Wikipedia page of the passage. We then perform simple string matching to extract all entity names in the question and the passage, and treat all matched entity names as entity annotations for their referent entities. We ignore an entity name if the name refers to multiple entities on the page. Further, to reduce noise, we also exclude an entity name if its *link probability*, the probability that the name appears as a hyperlink in Wikipedia, is lower than 1%. We use the March 2016 version of Wikipedia to collect the entity hyperlinks and the link probabilities of the entity names.