Scalable Zero-shot Entity Linking with Dense Entity Retrieval

Ledell Wu,[1] **Fabio Petroni,**[1] **Martin Josifoski,**[3*] **Sebastian Riedel,**[1,2] **Luke Zettlemoyer**[1]
[1]Facebook AI Research
{ledell, fabiopetroni, sriedel, lsz}@fb.com
[2]University College London
[3]Ecole Polytechnique Federale de Lausanne
martin.josifoski@epfl.ch

Abstract

This paper introduces a conceptually simple, scalable, and highly effective BERT-based entity linking model, along with an extensive evaluation of its accuracy-speed trade-off. We present a two-stage zero-shot linking algorithm, where each entity is defined only by a short textual description. The first stage does retrieval in a dense space defined by a bi-encoder that independently embeds the mention context and the entity descriptions. Each candidate is then re-ranked with a cross-encoder, that concatenates the mention and entity text. Experiments demonstrate that this approach is state of the art on recent zero-shot benchmarks (6 point absolute gains) and also on more established non-zero-shot evaluations (e.g. TACKBP-2010), despite its relative simplicity (e.g. no explicit entity embeddings or manually engineered mention tables). We also show that bi-encoder linking is very fast with nearest neighbour search (e.g. linking with 5.9 million candidates in 2 milliseconds), and that much of the accuracy gain from the more expensive cross-encoder can be transferred to the bi-encoder via knowledge distillation. Our code and models are available at https://github.com/facebookresearch/BLINK.

1 Introduction

Scale is a key challenge for entity linking; there are millions of possible entities to consider for each mention. To efficiently filter or rank the candidates, existing methods use different sources of external information, including manually curated mention tables (Ganea and Hofmann, 2017), incoming Wikipedia link popularity (Yamada et al., 2016), and gold Wikipedia entity categories (Gillick et al., 2019). In this paper, we show that BERT-based models set new state-of-the-art performance levels for large scale entity linking when used in a zero shot setup, where there is no external knowledge and a short text description provides the only information we have for each entity. We also present an extensive evaluation of the accuracy-speed trade-off inherent to large pre-trained models, and show is possible to achieve very efficient linking with modest loss of accuracy.

More specifically, we introduce a two stage approach for zero-shot linking (see Figure 1 for an overview), based on fine-tuned BERT architectures (Devlin et al., 2019). In the first stage, we do retrieval in a dense space defined by a bi-encoder that independently embeds the mention context and the entity descriptions (Humeau et al., 2019; Gillick et al., 2019). Each retrieved candidate is then examined more carefully with a cross-encoder that concatenates the mention and entity text, following Logeswaran et al. (2019). This overall approach is conceptually simple but highly effective, as we show through detailed experiments.

Our two-stage approach achieves a new state-of-the-art result on TACKBP-2010, with an over 30% relative error reduction. By simply reading the provided text descriptions, we are able to outperform previous methods that included many extra cues such as entity name dictionaries and link popularity. We also improve the state of the art on existing zero-shot benchmarks, including a nearly 6 point absolute gain on the recently introduced Wikia corpus (Logeswaran et al., 2019) and more than 7 point absolute gain on WikilinksNED Unseen-Mentions (Onoe and Durrett, 2019).

Finally, we do an extensive evaluation of the accuracy-speed trade-off inherent in our bi- and cross-encoder models. We show that the two stage methods scales well in a full Wikipedia setting, by linking against all the 5.9M Wikipedia entities for TACKBP-2010, while still outperforming existing model with much smaller candidate sets. We

[*]Work done during internship with Facebook.

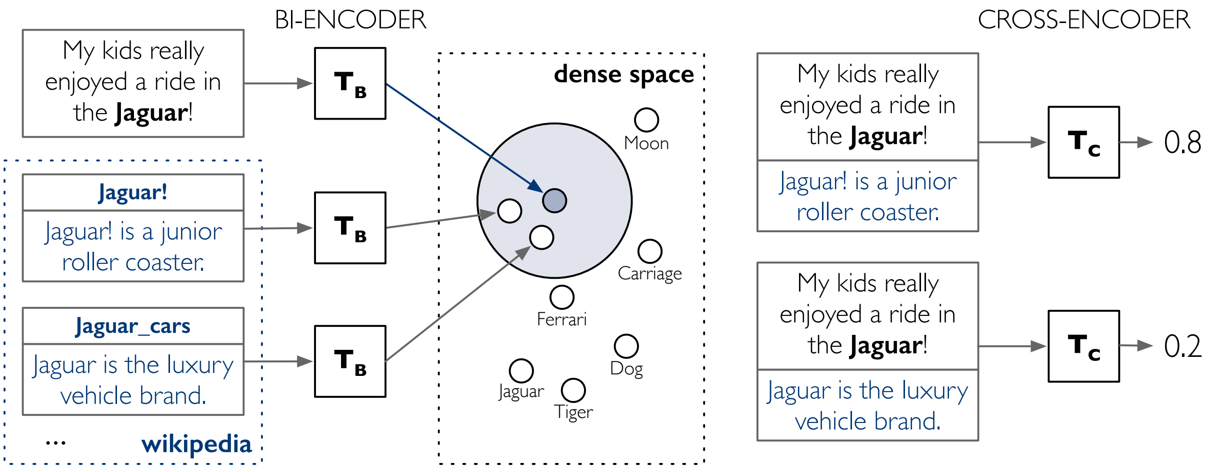
BI-ENCODER
My kids really enjoyed a ride in the Jaguar!
T_B
dense space
Moon
Jaguar!
Jaguar! is a junior roller coaster.
T_B
Carriage
Ferrari
Jaguar_cars
Jaguar is the luxury vehicle brand.
T_B
Dog
Jaguar
Tiger
...
wikipedia
CROSS-ENCODER
My kids really enjoyed a ride in the Jaguar!
Jaguar! is a junior roller coaster.
T_C
0.8
My kids really enjoyed a ride in the Jaguar!
Jaguar is the luxury vehicle brand.
T_C
0.2

