Interpretable Entity Representations through Large-Scale Typing

Yasumasa Onoe and **Greg Durrett**
Department of Computer Science
The University of Texas at Austin
{yasumasa, gdurrett}@cs.utexas.edu

Abstract

In standard methodology for natural language processing, entities in text are typically embedded in dense vector spaces with pre-trained models. The embeddings produced this way are effective when fed into downstream models, but they require end-task fine-tuning and are fundamentally difficult to interpret. In this paper, we present an approach to creating entity representations that are human readable and achieve high performance on entity-related tasks out of the box. Our representations are vectors whose values correspond to posterior probabilities over fine-grained entity types, indicating the confidence of a typing model's decision that the entity belongs to the corresponding type. We obtain these representations using a fine-grained entity typing model, trained either on supervised ultra-fine entity typing data (Choi et al., 2018) or distantly-supervised examples from Wikipedia. On entity probing tasks involving recognizing entity identity, our embeddings used in parameter-free downstream models achieve competitive performance with ELMo- and BERT-based embeddings in trained models. We also show that it is possible to reduce the size of our type set in a learning-based way for particular domains. Finally, we show that these embeddings can be post-hoc modified through a small number of rules to incorporate domain knowledge and improve performance.

1 Introduction

In typical neural NLP systems, entities are embedded in the same space as other words either in context-independent (Mikolov et al., 2013; Pennington et al., 2014) or in context-dependent ways (Peters et al., 2018; Devlin et al., 2019). Such approaches are powerful: pre-trained language models implicitly learn factual knowledge about those entities (Petroni et al., 2019; Roberts et al., 2020; Jiang et al., 2020) and these representations can be grounded in structured and human-curated knowledge bases (Logan et al., 2019; Levine et al., 2019; Peters et al., 2019; Zhang et al., 2019; Poerner et al., 2019; Xiong et al., 2020; Wang et al., 2020). However, these embeddings do not *explicitly* maintain representations of this knowledge, and dense entity representations are not directly interpretable. Knowledge probing tasks can be used to measure LMs' factual knowledge (Petroni et al., 2019), but designing the right probing task is another hard problem (Chen et al., 2019; Poerner et al., 2019), particularly if the probes are parameter-rich (Hewitt and Manning, 2019).

In this work, we explore a set of interpretable entity representations that are simultaneously human and machine readable. The key idea of this approach is to use fine-grained entity typing models with large type inventories (Ling and Weld, 2012; Gillick et al., 2014; Choi et al., 2018; Onoe and Durrett, 2020). Given an entity mention and context words, our typing model outputs a high-dimensional vector whose values are associated with predefined fine-grained entity types. Each value ranges between 0 and 1, corresponding to the confidence of the model's decision that the entity has the property given by the corresponding type. We use pre-trained Transformer-based entity typing models, trained either on a supervised entity typing dataset (Choi et al., 2018) or on a distantly-supervised dataset derived from Wikipedia categories (Onoe and Durrett, 2020). The type vectors from these models, which contain tens of thousands of types, are then used as contextualized entity embeddings in downstream tasks.

Past work has shown that such type-driven representations are useful for entity linking (Onoe and Durrett, 2020); we improve the quality of these representations, broaden the scope of where they can be applied, and show techniques to extend and debug them by exploiting their interpretable na-

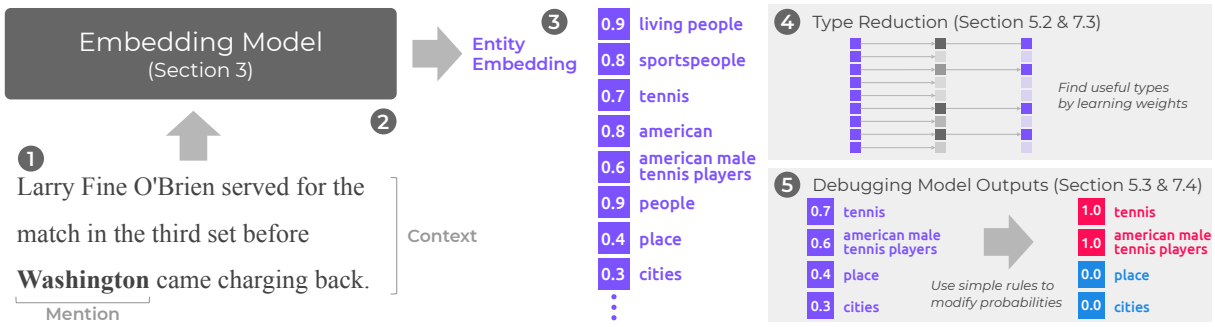

Figure 1: Interpretable entity representations. (1) A mention and its context are fed into (2) an embedding model. (3) An entity embedding vector consists of probabilities for corresponding types. (4) We can reduce the size of the type set for a particular downstream task in a learning-based way (Section 5.2). (5) We can also incorporate domain knowledge via rules to modify bad type probabilities and improve model performance (Section 5.3).

ture. Dense representations of entities have similarly been applied to entity linking (Yamada et al., 2016; Eshel et al., 2017), as well as relation extraction (Baldini Soares et al., 2019), entity typing (Ling et al., 2020), and question answering (Févry et al., 2020). Those approaches use millions of predefined entities, while our approach uses a much smaller number of types (10k or 60k). This makes it simultaneously more compact and also more flexible when generalizing to unknown entities.

