组会

曾双

2020.10.10

EMNLP 2020

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

Motivation

- 传统的实体表示（如TransE）将KB中的实体使用一个向量进行表示
  - 需要通过EL技术将文本中的实体链接到KB中的实体
  - 不能表示KB中不存在的实体
  - 实体的表示没有感知上下文

Motivation

- 基于transformer的预训练模型（BERT，RoBERTa，SpanBERT，MTB）得到的词的上下文表示(contextualized word representations ，CWR）可以缓解以上问题，但他们不适合表示实体，原因如下：
  - 他们预训练阶段输出的不是span-level的实体表示（因为他们是word-based的预训练），只能通过下游数据和任务来得到实体表示
  - “Ring” in “…. The Lord of the [MASK] ….”
  - 许多entity-related任务（relation classification， QA）需要实体间的关系推理

Method

- In this paper, we propose new pretrained contextualized representations of words and entities by developing LUKE (Language Understanding with Knowledge-based Embeddings).
- 与前人的主要不同在于：
  - 将entity看做一个token，预训练任务加入实体的预测，这样，基于transformer架构可以直接建模实体间的关系

Architecture

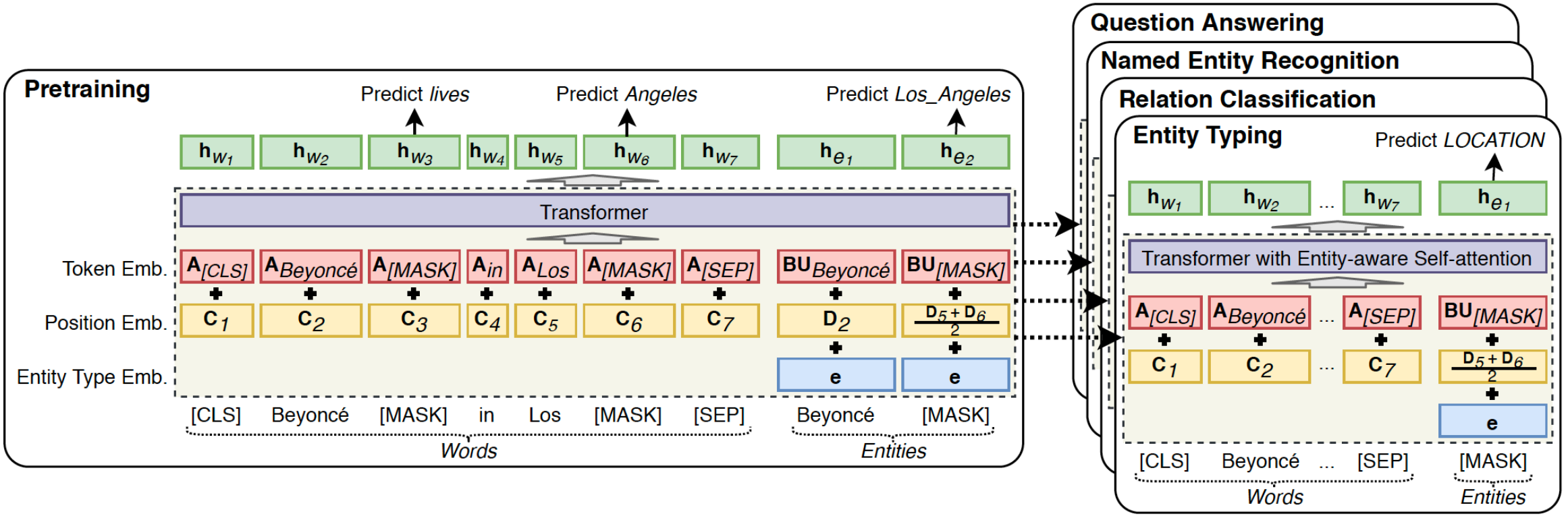

Figure 1: Architecture of LUKE using the input sentence "*Beyoncé lives in Los Angeles.*" LUKE outputs contextualized representation for each word and entity in the text. The model is trained to predict randomly masked words (e.g., *lives* and *Angeles* in the figure) and entities (e.g., *Los_Angeles*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

Method

- 预训练阶段，15%的概率随机mask掉句子中的entity，然后训练模型去预测mask掉的实体
- 在下游任务时，使用mask作为实体的输入，输出得到句子中某个实体的表示

- 在计算self-attention系数的时候，使用不同的Q矩阵来考虑query和key的token类别，以捕获word与entity、entity与entity的关系

Data Source

Donald Trump

From Wikipedia, the free encyclopedia

For other uses, see Donald Trump (disambiguation).