Figure 1: High level description of our zero-shot entity linking solution. From the top-left, the input gets encoded in the same dense space where all entities representations lie. A nearest neighbors search is then performed (depicted with a blue circle), k entities retrieved and supplied to the cross encoder. The latter attends over both input text and entities descriptions to produce a probability distribution over the candidates.

also show that bi-encoder linking is very fast with approximate nearest neighbor search (e.g. linking over 5.9 million candidates in 2 milliseconds), and that much of the accuracy gain from the more expensive cross-encoder can be transferred to the bi-encoder via knowledge distillation. We release our code and models, as well as a system to link entity mentions to all of Wikipedia (similar to TagME (Ferragina and Scaiella, 2011)).[1]

2 Related Work

We follow most recent work in studying entity linking with gold mentions.[2] The entity linking task can be broken into two steps: candidate generation and ranking. Prior work has used frequency information, alias tables and TF-IDF-based methods for candidate generation. For candidate ranking, He et al. (2013), Sun et al. (2015), Yamada et al. (2016), Ganea and Hofmann (2017), and Kolitsas et al. (2018) have established state-of-the-art results using neural networks to model context word, span and entity. There is also recent work demonstrating that fine-grained entity typing information helps linking (Raiman and Raiman, 2018; Onoe and Durrett, 2019; Khalife and Vazirgiannis, 2018).

Two recent results are most closely related to our work. Logeswaran et al. (2019) proposed the zero-shot entity linking task. They use cross-encoders for entity ranking, but rely on traditional IR-techniques for candidate generation and did not evaluate on large scale benchmarks such as TACKBP. Gillick et al. (2019) show that dense embeddings work well for candidate generation, but they did not do pre-training and included external category labels in their bi-encoder architectures, limiting their linking to entities in Wikipedia. Our approach can be seen as generalizing both of these lines of work, and showing for the first time that pre-trained zero-shot architectures are both highly accurate and computationally efficient at scale.

Humeau et al. (2019) studied different architectures to use deep pre-trained bidirectional transformers and performed detailed comparison of three different architectures, namely bi-encoder, poly-encoder, cross-encoder on tasks of sentence selection in dialogues. Inspired by their work, we use similar architectures to the problem of entity linking, and in addition, demonstrate that bi-encoder can be a strong model for retrieval. Instead of using the poly-encoder as a trade-off between cross-encoder and bi-encoder, we propose to train a bi-encoder model with knowledge distillation (Buciluundefined et al., 2006; Hinton et al., 2015) from a cross-encoder model to further improve the bi-encoder's performances.

3 Definition and Task Formulation

Entity Linking Given an input text document $\mathbf{D} = \{w_1, ..., w_r\}$ and a list of entity mentions $\mathbf{M_D} = \{m_1, ..., m_n\}$, the output of an entity

[1] Our code and models are available at `https://github.com/facebookresearch/BLINK`

[2] Kolitsas et al. (2018) study end-to-end linking. Our techniques should be applicable to this setting as well, but we leave this exploration to future work.

linking model is a list of mention-entity pairs $\{(m_i, e_i)\}_{i\in[1,n]}$ where each entity is an entry in a knowledge base (KB) (e.g. Wikipedia), $e \in \mathcal{E}$. We assume that the title and description of the entities are available, which is a common setting in entity linking (Ganea and Hofmann, 2017; Logeswaran et al., 2019). We also assume each mention has a valid gold entity in the KB, which is usually referred as *in-KB* evaluation. We leave the out-of-KB prediction (i.e. *nil* prediction) to future work.

Zero-shot Entity Linking We also study zero-shot entity linking (Logeswaran et al., 2019). Here the document setup is the same, but the knowledge base is separated in training and test time. Formally, denote $\mathcal{E}_{train}$ and $\mathcal{E}_{test}$ to be the knowledge base in training and test, we require $\mathcal{E}_{train} \cap \mathcal{E}_{test} = \emptyset$. The set of text documents, mentions, and entity dictionary are separated in training and test so that the entities being linked at test time are unseen.