We evaluate our embedding approach on benchmark tasks for entity representations. We use coreference arc prediction (CAP) and named entity disambiguation on CoNLL-YAGO, two tasks in the EntEval suite (Chen et al., 2019), as well as entity linking on WikilinksNED (Eshel et al., 2017), which covers broader entities and writing styles. We compare our approach against entity representations produced directly by pre-trained models. Our "out-of-the-box" entity representations combined with simple heuristics like dot product or cosine similarity can achieve competitive results on CAP without additional trainable parameters. On NED tasks, our approach outperforms all baselines by a substantial margin. We show that a much smaller type set can be distilled in a per-task fashion to yield similar performance. Finally, we show a proof-of-concept for how we can leverage the interpretability of our embeddings to "debug" downstream errors, a challenge for black-box models.

2 Interpretable Entity Representations

Our approach for producing entity representations is shown in Figure 1. For an entity mention in context,[1] we compute a vector of probabilities, each of which reflects (independently) the probability of an entity exhibiting a particular type. Types are predefined concepts that could be derived from existing knowledge bases. We hypothesize that real world entities can be represented as a combination of those concepts if we have a large and varied enough concept inventory. This representation can be used as a dense vector since the values are still continuous numbers (though restricted between 0 and 1). It is interpretable like a discrete feature vector, since each dimension has been named with the corresponding entity type.

We define $s = (w_1, ..., w_N)$ to denote a sequence of context words, and $m = (w_i, ..., w_j)$ to denote an entity mention span in s. The input word sequence s could be naturally co-occurring *context* words for the mention, or *descriptive* words such as might be found in a definition. The output variable is a vector $\mathbf{t} \in [0, 1]^{|\mathcal{T}|}$ whose values are probabilities corresponding to fine-grained entity types $\mathcal{T}$. These entity types are predefined and static, so their meanings are identical for all entities. Our goal here is to learn parameters θ of a function f_θ that maps the mention m and its context s to a vector $\mathbf{t}$, which capture salient features of the entity mention with the context.

We learn the parameters θ in a supervised manner. We use a labeled dataset $\mathcal{D} = \{(m, s, \mathbf{t}^*)^{(1)}, ..., (m, s, \mathbf{t}^*)^{(k)}\}$ to train an entity typing model. The gold labels $\mathbf{t}^*$ are obtained by manual annotation or distant-supervision techniques (Craven and Kumlien, 1999; Mintz et al., 2009). We select a predefined types $\mathcal{T}$ from modified Wikipedia categories, or we use an existing type set such as UFET (Choi et al., 2018) (discussed in Section 4).

We use the output vectors $\mathbf{t}$ as general purpose

[1] Our approach can also embed knowledge base entities, as discussed in Section 4.

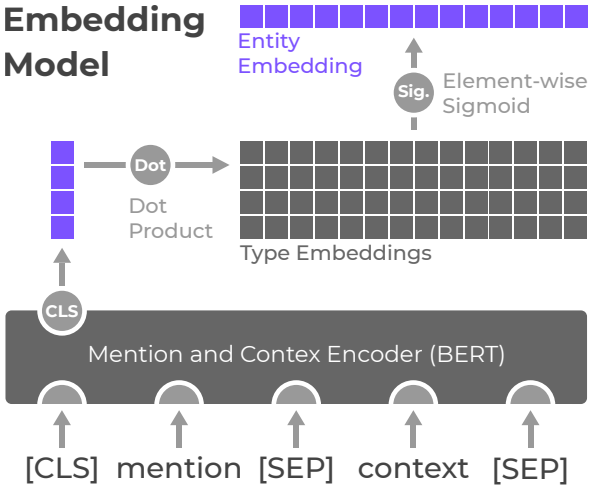

Figure 2: Embedding model architecture. We use BERT to embed the mention and context, then multiply by an output matrix and apply an elementwise sigmoid to compute posterior probabilities for each type.

entity representations in downstream tasks, combined with off-the-shelf similarity measures like dot product and cosine similarity. These representations can also be customized in a per-task way. The number of entity types can be reduced by 90% while maintaining similar performance (top right of Figure 1), as discussed in Section 5.2.

Another advantage of our interpretable embeddings is that they give us a hook to "debug" our downstream models. Debugging black-box models built on embeddings is typically challenging, but since our entity representations are directly interpretable, we can modify the output vectors $\mathbf{t}$ using our prior knowledge about entities (bottom right of Figure 1). For example, we might know that in the financial domain *Wall Street* does not refer to a location. We show that simple rules based on prior knowledge can improve performance further (discussed in Section 5.3); critically, this is done without having to annotate data in the target domain, giving system designers another technique for adapting these models.

3 Embedding Model

Our model f_θ to produce these embeddings is shown in Figure 2: it takes as input the mention m and its context s and predicts probabilities for predefined entity types $\mathcal{T}$. This is a Transformer-based typing model following the BERT model presented in Onoe and Durrett (2019). First, a Transformer-based encoder (Vaswani et al., 2017) maps the input variables, m and s, to an intermediate vector representation. A type embedding layer then projects the intermediate representation to a vector whose dimensions correspond to the entity types $\mathcal{T}$. Finally, we apply a sigmoid function on each real-valued score in the vector to obtain the posterior probabilities that form our entity representation $\mathbf{t}$ (top of the figure).