Donald John Trump (born June 14, 1946) is the 45th and current president of the United States. Before entering politics, he was a businessman and television personality.

Born and raised in Queens, New York City, Trump attended Fordham University for two years and received a bachelor's degree in economics from the Wharton School of the University of Pennsylvania. He became president of his father's real estate business in 1971, renamed it The Trump Organization, and expanded its operations to building or renovating skyscrapers, hotels, casinos, and golf courses. Trump later started various side ventures, mostly by licensing his name. Trump and his businesses have been involved in more than 4,000 state and federal legal actions, including six bankruptcies. He owned the Miss Universe brand of beauty pageants from 1996 to 2015, and produced and hosted the reality television series *The Apprentice* from 2003 to 2015.

Trump's political positions have been described as populist, protectionist, isolationist, and nationalist. He entered the 2016 presidential race as a Republican and was elected in a surprise electoral college victory over Democratic nominee Hillary Clinton while losing the popular vote.[a] He became the oldest first-term U.S. president[b] and the first without prior military or government service. His election and policies have sparked numerous protests. Trump has made many false or misleading statements during his campaign and presidency. The statements have been documented by fact-checkers, and the media have widely described the phenomenon as unprecedented in American politics. Many of his comments and actions have been characterized as racially charged or racist.

During his presidency, Trump ordered a travel ban on citizens from several Muslim-majority countries, citing security concerns; after legal challenges, the Supreme Court upheld the policy's third revision. He enacted a tax-cut package for individuals and businesses, rescinding the individual health insurance mandate penalty of the Affordable Care Act, but has failed to repeal and replace the ACA as a whole. He appointed Neil Gorsuch and Brett Kavanaugh to the Supreme Court. In foreign policy, Trump has pursued an America First agenda, withdrawing the U.S.

Donald Trump

45th President of the United States

Incumbent

Assumed office
January 20, 2017

- **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 $\mathbf{e} \in \mathbb{R}^D$.

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:

$$\mathbf{y}_i = \sum_{j=1}^{k} \alpha_{ij} \mathbf{V} \mathbf{x}_j$$

$$e_{ij} = \frac{\mathbf{K}\mathbf{x}_j^\top \mathbf{Q}\mathbf{x}_i}{\sqrt{L}}$$

$$\alpha_{ij} = \mathrm{softmax}(e_{ij})$$

预训练的时候只用Q，下游任务时才对其余Q进行参数更新！
下游任务时，其余Q初始化为预训练得到的Q

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.

Entity Typing数据集Open Entity，给定一个句子，识别句子中实体的类型。有9种类型的实体。micro F1

输入是一个句子所有词和句子中的一个实体的[MASK]，通过positional embedding来区分是哪个实体

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.

句子级RE数据集TACRED，给定一个句子，识别句子中不同实体的关系。有42种类型的关系。micro F1

输入是一个句子所有词和句子中的头尾实体的[HEAD]和[TAIL]，拼接头尾实体的输出表示进行多分类

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.

句子级NER数据集CoNLL-2003，给定一个句子，识别句子的实体及其类别。span-level F1

输入是一个句子所有词和句子某个枚举的span，拼接头尾词和实体的输出表示进行多分类

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.

完形填空风格的QA数据集ReCoRD (2008)，EM & token-level F1

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.

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.

抽取式QA数据集SQuAD 1.1，给定一个问题和段落，识别段落中的span作为答案。
EM & token-level F1

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.

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.

感想

- Entity-related的任务可以使用LUKE做初始化的表示，由于是基于RoBERTa-Large的参数进行训练的，共6亿参数，估计只能拿来做特征抽取不能做fine-tuning

- 目前word-based或entity-based的MLM mask策略，是随机mask的，能否改成定向的mask，只mask那些信息量比较大的word或者entity， 比如WordNet中邻居很丰富的词或者KG中邻居很丰富的实体，相当于在预训练阶段就隐式地去注入先验知识，这样迁移到下游任务的知识更好

Thanks!