4 Methodology

Figure 1 shows our overall approach. The bi-encoder uses two independent BERT transformers to encode model context/mention and entity into dense vectors, and each entity candidate is scored as the dot product of these vectors. The candidates retrieved by the bi-encoder are then passed to the cross-encoder for ranking. The cross-encoder encodes context/mention and entity in one transformer, and applies an additional linear layer to compute the final score for each pair.

4.1 Bi-encoder

Architecture We use a bi-encoder architecture similar to the work of Humeau et al. (2019) to model (mention, entity) pairs. This approach allows for fast, real-time inference, as the candidate representations can be cached. Both input context and candidate entity are encoded into vectors:

$$\boldsymbol{y_m} = \text{red}(T_1(\tau_m)) \tag{1}$$

$$\boldsymbol{y_e} = \text{red}(T_2(\tau_e)) \tag{2}$$

where τ_m and τ_e are input representations of mention and entity respectively, T_1 and T_2 are two transformers. $\text{red}(.)$ is a function that reduces the sequence of vectors produced by the transformers into one vector. Following the experiments in Humeau et al. (2019), we choose $\text{red}(.)$ to be the last layer of the output of the `[CLS]` token.

Context and Mention Modeling The representation of context and mention τ_m is composed of the word-pieces of the context surrounding the mention and the mention itself. Specifically, we construct input of each mention example as:

`[CLS]` ctxt$_l$ `[M`$_s$`]` mention `[M`$_e$`]` ctxt$_r$ `[SEP]`

where mention, ctxt$_l$, ctxt$_r$ are the word-pieces tokens of the mention, context before and after the mention respectively, and `[M`$_s$`]`, `[M`$_e$`]` are special tokens to tag the mention. The maximum length of the input representation is a hyperparameter in our model, and we find that small value such as 32 works well in practice (see Appendix A).

Entity Modeling The entity representation τ_e is also composed of word-pieces of the entity title and description (for Wikipedia entities, we use the first ten sentences as description). The input to our entity model is:

`[CLS]` title `[ENT]` description `[SEP]`

where title, description are word-pieces tokens of entity title and description, and `[ENT]` is a special token to separate entity title and description representation.

Scoring The score of entity candidate e_i is given by the dot-product:

$$s(m, e_i) = \boldsymbol{y_m} \cdot \boldsymbol{y_{e_i}} \tag{3}$$

Optimization The network is trained to maximize the score of the correct entity with respect to the (randomly sampled) entities of the same batch (Lerer et al., 2019; Humeau et al., 2019). Concretely, for each training pair (m_i, e_i) in a batch of B pairs, the loss is computed as:

$$\mathcal{L}(m_i, e_i) = -s(m_i, e_i) + \log \sum_{j=1}^{B} \exp\left(s(m_i, e_j)\right) \tag{4}$$

Lerer et al. (2019) presented a detailed analysis on speed and memory efficiency of using batched random negatives in large-scale systems. In addition to in-batch negatives, we follow Gillick et al. (2019) by using hard negatives in training. The hard negatives are obtained by finding the top 10 predicted entities for each training example. We add these extra hard negatives to the random in-batch negatives.

Inference At inference time, the entity representation for all the entity candidates can be pre-computed and cached. The inference task is then reduced to finding maximum dot product between mention representation and entity candidate representations. In Section 5.2.3 we present efficiency/accuracy trade-offs by exact and approximate nearest neighbor search using FAISS (Johnson et al., 2019) in a large-scale setting.

4.2 Cross-encoder

Our cross-encoder is similar to the ones described by Logeswaran et al. (2019) and Humeau et al. (2019). The input is the concatenation of the input context and mention representation and the entity representation described in Section 4.1 (we remove the `[CLS]` token from the entity representation). This allows the model to have deep cross attention between the context and entity descriptions. Formally, we use $y_{m,e}$ to denote our context-candidate embedding:

$$\boldsymbol{y_{m,e}} = \mathrm{red}(T_{\mathrm{cross}}(\tau_{m,e})) \tag{5}$$

where $\tau_{m,e}$ is the input representation of mention and entity, T_{cross} is a transformer and $red(.)$ is the same function as defined in Section 4.1.

Scoring To score entity candidates, a linear layer $\boldsymbol{W}$ is applied to the embedding $\boldsymbol{y_{m,e}}$:

$$s_{\mathrm{cross}}(m, e) = \boldsymbol{y_{m,e}} \boldsymbol{W} \tag{6}$$

Optimization Similar to methods in Section 4.1, the network is trained using a softmax loss to maximize $s_{\mathrm{cross}}(m_i, e_i)$ for the correct entity, given a set of entity candidates (same as in Equation 4).

Due to its larger memory and compute footprint, we use the cross-encoder in a re-ranking stage, over a small set (≤ 100) of candidates retrieved with the bi-encoder. The cross-encoder is not suitable for retrieval or tasks that require fast inference.

4.3 Knowledge Distillation

To better optimize the accuracy-speed trade-off, we also report knowledge distillation experiments that use a cross-encoder as a teacher for a bi-encoder model. We follow Hinton et al. (2015) to use a softmax with temperature where the target distribution is based on the cross-encoder logits.