Mention and Context Encoder We use pretrained BERT[2] (Devlin et al., 2019) for the mention and context encoder. This BERT-based encoder accepts as input a token sequence formatted as $\mathbf{x} =$ `[CLS]` m `[SEP]` s `[SEP]`, where the mention m and context s are chunked into WordPiece tokens (Wu et al., 2016). We encode the whole sequence using BERT and use the hidden vector at the `[CLS]` token as the mention and context representation: $\mathbf{h}_{\texttt{[CLS]}} = \textsc{BertEncoder}(\mathbf{x})$.

Type Embeddings This output layer is a single linear layer whose parameter matrix can be viewed as a matrix of type embeddings $\mathbf{E} \in \mathbb{R}^{|\mathcal{T}| \times d}$, where d is the dimension of the mention and context representation $\mathbf{h}_{\texttt{[CLS]}}$. We obtain the output probabilities $\mathbf{t}$ by multiplying $\mathbf{E}$ by $\mathbf{h}_{\texttt{[CLS]}}$, followed by an element-wise sigmoid function: $\mathbf{t} = \sigma\left(\mathbf{E} \cdot \mathbf{h}_{\texttt{[CLS]}}\right)$.[3] Similar to previous work (Choi et al., 2018; Onoe and Durrett, 2019), we assume independence between all entity type in $\mathcal{T}$.

One assumption in our approach is that the model's output probabilities are a meaningful measure of class membership. Past work (Desai and Durrett, 2020) has observed that this is true for other models involving BERT variants.

Training Following Choi et al. (2018), the loss is a sum of binary cross-entropy losses over all entity types $\mathcal{T}$ over all training examples $\mathcal{D}$. That is, we treat each type prediction for each example as an independent binary decision, with shared parameters in the BERT encoder.

4 Training Data

To train our entity typing model, we need labeled examples consisting of $(m, s, \mathbf{t}^*)$ triples. Although there are labeled typing datasets such as UFET

[2]We use BERT-large uncased (whole word masking) in our experiments. We experimented with RoBERTa (Liu et al., 2019) but found it to work less well.

[3]Note that this makes our entities occupy a d-dimensional subspace in the type representation logit space (pre-sigmoid). A different model could be used to combat this low-rankness. Regardless, the explicit type space has advantages in terms of out-of-the-box functionality as well as interpretability.

(Choi et al., 2018), getting large amounts of manually labeled data is expensive. Moreover, the UFET dataset contains instances of entities in context, so it is suitable for training models for *contextual embeddings*, but it doesn't have examples of definitions for *descriptive embeddings* (following the terminology of Chen et al. (2019)).

Therefore, we additionally use two distantly labeled entity typing datasets derived from Wikipedia. We leverage past work in using types derived from Wikipedia categories (Suchanek et al., 2007; Onoe and Durrett, 2020, inter alia), which contain type information and are widely annotated across Wikipedia articles. We select the appropriate dataset for each setting depending on task-specific requirements (see Section 6). For all datasets, we compute entity typing macro F1 using development examples (1k) to check model convergence.

Wiki-Context We collect a set of occurrences of typed entity mentions using hyperlinks in Wikipedia. Given a sentence with a hyperlink, we use the hyperlink as an entity mention m, the sentence as a context sentence s, and the Wiki categories of the destination page as the gold entity types $\mathbf{t}^*$. We use the preprocessing of Onoe and Durrett (2020) to modify the type set: they introduce more general categories into the Wikipedia category set by splitting existing complex categories. Following their work, we filter the resulting set to keep the 60,000 most frequent types. Scraping Wikipedia yields 6M training examples that cover a wide range of entities and entity types.

Wiki-Description Following a similar paradigm as for Wiki-Context, we create description-focused training examples from Wikipedia.[4] We use the same entity type set as the Wiki-Context dataset. We collect lead paragraphs from all Wikipedia pages and filter to keep examples that contain at least 1 entity type in the 60k entity types. We use the Wikipedia page title (usually boldfaced) in the lead paragraph as the entity mention m, and retain at most 100 words on either side to form the context s. The Wiki categories of the same page would be the gold entity types $\mathbf{t}^*$. We obtain 2M training examples after filtering. The size of entity type set is 60k.

[4]For tasks like entity linking, we could in principle just use gold type vectors for each entity, as in Onoe and Durrett (2020). However, the paradigm here matches that of Chen et al. (2019), and the descriptive entity embedding model we train can generalize to unseen descriptions at test time.

UFET This ultra-fine entity typing dataset is created by Choi et al. (2018). This dataset consists of 6k manually annotated examples. The entity mention spans could be named entities, nominal expressions, and pronouns while Wiki-based datasets mostly provide named entity mention spans. We use 5.5k examples for training and 500 examples for validation. Note that because our goal in this work is downstream task performance, we deviate from the standard train/dev/test splits of 2k/2k/2k in favor of higher performance.

5 Tailoring to a Task

Our interpretable entity embeddings are designed for general-purpose uses and intended to work "out-of-the-box". We first discuss two scenarios (tasks) and then describe two ways we can customize these representations for a downstream task: reducing the size of types and debugging model output using prior knowledge.

5.1 Tasks

Coreference Arc Prediction (CAP) This task focuses on resolving local coreference arcs. For each instance, two entity mention spans and their context are provided. The task is to predict if those two mention spans are coreferent or not, so this is a binary classification problem.[5]

Named Entity Disambiguation (NED) NED is the task of connecting entity mentions in text with real world entities in a knowledge base, including disambiguating between sometimes highly related candidates (e.g., the same movie produced in different years). We use the local resolution setting where each instance features a single entity mention span in the input text and several possible candidates. We consider the setting where descriptions for candidates entities are available (e.g., the first sentence of the Wikipedia article).