Concretely, let z be a vector of logits for set of entity candidates and T a temperature, and $\sigma(z, T)$ a (tempered) distribution over the entities with

$$\sigma(z, T) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}. \tag{7}$$

Then the overall loss function, incorporating both distillation and student losses, is calculated as

$$\mathcal{L}_{dist} = \mathcal{H}(\sigma(z_t; \tau), \sigma(z_s; \tau)) \tag{8}$$

$$\mathcal{L}_{st} = \mathcal{H}(e, \sigma(z_s; 1)) \tag{9}$$

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{st} + (1 - \alpha) \cdot \mathcal{L}_{dist} \tag{10}$$

where e is the ground truth label distribution with probability 1 for the gold entity, $\mathcal{H}$ is the cross-entropy loss function, and α is coefficient for mixing distillation and student loss $\mathcal{L}_{st}$. The student logits z_s are the output of the bi-encoder scoring function $s(m, e_i)$, the teacher logits the output of the cross-encoder scoring funcion $s_{\mathrm{cross}}(m, e)$.

5 Experiments

In this section, we perform an empirical study of our model on three challenging datasets.

5.1 Datasets

The Zero-shot EL dataset was constructed by Logeswaran et al. (2019) from Wikia.[3] The task is to link entity mentions in text to an entity dictionary with provided entity descriptions, in a set of domains. There are 49K, 10K, and 10K examples in the train, validation, test sets respectively. The entities in the validation and test sets are from different domains than the train set, allowing for evaluation of performance on entirely unseen entities. The entity dictionaries cover different domains and range in size from 10K to 100K entities.

TACKBP-2010 is widely used for evaluating entity linking systems Ji et al. (2010).[4] Following prior work, we measure *in-KB* accuracy (P@1). There are 1,074 and 1,020 annotated mention/entity pairs derived from 1,453 and 2,231 original news and web documents on training and evaluation dataset, respectively. All the entities are from the TAC Reference Knowledgebase which contains 818,741 entities with titles, descriptions and other meta info.

[3] `https://www.wikia.com.`

[4] `https://tac.nist.gov`

Method	Train	Validation	Test
BM25	76.86	76.22	69.13
Ours (bi-encoder)	93.12	91.44	82.06

Table 1: Recall@64 (%) on Zero-shot EL dataset, for the BM25 approach and our dense space bi-encoder based retrieval. Results on Train/Valideation/Test set reported.

WikilinksNED Unseen-Mentions was created by Onoe and Durrett (2019) from the original WikilinksNED dataset (Eshel et al., 2017), which contains a diverse set of ambiguous entities spanning a variety of domains. In the Unseen-Mentions version, no mentions in the validation and test sets appear in the training set. The train, validation and test sets contain 2.2M, 10K, and 10K examples respectively. In this setting, the definition of unseen-mentions is different from that in zero-shot entity linking: entities in the test set can be seen in the training set. However, in both definitions no (mention, entity) pairs from test set are observed in the training set. In the unseen-mentions test set, about 25% of the entities appear in training set.

5.2 Evaluation Setup and Results

We experiment with both BERT-base and BERT-large (Devlin et al., 2019) for our bi-encoders and cross-encoders. The details of training infrastructure and hyperparameters can be found in Appendix A. All models are implemented in PyTorch[5] and optimizied with Adam (Kingma and Ba, 2014). We use (base) and (large) to indicate the version of our model where the underlying pretrained transformer model is BERT-base and BERT-large, respectively.

5.2.1 Zero-shot Entity Linking

First, we train our bi-encoder on the training set, initializing each encoder with pre-trained BERT base. Hyper-parameters are chosen based on Recall@64 on validation datase. For specifics, see Appendix A.2. Our bi-encoder achieves much higher recall than BM25, as shown in Figure 2. Following Logeswaran et al. (2019), we use the top 64 retrieved candidates for the ranker, and we report Recall@64 on train, validation and test in Table 1.

After training the bi-encoder for candidate generation, we train our cross-encoder (initialized with pre-trained BERT) on the top 64 retrieved candidates from bi-encoder for each sample on the training set, and evaluate the cross-encoder on the test dataset. Overall, we are able to obtain a much better end-to-end accuracy, as shown in Table 2, largely due to the improvement on the retrieval stage.

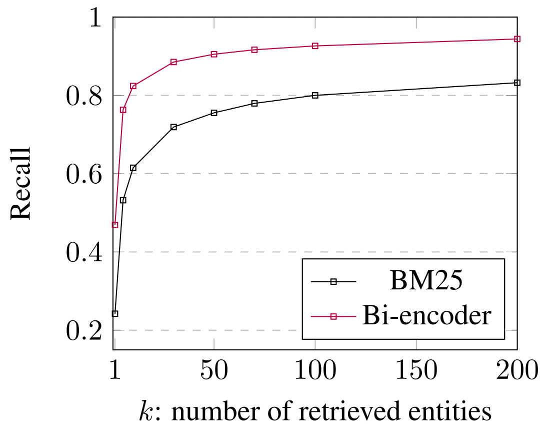

Figure 2: Top-k entity retrieval recall on validation dataset of Zero-shot EL dataset

Method	U.Acc.
Logeswaran et al. (2019)	55.08
Logeswaran et al. (2019)(domain)[†]	56.58
Ours (base)	61.34
Ours (large)	63.03

Table 2: Performance on test domains on the Zero-shot EL dataset. U.Acc. represents the unnormalized accuracy. † indicates model trained with domain adaptive pre-training on source and target domain. Average performance across a set of worlds is computed by macro-averaging.