5.2 Type Reduction

The type sets we consider in this work are very large, consisting of 10k or 60k types. Although larger type sets provides more precise entity representations, these may have redundant types or types which are unimportant for a particular domain. For both statistical and computational efficiency, we

[5]The mentions in this case are always drawn from the same or adjacent sentences, so constraints from saliency that would need to be incorporated in a full coreference system are less relevant in this setting.

would like to compute the types useful for a downstream task in a data-driven way.

For all tasks we consider in this work, our model will depend chiefly on a function $\text{sim}(\mathbf{t}_1, \mathbf{t}_2)$ for two different type vectors. These type vectors are computed from mention and context pairs using the trained entity typing model $\mathbf{t} = f_\theta(m, s)$. In experiments, we will use either dot product or cosine similarity as our similarity function.

Our approach to compression involves learning a sparse trainable mask that restricts the set of types considered. We parameterize the dot product[6] and cosine similarity[7] operations with a weight matrix $\mathbf{W}$, a diagonal matrix $\text{diag}(w_1, w_2, ..., w_{|\mathcal{T}|})$ whose components correspond to the entity types in $\mathcal{T}$. The parameters $\mathbf{W}$ can be learned directly on downstream tasks (e.g., CAP and NED). Note that in the cosine scoring function, we clip these parameter values to be between 0 and 1. We train with the standard downstream task objective, but with an additional L_1 regularization term applied to $\mathbf{W}$ (Tibshirani, 1994) to encourage the $\mathbf{W}$ values to be sparse.

This approach naturally leads to around $20-35\%$ sparsity in the vector $\text{diag}(w_1, w_2, ..., w_{|\mathcal{T}|})$ with settings of the regularization parameter we found effective. In practice, to achieve a higher level of sparsity, we further reduce the entity type set based on the magnitude of $\mathbf{W}$ (e.g., keep the 10% of types with the highest values). Finally, we use the reduced entity types for further experiments on the target task.

5.3 Debuggability

Our interpretable entity representations allow us to more easily understand when our models for downstream tasks make incorrect predictions, typically by mischaracterizing an ambiguous entity in context by assigning incorrect probabilities to the entity types. As an example from the CoNLL-YAGO NED dataset, we observe that our model gets confused if the mention span *Spain* should refer to *Women's national tennis team* or *Men's national tennis team*. If we are trying to adapt to this scenario without using in-domain annotated data, a domain expert may nevertheless be able to articulate a rule to fix this error. Such a rule might be: whenever *Fed Cup* (the international team competition in women's tennis) appears in the context, we assign 1 to a collection of relevant entity types such as `women's` and 0 to irrelevant types such as `davis cup teams` (the international team competition in men's tennis). Critically, because our representations have interpretable axes, we can more easily transform our entity representations and incorporate this kind of domain knowledge.

[6] $\text{sim}_{\text{dot}}(\mathbf{t}_1, \mathbf{t}_2) = \mathbf{t}_1{}^\top \mathbf{W}\, \mathbf{t}_2$

[7] $\text{sim}_{\text{cos}}(\mathbf{t}_1, \mathbf{t}_2) = \frac{\mathbf{t}_1^\top \mathbf{W}\, \mathbf{t}_2}{\sqrt{\mathbf{t}_1^\top \mathbf{W}\, \mathbf{t}_1}\sqrt{\mathbf{t}_2^\top \mathbf{W}\, \mathbf{t}_2}}$

6 Experimental Setup

We evaluate the "out-of-the-box" quality of our entity representations and baselines on two entity probing tasks as discussed in the previous section.

6.1 Datasets

Coreference Arc Prediction (CAP) We use the English CAP dataset derived from PreCo (Chen et al., 2018) by Chen et al. (2019). The creators of the dataset partition the data by cosine similarity of GloVe (Pennington et al., 2014) embeddings of mention spans and balance the number of positive and negative examples in each bucket, so that models do not solve the task by capturing surface features of entity mention spans. The original data split provides 8k examples for each of the training, development, and test sets.

Named Entity Disambiguation (NED) We use the standard English CoNLL-YAGO benchmark (Hoffart et al., 2011) preprocessed by Chen et al. (2019). For each entity mention, at most 30 candidate entities are selected using the CrossWikis dictionary (Spitkovsky and Chang, 2012). This dataset contains 18.5k training, 4.8k dev, and 4.5k test examples from newswire text, so the variety of entities and the writing styles are limited. For this reason, we create another NED dataset from WikilinksNED (Eshel et al., 2017), which includes a wide range of entities and diverse writing styles from scraped English web text linking to Wikipedia. We limit the number of candidate entities to 3 for each instance, which still makes a challenging benchmark. We create 5k training, 1k dev, and 1k test examples and call this dataset WLNED. In both CoNLL-YAGO and WLNED, we form descriptions of candidate entities using the Wiki-Context data, but do not use any structural information from Wikipedia (hyperlinks, etc.).

6.2 Baselines

Figure 3 schematically shows the use of our model compared to baselines, which we now describe.