We also report cross-encoder performance on the same retrieval method (BM25) used by Logeswaran et al. (2019) in Table 3, where the performance is evaluated on the subset of test instances for which the gold entity is among the top 64 candidates retrieved by BM25. We observe that our cross-encoder obtains slightly better results than reported by Logeswaran et al. (2019), likely due to implementation and hyper-parameter details.

5.2.2 TACKBP-2010

Following prior work (Sun et al., 2015; Cao et al., 2018; Gillick et al., 2019; Onoe and Durrett, 2019), we pre-train our models on Wikipedia[6] data. Data and model training details can be found in Appendix A.1.

[5]https://pytorch.org

[6]https://www.wikipedia.org/

Method	Valid	Test
TF-IDF†	26.06	
Ganea and Hofmann (2017)†	26.96	-
Gupta et al. (2017)†	27.03	-
Logeswaran et al. (2019)	76.06	75.06
Ours (base)	78.24	76.58

Table 3: Normalized accuracy on validation and test set on Zero-shot EL, where the performance is evaluated on the subset of test instances for which the gold entity is among the top-k candidates retrieved during candidate generation. † indicates methods re-implemented by Logeswaran et al. (2019).

After training our model on Wikipedia, we fine-tune the model on the TACKBP-2010 training dataset. We use the top 100 candidates retrieved by the bi-encoder as training examples for the cross-encoder, and chose hyper-parameters based on cross validation. We report accuracy results in Table 4. For ablation studies, we also report the following versions of our model:

1. bi-encoder only: we use bi-encoder for candidate ranking instead of cross-encoder.

2. Full Wikipedia: we use 5.9M Wikipedia articles as our entity Knowlegebase, instead of TACKBP Reference Knowledgebase.

3. Full Wikipedia w/o finetune: same as above, without fine-tuning on the TACKBP-2010 training set.

As expected, the cross-encoder performs better than the bi-encoder on ranking. However, both models exceed state-of-the-art performance levels, demonstrating that the overall approach is highly effective. We observe that our model also performs well when we change the underlying Knowledgebase to full Wikipedia, and even without fine-tuning on the dataset. In Table 5 we show that our bi-encoder model is highly effective at retrieving relevant entities, where the underlying Knowledgebase is full Wikipedia.

There are however many other cues that could potentially be added in future work. For example, Khalife and Vazirgiannis (2018) report 94.57% precision on the TACKBP-2010 dataset. However, their method is based on the strong assumption that a gold fine-grained entity type is given for each mention (and they do not attempt to do entity type prediction). Indeed, if fine-grained entity type information is given by an oracle at test time, then Raiman and Raiman (2018) reports 98.6% accuracy on TACKBP-2010, indicating that improving fine-grained entity type prediction would likely improve entity linking. Our results is achieved without gold fine-grained entity type information. Instead, our model learns representations of context, mention and entities based on text only.

Method	Accuracy
He et al. (2013)	81.0
Sun et al. (2015)	83.9
Yamada et al. (2016)†	85.5
Globerson et al. (2016)†	87.2
Sil et al. (2018)	87.4
Nie et al. (2018)†	89.1
Raiman and Raiman (2018)	90.9
Cao et al. (2018)†	91.0
Gillick et al. (2019)	87.0
Ours	94.5
Ours (bi-encoder only)	92.9
Ours (full Wiki)	92.8
Ours (full Wiki, w/o finetune)	91.5

Table 4: Accuracy scores of our proposed model and models from prior work on TACKBP-2010. † indicates methods doing *global* resolution of all mentions in a document. Our work focuses on *local* resolution where each mention is modeled independently.

Method	Recall@100
AT-Prior†	89.5
AT-Ext†	91.7
BM25†	68.9
Gillick et al. (2019)	96.3
Ours (full wiki)	98.3

Table 5: Retrieval evaluation comparison for TACKBP-2010. † indicates alias table and BM25 baselines implemented by (Gillick et al., 2019). AT-Prior: alias table ordered by prior probabilities; AT-Ext: alias table extended with heuristics.

5.2.3 WikilinksNED Unseen-Mentions

Similarly to the approach described in Section 5.2.2, we train our bi-encoder and cross-encoder model first on Wikipedia examples, then fine-tune on the training data from this dataset. We also present our model trained on Wikipedia examples

Method	**Training**	**Test**
MOST FREQUENT	Wiki	54.1
COSINE SIMILARITY	Wiki	21.7
GRU+ATTN (Mueller and Durrett, 2018)	in-domain	41.2
GRU+ATTN	Wiki	43.4
CBoW+WORD2VEC	in-domain	43.0
CBoW+WORD2VEC	Wiki	38.0
Onoe and Durrett (2019)	Wiki	62.2
Ours	in-domain	74.7
Ours	Wiki	75.2
Ours	Wiki (bi-encoder)	71.5
Ours	Wiki and in-domain	76.8

Table 6: Accuracy on the WikilinksNED Unseen-Mentions test set. The numbers of baseline models are from (Onoe and Durrett, 2019). The column **Training** indicates the source of data used in training: *Wiki* means Wikipedia examples; *in-domain* means examples in the training set.

and applied directly on the test set as well as our model trained on this dataset directly without training on Wikipedia examples. We report our models' performance of accuracy on the test set in Table 6, along with baseline models presented from Onoe and Durrett (2019). We observe that our model out-performs all the baseline models.

Inference time efficiency To illustrate the efficiency of our bi-encoder model, we profiled retrieval speed on a server with Intel Xeon CPU E5-2698 v4 @ 2.20GHz and 512GB memory. At inference time, we first compute all entity embeddings for the pool of 5.9M entities. This step is resource intensive but can be paralleled. On 8 Nvidia Volta v100 GPUs, it takes about 2.8 hours to compute all entity embeddings. Given a query of mention embedding, we use FAISS (Johnson et al., 2019) IndexFlatIP index type (exact search) to obtain top 100 entity candidates. On the WikilinksNED Unseen-Mentions test dataset which contains 10K queries, it takes 9.2 ms on average to return top 100 candidates per query in batch mode.

We also explore the approximate search options using FAISS. We choose the IndexHNSWFlat index type following Karpukhin et al. (2020). It takes additional time in index construction while reduces the average time used per query. In Table 7, we see that $HNSW_1$[7] reduces the average query time to 2.6 ms with less than 1.2% drop in accuracy and recall, and $HNSW_2$[8] further reduce the query time to 1.4 ms with less than 2.1% drop.

Method	Acc	R@10	R@30	R@100	ms/q
Ex. Search	71.5	92.7	95.4	96.7	9.2
$HNSW_1$	71.1	91.6	94.2	95.5	2.6
$HNSW_2$	70.7	91.0	93.9	94.6	1.4

Table 7: Exact and approximate candidate retrieval using FAISS. Last column: average time per query (ms).

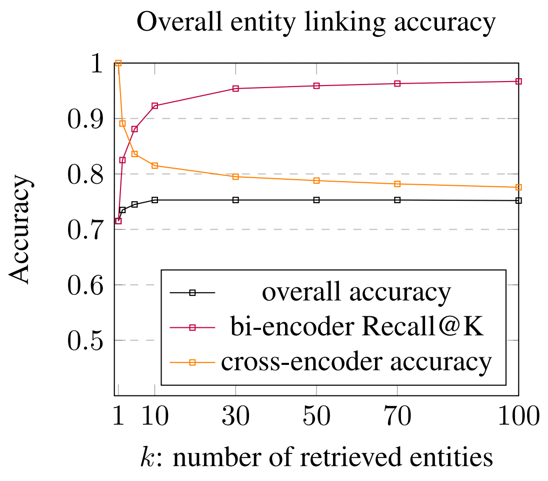

Figure 3: Overall model accuracy based on different choices of k (number of retrieved entities from bi-encoder), on the Unseen-Mentions dataset.

Influence of number of candidates retrieved In a two-stage entity linking systems, the choice of number of candidates retrieved influences the overall model performance. Prior work often used a fixed number of k candidates where k ranges from 5 to 100 (for instance, Yamada et al. (2016) and Ganea and Hofmann (2017) choose $k = 30$, (Logeswaran et al., 2019) choose $k = 64$). When k is larger, the recall accuracy increases, however, the ranking stage accuracy is likely to decrease. Further, increasing k would often increase the run-time on the ranking stage. We explore different choices of k in our model, and present the $recall@K$ curve, ranking stage accuracy and overall accuracy in Figure 3. Based on the overall accuracy, we found that $k = 10$ is optimal.

5.3 Knowledge Distillation

In this section, we present results on knowledge distillation, using our cross-encoder as a teacher model and bi-encoder as a student model.

[7]Neighbors to store per node: 128, construction time search depth: 200, search depth: 256; construction time: 2.1h.

[8]Neighbors to store per node: 128, construction time search depth: 200, search depth: 128; construction time: 1.8h.

Mention	Bi-encoder	Cross-encoder
But surely the biggest surprise is **Ronaldo**'s drop in value, despite his impressive record of 53 goals and 14 assists in 75 appearances for Juventus.	Ronaldo (Brazilian footballer)	**Cristiano Ronaldo**
... they spent eleven days in the United Kingdom and Spain, photographing things like **Gothic** statues, bricks, and stone pavements for use in textures.	Gothic fiction	**Gothic art**
To many people in many cultures, music is an important part of their way of life. **Ancient Greek** and Indian philosophers defined music as tones ...	**Acient Greek**	Ancient Greek philosophy

Table 8: Examples of top entities predicted by Bi-encoder model and Cross-encoder model. Mentions in the examples are written in **ornage** and the correct entity prediction in **bold**.