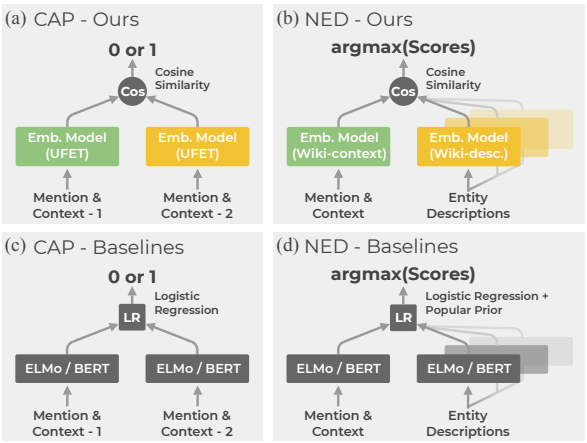

Figure 3: Overview of our downstream architectures. Our method simply computes cosine similarity and uses it as a score for each task, not introducing any new parameters. Our baselines use a trainable logistic regression layer over pre-trained embeddings to make classification decisions.

Entity Embeddings We create entity representations of a mention span m and a context s using ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). We largely follow the embedding procedure of Chen et al. (2019). Their downstream models use trainable weights to combine the vectors from the pre-trained model layers; we use this in the baselines as well except for the results in Table 4 and for ELMo. Note that we do not fine tune the ELMo and BERT parameters of the baselines, since our focus is on *general* entity representations that can work off-the-shelf rather than task specific entity representations. Our approach does not use task specific fine tuning either.

ELMo We run ELMo on the entire sentence s and combine the three layer outputs using uniform weights. Then, we average contextualized vectors of the mention span m to obtain the entity representation.

BERT Base We concatenate an entity mention m and its context s and feed it into BERT-base. We compute the weighted sum of the `[CLS]` vectors[8] from all 13 layers and use it as an entity representation.

BERT Large Similar to the BERT-base baseline, we feed an entity mention m and its context s into BERT-large and average the `[CLS]` vectors from all 25 layers.

KnowBert-W+W KnowBert is built on top of BERT-base by adding an internal entity linker. We use KnowBert-W+W, which has been trained on Wikipedia and WordNet (Fellbaum, 1998) as an *embedding model* that incorporates external information; note that we are *not* using this as an entity linking system, even for NED. Similar to other BERT baselines, we feed a mention span m and context s, and we use the weighted sum of the `[CLS]` vectors from all 15 layers.

[8]We tried pooling span representations like for ELMo and saw similar results.

Model	Test Acc.
GloVe (Chen et al., 2019)	71.9
ELMo (Chen et al., 2019)	80.2
BERT Base → LR (Chen et al., 2019)	80.6
BERT Large → LR (Chen et al., 2019)	79.1
KnowBert embeddings → LR	**81.5**
EntEmbeddings → Cosine	80.2

Table 1: Accuracy on the CAP test set. All baselines use logistic regression (LR) trained on the CAP training set. Ours predicts based on cosine similarity (no additional training required).

Classification Layer for Baselines Following Chen et al. (2019), we train a simple classifier to make final predictions. Our feature vector of two entity representations x_1 and x_2 is a concatenation of x_1, x_2, element-wise product, and absolute difference: $[x_1, x_2, x_1 \odot x_2, |x_1 - x_2|]$. These are depicted in Figure 3 as "LR" blocks.

This classifier is used for baselines only. Our approach only uses dot product or cosine similarity and does not require additional training. Since the size of our embeddings is generally larger, we avoid using a classifier in our setting because it would introduce more parameters than the baseline models, making fair comparison difficult.

6.3 Embedding Model Hyperparameters

We use pre-trained BERT-large uncased (24-layer, 1024-hidden, 16-heads, 340M parameters, whole word masking) (Devlin et al., 2019) for our mention and context encoder. All BERT hyperparameters are unchanged. The entity embedding matrix contains 10M (UFET type set) or 60M (Wiki type set) parameters. We train our models with batch size 32 (8×4 gradient accumulation steps) using one NVIDIA V100 GPU for a week. We use the AdamW optimizer (Kingma and Ba, 2014; Loshchilov and Hutter, 2018) with learning rate 2e-5 for BERT parameters and learning rate 1e-3 for the type embedding matrix. We use HuggingFace's Transformers library (Wolf et al., 2019) to

Model	Dev Acc.
BERT BASE → Cosine	54.4
BERT LARGE → Cosine	52.7
Mention and Context Rep. → Cosine	69.0
EntEmbeddings → Cosine	80.1

Table 2: "Out-of-the-box" accuracy on the CAP development set. We compare performance of BERT-base, BERT-large, and the mention and context representation of the embedding model with ours, using just cosine similarity and no classifier.

implement our models.[9]

7 Results and Discussion

7.1 Coreference Arc Prediction (CAP)

We compute our embeddings from an entity typing model trained on the UFET dataset (Choi et al., 2018) for CAP (10k types). We choose this dataset because many of mention spans in the CAP examples are nominal expressions or pronouns, and the Wiki-Context dataset includes almost entirely mentions of proper nouns. To make a prediction if two mentions are coreferent, we compute $\text{sim}_{\cos}(\mathbf{t}_1, \mathbf{t}_2)$ over the type vectors for each mention and check if this is greater than a threshold, which we set to 0.5.

Only our baselines use the CAP training set; our model does not train on this data. We compare our approach with the baselines described above as reported in Chen et al. (2019). Note that they use two different types of entity representations: one based on entity descriptions and another based on entity names only.