We experiment knowledge distillation on the TACKBP-2010 and the WikilinksNED Unseen-Mentions dataset. We use the bi-encoder pretrained on Wikipedia as the student model, and fine-tune it on each dataset with knowledge distillation from the teacher model, which is the best performing cross-encoder model pretrained on Wikipedia and fine-tuned on the dataset.

We also fine-tune the student model in our experiments on each dataset, without the knowledge distillation component, as baseline models. As we can see in Table 9, the bi-encoder model trained with knowledge distillation from cross-encoder outperforms the bi-encoder without knowledge distillation, providing another point in the accuracy-speed trade-off curve for these architectures.

Dataset	bi-encoder	teacher	bi-encoder-KD
Unseen	74.4	76.8	75.7
TAC2010	92.9	94.5	93.5

Table 9: Knowledge Distillation Results. The teacher model is the cross-encoder, and bi-encoder-KD is the bi-encoder model trained with knowledge distillation.

6 Qualitative Analysis

Table 8 presents some examples from our bi-encoder and cross-encoder model predictions, to provide intuition for how these two models consider context and mention for entity linking.

In the first example, we see that the bi-encoder mistakenly links "Ronaldo" to the Brazilian football player, while the cross-encoder is able to use context word "Juventus" to disambiguate. In the second example, the cross-encoder is able to identify from context that the sentence is describing art instead of fiction, where the bi-encoder failed. In the third example, the bi-encoder is able to find the correct entity "Ancient Greek,"; where the cross-encoder mistakenly links it to the entity "Ancient Greek philosophy," likely because that the word "philosophers" is in context. We observe that cross-encoder is often better at utilizing context information than bi-encoder, but can sometimes make mistakes because of misleading context cues.

7 Conclusion

We proposed a conceptually simple, scalable, and highly effective two stage approach for entity linking. We show that our BERT-based model outperforms IR methods for entity retrieval, and achieved new state-of-the-art results on recently introduced zero-shot entity linking dataset, WikilinksNED Unseen-Mentions dataset, and the more established TACKBP-2010 benchmark, without any task-specific heuristics or external entity knowledge. We present evaluations of the accuracy-speed trade-off inherent to large pre-trained models, and show that it is possible to achieve efficient linking with modest loss of accuracy. Finally, we show that knowledge distillation can further improve bi-encoder model performance. Future work includes:

- Enriching entity representations by adding entity type and entity graph information;
- Modeling coherence by jointly resolving mentions in a document;
- Extending our work to other languages and other domains;
- Joint models for mention detection and entity linking.

Acknowledgements

We thank our colleagues Marjan Ghazvininejad, Kurt Shuster, Terra Blevins, Wen-tau Yih and Jason Weston for fruitful discussions.

References

Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA. Association for Computing Machinery.

Yixin Cao, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Neural collective entity linking. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 675–686.

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.

Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuya Yamada, and Omer Levy. 2017. Named entity disambiguation for noisy text. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 58–68, Vancouver, Canada. Association for Computational Linguistics.

Paolo Ferragina and Ugo Scaiella. 2011. Fast and accurate annotation of short texts with wikipedia pages. *IEEE software*, 29(1):70–75.

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, Copenhagen, Denmark. Association for Computational Linguistics.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. *arXiv preprint arXiv:1909.10506*.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 621–631.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–34, Sofia, Bulgaria. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *stat*, 1050:9.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)*, volume 3, pages 3–3.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering.

Sammy Khalife and Michalis Vazirgiannis. 2018. Scalable graph-based individual named entity identification. *CoRR*, abs/1811.10547.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529, Brussels, Belgium. Association for Computational Linguistics.

Adam Lerer, Ledell Wu, Jiajun Shen, Timothée Lacroix, Luca Wehrstedt, Abhijit Bose, and Alexander Peysakhovich. 2019. Pytorch-biggraph: A large-scale graph embedding system. In *Proceedings of the 2nd SysML Conference*.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460, Florence, Italy. Association for Computational Linguistics.

David Mueller and Greg Durrett. 2018. Effective use of context in noisy entity linking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1024–1029.

Feng Nie, Yunbo Cao, Jinpeng Wang, Chin-Yew Lin, and Rong Pan. 2018. Mention and entity description co-attention for entity disambiguation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yasumasa Onoe and Greg Durrett. 2019. Fine-grained entity typing for domain independent entity linking. *arXiv preprint arXiv:1909.05780*.

Jonathan Raphael Raiman and Olivier Michel Raiman. 2018. Deeptype: multilingual entity linking by neural type system evolution. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual entity linking. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

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, Berlin, Germany. Association for Computational Linguistics.