Table 1 compares test accuracy on the CAP task. Although the KNOWBERT baseline achieves the highest accuracy, our entity representations reach comparable accuracy, 80.2, without training an additional classifier. This validates our hypothesis that these embeddings are useful out-of-the-box and contain as much information as BERT-based embeddings, despite the constraints imposed by their explicit, interpretable structure.

To further investigate the gains of our interpretable entity embeddings, we compare out-of-the-box performance (i.e., using cosine similarity instead of a task specific classifier) of BERT-base, BERT-large, and the hidden layer of the embedding model with ours. Table 2 shows development accuracy on CAP. BERT-base and BERT-large barely outperform random guessing (i.e., 50%); we experimented with both the [CLS] and pooling methods and found pooling to work slightly better. We also compare to the mention and context representations $\mathbf{h}_{\texttt{[CLS]}}$ of our entity typing model. This latent representation is clearly better than the BERT representations but still underperforms our EntEmbeddings by 10 points.

[9]Code and datasets used in our experiments are available at https://github.com/yasumasaonoe/InterpretableEntityRepresentation.

Model	Test Acc.
MOST FREQUENT	58.2
ELMO Description	63.4
ELMO Name	71.2
BERT BASE Description	64.7
BERT BASE Name	74.3
BERT LARGE Description	64.6
BERT LARGE Name	74.8
EntEmbeddings → Cosine	**84.8**

Table 3: Accuracy on the CoNLL-YAGO test set in the EntEval setting (Chen et al., 2019). All baselines are from Chen et al. (2019) and use logistic regression (LR) trained on the CoNLL-YAGO training set and the prior probability. Ours predicts based on cosine similarity (no additional training required).

7.2 Named Entity Disambiguation (NED)

We use the entity typing model trained on the Wiki-Context data (see Section 4) to get the mention and context representation $\mathbf{t}$. In the CoNLL-YAGO setting, similar to past work (Onoe and Durrett, 2019; Févry et al., 2020), we prepend the document title and the first sentence to the input to enrich the context information. To obtain the candidate representations $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_j, ...\}$, we use the model trained on the Wiki-Description data, which is specialized for entity descriptions (see Section 4) similar to Gillick et al. (2019). We choose Wikipedia datasets here because UFET does not support entity descriptions. We rank the candidate entities based on cosine similarity between $\mathbf{t}$ and $\mathbf{c}_j$, and the entity with the highest score is our model's prediction.

The MOST FREQUENT baseline chooses the most frequently observed entity for a given mention as a prediction, based on a prior probability p_{prior} computed from link counts on Wikipedia. All baselines except MOST FREQUENT combine the classifier output and the prior probability to make a prediction: $\arg\max_c \left[p_{\text{prior}}(c) + p_{\text{classifier}}(c)\right]$.[10]

Table 3 lists test accuracy on the CoNLL-YAGO

[10]We adapt this technique from Chen et al. (2019) to be consistent with their setting.

Model	Test Acc.
MOST FREQUENT	64.6
ELMO embeddings → LR + prior	71.6
BERT BASE embeddings → LR + prior	69.6
BERT LARGE embeddings → LR + prior	69.1
KNOWBERT embeddings → LR + prior	71.3
EntEmbeddings → Cosine	**75.6**

Table 4: Accuracy on the WLNED test set. All baselines use logistic regression (LR) trained on the WLNED training set and the prior probability. Ours predicts based on cosine similarity (no additional training required).

Task	#Types Dev Acc.	reduced change
CAP	10k ⟶ 1k 80.1 ⟶ 78.9	90% −1.2
CoNLL-YAGO	60k ⟶ 5k 85.3 ⟶ 85.0	92% −0.3

Table 5: Accuracy on the development sets before and after applying type reduction.

	Dev Acc.
EntEmbeddings → Cosine	85.3
EntEmbeddings + **Debug** → Cosine	87.0

Table 6: Accuracy on the CoNLL-YAGO development set before and after applying debugging rules.

data. Our approach outperforms all baselines, indicating that our entity representations include useful information about entities out-of-the-box. Such a performance gap is expected since our entity representations can directly encode some factual knowledge from Wikipedia. However, these results also imply that pre-trained LMs *do not* have enough factual information out-of-the-box; they may rely on in-domain fine-tuning to achieve high performance in the target domain, and often fail to generalize to new settings.

Note that while these accuracies are significantly below the supervised state-of-the-art (95%), they are competitive with the "zero-shot" entity results from recent past work (Gupta et al., 2017; Onoe and Durrett, 2020).

Table 4 shows test accuracy on the WLNED data. The general trend is similar to the CoNLL-YAGO results, and our approach outperforms all baselines. ELMO embeddings achieve the highest accuracy, closely followed by KNOWBERT embeddings.

7.3 Reducing the Number of Types

We show that our approach from Section 5.2 effectively prunes unnecessary types, and it leads to a compact task-specific entity typing model.

For the CAP dataset, we train a bilinear model with the dot scoring function and keep the top 1k types by their weights in $\mathbf{W}$ as the new type set. As can be seen in Table 5, the reduced type set only results in a reduction of 1.2% in development accuracy after removing 90% of types.

To learn the type reduction in the CoNLL-YAGO setting, we convert the CoNLL-YAGO training data to a binary classification problem for simplicity by choosing positive and random negative entities. We train a model with the cosine scoring function and keep the top 5k types by weight as described in Section 5.2. In Table 5, the reduced type set achieves the comparable development accuracy only using around 10% of the original entity types.