A Training details and hyper-parameters Optimization

- Computing infrastructure: we use 8 Nvidia Volta v100 GPUs for model training.
- Bounds for each hyper parameter: see Table 10. In addition, for our bi-encoders, we use a max number of tokens of $[32, 64, 128]$ for context/mention encoder and 128 for candidate encoder. In our knowledge distillation experiments, we set $\alpha = 0.5$, and T in $[2, 5]$. We use grid search for hyperparameters, for a total number of 24 trials.
- Number of model parameters: see Table 11.
- For all our experiments we use accuracy on validation set as criterion for selecting hyper-parameters.

Parameter	Bounds
Learning rate	$[2e^{-6}, 5e^{-6}, 1e^{-5}, 2e^{-5}]$
Bi-encoder batch size	$[128, 256]$
Cross-encoder batch size	$[1, 5]$

Table 10: Bounds of hyper-parameters in our models

Model	Number of parameters
Bi-encoder (base)	220M
Cross-encoder (base)	110M
Bi-encoder (large)	680M
Cross-encoder (large)	340M

Table 11: Number of parameters in our models

A.1 Training on Wikipedia data

We use Wikipedia data to train our models first, then fine-tune it on specific dataset. This approach is used in our experiments on TACKBP-2010 and WikilinksNED Unseen-Mentions datasets.

We use the May 2019 English Wikipedia dump which includes 5.9M entities, and use the hyperlinks in articles as examples (the anchor text is the mention). We use a subset of all Wikipedia linked mentions as our training data for the bi-encoder model (A total of 9M examples). We use a holdout set of 10K examples for validation. We train our cross-encoder model based on the top 100 retrieved results from our bi-encoder model on Wikipedia data. For the training of the cross-encoder model, we further down-sample our training data to obtain a training set of 1M examples.

Bi-encoder (large) model Hyperparameter configuration for best model: learning rate=$1e^{-5}$, batch size=128, max context tokens=32. Average runtime for each epoch: 17.5 hours/epoch, trained on 4 epochs.

Cross-encoder (large) model Hyperparameter configuration for best model: learning rate=$2e^{-5}$, batch size=1, max context tokens=32. Average runtime for each epoch: 37.2 hours/epoch, trained on 1 epoch.

A.2 Zero-shot Entity Linking Dataset

Dataset available at https://github.com/lajanugen/zeshel. There are 49K, 10K, and 10K examples in the train, validation, test sets respectively. Training details:

Bi-encoder (base) model Hyperparameter configuration for best model: learning rate=$2e^{-5}$, batch size=128, max context tokens=128. Average runtime: 28.2 minutes/epoch, trained on 5 epochs.

Bi-encoder (large) model Hyperparameter configuration for best model: learning rate=$1e^{-5}$, batch size=128, max context tokens=128. Average runtime: 38.2 minutes/epoch, trained on 5 epochs.

Cross-encoder (base) model Hyperparameter configuration for best model: learning rate=$1e^{-5}$, batch size=1, max context tokens=128. Average runtime: 2.6 hours/epoch, trained on 2 epochs.

Cross-encoder (large) model Hyperparameter configuration for best model: learning rate=$1e^{-5}$, batch size=1, max context tokens=128. Average runtime: 8.5 hours/epoch, trained on 2 epochs.

A.3 TACKBP-2010 Dataset

Dataset available at https://catalog.ldc.upenn.edu/LDC2018T16. There are 1,074 and 1,020 annotated examples in the train and test sets respectively. We use a 10-fold cross-validation from training set. Training details:

Bi-encoder (large) model Hyperparameter configuration for best model: learning rate=$2e^{-6}$, batch size=128, max context tokens=32. Average runtime: 9.0 minutes/epoch, trained on 10 epochs.

Bi-encoder (large) model with Knowledge Distillation Hyperparameter configuration for best model: learning rate=$2e^{-5}$, batch size=128, max context tokens=32, $T = 2, \alpha = 0.5$. Average runtime: 11.2 minutes/epoch, trained on 10 epochs.

Cross-encoder (large) model Hyperparameter configuration for best model: learning rate=$1e^{-5}$, batch size=1, max context tokens=128. Average runtime: 20.4 minutes/epoch, trained on 10 epochs.

A.4 WikilinksNED Unseen-Mentions Dataset

The train, validation and test sets contain 2.2M, 10K, and 10K examples respectively. We use a subset of 100K examples to fine-tune our model on this dataset, as we found more examples do not help. Training details:

Bi-encoder (large) model Hyperparameter configuration for best model: learning rate=$2e^{-6}$, batch size=128, max context tokens=32. Average runtime for each epoch: 3.2 hours/epoch, trained on 1 epochs.

Bi-encoder (large) model with Knowledge Distillation Hyperparameter configuration for best model: learning rate=$5e^{-6}$, batch size=128, max context tokens=32, $T = 2, \alpha = 0.5$. Average runtime: 6.5 hours/epoch, trained on 1 epochs.

Cross-encoder (large) model Hyperparameter configuration for best model: learning rate=$2e^{-6}$, batch size=5, max context tokens=128. Average runtime: 4.2 hours/epoch, trained on 1 epochs.