Combined, these results show that the computational tractability of our approach can be improved given a specific downstream task. While our large type vectors are domain-general, they can be specialized and made sparse for specific applications.

7.4 Debugging Model Outputs

We investigate if simple rules crafted using domain knowledge can further fix errors as discussed in Section 5.3. For CoNLL-YAGO, we create 11 rules and directly modify probabilities for certain types in entity representations $\mathbf{t}$. These rules are based on our observations of errors, in the same way that a user might want to inject domain knowledge while debugging their system. As described in Section 5.3, this allows us to encode specific knowledge about domain entities (e.g., the particulars of championships for men's vs. women's tennis) that is unlikely to be encoded in our pre-trained model. The full set of rules is listed in Appendix A.

Table 6 shows that by applying our 11 rules, which *only* modify our type embeddings post-hoc, the development accuracy goes up by 1.7 points. Also note that such rule-based embedding changes are not task-specific, and change the embeddings for *any* additional prediction task we wish to run on this data, whether it's entity linking, coreference, relation extraction, or more. Note that only a few tens of types are active and contribute to the final score substantially on any given example, so these can be handled with a relatively small set of rules. We believe that more generally, this could be a recipe for injecting knowledge when porting the

system to new domains, as opposed to annotating training data.

7.5 Analysis: Entity Typing Performance

One important factor for our model is the performance of the underlying entity typing model. Table 7 shows the entity typing results on the development set of Wiki-Context, Wiki-Description, and UFET. On Wiki-Context, our entity typing model achieves 82.0 F1, which is fairly high given that there are 60,000 labels. All Wiki-Description development examples are unseen during the training time; thus, F1 is lower compared to Wiki-Context. The results on UFET are not directly comparable with past work[11] since we combine parts of the dev and test set in with the training set to have more training data.

Overall, a BERT-based entity typing model can handle large number of entity types (10k or 60k) well. Some of the high performance here can be attributed to memorizing common entities in the training data. However, we argue that this memorization is not necessarily a bad thing when the embeddings still generalize to work well on less frequent entities and in scenarios like CAP.

Model	#Types	P	R	F1
WIKI-CONTEXT	60k	86.7	77.7	82.0
WIKI-DESCRIPTION	60k	77.6	71.2	74.2
UFET	10k	54.6	40.5	46.5

Table 7: Macro-averaged P/R/F1 on the development sets.

8 Related Work

Some past work learns static vectors for millions of predefined entities. Yamada et al. (2016) and Eshel et al. (2017) embed words and entities in the same continuous space particularly for NED. Ling et al. (2020) learn general purpose entity embeddings from context and entity relationships in a knowledge base while Févry et al. (2020) does not rely on that structured information about entities. Our approach only stores type embeddings which can be substantially smaller than the entity embedding matrix.

Entity typing information has been used across a range of NLP tasks, including models for entity linking and coreference (Durrett and Klein, 2014). In entity linking specifically, typing has been explored for cross-domain entity linking (Gupta et al., 2017; Onoe and Durrett, 2020). Past work by Raiman and Raiman (2018) has also explored learning a type system for this task. Our approach to learning types starts from a large set and filters it down, which is a simpler problem. A range of approaches have also considered augmenting pre-trained models with type information (Peters et al., 2019); however, in these models, the types inform dense embeddings which are still uninterpretable.

A related thrust of the literature has looked at understanding entities using interpretable embeddings based around feature norms (McRae et al., 2005); this has advantages for learning in few-shot setups (Wang et al., 2017). However, most of this past work has used embeddings that are much lower-dimensional than ours, and don't necessarily to scale to broad-domain text or all of Wikipedia.

Another line of past work tests if type information or other knowledge is captured by pre-trained LMs. Peters et al. (2018) report that ELMo performs well on word sense disambiguation and POS tagging. Some other work also investigates models' ability to induce syntactic information by measuring accuracy of a probe (Zhang and Bowman, 2018; Hewitt and Manning, 2019; Hewitt and Liang, 2019). However, there is significant uncertainty about how to calibrate such probing results (Voita and Titov, 2020); our model's representations are more directly interpretable and don't require post-hoc probing.

Lastly, our work is distinct from SPINE (Subramanian et al., 2017), a past technique for learning sparse interpretable embeddings. However, this technique requires an additional step to reveal their interpretability. Each dimension of our entity representations has a name (i.e., a fine-grained type) with a probability, and thus it is immediately interpretable.

9 Conclusion

In this work, we presented an approach to creating interpretable entity representations that are human readable and achieve high performance on entity-related tasks out of the box. We show that it is possible to reduce the size of our type set in a learning-based way for particular domains. In addition, these embeddings can be post-hoc modified through simple rules to incorporate domain knowledge and improve performance.

[11] The SOTA performance on the original split is around 40 F1.

Acknowledgments

Thanks to the anonymous reviewers for their helpful comments, members of the UT TAUR lab for helpful discussion, Pengxiang Cheng for constructive suggestions, and Mingda Chen for providing the details of experiments. This work was partially supported by NSF Grant IIS-1814522, NSF Grant SHF-1762299, a gift from Arm, and an equipment grant from NVIDIA. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources used to conduct this research.

This material is also based on research that is in part supported by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory (AFRL), DARPA, or the U.S. Government.

References

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the Blanks: Distributional Similarity for Relation Learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hong Chen, Zhenhua Fan, Hao Lu, Alan Yuille, and Shu Rong. 2018. PreCo: A large-scale dataset in preschool vocabulary for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mingda Chen, Zewei Chu, Yang Chen, Karl Stratos, and Kevin Gimpel. 2019. EntEval: A holistic evaluation benchmark for entity representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-Fine Entity Typing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mark Craven and Johan Kumlien. 1999. Constructing Biological Knowledge Bases by Extracting Information from Text Sources. In *Proceedings of ISMB*.

Shrey Desai and Greg Durrett. 2020. Calibration of Pre-trained Transformers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Greg Durrett and Dan Klein. 2014. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. *Transactions of the Association for Computational Linguistics (TACL)*, 2:477–490.

Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuya Yamada, and Omer Levy. 2017. Named Entity Disambiguation for Noisy Text. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. 2020. Entities as Experts: Sparse Memory Access with Entity Supervision. *ArXiv*, abs/2004.07202.

Thibault Févry, Nicholas FitzGerald, Livio Baldini Soares, and Tom Kwiatkowski. 2020. Empirical evaluation of pretraining strategies for supervised entity linking. In *Automated Knowledge Base Construction (AKBC)*.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-Dependent Fine-Grained Entity Type Tagging. *CoRR*, abs/1412.1820.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning Dense Representations for Entity Retrieval. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity Linking via Joint Encoding of Types, Descriptions, and Context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John Hewitt and Percy Liang. 2019. Designing and Interpreting Probes with Control Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know? *Transactions of the Association for Computational Linguistics (TACL)*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2019. Sensebert: Driving some sense into bert. *ArXiv*, abs/1908.05646.

Jeffrey Ling, Nicholas FitzGerald, Zifei Shan, Livio Baldini Soares, Thibault Févry, D. Weiss, and Tom Kwiatkowski. 2020. Learning Cross-Context Entity Representations from Text. *ArXiv*.

Xiao Ling and Daniel S. Weld. 2012. Fine-Grained Entity Recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

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*, abs/1907.11692.

Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101.

Ken McRae, George S. Cree, Mark S. Seidenberg, , and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, 37(4):547–559.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Yasumasa Onoe and Greg Durrett. 2019. Learning to Denoise Distantly-Labeled Data for Entity Typing. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Yasumasa Onoe and Greg Durrett. 2020. Fine-Grained Entity Typing for Domain Independent Entity Linking. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

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 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2019. BERT is Not a Knowledge Base (Yet): Factual Knowledge vs. Name-Based Reasoning in Unsupervised QA. *ArXiv*, abs/1911.03681.

Jonathan Raiman and Olivier Raiman. 2018. DeepType: Multilingual Entity Linking by Neural Type System Evolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model? *ArXiv*, abs/2002.08910.

Valentin I. Spitkovsky and Angel X. Chang. 2012. A Cross-Lingual Dictionary for English Wikipedia Concepts. In *Proceedings of LREC*.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard H. Hovy. 2017. SPINE: Sparse Interpretable Neural Embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A Core of Semantic Knowledge. In *Proceedings of World Wide Web Conference (WWW)*.

Robert Tibshirani. 1994. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*.

Elena Voita and Ivan Titov. 2020. Information-Theoretic Probing with Minimum Description Length. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Cuihong Cao, Daxin Jiang, and Ming Zhou. 2020. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. *ArXiv*, abs/2002.01808.

Su Wang, Stephen Roller, and Katrin Erk. 2017. Distributional Modeling on a Diet: One-shot Word Learning from Text Only. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*, abs/1910.03771.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv*, abs/1609.08144.

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 (ICLR)*.

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 Conference on Computational Natural Language Learning (CoNLL)*.

Kelly Zhang and Samuel Bowman. 2018. Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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 Annual Meeting of the Association for Computational Linguistics (ACL)*.

Appendix A: Debugging Rules

(See Table 8)

Rule	Types to be set to 1	Types to be set to 0
fed cup is in the context.	women's, tennis, teams, sports	davis cup teams, davis cup
soccer is in the context.	football	uefa member associations
cricket is in the context.	england in international cricket, men's, national cricket teams, in english cricket	women's, women, women's national cricket teams, football
tennis is in the context, and the mention is *washington*.	tennis, living people	cities, in washington (state), in washington, d.c, established, establishments, capital districts and territories, populated, 'places
The mention is *wall street*.	exchanges, stock	streets, tourist
soccer and *1996* are in the context, and the mention is *world-cup*.	1998	1996
baseball and *new york* are in the context, and the mention is *chicago*.	chicago white sox	chicago cubs
yeltsin is in the context, and the mention is *lebed*.	living people, of russia	
venice festival is in the context, and the mention is *jordan*.	living people, people, irish, irish male novelists, 1950 births, male screenwriters, bafta winners (people), writers, for best director winners, people from dublin (city), 20th-century irish novelists	member states of the organisation of islamic cooperation, of the organisation of islamic cooperation, in jordan, territories, countries, states, of the arab league, member, western asian countries, member states of the arab league, member states of the united nations, jordan, tourism
baseball is in the context.	major, baseball	soccer, football, major league soccer, professional sports leagues in canada, professional, in the united states, in canada
squash is in the context, and the mention is *jansher*.	1969 births	1963 births

Table 8: Debugging rules applied for the CoNLL-YAGO development set.